REFLECT-THEN-PLAN: OFFLINE MODEL-BASED PLANNING THROUGH A *Doubly Bayesian* LENS

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline reinforcement learning (RL) is essential when online exploration is costly or unsafe, but it often struggles with high epistemic uncertainty due to limited data. Existing methods learn fixed conservative policies, which limit adaptivity and generalization. To tackle these challenges, we propose **Reflect-then-Plan (RefPlan)**, a novel *doubly Bayesian* approach for offline model-based (MB) planning that enhances offline-learned policies for improved adaptivity and generalization. RefPlan integrates uncertainty modeling and MB planning in a unified probabilistic framework, recasting planning as Bayesian posterior estimation. During deployment, it updates a belief distribution over environment dynamics based on real-time observations. By incorporating this uncertainty into MB planning via marginalization, RefPlan derives plans that account for unknowns beyond the agent's limited knowledge. Empirical results on standard benchmarks show that RefPlan significantly improves the performance of conservative offline RL policies. In particular, RefPlan maintains robust performance under high epistemic uncertainty and limited data, while demonstrating resilience to changing environment dynamics, improving the flexibility, generalizability, and robustness of offline-learned policies.

1 INTRODUCTION

Recent years have seen significant progress in offline reinforcement learning (RL), in which a learner has to learn a performant policy from a static dataset of experiences (Levine et al., 2020; Kumar et al., 2020; An et al., 2021; Kostrikov et al., 2022). This is particularly appealing in scenarios where online exploration is costly or unsafe (Yu et al., 2018; Kalashnikov et al., 2018; Boute et al., 2022).

The agent's inability to gather more experiences have severe implications. In particular, it becomes practically impossible to precisely identify the true Markov decision process (MDP) with a limited dataset, as it only covers a portion of the entire state-action space, leading to high *epistemic uncertainty* for states and actions outside the data distribution. Most offline RL methods aim to learn a conservative policy that stays close to the data distribution, thus steering away from high epistemic uncertainty.

While incorporating conservatism into offline learning has proven effective (Jin et al., 2021; Yu et al., 2020; Kumar et al., 2020), it can result in overly restrictive policies that lack generalizability. Most methods learn a Markovian policy that relies solely on the current state, leading the agent to potentially take poor actions in unexpected states during evaluation. Model-based (MB) planning can enhance the agent's responsiveness during evaluation (Sikchi et al., 2021; Argenson & Dulac-Arnold, 2021; Zhan et al., 2022), but it still primarily addresses epistemic uncertainty through conservatism.

Noting this challenge, Chen et al. (2021) and Ghosh et al. (2022) propose to learn an *adaptive* policy that can reason about the environment and accordingly react at evaluation. Essentially, they formulate the offline RL problem as a partially observable MDP (POMDP)—where the partial observability relates to the agent's epistemic uncertainty, aka *Epistemic POMDP* (Ghosh et al., 2021). Thus, learning an adaptive policy involves approximately inferring the *belief state* from the history of transitions experienced by the agent and allowing the policy to condition on this belief state.

While learning an adaptive policy can help make the agent more flexible and generalizable, it still heavily depends on the training phase. Our empirical evaluation demonstrates that a learned policy—whether it be adaptive or fixed—can be significantly strengthened by incorporating MB planning.

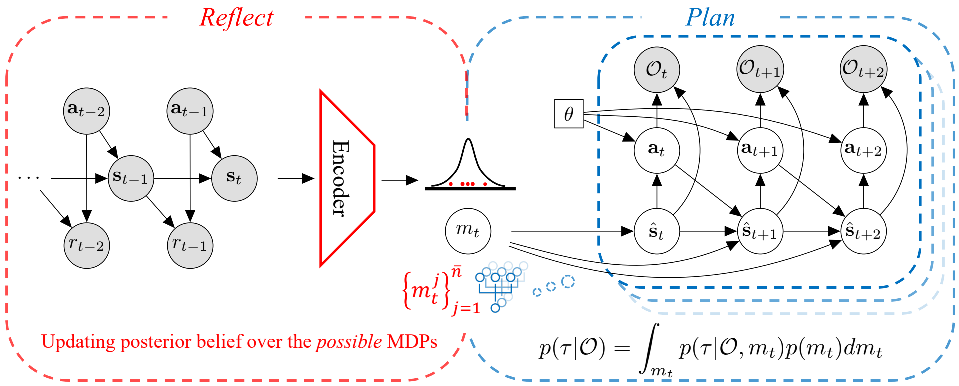


Figure 1: Schematic illustration of RefPlan. (*Reflect*) At time t, RefPlan utilizes real-time agent experiences $\tau_{:t} = (\mathbf{s}_0, \mathbf{a}_0, r_0, \ldots, \mathbf{s}_t)$ to infer the posterior belief m_t over environments using a variational autoencoder. Unlike prior methods, RefPlan learns diverse dynamics models conditioned on m_t, capturing different transition and reward functions. (*Plan*) Offline planning is framed as probabilistic inference, where the posterior over optimal plans $p(\tau|\mathcal{O})$ (with $\mathcal{O}$ denoting optimality variables in the control-as-inference framework) is inferred. A prior $p(\tau)$ is incorporated by learning π_θ via offline policy learning. By marginalizing m_t via Monte Carlo sampling, RefPlan addresses epistemic uncertainty, enhancing π_θ for better adaptivity and generalizability.

However, existing MB planning methods fall short in adequately addressing the agent's epistemic uncertainty, and it remains elusive how one can effectively incorporate the uncertainty into planning.

We propose ***Reflect-then-Plan (RefPlan)***, a novel *doubly Bayesian* approach for offline MB planning. RefPlan combines epistemic uncertainty modeling with MB planning in a unified probabilistic framework, inspired by the control-as-inference paradigm (Levine, 2018). RefPlan adapts Bayes-adaptive deep RL techniques (Zintgraf et al., 2020; Dorfman et al., 2021) to infer a posterior belief distribution from past experiences during test time (*Reflect*). To harness this uncertainty for planning, we recast planning as Bayesian posterior estimation (*Plan*). By marginalizing over the agent's epistemic uncertainty, RefPlan effectively considers a range of possible scenarios beyond the agent's immediate knowledge, resulting in a posterior distribution over optimized plans under the learned model (Figure 1).

In our experiments, we demonstrate that RefPlan can be integrated with various offline RL policy learning algorithms to consistently boost their test-time performance in standard offline RL benchmark domains (Fu et al., 2020). RefPlan not only maintains robust performance under high epistemic uncertainty but also shows superior resilience when the environment dynamics change or when data availability is limited, outperforming compared methods in these challenging scenarios.

2 Related Work

Offline RL In offline RL, policy distribution shift is a major challenge, leading to instabilities like extrapolation errors and value overestimation (Kumar et al., 2019; Fujimoto et al., 2019). To address this, various approaches introduce conservatism. For instance, policy constraint methods constrain the learned policy's deviation from the behavior policy (Wu et al., 2019; Kumar et al., 2019; Fujimoto & Gu, 2021). Value-based approaches penalize the values of out-of-distribution (OOD) actions (Kumar et al., 2020; An et al., 2021). One can also avoid querying OOD actions by learning the value function solely from in-dataset samples and distilling a policy (Kostrikov et al., 2022).

MB offline policy learning methods learn a dynamics model from batch data, then use the model to generate imaginary rollouts to augment the offline dataset. To mitigate the risk of exploiting errors in the model for policy optimization, model uncertainty—heuristically estimated from ensemble dynamics models—can be penalized in rewards (Yu et al., 2020; Kidambi et al., 2021; Lu et al., 2021). Alternatively, values of model-generated samples can be minimized (Yu et al., 2021). Adversarial dynamics models can also discourage the learner from choosing OOD actions (Rigter et al., 2022).

Typically, these offline policies are fixed after training, but Ghosh et al. (2021; 2022) show that fixed policies can fail under high epistemic uncertainty, highlighting the need for adaptive policies. APE-V (Ghosh et al., 2022) addresses this by maintaining a value ensemble to approximate the distribution over possible environments, adapting the policy based on this ensemble during evaluation. MAPLE (Chen et al., 2021) uses an RNN to encode the agent's history into a dense vector, allowing the policy to adapt by conditioning on this history. MAPLE also utilizes an ensemble dynamics model to expose the adaptive policy to diverse simulated environments, enhancing its robustness to uncertainty.

Model-based planning for offline RL MB planning can add responsiveness at test time. For example, MBOP (Argenson & Dulac-Arnold, 2021) uses model predictive control (MPC) with MPPI (Williams et al., 2015), a trajectory optimization (TrajOpt) method, modifying it for offline setups by using a behavior-cloning (BC) policy for trajectory generation. Uncertain rollouts can be filtered out based on the ensemble disagreement (Zhan et al., 2022).

LOOP (Sikchi et al., 2021) enhances offline-learned policies with MB planning, achieving superior performance than MBOP. It approaches offline MB planning using KL-regularized optimization but only addresses epistemic uncertainty by penalizing ensemble variance in rewards during TrajOpt. In contrast, RefPlan is derived from a Bayesian perspective, which explicitly accounts for the agent's epistemic uncertainty, resulting in better generalization and stronger performance.

Probabilistic interpretation of MB planning The control-as-inference framework (Levine, 2018; Abdolmaleki et al., 2018) offers a probabilistic perspective on control and RL problems. Within the context of MB planning, this framework naturally leads to sampling-based solutions (Piché et al., 2019; Okada & Taniguchi, 2020). For instance, Okada & Taniguchi (2020) demonstrated that various sampling-based TrajOpt algorithms can be derived from this probabilistic view. Janner et al. (2022) introduced a diffusion-based planner that utilizes the control-as-inference framework to derive a perturbation distribution, embedding reward signals into the diffusion sampling process. However, to the best of our knowledge, we are the first to propose an offline MB planning algorithm that integrates an offline-learned policy as a prior within a Bayesian framework and explicitly accounts for the epistemic uncertainty during planning, all within a unified probabilistic formulation.

Bayesian RL and epistemic POMDP Bayesian RL (Ghavamzadeh et al., 2015) and Bayes-adaptive MDP (BAMDP) (Duff, 2002) tackle the problem of learning optimal policies in unknown MDPs. A BAMDP can be reformulated as a belief-state MDP, where the belief state acts as a sufficient statistic summarizing the agent's history (Guez et al., 2012). This belief-state representation highlights BAMDP as a specific instance of a POMDP (Kaelbling et al., 1998). Building on this, Zintgraf et al. (2020) framed meta-RL as a BAMDP and proposed VariBAD, a variational inference-based method for approximating the belief distribution over possible environments, enabling the optimization of meta-policies.

Relatedly, Ghosh et al. (2021) introduced the concept of *epistemic POMDP*, where an agent's epistemic uncertainty—stemming from factors such as incomplete exploration or ambiguity in task specification—induces partial observability. Unlike BAMDPs, which primarily focus on online learning and asymptotic regrets, epistemic POMDPs emphasize the agent's performance during a single evaluation episode, making them especially relevant for test-time generalization. Notably, Ghosh et al. (2022) observed that offline RL problems in a single-task setting can also be conceptualized as epistemic POMDPs. This arises because static offline datasets typically cover only a subset of the state-action space, introducing partial observability regarding true environment dynamics outside the offline data distribution.

In this work, we similarly adopt the epistemic POMDP perspective for addressing single-task offline RL. However, unlike prior approaches, our focus is on MB planning. Specifically, we aim to enhance policies learned through offline RL by addressing the agent's epistemic uncertainty, thereby enabling more effective generalization during deployment.

3 PRELIMINARIES

We study RL in the framework of *Markov decision processes* (MDPs) that are characterized by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, d_0, \gamma)$. The state and action spaces ($\mathcal{S}$ and $\mathcal{A}$, respectively) are continuous,

$T(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ is the transition probability distribution, $r(\mathbf{s}, \mathbf{a})$ is the reward function, d_0 is the initial state distribution, and $\gamma \in [0, 1]$ is the discount factor. The *model* of the environment refers to the transition and reward functions. The goal of RL is to find an optimal policy π^* which maximizes the expected discounted return, $\mathbb{E}_{\mathbf{s}_0 \sim d_0, \mathbf{s}_t \sim T, \mathbf{a}_t \sim \pi^*}[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$.

Offline MB planning In offline RL, we have a dataset $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}_{i=1}^{N}$ collected by some behavior policy β. MB methods learn a parameterized predictive model $\hat{p}_\psi(\mathbf{s}', r|\mathbf{s}, \mathbf{a})$, usually trained via maximum likelihood estimation (MLE) to minimize $L(\psi) = \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}',r)\sim\mathcal{D}}[-\log \hat{p}_\psi(\mathbf{s}', r|\mathbf{s}, \mathbf{a})]$. Imaginary data $\mathcal{D}_{\text{model}}$ sampled by $\hat{p}_\psi$ can be used together with $\mathcal{D}$ for offline policy learning.

In this work, however, our focus is on using learned models for planning at test time. MB planning methods commonly use MPC, where at each time step, a TrajOpt method *re-plans* and optimizes the action sequence $\mathbf{a}^*_{t:t+H}$ to maximize the expected H-step return under the learned model $\hat{p}_\psi$, while incorporating a value function V_ϕ to account for long-term rewards (Lowrey et al., 2018). I.e.,

$$\mathbf{a}^*_{t:t+H} = \underset{\mathbf{a}_{t:t+H}}{\arg\max}\ \mathbb{E}_{\hat{p}_\psi}\left[R_H(\mathbf{s}_t, \mathbf{a}_{t:t+H})\right], \tag{1}$$

where $R_H(\mathbf{s}_t, \mathbf{a}_{t:t+H}) := \sum_{h=0}^{H-1} \gamma^h \hat{r}_\psi(\hat{\mathbf{s}}_{t+h}, \mathbf{a}_{t+h}) + \gamma^H V_\phi(\hat{\mathbf{s}}_{t+H})$ is the return of a candidate action sequence $\mathbf{a}_{t:t+H} = (\mathbf{a}_t, \ldots, \mathbf{a}_{t+H-1})$ under $\hat{p}_\psi$.

MPPI (Williams et al., 2015; Nagabandi et al., 2019) is a TrajOpt algorithm that samples $\bar{N}$ plans, $\{\mathbf{a}^n_{t:t+H}\}_{n=1}^{\bar{N}}$, and weighs them by their MB returns using a softmax with inverse temperature κ, giving higher weights to higher-return trajectories. The optimized action is $\mathbf{a}^*_{t+h} = \frac{\sum_{n=1}^{\bar{N}} \exp(\kappa R^n_H) \cdot \mathbf{a}^n_{t+h}}{\sum_{n=1}^{\bar{N}} \exp(\kappa R^n_H)}$, where R^n_H is the return of the nth trajectory. MBOP (Argenson & Dulac-Arnold, 2021; Zhan et al., 2022) adapts MPPI for offline settings by sampling actions from a BC policy with smoothing.

The control-as-inference framework The control-as-inference framework reformulates RL as a probabilistic inference problem (Levine, 2018). This is achieved by introducing auxiliary binary optimality variables $\mathcal{O}_t$, where $\mathcal{O}_t = 1$ indicates that the state-action pair $(\mathbf{s}_t, \mathbf{a}_t)$ is *optimal*. Formally, we define the likelihood of optimality of a trajectory $\tau_{t:t+H} = (\mathbf{s}_t, \mathbf{a}_t, \ldots, \mathbf{s}_{t+H})$ below.

Definition 1 (The optimality likelihood)**.** *For $\tau_{t:t+H}$, let $\mathcal{O} = 1$ if all time steps are optimal (i.e., $\mathcal{O}_{t+h} = 1\ \forall h$). The optimality likelihood of τ is given by:*

$$p(\mathcal{O} = 1|\tau) \propto \prod_h p(\mathcal{O}_{t+h} = 1|\mathbf{s}_{t+h}, \mathbf{a}_{t+h}). \tag{2}$$

To relate optimality to rewards, we can assume $p(\mathcal{O}_{t+h}|\mathbf{s}_{t+h}, \mathbf{a}_{t+h}) \propto \exp(\kappa \cdot r_{t+h})$, where $\kappa > 0$ is an inverse temperature parameter, leading to $p(\mathcal{O}|\tau) \propto \exp(\kappa \cdot \sum_h r_h)$.[1] With this, the expected return maximization is recast as inferring the posterior over trajectories under the given probabilistic graphical model (PGM) (see Figure 2, left) given that all time steps are optimal:

$$p(\tau|\mathcal{O}) \propto p(\tau, \mathcal{O}) = p(\mathbf{s}_t) \prod_{h=1}^{H} p(\mathcal{O}_{t+h}|\mathbf{s}_{t+h}, \mathbf{a}_{t+h}) p(\mathbf{s}_{t+h+1}|\mathbf{s}_{t+h}, \mathbf{a}_{t+h}). \tag{3}$$

Here, the prior over actions is often assumed to be implicitly captured by the reward function or treated as a uniform improper prior (Levine, 2018; Piché et al., 2019). However, as we will demonstrate in Section 4.1, we explicitly model the prior over actions to formalize an offline MB planning framework, enabling the enhancement of an offline-learned policy through MB planning.

Epistemic POMDP, Bayes-adaptive MDP, and Offline RL A partially observable MDP (POMDP) extends MDPs to scenarios with incomplete state information. POMDPs can be reformulated as belief-state MDPs, where a *belief*—a probability distribution over states—represents the uncertainty over states given the agent's prior observations (Kaelbling et al., 1998).

Unlike an ordinary POMDP, *epistemic POMDPs* address generalization to unseen test conditions in RL (Ghosh et al., 2021). In this scenario, the agent experiences partial observability entirely due to its

[1] We use an exponential function for the optimality likelihood but note that other monotonic functions are also possible (Okada & Taniguchi, 2020). Also for brevity, we denote $\mathcal{O}_t = 1$ as $\mathcal{O}_t$ and $\tau_{t:t+H}$ as τ.

epistemic uncertainty about the identity of the true environment $\mathcal{M}$ at test time. Specifically, at test time, the agent's goal is to maximize the expected return $\mathbb{E}_{\mathcal{M}\sim p(\mathcal{M}|\mathcal{D})}[\sum_t \gamma^t r_t]$ under the posterior $p(\mathcal{M}|\mathcal{D})$ obtained after observing the train data $\mathcal{D}$. Thus, an epistemic POMDP is an instance of a Bayes-adaptive MDP (BAMDP) (Duff, 2002; Kaelbling et al., 1998). Unlike BAMDP, which tackles learning to act optimally under the uncertainty over true MDPs, epistemic POMDP focuses on the agent's test time evaluation performance rather than online learning. For a thorough definition of BAMDPs, please check Appendix A.1.

In this work, we view offline RL as epistemic POMDP, following Ghosh et al. (2022), drawing connections to Bayesian approaches. That is, limited coverage of the state-action space in the offline dataset induces epistemic uncertainty about dynamics beyond the data distribution. Failure to manage this uncertainty can result in catastrophic outcomes, particularly when an offline-trained agent encounters unseen states or slightly altered dynamics during deployment, leading to arbitrarily poor performance.

To address these challenges, we can leverage the BAMDP reformulation of epistemic POMDPs. This reformulation enables reasoning over the agent's uncertainty through a prior belief $b_0 = p(\mathcal{M})$, updated to a posterior $b_t = p(\mathcal{M}|\tau_{:t})$ as new experiences $\tau_{:t}$ are gathered during deployment. However, computing the exact posterior belief is generally intractable. Therefore, in Section 4, we tackle this challenge by approximating the belief distribution through variational inference techniques adapted from meta-RL approaches (Zintgraf et al., 2020; Dorfman et al., 2021).

4 RefPlan: A Probabilistic Framework for Offline Planning

In this section, we now seek to address the following question:

> *How can a learned model be utilized at test time to* ***enhance the performance*** *of an offline-trained agent and enable it to* ***account for its epistemic uncertainty****?*

To tackle this, we introduce ***RefPlan***: a novel probabilistic framework for MB planning that leverages learned models and allows the agent to reason with its uncertainty during deployment.

In Section 4.1, we derive a sampling-based offline MB planning algorithm rooted in a probabilistic inference perspective, demonstrating how this approach can enhance the capabilities of any offline-trained policy. Next, we delve into the epistemic POMDP formulation of the offline RL problem in Section 4.2, outlining how the agent's epistemic uncertainty can be effectively captured and represented. We introduce variational learning techniques for estimating the agent's uncertainty over the environment dynamics. Finally, in Section 4.3, we unify these concepts, presenting how RefPlan integrates epistemic uncertainty into the MB planning process, enabling the agent to plan under the learned models and adapt in real-time while accounting for its uncertainty.

4.1 Offline Model-Based Planning as Probabilistic Inference

We recast offline MB planning within the *control-as-inference* framework, allowing us to treat planning as a posterior inference problem. This approach enables the agent to optimize its actions by reasoning over the learned dynamics model and prior knowledge obtained from offline training. Central to this formulation is the use of a *prior policy*, which guides the agent's plans based on knowledge learned during offline training.

We start by formalizing the concept of a prior policy, which lay the basis for the Bayesian formulation of the offline MB planning process, together with the optimality likelihood defined in Definition 1.

Definition 2 (Prior policy)**.** *A prior policy* $\pi_{\mathrm{p}} : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ *is a policy learned from an offline RL algorithm* $\mathfrak{L}$ *using the dataset* $\mathcal{D}$*.*

The prior policy, parameterized by θ, is provided by an offline learning algorithm $\mathfrak{L}$, such as CQL (Kumar et al., 2020) or BC, and must be considered by the offline MB planner when optimizing the planning objective in (1).

In the offline setting, we aim to enhance the prior policy π_{p} via MB planning at test time by inferring the posterior over $\mathbf{a}_{t:t+H}$, conditioned on the optimality observations $\mathcal{O}_{t+h}$ predicted by the learned model $\hat{p}_{\psi}$. At time t, we seek to compute $p(\mathbf{a}_{t:t+H}|\mathcal{O})$, as shown in Figure 2 (middle).

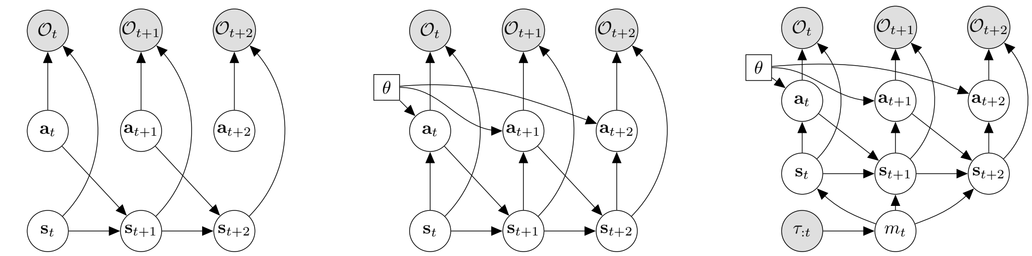


Figure 2: PGMs for the control-as-inference framework, offline MB planning, and RefPlan. **(Left)** States evolve within the learned model, with actions and states influencing optimality. Optimality variables act like observations in a hidden Markov model, framing planning as inferring the posterior over actions given optimality. **(Middle)** In offline MB planning, actions follow the prior policy π_{p}: $\mathbf{a}_t \sim \pi_{\mathrm{p}}(\cdot|\mathbf{s}_t;\theta)$. **(Right)** RefPlan uses past experiences $\tau_{:t}$ to infer m_t, the agent's belief about the environment, and computes the expected optimal action sequence by marginalizing over m_t.

The key distinction in this setup from the original control-as-inference framework is the inclusion of the prior policy, which serves as a source for action sampling during planning. Given the prior policy π_{p} and the learned model $\hat{p}_\psi$, we can define the prior distribution over the trajectory τ as follows:

$$p(\tau) = \prod_{h=0}^{H-1} \pi_{\mathrm{p}}(\mathbf{a}_{t+h}|\mathbf{s}_{t+h})\hat{p}_\psi(\mathbf{s}_{t+h+1}|\mathbf{s}_{t+h},\mathbf{a}_{t+h}). \tag{4}$$

Sampling trajectories from this prior, $p(\tau)$, is straightforward through forward sampling, where actions are drawn from π_{p} and state transitions are generated using $\hat{p}_\psi$.

Computing the exact posterior $p(\tau|\mathcal{O})$ is intractable due to the difficulty of calculating the marginal $p(\mathcal{O})$. However, importance sampling offers a practical method to estimate the posterior expectation over $\mathbf{a}_{t:t+H}$. To demonstrate, we first expand the posterior using Bayes' rule:

$$p(\tau|\mathcal{O}) \;\propto\; p(\mathcal{O}|\tau)p(\tau) \;\propto\; \exp\left(\kappa\sum_{h=0}^{H-1} r_{t+h}\right)\left[\prod_{h=0}^{H-1}\hat{p}_\psi(\mathbf{s}_{t+h+1}|\mathbf{s}_{t+h},\mathbf{a}_{t+h})\pi_{\mathrm{p}}(\mathbf{a}_{t+h}|\mathbf{s}_{t+h})\right]. \tag{5}$$

Then, we can estimate the expected value of an arbitrary function $f(\mathbf{a}_{t:t+H})$ under $p(\tau|\mathcal{O})$. That is,

$$\begin{aligned}\mathbb{E}_{p(\tau|\mathcal{O})}[f(\mathbf{a}_{t:t+H})] &= \int_\tau f(\mathbf{a}_{t:t+H})\, p(\tau|\mathcal{O})\, d\tau = \int_\tau f(\mathbf{a}_{t:t+H})\, \frac{p(\mathcal{O}|\tau)\, p(\tau)}{p(\mathcal{O})}\, d\tau \\ &= \int_\tau f(\mathbf{a}_{t:t+H})\, \frac{p(\mathcal{O}|\tau)}{p(\mathcal{O})}\, p(\tau)\, d\tau = \int_\tau f(\mathbf{a}_{t:t+H})\, \frac{\alpha\cdot\exp\left(\kappa\sum_h r_{t+h}\right)}{p(\mathcal{O})}\, p(\tau)\, d\tau \\ &= \frac{\mathbb{E}_{p(\tau)}\left[f(\mathbf{a}_{t:t+H})\ \exp\left(\kappa\sum_h r_{t+h}\right)\right]}{\mathbb{E}_{p(\tau)}\left[\exp\left(\kappa\sum_h r_{t+h}\right)\right]}.\end{aligned} \tag{6}$$

In the last step, we used $p(\mathcal{O}) = \int_\tau p(\mathcal{O}|\tau)p(\tau)d\tau = \alpha\,\mathbb{E}_{p(\tau)}\left[\exp\left(\kappa\sum_h r_{t+h}\right)\right]$ and the proportionality coefficient $\alpha > 0$ cancels out.

Thus, the posterior expectation over $\mathbf{a}_{t:t+H}$ can be obtained with $f(\mathbf{a}_{t:t+H}) = \mathbf{a}_{t:t+H}$ as below.

$$\mathbb{E}_{p(\tau|\mathcal{O})}[\mathbf{a}_{t:t+H}] = \frac{\mathbb{E}_{p(\tau)}\left[\mathbf{a}_{t:t+H}\ \exp\left(\kappa\sum_h r_{t+h}\right)\right]}{\mathbb{E}_{p(\tau)}\left[\exp\left(\kappa\sum_h r_{t+h}\right)\right]} \tag{7}$$

$$\approx \sum_{n=1}^{\bar{N}}\left(\frac{\exp\left(\kappa\sum_h r^n_{t+h}\right)}{\sum_{i=1}^{\bar{N}}\exp\left(\kappa\sum_h r^i_{t+h}\right)}\right)\mathbf{a}^n_{t:t+H}. \tag{8}$$

That is, we estimate the posterior mean by sampling $\bar{N}$ trajectories from $p(\tau)$ with π_{p} and $\hat{p}_\psi$, then computing the weighted sum of the actions. Each weight $w^n := \frac{\exp(\kappa\sum_h r_{t+h})}{\sum_{i=1}^{\bar{N}}\exp(\kappa\sum_h r^i_{t+h})}$ is proportional to the exponentiated MB return of the nth trajectory, assigning higher weights to plans with better returns. This helps the agent select actions likely to improve on those from the prior policy.

We note that (8) can also be derived from an optimization perspective. Specifically, LOOP (Sikchi et al., 2021) constrains the distribution over plans by minimizing the KL divergence from the prior policy. In LOOP, the variance of values generated by the model ensemble is penalized to mitigate uncertainty; however, the agent's epistemic uncertainty is not explicitly modeled and fully addressed. By contrast, by viewing offline RL as an epistemic POMDP and formulating it as a probabilistic inference problem, we can directly incorporate the agent's epistemic uncertainty into MB planning by approximately learning the belief distribution, which we delve into in the next part.

4.2 Learning Epistemic Uncertainty via Variational Inference

Although offline RL can be framed as a BAMDP, obtaining an exact posterior belief update is impractical. Inspired by Zintgraf et al. (2020) and Dorfman et al. (2021), we introduce a latent variable m to approximate the underlying MDP. We assume that knowing the posterior distribution $p(m|\tau_{:t})$ is sufficient for planning under epistemic uncertainty. As a result, transitions and rewards are assumed to depend on m, i.e., $T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, m)$ and $r(\mathbf{s}_t, \mathbf{a}_t, m)$. When $p(m|\tau_{:t})$ is accurate and $\tau_{:t}$ is in-distribution, T and r will closely match the transitions in $\mathcal{D}$. For OOD $\tau_{:t}$, the posterior over m captures epistemic uncertainty, allowing T and r to model diverse possible scenarios.

Given a trajectory $\tau_{:t}$, consider the task of maximizing its likelihood, conditioned on the actions. Conditioning on the actions is essential because they are generated by a policy—β during training and π_p at evaluation—and are not modeled by the environment. Although directly optimizing the likelihood $p(\mathbf{s}_0, r_0, \mathbf{s}_1, r_1, \ldots, \mathbf{s}_{t+1}|\mathbf{a}_0, \ldots, \mathbf{a}_t)$ is intractable, we can maximize the ELBO as in VariBAD by introducing an encoder q_φ and a decoder $\hat{p}_\psi$:

$$\log p(\mathbf{s}_0, r_0, \ldots, \mathbf{s}_{t+1}|\mathbf{a}_0, \ldots, \mathbf{a}_t) = \log \int_{m_t} p(\mathbf{s}_0, r_0, \ldots, \mathbf{s}_{t+1}, m_t \mid \mathbf{a}_0, \ldots, \mathbf{a}_t)\, dm_t \tag{9}$$
$$= \log \mathbb{E}_{m_t \sim q_\varphi(\cdot|\tau_{:t})}\left[\frac{p(\mathbf{s}_0, r_0, \ldots, \mathbf{s}_{t+1}, m_t \mid \mathbf{a}_0, \ldots, \mathbf{a}_t)}{q_\varphi(m_t|\tau_{:t})}\right]$$
$$\geq \mathbb{E}_{m_t \sim q_\varphi(\cdot|\tau_{:t})}[\log \hat{p}_\psi(\mathbf{s}_0, \ldots, \mathbf{s}_{t+1}|m_t, \mathbf{a}_0, \ldots, \mathbf{a}_t)] - KL(q_\varphi(m_t|\tau_{:t})||p(m_t)) = ELBO_t(\varphi, \psi).$$

The encoder q_φ is parameterized as an RNN followed by a fully connected layer that outputs Gaussian parameters $\mu(\tau_{:t})$ and $\log \sigma^2(\tau_{:t})$. Thus, $m_t \sim q_\varphi(\cdot|\tau_{:t}) = \mathcal{N}\left(\mu(\tau_{:t}), \sigma^2(\tau_{:t})\right)$. The KL term regularizes the posterior with the prior $p(m_t)$, which is a standard normal at $t = 0$ and the previous posterior $q_\varphi(\cdot|\tau_{:t-1})$ for subsequent time steps. The decoder $\hat{p}_\psi$ learns the transition dynamics and reward function of the true MDP. This becomes clear when we observe that the first term in $ELBO_t$ corresponds to the reconstruction loss, which can be decomposed as follows:

$$\log \hat{p}_\psi(\mathbf{s}_0, r_0, \ldots, \mathbf{s}_{t+1}|m_t, \mathbf{a}_0, \ldots, \mathbf{a}_t) = \log p(\mathbf{s}_0|m_t) + \sum_{h=0}^{t} \left[\log \hat{p}_\psi(\mathbf{s}_{h+1}|\mathbf{s}_h, \mathbf{a}_h, m_t) + \log \hat{p}_\psi(r_{h+1}|\mathbf{s}_h, \mathbf{a}_h, m_t)\right]. \tag{10}$$

Here, $\hat{p}_\psi$ learns to predict future states and rewards conditioned on the latent variable m_t. The encoder captures the agent's epistemic uncertainty, while the decoder provides predictions about the environment under different latent variables m_t. To sum up, we train a variational autoencoder (VAE) via $\max_{\phi,\psi}\ \mathbb{E}_\mathcal{D}\left[\sum_{t=0}^{T}\ ELBO_t(\phi, \psi)\right]$ using trajectories sampled from the offline dataset $\mathcal{D}$.

Unlike VariBAD, where the decoder is only used to train the encoder, we also use $\hat{p}_\psi$ for MB planning. To improve $\hat{p}_\psi$'s accuracy, we employ a two-stage training procedure: first, the VAE is trained with the ELBO objective; then, the encoder is frozen and $\hat{p}_\psi$ is further finetuned using the MLE objective:

$$L(\psi) = \mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_{h=0}^{H-1} \mathbb{E}_{m_h \sim q_\varphi(\cdot|\tau_{:h})}[-\log \hat{p}_\psi(\mathbf{s}_{h+1}, r_h|\mathbf{s}_h, \mathbf{a}_h, m_h)]\right]. \tag{11}$$

Trajectory segments of length H are sampled from the offline dataset. At each step $h \in [0, H)$, the encoder $q_\varphi(\cdot|\tau_{:h})$ samples m_h, enabling computation of the inner expectation in (11) and refining $\hat{p}_\psi$ for improved predictions.

4.3 Integrating Epistemic Uncertainty into Model-Based Planning

Building on the probabilistic inference formulation of offline MB planning and the representation of epistemic uncertainty via variational inference in the BAMDP framework, we introduce RefPlan. This offline MB planning algorithm integrates epistemic uncertainty into the planning process, improving decision-making and enhancing the performance of any offline-learned prior policy during test time.

Assume we have a posterior sample $m_t \sim q_\varphi(m|\tau_{:t})$, representing the agent's belief about the environment at time t. Our goal is to use this posterior to enhance test-time planning. In Section 4.1, we have computed $p(\tau|\mathcal{O})$ using the learned models $\hat{p}_\psi$ and the prior policy π_p. By introducing the latent variable m to capture epistemic uncertainty, we extend the transition and reward functions to depend on m, giving the dynamics $\hat{p}_\psi(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, m_t)$ and rewards $r(\mathbf{s}_t, \mathbf{a}_t, m_t)$, resulting in the following conditional trajectory distribution:

$$\begin{aligned} p(\tau|\mathcal{O}, m_t) &\propto p(\mathcal{O}|\tau, m_t)p(\tau|m_t) \\ &\propto \exp\left(\kappa \sum_{h=0}^{H-1} r(\mathbf{s}_{t+h}, \mathbf{a}_{t+h}, m_t)\right) \left[\prod_{h=0}^{H-1} \hat{p}_\psi(\mathbf{s}_{t+h+1}|\mathbf{s}_{t+h}, \mathbf{a}_{t+h}, m_t)\pi_\mathrm{p}(\mathbf{a}_{t+h}|\mathbf{s}_{t+h})\right]. \end{aligned}$$

Thus, we can apply the sampling-based posterior estimation provided in (8) to approximate the conditional expectation $\mathbb{E}_{p(\tau|\mathcal{O},m_t)}[\mathbf{a}_{t:t+H}]$.

A practical approach to handle epistemic uncertainty is to marginalize over the latent variable m_t, effectively averaging over possible scenarios. This results in the marginal posterior distribution $p(\tau|\mathcal{O})$. Although directly computing this marginal posterior is challenging, we can estimate the expectation of optimal plans using the law of total expectation:

$$\mathbb{E}_{p(\tau|\mathcal{O})}[\mathbf{a}_{t:t+H}] = \mathbb{E}_{m_t \sim q_\varphi(\cdot|\tau_{:t})}\left[\mathbb{E}_{p(\tau|\mathcal{O},m_t)}[\mathbf{a}_{t:t+H} \mid m_t]\right]. \tag{12}$$

The inner expectation follows (8), with states and rewards sampled from $\hat{p}_\psi$, conditional on m_t. The outer expectation over m_t is computed using Monte Carlo sampling with $\bar{n}$ samples, giving us:

$$\mathbb{E}_{p(\tau|\mathcal{O})}[\mathbf{a}_{t:t+H}] \approx \frac{1}{\bar{n}} \sum_{j=1}^{\bar{n}} \left[\sum_{n=1}^{\bar{N}} \left(\frac{\exp\left(\kappa \sum_h r_{t+h}^{n,j}\right)}{\sum_{i=1}^{\bar{N}} \exp\left(\kappa \sum_h r_{t+h}^{i,j}\right)}\right) \mathbf{a}_{t:t+H}^n\right], \tag{13}$$

where $r_{t+h}^{n,j} = r(\mathbf{s}_{t+h}^{n,j}, \mathbf{a}_{t+h}^n, m_t^j)$ and $\mathbf{s}_{t+h+1}^{n,j} \sim \hat{p}_\psi(\cdot|\mathbf{s}_{t+h}^{n,j}, \mathbf{a}_{t+h}^n, m_t^j)$. Figure 2 (right) illustrates how RefPlan leverages the agent's past experiences $\tau_{:t}$ to shape epistemic uncertainty through the latent variable m_t and enhances the prior policy π_p through posterior inference. Algorithm 2 in the appendix summarizes RefPlan.[2] Additionally, following Sikchi et al. (2021), we apply an uncertainty penalty based on the variance of the returns predicted by the learned model ensemble.

5 Experiments

In this part, we answer the following research questions: **(RQ1)** How does RefPlan perform when the agent is initialized in a way that induces high epistemic uncertainty due to OOD states? **(RQ2)** Can RefPlan effectively improve policies learned from diverse offline policy learning algorithms? **(RQ3)** How does RefPlan perform when trained on limited offline datasets that increase epistemic uncertainty by restricting the datasets' coverage of the state-action space? **(RQ4)** How robust is RefPlan when faced with shifts in environment dynamics at test time?

We evaluate these RQs using the D4RL benchmark (Fu et al., 2020) and its variations, focusing on locomotion tasks in *HalfCheetah*, *Hopper*, and *Walker2d* environments, each with five configurations: *random* (R), *medium* (M), *medium-replay* (MR), *medium-expert* (ME), and *full-replay* (FR).

Baselines: RefPlan is designed to improve any offline learned policy through planning. We evaluate prior policies using model-free methods (CQL, EDAC) and MB methods (MOPO, COMBO, MAPLE). Among offline MB planning methods, we use LOOP, which is designed to enhance prior policies and outperforms methods like MBOP. Therefore, for each prior policy, we compare its original performance to its performance when augmented with LOOP or RefPlan for test-time planning.

[2] Direct planning with sampling methods like SIR (Skare et al., 2003) may be better for multi-modal problems. However, our approach using (13) yields strong empirical results, so we leave direct sampling for future work.

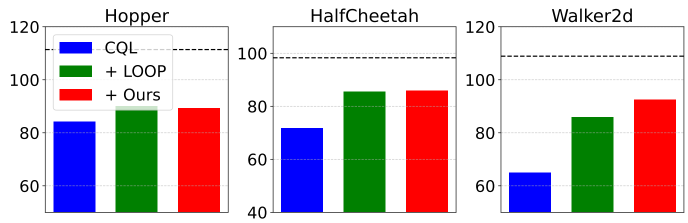


Figure 3: CQL's performance when trained on ME and tested in OOD states from R. '+ LOOP' and '+ Ours' show improvements with LOOP and RefPlan, while dotted lines indicate original performance.

Table 1: Normalized scores of offline RL algorithms on D4RL MuJoCo Gym environments (3 seeds). For each prior policy, we show its original performance and its performance augmented with LOOP or RefPlan (Ours) for MB planning during testing. **Bold** indicates the best performance, while underline denotes cases where confidence intervals significantly overlap between two methods.

		CQL			EDAC			MOPO			COMBO			MAPLE		
		Orig	LOOP	Ours	Orig	LOOP	Ours	Orig	LOOP	Ours	Orig	LOOP	Ours	Orig	LOOP	Ours
Hopper	R	1.0	1.1	1.2	23.6	23.5	23.5	32.2	32.4	32.4	6.3	6.2	6.0	31.5	31.8	31.6
	M	66.9	73.9	**85.1**	101.5	101.5	101.5	66.9	67.5	67.7	60.9	67.9	**77.2**	29.4	**33.7**	32.8
	MR	94.6	97.5	**98.1**	100.4	101.0	101.1	90.3	93.6	**94.5**	101.1	101.4	101.8	61.0	77.7	**82.6**
	ME	111.4	111.6	**112.1**	106.7	104.7	**109.9**	91.3	82.7	**96.5**	105.6	78.4	**107.8**	46.9	53.4	**57.8**
	FR	104.2	106.2	**107.6**	106.6	107.0	107.2	73.2	55.6	**77.2**	**89.9**	54.9	84.1	79.1	77.0	**91.7**
HalfCheetah	R	19.9	21.4	21.2	22.5	25.8	25.9	29.8	31.5	**33.0**	40.3	40.0	40.7	33.5	34.9	35.0
	M	47.4	**57.1**	56.5	63.8	**73.0**	71.4	42.8	58.4	**59.8**	67.2	73.2	**77.4**	68.8	72.9	**74.6**
	MR	47.0	52.1	**54.1**	61.8	66.9	66.5	70.6	71.8	**73.8**	73.0	71.2	**75.0**	71.5	74.7	**76.3**
	ME	98.3	104.0	**108.5**	100.8	107.1	**108.8**	73.5	94.5	**96.6**	97.6	110.3	110.3	64.0	91.9	**92.8**
	FR	77.5	81.8	**86.7**	81.7	87.5	**88.5**	81.7	88.2	**90.8**	71.8	82.6	**86.3**	66.8	87.8	**90.2**
Walker2d	R	0.1	0.1	0.3	17.5	13.4	**21.7**	13.3	12.4	13.1	4.1	3.0	4.3	21.8	21.8	21.9
	M	77.1	84.4	**86.2**	77.6	91.7	**93.2**	82.0	79.1	**85.9**	71.2	81.1	**87.4**	88.3	89.7	**91.6**
	MR	63.5	81.9	**93.6**	85.0	86.0	86.4	81.7	85.2	**88.3**	88.0	89.5	**93.3**	85.0	89.5	**91.2**
	ME	108.9	111.4	111.8	98.5	97.4	**116.0**	51.9	49.0	**68.1**	108.3	111.1	**112.7**	111.8	112.9	**114.0**
	FR	96.6	99.4	**101.3**	98.0	98.3	**99.7**	90.5	92.6	**93.2**	78.1	83.0	**99.5**	94.2	96.7	**98.4**

Metrics: For RQ1-RQ3, we compare normalized scores averaged over 3 seeds, with 100 for online SAC and 0 for a random policy, scaled linearly in between. For RQ4, we report average returns.

5.1 RefPlan handles epistemic uncertainty from OOD states

To address RQ1, we assessed RefPlan's robustness under high epistemic uncertainty caused by OOD initialization. Prior policies were trained on the ME dataset and evaluated on the states from the R dataset. We tested three prior policies: CQL (Figure 3), MAPLE (Figure 7), and COMBO (Figure 8).

Across all environments, RefPlan consistently mitigated performance degradation due to OOD initialization, with particularly notable improvements in *HalfCheetah* and *Walker2d*. For instance, when MAPLE was used as the prior policy in *HalfCheetah*, RefPlan outperformed the original policy (dotted line in Figure 7). In *Walker2d*, RefPlan boosted performance by 16.4%, 31.4%, and 42.5% for COMBO, MAPLE, and CQL, respectively. Although the gains were more modest in *Hopper*, RefPlan still reduced performance drops. Overall, RefPlan showed strong resilience under high epistemic uncertainty caused by OOD initialization.

5.2 RefPlan enhances any offline-learned policies

To address RQ2, we evaluated the normalized score metric across five offline policy learning algorithms. Table 1 shows that RefPlan outperformed baselines in 10 (CQL), 7 (EDAC), 12 (MOPO), 9 (COMBO), and 12 (MAPLE) of 15 tasks, matching performance in the others. Both MB planning methods, LOOP and RefPlan, improved performance, with RefPlan showing a more substantial gain. On average, RefPlan enhanced prior policy performance by 11.6%, compared to LOOP's 5.3%. Furthermore, Figure 6 in Appendix B shows that RefPlan consistently outperforms LOOP with non-overlapping confidence intervals under the RLiable evaluation (Agarwal et al., 2022). These

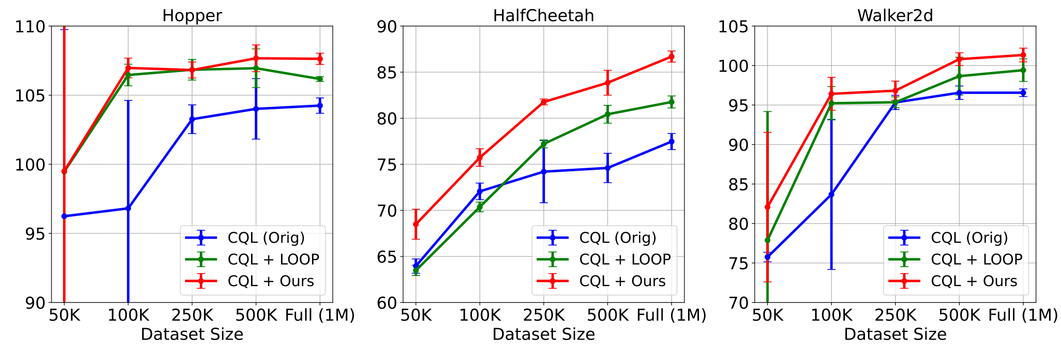


Figure 4: Performance comparison of RefPlan and LOOP across different dataset sizes in *Hopper*, *HalfCheetah*, and *Walker2d* environments using the FR dataset, which contains 1M samples. We use CQL as the prior policy learning algorithm, and the results represent the average and standard error calculated from three random seeds.

results demonstrate RefPlan's superior ability to enhance various offline policy learning algorithms by explicitly accounting for epistemic uncertainty during planning.

5.3 Performance with limited offline data with varying dataset sizes

With limited data, the agent faces increased epistemic uncertainty. A key question is whether RefPlan can better handle these scenarios with constrained data (RQ3). To explore this, we randomly subsample 50K, 100K, 250K, and 500K transition samples from the FR dataset for each environment. We then train the prior policy using CQL and compare its performance with that achieved when enhanced by either LOOP or RefPlan. As shown in Figure 4, RefPlan consistently demonstrates greater resilience to limited data, outperforming the baselines across all three environments.

5.4 RefPlan is more robust to changing dynamics?

To address RQ4, we evaluated RefPlan in the *HalfCheetah* environment under varying dynamics, including *disabled joint*, *hill*, slopes (*gentle* and *steep*), and *field*, following the approach of Clavera et al. (2019) (Appendix D). High epistemic uncertainty arises when dynamics differ from those seen during prior policy training. We trained the prior policy using the FR dataset, which contains the most diverse trajectories, and used MAPLE for its adaptive policy learning. Table 5 shows that while MAPLE struggled with changed dynamics, MB planning methods improved performance. RefPlan achieved the best results across all variations but still faced notable drops, especially in the hill and gentle environments. Data augmentation for single-task offline RL could enhance adaptability, a topic for future work.

Figure 5: Average returns on *HalfCheetah* with dynamics changes.

Task	Orig	LOOP	Ours
joint	5295	6088	**6190**
hill	327.1	949.7	**1224**
gentle	1087	2363	**2435**
steep	2123	3245	**6238**
field	1205	2774	**3345**

6 Conclusion

In this paper, we introduced **RefPlan** (Reflect-then-Plan), a novel *doubly Bayesian* approach to offline model-based planning that integrates epistemic uncertainty modeling with model-based planning in a unified probabilistic framework. Our method enhances offline RL by explicitly accounting for epistemic uncertainty, a common challenge in offline settings where data coverage is often incomplete. Through extensive experiments on standard offline RL benchmarks, we demonstrated that RefPlan consistently outperforms existing methods, particularly under challenging conditions of OOD initialization, limited data availability, and changing environment dynamics, making it a valuable tool for more reliable and adaptive offline RL. Future work could extend RefPlan to more complex models and environments.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice, 2022. URL https://arxiv.org/abs/2108.13264.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=ZUvaSolQZh3.

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OMNB1G5xzd4.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Robert N. Boute, Joren Gijsbrechts, Willem van Jaarsveld, and Nathalie Vanvuchelen. Deep reinforcement learning for inventory control: A roadmap. *European Journal of Operational Research*, 298(2):401–412, 2022. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2021.07.016. URL https://www.sciencedirect.com/science/article/pii/S0377221721006111.

Xiong-Hui Chen, Yang Yu, Qingyang Li, Fan-Ming Luo, Zhiwei Tony Qin, Shang Wenjie, and Jieping Ye. Offline model-based adaptable policy learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=lrdXc17jm6.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/file/3de568f8597b94bda53149c7d7f5958c-Paper.pdf.

Ignasi Clavera, Anusha Nagabandi, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyztsoC5Y7.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta reinforcement learning – identifiability challenges and effective data collection strategies. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=IBdEfhLveS.

Michael O'Gordonz Duff. *Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts Amherst, Amherst, USA, 2002.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020.

Scott Fujimoto and Shixiang Gu. A minimalist approach to offline reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Q32U7dzWXpc.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2052–2062. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/fujimoto19a.html.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Found. Trends Mach. Learn.*, 8(5–6):359–483, November 2015. ISSN 1935-8237.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 25502–25515. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/d5ff135377d39f1de7372c95c74dd962-Paper.pdf.

Dibya Ghosh, Anurag Ajay, Pulkit Agrawal, and Sergey Levine. Offline RL policies should be trained to be adaptive. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7513–7530. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/ghosh22a.html.

Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/35051070e572e47d2c26c241ab88307f-Paper.pdf.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 17–23 Jul 2022.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5084–5096. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/jin21e.html.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1–2):99–134, may 1998. ISSN 0004-3702.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 651–673. PMLR, 29–31 Oct 2018. URL https://proceedings.mlr.press/v87/kalashnikov18a.html.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel : Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/file/c2073ffa77b5357a498057413bb09d3a-Paper.pdf.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control, 2018. URL https://arxiv.org/abs/1811.01848.

Cong Lu, Philip Ball, Jack Parker-Holder, Michael Osborne, and S Roberts. Revisiting design choices in offline model based reinforcement learning, 2021. URL https://openreview.net/forum?id=UfX6vkvhtl.

Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation, 2019. URL https://arxiv.org/abs/1909.11652.

Masashi Okada and Tadahiro Taniguchi. Variational inference mpc for bayesian model-based reinforcement learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 258–272. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/okada20a.html.

Alexandre Piché, Valentin Thomas, Cyril Ibrahim, Yoshua Bengio, and Chris Pal. Probabilistic planning with sequential monte carlo methods. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByetGn0cYX.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *arXiv preprint arXiv:2204.12581*, 2022.

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=1GNV9SW95eJ.

Øivind Skare, Erik Bølviken, and Lars Holden. Improved sampling-importance resampling and reduced bias importance sampling. *Scandinavian Journal of Statistics*, 30(4):719–737, 2003. doi: https://doi.org/10.1111/1467-9469.00360. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-9469.00360.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling, 2015.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *CoRR*, abs/1911.11361, 2019. URL http://arxiv.org/abs/1911.11361.

Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *CoRR*, abs/1805.04687, 2018. URL http://arxiv.org/abs/1805.04687.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: conservative offline model-based policy optimization. *CoRR*, abs/2102.08363, 2021.

Xianyuan Zhan, Xiangyu Zhu, and Haoran Xu. Model-based offline planning with trajectory pruning. In Lud De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 3716–3722. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/516. URL https://doi.org/10.24963/ijcai.2022/516. Main Track.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning, 2020. URL https://arxiv.org/abs/1910.08348.

A Additional Background

A.1 Bayes-Adaptive Markov Decision Processes

Bayes-Adaptive Markov Decision Processes (BAMDPs) (Duff, 2002) extend the standard MDP framework by explicitly incorporating uncertainty over the transition and reward functions. In a BAMDP, instead of assuming that the transition dynamics $T(\mathbf{s}'|\mathbf{s},\mathbf{a})$ and reward function $r(\mathbf{s},\mathbf{a})$ are known and fixed, we assume that they are drawn from an unknown distribution. The agent maintains a posterior belief over these functions and updates it as new data are collected through interaction with the environment.

To illustrate, consider a simple case where we have finite and discrete state and action spaces with $|\mathcal{S}| = n_s$ and $|\mathcal{A}| = n_a$; hence, a state can be represented with an integer, i.e., $s = i$ for $i = 1, \ldots, n_s$, and similarly for the actions. While the reward function $r(s, a)$ is assumed to be known, we are uncertain about the transition probabilities $T(s'|s, a)$. We can model this uncertainty by placing a prior distribution over the transition probabilities, typically using a Dirichlet prior, which is conjugate to the multinomial likelihood of observing transitions between states.

For each state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, the transition probabilities $T(s'|s, a)$ are parameterized by a multinomial distribution:

$$T(s'|s,a) \sim \mathrm{Multinomial}(\boldsymbol{\theta}_{s,a,s'}), \tag{14}$$

where $\boldsymbol{\theta}_{s,a} = (\boldsymbol{\theta}_{s,a,1}, \ldots, \boldsymbol{\theta}_{s,a,n_s})$ represents the probabilities of transitioning from state s to any state $s' \in \mathcal{S}$ under action a. These parameters follow a Dirichlet distribution:

$$\boldsymbol{\theta}_{s,a} \sim \mathrm{Dirichlet}(\alpha_{s,a}), \tag{15}$$

where $\alpha_{s,a} = (\alpha_{s,a,1}, \ldots, \alpha_{s,a,n_s}) > 0$ are the Dirichlet hyperparameters.

Initially, the agent holds a prior belief about the transition probabilities, represented by the Dirichlet hyperparameters $\alpha_{s,a}$ for all state-action pairs. As the agent interacts with the environment and observes transitions of the form (s, a, s'), it updates its posterior belief by simply updating the corresponding Dirichlet hyperparameters. Specifically, when the agent observes a transition from state s to state s' under action a, the corresponding Dirichlet hyperparameter is updated as:

$$\alpha_{s,a,s'} \leftarrow \alpha_{s,a,s'} + 1, \tag{16}$$

while all other Dirichlet hyperparameters remain unchanged. This process of updating the Dirichlet hyperparameters fully captures the agent's experiences; hence, these hyperparameters act as sufficient statistics for the agent's belief about the environment.

By transforming the BAMDP into a belief-state MDP, where the belief state $b_t = p(\boldsymbol{\theta}|\tau_{:t})$ is a distribution over transition probabilities conditioned on the observed trajectory $\tau_{:t} = (s_0, a_0, s_1, \ldots, s_t)$, the agent can solve the problem using standard MDP solution methods. The augmented state space, or hyper-state space, includes both the *physical* state $s \in \mathcal{S}$ and the belief state $b \in \mathcal{B}$. In this simple finite state-action example, the belief state corresponds to the Dirichlet hyperparameters α.

The transition dynamics of the resulting belief-state MDP are fully known and can be written as:

$$\begin{aligned} T(\bar{s}'|\bar{s},a) = T(s',\alpha'|s,\alpha,a) &= T(s'|s,a,\alpha)p(\alpha'|s,\alpha,a) && (17) \\ &= \frac{\alpha_{s,a,s'}}{\sum_{s''\in\mathcal{S}} \alpha_{s,a,s''}} \mathbb{I}\left(\alpha'_{s,a,s'} = \alpha_{s,a,s'} + 1\right), && (18) \end{aligned}$$

where $\mathbb{I}(\cdot)$ is the indicator function. This transformation turns the BAMDP into a fully observable MDP in the hyper-state space, which allows the use of standard, e.g., DP methods to compute an optimal policy.

However, the computational complexity of solving the BAMDP grows quickly with the number of states and actions. If the states are fully connected (i.e., $p(s'|s,a) > 0,\ \forall s, a, s'$), the number of reachable belief states increases exponentially over time, making exact solutions intractable for even moderately sized problems.

For a comprehensive overview of solution methods for BAMDPs, we refer readers to the survey by Ghavamzadeh et al. (2015). In this work, we have utilized variational inference techniques from Zintgraf et al. (2020) and Dorfman et al. (2021) to approximate the agent's posterior belief over the environment dynamics, $p(b|\tau_{:t})$, based on past experiences.

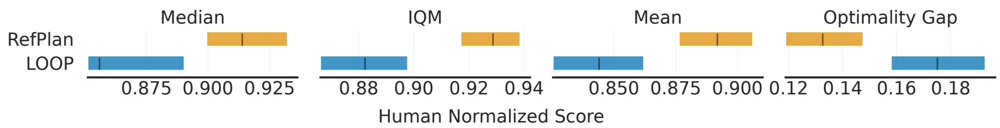


Figure 6: RLiable (Agarwal et al., 2022) comparison of RefPlan and LOOP. Across all four metrics, RefPlan demonstrates superior performance with non-overlapping confidence intervals, highlighting statistically significant improvements over LOOP.

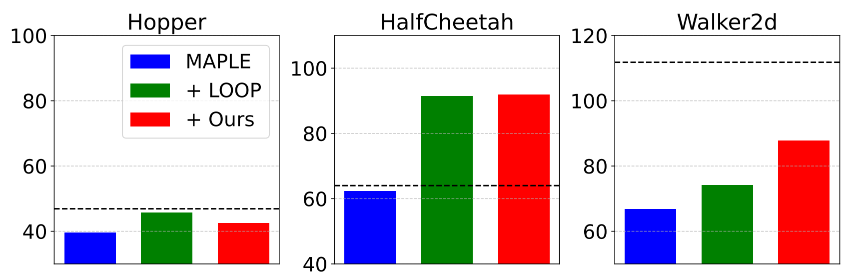


Figure 7: MAPLE's performance when trained on ME and tested in OOD states from R. '+ LOOP' and '+ Ours' show improvements with LOOP and RefPlan, while dotted lines indicate original performance.

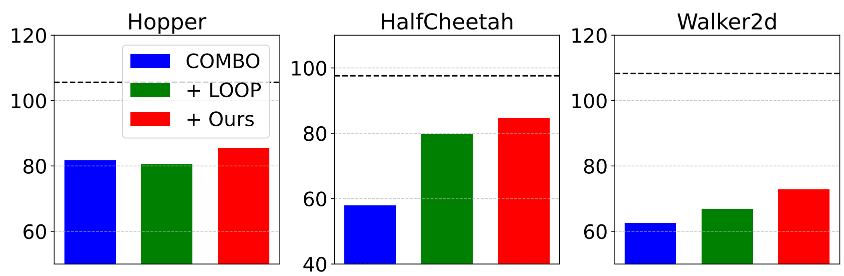


Figure 8: COMBO's performance when trained on ME and tested in OOD states from R. '+ LOOP' and '+ Ours' show improvements with LOOP and RefPlan, while dotted lines indicate original performance.

B Additional Results

B.1 Performance comparison of RefPlan vs. LOOP

In order to make a more statistically rigorous comparison between RefPlan and LOOP, we leverage RLiable (Agarwal et al., 2022), a framework designed for robust evaluation of reinforcement learning algorithms. RLiable focuses on statistically sound aggregate metrics, such as the median, interquartile mean (IQM), mean, and optimality gap, which provide a comprehensive view of algorithm performance across tasks. By using bootstrapping with stratified sampling, RLiable also estimates confidence intervals, ensuring that comparisons are not skewed by outliers or noise.

We applied RLiable to compare RefPlan and LOOP across the tested environments and prior policy setups (Figure 6). Across all metrics, RefPlan consistently outperformed LOOP, with non-overlapping confidence intervals, indicating statistically significant improvements.

B.2 RefPlan handles epistemic uncertainty from OOD states

To address RQ1, we assessed the robustness of RefPlan under conditions of high epistemic uncertainty arising from OOD initialization. Specifically, we began by training a prior policy on the ME dataset and then tested it in states sampled from the R dataset, which are OOD. The evaluation included

Table 2: Performance comparison of RefPlan against baseline methods on Hopper, HalfCheetah, and Walker2d tasks using MOPO and COMBO for offline policy optimization. The table evaluates original policies (Orig), policies trained with Non-Markovian (NM) dynamics models (NM (Train)), NM-trained policies combined with RefPlan for planning (NM (Train) + RefPlan), and RefPlan using original policies as priors. Results demonstrate RefPlan's ability to improve test-time performance across different dynamics models and environments.

		Orig	NM (Train)	NM (Train) +RefPlan	RefPlan
Hopper	M	66.9	-	-	**67.7**
	MR	90.3	93.2	**98.18**	94.5
	ME	91.3	-	-	**96.5**
HalfCheetah	M	42.8	40.6	**66.45**	59.8
	MR	70.6	53.2	72.46	**73.8**
	ME	73.5	71.6	**100.34**	96.6
Walker2d	M	82.0	60.6	72.73	**85.9**
	MR	81.7	53.3	79.75	**88.3**
	ME	51.9	42.4	64.59	**68.1**

		Orig	NM (Train)	NM (Train) +RefPlan	RefPlan
Hopper	M	60.9	52.2	62.30	**77.2**
	MR	101.1	44.9	61.90	**101.8**
	ME	105.6	27.3	39.23	**107.8**
HalfCheetah	M	67.2	30.3	41.61	**77.4**
	MR	73.0	47.6	59.54	**75.0**
	ME	97.6	93.5	**109.25**	**110.3**
Walker2d	M	71.2	79.1	**89.43**	87.4
	MR	88.0	80.4	91.01	**93.3**
	ME	108.3	36.7	38.47	**112.7**

three prior policies: CQL (Figure 3), MAPLE (Figure 7), and COMBO (Figure 8). Across all environments and prior policies, RefPlan consistently mitigated performance drops, with the benefits being particularly notable in the *HalfCheetah* and *Walker2d* environments. For instance, when using MAPLE as the prior policy in *HalfCheetah*, the agent enhanced with MB planning significantly outperformed the original policy (represented by the dotted line in Figure 7). In the Walker2d environment, RefPlan boosted the performance of the prior policies by 16.4%, 31.4%, and 42.5% for COMBO, MAPLE, and CQL, respectively. Although the improvements in *Hopper* were more modest, MB planning methods still reduced performance deterioration. Overall, agents trained on a narrow data distribution experienced performance drops when exposed to unknown states, but MB planning approaches, particularly RefPlan, demonstrated significant resilience under high epistemic uncertainty.

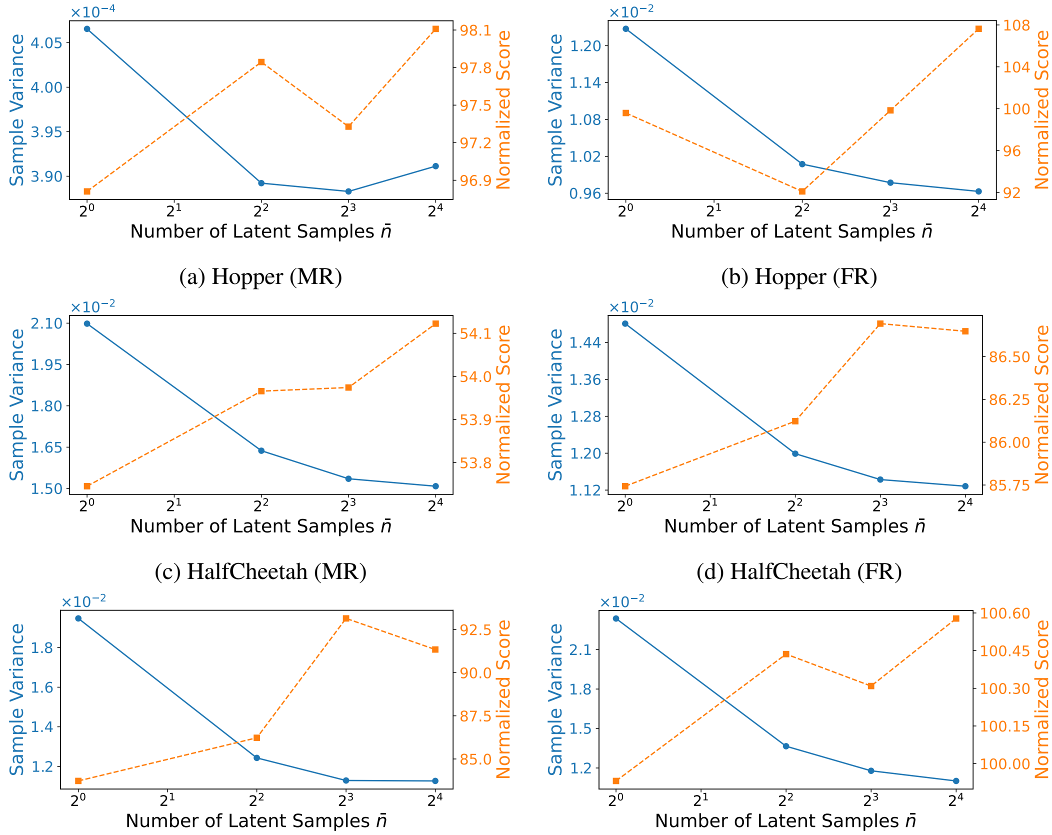
B.3 Performance comparison: non-Markovian dynamics model for training vs. planning

The experiments presented in Table 2 aims to evaluate the effectiveness of RefPlan in leveraging the VAE dynamics—consisting of the variational encoder q_ϕ and the probabilistic ensemble decoder $\hat{p}_\psi$ (Figure 10)—for planning at test time. Specifically, these experiments compare the following approaches:

- "Orig": the original prior policy trained using MOPO or COMBO.
- "NM (Train)": the policy trained using a non-Markovian (NM) VAE dynamics model during offline policy optimization via MOPO or COMBO.
- "NM (Train) + RefPlan ": the RefPlan agent that uses the policies trained using NM dynamics models as priors.
- "RefPlan ": the RefPlan agent that uses the original prior policies as priors.

The results demonstrate several key findings. First, RefPlan consistently outperforms NM (Train) across all environments and datasets, confirming that the VAE dynamics models are significantly more effective when used for planning at test time rather than during offline policy training. This highlights RefPlan's ability to explicitly handle epistemic uncertainty, leveraging the agent's real-time history to infer the underlying MDP dynamics.

Second, in MOPO results, NM (Train) diverged or underperformed in several cases. This suggests that the heuristically estimated model uncertainty used in MOPO is not well-suited for integrating with the VAE dynamics models during offline training. Even with large penalty parameters, the value function diverged in the Hopper tasks, indicating a fundamental limitation in using NM models with MOPO for policy optimization. By contrast, COMBO results did not exhibit these issues, suggesting that COMBO's framework is better equipped to incorporate such dynamics models during training.



(a) Hopper (MR) (b) Hopper (FR)

(c) HalfCheetah (MR) (d) HalfCheetah (FR)

(e) Walker2d (MR) (f) Walker2d (FR)

Figure 9: The sample variance and the performance vs. the number of latent samples of RefPlan, evaluated from three environments with the MR and FR datasets using CQL as the prior policy.

Finally, applying RefPlan to policies trained with NM dynamics models (NM (Train) + RefPlan) further boosted test-time performance, often by substantial margins. This demonstrates that even when NM dynamics models introduce suboptimality during offline training, RefPlan can recover and enhance the policy's performance through effective planning at test time. Across all environment-dataset combinations, RefPlan provides robust improvements over both the original and NM (Train)-optimized policies, further validating its capability to address epistemic uncertainty and improve the generalization of offline-learned policies.

B.4 Evaluating the impact of the number of latent samples on variances and performance

This experiment evaluates how the sample variance of the marginal action posterior mean from (13) changes with the number of latent samples ($\bar{n}$) used in the outer expectation. At each time step, we compute the posterior mean K times, calculate its variance averaged across action dimensions, and report the running average over a 1,000-step episode. Results are averaged over three random seeds, with CQL as the prior policy, across three environments (Hopper, HalfCheetah, Walker2d) and two dataset configurations (MR, FR).

The figures show that as $\bar{n}$ increases, the average sample variance decreases, with $\bar{n} = 1$ consistently yielding the highest variance. Performance, measured as normalized scores, generally improves with increasing $\bar{n}$, suggesting a positive correlation between reduced variance and higher performance. However, while reduced variance likely contributes to this improvement, further investigation is needed to confirm causality, as other factors may also play a role.

Algorithm 2 RefPlan: Offline MB Planning as Probabilistic Inference

1: **Input:** $\tau_{:t} = (\mathbf{s}_0, \mathbf{a}_0, r_0, \ldots, \mathbf{s}_t)$, $\hat{p}_\psi$, q_ϕ, π_p, $\hat{Q}$, H, $\bar{N}$, $\bar{n}$, κ
2: $\mu_t,\ \sigma_t \leftarrow q_\phi(\cdot|\tau_{:t})$ ▷ Get the Gaussian parameters
3: $\{m_t^j\}_{j=1}^{\bar{n}} \sim \mathcal{N}(\mu_t, \sigma_t^2)$ ▷ Sample $\bar{n}$ latent vectors from the approximate posterior
4: **for** $n = 1, \ldots, \bar{N}$ **do**
5:     **for** $h = 0, \ldots, H-1$ **do**
6:         $\mathbf{a}_{t+h}^n \sim \pi_\mathrm{p}(\cdot|\mathbf{s}_{t+h})$ ▷ Sample prior action sequence
7:         $\mathbf{s}_{t+h+1}^n \sim \hat{p}_\psi(\cdot|\mathbf{s}_{t+h}^n, \mathbf{a}_{t+h}^n, \mu_t)$ ▷ Sample the next state from model using μ_t
8:     **end for**
9:     **for** $j = 1, \ldots, \bar{n}$ **do**
10:         $\mathbf{s}_t^{n,j} \leftarrow \mathbf{s}_t$
11:         **for** $h = 0, \ldots, H-1$ **do**
12:             $\mathbf{s}_{t+h+1}^{n,j} \sim \hat{p}_\psi(\cdot|\mathbf{s}_{t+h}^{n,j}, \mathbf{a}_{t+h}^n, m_t^j)$ ▷ Sample next state from model using m_t^j
13:             $r_{t+h}^{n,j} \leftarrow r(\mathbf{s}_{t+h}^{n,j}, \mathbf{a}_{t+h}^n, m_t^j)$ ▷ Compute the reward using m_t^j
14:         **end for**
15:     **end for**
16: **end for**
17: Compute $\mathbb{E}_{p(\tau|\mathcal{O})}[\mathbf{a}_{t:t+H}]$ with (13)
18: **return** $\mathbb{E}_{p(\tau|\mathcal{O})}[\mathbf{a}_{t:t+H}]$ ▷ Return the plan to be used in line 7 of Algorithm 1

C Algorithm Details

C.1 Algorithm Summary

RefPlan is designed to enhance any offline RL policy by incorporating MB planning that accounts for epistemic uncertainty. The algorithm operates in two primary stages: pretraining (Appendix C.3) and test-time planning.

Pretraining stage The first step is to train a prior policy π_p using any offline RL algorithm. In parallel, a VAE is trained using the ELBO objective in (9), where the encoder captures the agent's epistemic uncertainty and the decoder learns the environment dynamics. See Appendix C.3 for more details.

Test-time planning stage During evaluation, the agent employs MPC (Algorithm 1), where RefPlan serves as the trajectory optimization subroutine. At each time step t, the agent gathers its history $\tau_{:t}$ and encodes it into a latent variable m_t using the pretrained encoder (line 2 of Algorithm 2). This latent variable encapsulates the agent's current belief about the environment, reflecting epistemic uncertainty.

Algorithm 1 Offline MB Planning

1: **Input:** $\hat{p}_\psi$, V_ϕ, $\mathcal{D}$, π_θ, $\mathfrak{L}$
2: Train $\hat{p}_\psi$ with $\mathcal{D}$ via MLE
3: Train V_ϕ and π_θ with $\mathfrak{L}$ and $\mathcal{D}$
4: $t \leftarrow 1$
5: **repeat**
6:     Observe $\mathbf{s}_t$
7:     $\mathbf{a}_{t:t+H}^* \leftarrow$TrajOpt$(\mathbf{s}_t, \hat{p}_\psi, \pi_\theta, V_\phi)$
8:     Take $\mathbf{a}_t^*$, observe $\mathbf{s}_{t+1}, r_t$
9:     $t \leftarrow t+1$
10: **until** episode terminates

Then, we first generate $\bar{N}$ prior plans with the prior policy and the learned model (lines 5-8). Each plan has the length of H, and we use μ_t to condition $\hat{p}_\psi$ at this stage. Optionally, we add a Gaussian noise to the actions sampled by π_p, following Argenson & Dulac-Arnold (2021); Sikchi et al. (2021).

Once the prior plans are prepared, we rollout the plans under the learned model to generate multiple trajectories. That is, for each sampled m_t, we obtain $\bar{N}$ trajectories (lines 9-14). These trajectories are then used to estimate the optimal plan, conditioned on m_t^j. We marginalize out the latent variable via Monte-Carlo expectation using the law of total expectation.

Finally, the first action from the optimized plan is selected and executed in the environment. This process repeats at each subsequent time step, with the agent continuously updating its belief state and re-optimizing its plan based on new observations.

Table 3: Hyperparameters for Model Architecture and Training

Architecture Hyperparameters	**Value**
Task Embedding Dimension	16
State Embedding Dimension	16
Action Embedding Dimension	16
Reward Embedding Dimension	4
GRU Hidden Dimension	256
Decoder Network Architecture	Fully connected, [200, 200, 200, 200] with skip connection
Decoder ensemble size	20
Decoder number of elite models	14
Training Hyperparameters	**Value**
KL Weight Coefficient	0.1
Input Normalization	True
Learning Rate	0.001
Weight Decay	0.01
Optimizer	AdamW
Batch Size	64

C.2 Architecture

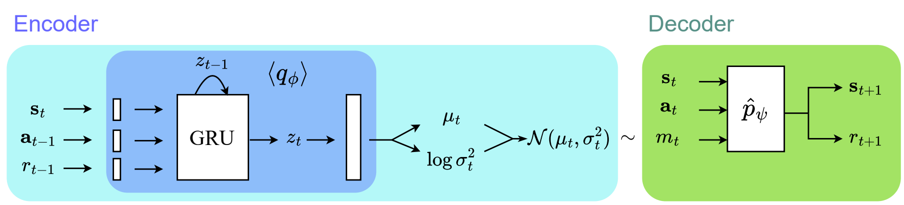


Figure 10: A schematic illustration of the architecture of RefPlan. We use the same encoder architecture as in VariBAD (Zintgraf et al., 2020), which consists of a GRU model and a fully connected layer. Unlike VariBAD, which uses the decoder only for training the encoder, we employ a two-stage training procedure (Appendix C.3) to learn a decoder that is directly used for planning at test time. The decoder network reconstructs the past trajectory and predicts the next state but does not attempt to predict the entire future trajectory as in the prior work (see also Eq.(10)).

Figure 10 illustrates the architecture of RefPlan. For the encoder, we adopt the architecture from VariBAD (Zintgraf et al., 2020), with a few minor modifications to the hyperparameters. The encoder utilizes a GRU network to encode the agent's history and outputs the parameters of a Gaussian distribution representing the latent variable m_t.

At time $t = 0$, we initialize $z_{-1} = 0$ and $\mathbf{a}_{-1} = 0$. The state $\mathbf{s}_t$, the previous action $\mathbf{a}_{t-1}$, and the previous reward r_{t-1} are first embedded into their respective latent spaces using distinct linear layers, each followed by ReLU activation. These embedded vectors, along with the hidden state from the previous time step z_{t-1}, are then processed by the GRU, which outputs the updated hidden state z_t. This hidden state is subsequently linearly projected onto the task embedding space to obtain the mean (μ_t) and log variance ($\log \sigma_t^2$) of the Gaussian distribution for the latent variable at the current time step.

Since the decoder plays a critical role in test-time planning, we follow established practices from prior work and implement the decoder using a probabilistic ensemble network (Chua et al., 2018; Janner et al., 2019; Yu et al., 2020; 2021; Chen et al., 2021). Specifically, the ensemble consists of 20 models, from which we select the 14 *elite* models that achieve the lowest validation loss during training. The decoder network conditions on a latent sample $m_t \sim \mathcal{N}(\mu_t, \sigma_t^2)$, along with $\mathbf{s}_t$ and $\mathbf{a}_t$, to predict the next state $\mathbf{s}_{t+1}$ and reward r_{t+1}. The hyperparameters associated with the architecture are summarized in Table 3.

Table 4: Reproducing the reported performances of offline policy learning algorithms on the D4RL MuJoCo tasks. *Numbers reported in An et al. (2021).

		CQL		EDAC		MOPO		COMBO		MAPLE	
		Paper	Rep.	Paper	Rep.	Paper	Rep.	Paper	Rep.	Paper	Rep.
Hopper	R	10.8	1.0	25.3	23.6	11.7	32.2	17.9	6.3	10.6	31.5
	M	86.6	66.9	101.6	101.5	28.0	66.9	97.2	60.9	21.1	29.4
	MR	48.6	94.6	101.0	100.4	67.5	90.3	89.5	101.1	87.5	61.0
	ME	111.0	111.4	110.7	106.7	23.7	91.3	111.1	105.6	42.5	46.9
	FR	101.9*	104.2	105.4	106.6	-	73.2	-	89.9	-	79.1
HalfCheetah	R	35.4	19.9	28.4	22.5	35.4	29.8	38.8	40.3	38.4	33.5
	M	44.4	47.4	65.9	63.8	42.3	42.8	54.2	67.2	50.4	68.8
	MR	46.2	47.0	61.3	61.8	53.1	70.6	55.1	73.0	59.0	71.5
	ME	62.4	98.3	106.3	100.8	63.3	73.5	90.0	97.6	63.5	64.0
	FR	76.9*	77.5	84.6	81.7	-	81.7	-	71.8	-	66.8
Walker2d	R	7.0	0.1	16.6	17.5	13.6	13.3	7.0	4.1	21.7	21.8
	M	74.5	77.1	92.5	77.6	17.8	82.0	81.9	71.2	56.3	88.3
	MR	32.6	63.5	87.1	85.0	39.0	81.7	56.0	88.0	76.7	85.0
	ME	98.7	108.9	114.7	98.5	44.6	51.9	103.3	108.3	73.8	111.8
	FR	94.2*	96.6	99.8	98.0	-	90.5	-	78.1	-	94.2

C.3 Pretraining

RefPlan requires two stages of pretraining. First, we use an off-the-shelf offline RL algorithm to train a prior policy π_{p}. In our experiments, we evaluated several algorithms, including CQL (Kumar et al., 2020), EDAC (An et al., 2021), MOPO (Yu et al., 2020), COMBO (Yu et al., 2021), and MAPLE (Chen et al., 2021), though any offline RL policy learning algorithm could be utilized.

Second, we train the encoder q_ϕ and the decoder $\hat{p}_\psi$. The encoder q_ϕ is trained using the ELBO loss as defined in (9). The decoder $\hat{p}_\psi$ is trained to reconstruct the past and to predict the next state, conditioned on the sample m_t the current state $\mathbf{s}_t$, and the action $\mathbf{a}_t$. This training constitutes the first phase of dynamics learning. During this step, the encoder learns a latent representation that captures essential information for reconstructing the trajectory. Unlike VariBAD, where the decoder is trained to reconstruct the entire trajectory including future states, we found that focusing on the past and the next state improves the decoder's performance.

After completing the first training phase, we freeze the encoder network parameters and proceed to the second phase. In this phase, we fine-tune the decoder network $\hat{p}_\psi$ to accurately predict the next state given m_t, $\mathbf{s}_t$, and $\mathbf{a}_t$. This is achieved using the loss function defined in (11), which we reiterate here for clarity:

$$L(\psi) = \mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_{h=0}^{H-1} \mathbb{E}_{m_h\sim q_\phi(\cdot|\tau_{:h})}[-\log \hat{p}_\psi(\mathbf{s}_{h+1}, r_h|\mathbf{s}_h, \mathbf{a}_h, m_h)]\right]. \tag{19}$$

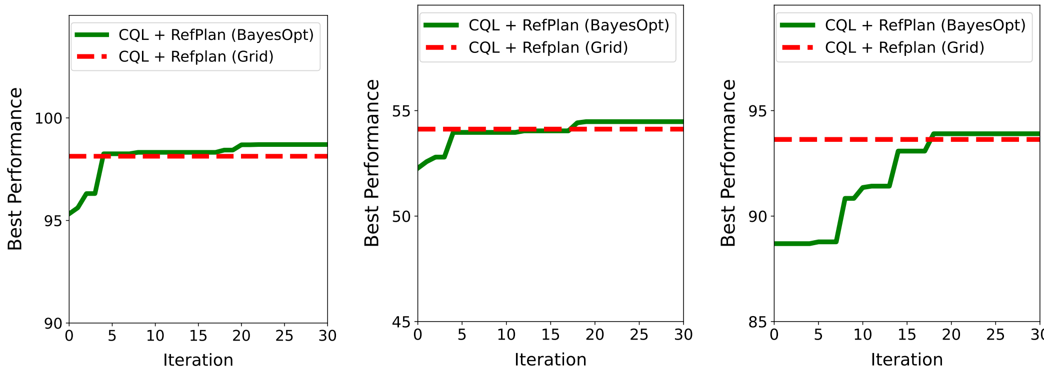
The second training phase ensures that the learned dynamics model, $\hat{p}_\psi$, accurately predicts the next state. This two-stage approach allows RefPlan to maintain an effective dynamics moel for planning at test time, unlike VariBAD, where the decoder is discarded after training the VAE.

D Experimental Details

D.1 Experimental settings

D4RL MuJoCo environments & datasets We use the v2 version for each dataset as provided by the D4RL library (Fu et al., 2020).

Evaluation under high epistemic uncertainty due to OOD initialization (RQ1) To address RQ1, we assessed a policy trained on the ME dataset of each MuJoCo environment by initializing the agent from a state randomly selected from the R dataset. The results, presented in Figure 3, 7, and 8 are averaged over 3 seeds. For a fair comparison, the same initial state was used across all methods being compared—the prior policy, LOOP, and RefPlan—under the same random seed.



(a) Hopper (MR) (b) HalfCheetah (MR) (c) Walker2d (MR)

Figure 11: Best performance vs. the number of BayesOpt iterations, using CQL as a prior policy on the MR datasets across three environments.

Benchmarking on D4RL tasks (RQ2) To generate the benchmark results shown in Table 1, we first trained the five baseline policies on each dataset across the three environments. The focus of our analysis is on the performance improvements of these prior policies when augmented with either LOOP or RefPlan as an MB planning algorithm during the evaluation phase. *Thus, our approach is designed to be complementary to any offline policy learning algorithms, making the relative performance gains more relevant than the absolute performance of each algorithm.* Nevertheless, we aimed to closely replicate the original policy performance reported in prior studies. Table 4 compares our reproduced results with those originally reported. Overall, our implementation closely matches the original performances, often exceeding them significantly across various datasets. However, in some cases, our reproduced policy checkpoints underperformed compared to the originally reported results, such as CQL on the R datasets, EDAC on Walker2d M and ME datasets, COMBO on the Hopper R and M datasets, and MAPLE on the Hopper MR dataset. We will make our code publicly available upon acceptance.

Varying dataset sizes (RQ3) In Figure 4, we present the normalized average return scores for CQL and its enhancements with either LOOP or RefPlan as we vary the dataset size from 50K to 500K. We conducted these experiments using the FR dataset across three environments, which originally contains 1M transition samples. To create the smaller datasets, we randomly subsampled trajectories. If the subsampled data exceeded the desired dataset size, we trimmed the last trajectory accordingly. For CQL training, we applied the same hyperparameters as those used for the full FR dataset.

Changing dynamics (RQ4) To explore RQ4, we adapted the HalfCheetah environment following the approach of Clavera et al. (2019), introducing five variations: *disabled joint*, *hill*, *gentle* slope, *steep* slope, and *field*. These variations were implemented using the code from `https://github.com/iclavera/learning_to_adapt`. Unlike the original work, which focuses on meta-RL, our study addresses an offline RL problem within a single task framework. Hence, to make the tasks easier, we modified the `height` parameter for most variations, excluding the *disabled joint* task. The specific adjustments to the height parameters are detailed in Table 5. These changes were intended to create more manageable tasks while still providing a meaningful challenge for the offline RL algorithms.

Table 5: Environment configuration for the HalfCheetah variations used in RQ4 experiments, showing the original and modified `height` parameter values for each task.

Task	Original	Modified
hill	0.6	0.2
gentle	1	0.2
steep	4	0.5
field	$Uniform(0.2, 1)$	$Uniform(0.05, 0.4)$

Table 6: Hyperparameters used for MAPLE + RefPlan used on D4RL MuJoCo Gym environments

	Hopper					HalfCheetah					Walker2D				
	H	σ	κ	$\bar{n}$	p	H	σ	κ	$\bar{n}$	p	H	σ	κ	$\bar{n}$	p
random	2	0.01	10.0	1	1.0	4	0.05	10.0	16	1.0	2	0.05	0.1	16	1.0
medium	2	0.01	0.1	8	0.1	4	0.01	5.0	16	0.5	2	0.05	10.0	1	0.1
med-replay	4	0.01	5.0	1	0.1	4	0.01	10.0	8	0.1	4	0.05	0.1	16	0.1
med-expert	2	0.01	0.1	1	1.0	4	0.01	10.0	4	0.1	2	0.01	10.0	8	0.5
full-replay	2	0.01	10.0	1	0.5	2	0.01	5.0	16	0.1	2	0.05	5.0	1	1.0

Table 7: Hyperparameters used for COMBO + RefPlan used on D4RL MuJoCo Gym environments

	Hopper					HalfCheetah					Walker2d				
	H	σ	κ	$\bar{n}$	p	H	σ	κ	$\bar{n}$	p	H	σ	κ	$\bar{n}$	p
random	2	0.05	0.1	16	0.1	2	0.05	0.1	4	0.1	4	0.01	0.5	16	1.0
medium	2	0.01	10.0	16	0.5	4	0.05	5.0	16	0.1	4	0.05	1.0	16	0.5
med-replay	4	0.01	0.5	8	1.0	2	0.01	5.0	4	0.1	4	0.01	5.0	16	0.1
med-expert	4	0.01	0.5	8	1.0	2	0.05	5.0	16	0.1	4	0.01	10.0	16	0.1
full-replay	4	0.01	0.1	8	0.5	4	0.01	10.0	4	0.1	4	0.05	1.0	8	0.5

Table 8: Hyperparameters used for MOPO + RefPlan used on D4RL MuJoCo Gym environments

	Hopper					HalfCheetah					Walker2d				
	H	σ	κ	$\bar{n}$	p	H	σ	κ	$\bar{n}$	p	H	σ	κ	$\bar{n}$	p
random	2	0.01	5.0	1	0.1	4	0.01	10.0	8	0.1	2	0.05	0.1	8	1.0
medium	2	0.05	5.0	1	0.1	4	0.05	10.0	4	0.1	2	0.05	5.0	4	1.0
med-replay	4	0.05	5.0	16	0.1	2	0.05	10.0	4	0.1	4	0.01	1.0	16	0.1
med-expert	4	0.05	1.0	8	0.1	4	0.01	10.0	16	0.1	4	0.01	10.0	16	1.0
full-replay	2	0.05	5.0	16	1.0	4	0.05	5.0	16	0.1	4	0.05	10.0	1	0.1

Table 9: Hyperparameters used for CQL + RefPlan used on D4RL MuJoCo Gym environments

	Hopper					HalfCheetah					Walker2d				
	H	σ	κ	$\bar{n}$	p	H	σ	κ	$\bar{n}$	p	H	σ	κ	$\bar{n}$	p
random	4	0.01	10.0	1	0.1	4	0.05	5.0	1	0.1	4	0.05	10.0	16	1.0
medium	2	0.01	5.0	16	0.5	2	0.01	5.0	1	0.5	2	0.05	10.0	16	1.0
med-replay	4	0.05	0.1	8	0.1	4	0.01	5.0	16	0.1	4	0.05	1.0	4	0.1
med-expert	4	0.01	1.0	1	0.5	2	0.01	5.0	8	0.1	2	0.01	5.0	8	0.1
full-replay	4	0.05	10.0	16	0.1	2	0.01	5.0	8	0.1	4	0.01	5.0	8	0.1

Table 10: Hyperparameters used for EDAC + RefPlan used on D4RL MuJoCo Gym environments

	Hopper					HalfCheetah					Walker2d				
	H	σ	κ	$\bar{n}$	p	H	σ	κ	$\bar{n}$	p	H	σ	κ	$\bar{n}$	p
random	4	0.05	10.0	16	0.5	4	0.05	5.0	8	0.1	2	0.05	10.0	4	0.1
medium	2	0.01	10.0	16	0.1	4	0.05	10.0	4	0.1	4	0.05	10.0	1	0.1
med-replay	2	0.05	10.0	1	0.1	4	0.05	10.0	8	0.1	4	0.05	5.0	16	0.1
med-expert	2	0.01	1.0	16	0.5	2	0.05	5.0	8	0.1	4	0.05	5.0	16	0.1
full-replay	4	0.05	10.0	4	0.1	2	0.05	10.0	8	0.1	2	0.05	10.0	1	0.1

D.2 HYPERPARAMETERS

Table 6-10 outline the hyperparameters used for RefPlan across the five prior policies discussed in Section 5. We conducted a grid search over the following hyperparameters: the planning horizon $H \in \{2, 4\}$, the standard deviation of the Gaussian noise $\sigma \in \{0.01, 0.05\}$, the inverse temperature parameter $\kappa \in \{0.1, 0.5, 1.0, 5.0, 10.0\}$, the number of latent samples $\bar{n} \in \{1, 4, 8, 16\}$, and the value uncertainty penalty $p \in \{0.1, 0.5, 1.0\}$. Our findings indicate that κ and $\bar{n}$ are the most influential hyperparameters, while the others have a comparatively minor effect on performance. For LOOP, we conducted a similar grid search over the same hyperparameters, excluding $\bar{n}$, which is specific to RefPlan.

Table 11: Per-epoch runtimes for VAE pretraining on the ME dataset.

Hopper		HalfCheetah		Walker2d	
q_ϕ	$\hat{p}_\psi$	q_ϕ	$\hat{p}_\psi$	q_ϕ	$\hat{p}_\psi$
55.3s	39.8s	53.2s	40.7s	54.6s	40.7s

Table 12: Runtime per environment step for RefPlan during evaluation in the HalfCheetah environment.

H \ $\bar{n}$	1	2	3	4
2	7.9×10^{-3}s	8.7×10^{-3}s	9.3×10^{-3}s	1.0×10^{-2}s
4	1.5×10^{-2}s	1.6×10^{-2}s	1.8×10^{-2}s	1.9×10^{-2}s

In addition, we used Bayesian optimization (BayesOpt, Snoek et al. (2012)), implemented in W&B (Biewald, 2020), to explore the challenge of identifying optimal hyperparameters for RefPlan. Figure 11 compares the number of iterations required for BayesOpt to achieve or surpass the performance of the best hyperparameter configuration found via grid search in each environment. Specifically, we used CQL as the prior policy and the MR dataset from three environments. Notably, BayesOpt required fewer than 20 iterations to exceed the performance reported in Table 1.

D.3 Computational Costs of RefPlan

In this section, we provide a detailed discussion of the computational costs associated with deploying RefPlan. As outlined in Appendix C.3, RefPlan requires the following pretrained components: a prior policy π_{p}, an encoder q_ϕ, and a decoder $\hat{p}_\psi$. Since the prior policy is trained using standard offline policy learning algorithms (e.g., CQL, EDAC, MOPO, COMBO, and MAPLE), which are not our contributions, we focus on reporting the computational costs associated with training the VAE model and executing the planning stage. All experiments were conducted on a single machine equipped with an RTX 3090 GPU.

VAE Pretraining Table 11 presents the per-epoch runtimes for VAE pretraining in the three environments. The reported runtimes correspond to datasets with 2M transition samples, the largest dataset size used in our experiments. Both the VAE pretraining and decoder fine-tuning phases were executed for up to 200 epochs or until the validation loss ceased to improve for 5 consecutive epochs, whichever occurred first.

Test-Time Planning At test time, planning with RefPlan involves selecting hyperparameters as detailed in Appendix D.2. Among these, the planning horizon H and the number of latent samples $\bar{n}$ influence runtime. Specifically, the computational cost scales linearly with H, which is an inherent property of planning algorithms. However, the cost increases sub-linearly with $\bar{n}$, as shown in Table 12. For example, with $H = 4$ and $\bar{n} = 4$, the agent achieves approximately 53 environment steps per second. We hypothesize that further optimization of PyTorch tensor operations to fully exploit GPU parallelism could yield even better computational performance, particularly with respect to $\bar{n}$.