

CONSERVATIVE BAYESIAN MODEL-BASED VALUE EXPANSION FOR OFFLINE POLICY OPTIMIZATION

Jihwan Jeong^{1,3*}, Xiaoyu Wang^{1*}, Michael Gimelfarb^{1,3}, Hyunwoo Kim^{2†}, Baher Abdulhai¹
& Scott Sanner^{1,3†}

¹University of Toronto, ²LG AI Research, ³Vector Institute
{jihwan.jeong, cnxiaoyu.wang, mike.gimelfarb}@mail.utoronto.ca,
hwkim@lgresearch.ai, baher.abdulhai@utoronto.ca,
ssanner@mie.utoronto.ca

ABSTRACT

Offline reinforcement learning (RL) addresses the problem of learning a performant policy from a fixed batch of data collected by following some behavior policy. Model-based approaches are particularly appealing in the offline setting since they can extract more learning signals from the logged dataset by learning a model of the environment. However, the performance of existing model-based approaches falls short of model-free counterparts, due to the compounding of estimation errors in the learned model. Driven by this observation, we argue that it is critical for a model-based method to understand when to trust the model and when to rely on model-free estimates, and how to act conservatively w.r.t. both. To this end, we derive an elegant and simple methodology called conservative Bayesian model-based value expansion for offline policy optimization (CBOP), that trades off model-free and model-based estimates during the policy evaluation step according to their epistemic uncertainties, and facilitates conservatism by taking a lower bound on the Bayesian posterior value estimate. On the standard D4RL continuous control tasks, we find that our method significantly outperforms previous model-based approaches: e.g., MOPO by 116.4%, MOREL by 23.2% and COMBO by 23.7%. Further, CBOP achieves state-of-the-art performance on 11 out of 18 benchmark datasets while doing on par on the remaining datasets.

1 INTRODUCTION

Fueled by recent advances in supervised and unsupervised learning, there has been a great surge of interest in data-driven approaches to reinforcement learning (RL), known as *offline RL* (Levine et al., 2020). In offline RL, an RL agent must learn a good policy entirely from a logged dataset of past interactions, without access to the real environment. This paradigm of learning is particularly useful in applications where it is prohibited or too costly to conduct online trial-and-error explorations (e.g., due to safety concerns), such as autonomous driving (Yu et al., 2018), robotics (Kalashnikov et al., 2018), and operations research (Boute et al., 2022).

However, because of the absence of online interactions with the environment that give correcting signals to the learner, direct applications of *online* off-policy algorithms have been shown to fail in the *offline* setting (Fujimoto et al., 2019; Kumar et al., 2019; Wu et al., 2019; Kumar et al., 2020). This is mainly ascribed to the distribution shift between the learned policy and the *behavior policy* (data-logging policy) during training. For example, in Q -learning based algorithms, the distribution shift in the policy can incur uncontrolled overestimation bias in the learned value function. Specifically, positive biases in the Q function for out-of-distribution (OOD) actions can be picked up during policy maximization, which leads to further deviation of the learned policy from the behavior policy, resulting in a vicious cycle of value overestimation. Hence, the design of offline RL algorithms revolves around how to counter the adverse impacts of the distribution shift while achieving improvements over the data-logging policy.

*Equal contribution

†Corresponding authors

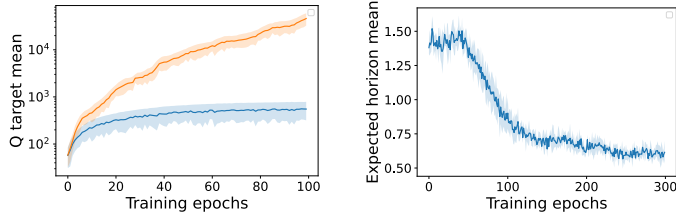


Figure 1: Prevention of value overestimation & adaptive reliance on model-based value predictions. (*Left*) We leverage the full posterior over the target values to prevent value overestimation during offline policy learning (blue). Without conservatism incorporated, the target value diverges (orange). (*Right*) We can automatically adjust the level of reliance on the model-based and bootstrapped model-free value predictions based on their respective uncertainty during model-based value expansion. The ‘expected horizon’ ($\mathbb{E}[h] = \sum_h w_h \cdot h$, $\sum_h w_h = 1$) shows an effective model-based rollout horizon during policy optimization. $\mathbb{E}[h]$ is large at the beginning, but it gradually decreases as the model-free value estimates improve over time. The figures were generated using the *hopper-random* dataset from D4RL (Fu et al., 2020).

In this work, we consider model-based (MB) approaches since they allow better use of a given dataset and can provide better generalization capability (Yu et al., 2020; Kidambi et al., 2020; Yu et al., 2021; Argenson & Dulac-Arnold, 2021). Typically, MB algorithms — e.g., MOPO (Yu et al., 2020), MOREL (Kidambi et al., 2020), and COMBO (Yu et al., 2021) — adopt the Dyna-style policy optimization approach developed in online RL (Janner et al., 2019; Sutton, 1990). That is, they use the learned dynamics model to generate rollouts, which are then combined with the real dataset for policy optimization.

We hypothesize that we can make better use of the learned model by employing it for target value estimation during the policy evaluation step of the actor-critic method. Specifically, we can compute h -step TD targets through dynamics model rollouts and bootstrapped terminal Q function values. In online RL, this MB value expansion (MVE) has been shown to provide a better value estimation of a given state (Feinberg et al., 2018). However, the naïve application of MVE does not work in the offline setting due to model bias that can be exploited during policy learning.

Therefore, it is critical to trust the model only when it can reliably predict the future, which can be captured by the epistemic uncertainty surrounding the model predictions. To this end, we propose CBOP (Conservative Bayesian MVE for Offline Policy Optimization) to control the reliance on the model-based and model-free value estimates according to their respective uncertainties, while mitigating the overestimation errors in the learned values. Unlike existing MVE approaches (e.g., Buckman et al. (2018)), CBOP estimates the *full* posterior distribution over a target value from the h -step TD targets for $h = 0, \dots, H$ sampled from ensembles of the state dynamics and the Q function. The novelty of CBOP lies in its ability to fully leverage this uncertainty in two related ways: (1) by deriving an adaptive weighting over different h -step targets informed by the posterior uncertainty; and (2) by using this weighting to derive *conservative* lower confidence bounds (LCB) on the target values that mitigates value overestimation. Ultimately, this allows CBOP to reap the benefits of MVE while significantly reducing value overestimation in the offline setting (Figure 1).

We evaluate CBOP on the D4RL benchmark of continuous control tasks (Fu et al., 2020). The experiments show that using the conservative target value estimate significantly outperforms previous model-based approaches: e.g., MOPO by 116.4%, MOREL by 23.2% and COMBO by 23.7%. Further, CBOP achieves state-of-the-art performance on 11 out of 18 benchmark datasets while doing on par on the remaining datasets.

2 BACKGROUND

We study RL in the framework of *Markov decision processes* (MDPs) that are characterized by a tuple $(\mathcal{S}, \mathcal{A}, T, r, d_0, \gamma)$; here, \mathcal{S} is the state space, \mathcal{A} is the action space, $T(s'|s, \mathbf{a})$ is the transition function, $r(s, \mathbf{a})$ is the immediate reward function, d_0 is the initial state distribution, and $\gamma \in [0, 1]$ is the discount factor. Specifically, we call the transition and reward functions the *model* of the environment, which we denote as $f = (T, r)$. A *policy* π is a mapping from \mathcal{S}

to \mathcal{A} , and the goal of RL is to find an optimal policy π^* which maximizes the expected cumulative discounted reward, $\mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t} [\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$, where $\mathbf{s}_0 \sim d_0$, $\mathbf{s}_t \sim T(\cdot | \mathbf{s}_{t-1}, \mathbf{a}_{t-1})$, and $\mathbf{a}_t \sim \pi^*(\cdot | \mathbf{s}_t)$. Often, we summarize the quality of a policy π by the state-action value function $Q^\pi(\mathbf{s}, \mathbf{a}) := \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t} [\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) | \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}]$, where $\mathbf{a}_t \sim \pi(\cdot | \mathbf{s}_t) \forall t > 0$.

Off-policy actor-critic methods, such as SAC (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018), have enjoyed great successes in complex continuous control tasks in deep RL, where parameterized neural networks for the policy π_θ (known as actor) and the action value function Q_ϕ (known as critic) are maintained. Following the framework of the *generalized policy iteration* (GPI) (Sutton & Barto, 2018), we understand the actor-critic algorithm as iterating between (i) policy evaluation and (ii) policy improvement. Here, policy evaluation typically refers to the calculation of $Q_\phi(\mathbf{s}, \pi_\theta(\mathbf{s}))$ for the policy π_θ , while the improvement step is often as simple as maximizing the currently evaluated Q_ϕ ; i.e., $\max_{\theta} \mathbb{E}_{\mathbf{s} \sim \mathcal{D}} [Q_\phi(\mathbf{s}, \pi_\theta(\mathbf{s}))]$ (Fujimoto et al., 2018).

Policy Evaluation At each iteration of policy learning, we evaluate the current policy π_θ by minimizing the *mean squared Bellman error* (MSBE) with the dataset \mathcal{D} of previous state transitions:

$$\mathcal{L}(\phi, \mathcal{D}) = \text{MSBE} := \mathbb{E}_{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}') \sim \mathcal{D}} \left[(y(\mathbf{s}, \mathbf{a}, \mathbf{s}') - Q_\phi(\mathbf{s}, \mathbf{a}))^2 \right], \quad (1)$$

$$y(\mathbf{s}, \mathbf{a}, \mathbf{s}') = r(\mathbf{s}, \mathbf{a}) + \gamma Q_{\phi'}(\mathbf{s}', \mathbf{a}'), \quad \mathbf{a}' \sim \pi_\theta(\cdot | \mathbf{s}') \quad (2)$$

where $y(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ is the *TD target* at each (\mathbf{s}, \mathbf{a}) , towards which Q_ϕ is regressed. A separate *target* network $Q_{\phi'}$ is used in computing y to stabilize learning (Mnih et al., 2015). Off-policy algorithms typically use some variations of (2), e.g., by introducing the clipped double- Q trick (Fujimoto et al., 2018), in which $\min_{j=1,2} Q_{\phi'_j}(\mathbf{s}', \mathbf{a}')$ is used instead of $Q_{\phi'}(\mathbf{s}', \mathbf{a}')$ to prevent value overestimation.

Model-based Offline RL In the offline setting, we are given a fixed set of transitions, \mathcal{D} , collected by some *behavior* policy π_β , and the aim is to learn a policy π that is better than π_β . In particular, offline *model-based* (MB) approaches learn the model $\hat{f} = (\hat{T}, \hat{r})$ of the environment using \mathcal{D} to facilitate the learning of a good policy. Typically, \hat{f} is trained to maximize the log-likelihood of its predictions. Though MB algorithms are often considered capable of better generalization than their *model-free* (MF) counterparts by leveraging the learned model, it is risky to trust the model for OOD samples. Hence, MOPO (Yu et al., 2020) and MOREL (Kidambi et al., 2020) construct and learn from a pessimistic MDP where the model uncertainty in the next state prediction is penalized in the reward. Criticizing the difficulty of accurately computing well-calibrated model uncertainty, COMBO (Yu et al., 2021) extends CQL (Kumar et al., 2020) to the model-based regime by regularizing the value function on OOD samples generated via model rollouts. These methods follow the *Dyna-style* policy learning where model rollouts are used to augment the offline dataset (Sutton, 1990; Janner et al., 2019).

Model-based Value Expansion (MVE) for Policy Optimization An alternative to the aforementioned *Dyna-style* approaches is MVE (Feinberg et al., 2018), which is arguably better suited to seamlessly integrating the power of both MF and MB worlds. In a nutshell, MVE attempts to more accurately estimate the TD target in (2) by leveraging a model of the environment, which can lead to more efficient policy iteration. Specifically, we can use the h -step MVE target $\hat{R}_h(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ for $y(\mathbf{s}, \mathbf{a}, \mathbf{s}')$:

$$\hat{y}(\mathbf{s}, \mathbf{a}, \mathbf{s}') = \hat{R}_h(\mathbf{s}, \mathbf{a}, \mathbf{s}') := \sum_{t=0}^h \gamma^t \hat{r}_t(\hat{\mathbf{s}}_t, \hat{\mathbf{a}}_t) + \gamma^{h+1} Q_{\phi'}(\hat{\mathbf{s}}_{h+1}, \hat{\mathbf{a}}_{h+1}), \quad (3)$$

$$(\hat{\mathbf{s}}_0, \hat{\mathbf{a}}_0, \hat{r}_0, \hat{\mathbf{s}}_1) = (\mathbf{s}, \mathbf{a}, r, \mathbf{s}'), \quad \hat{\mathbf{s}}_t \sim \hat{T}(\cdot | \hat{\mathbf{s}}_{t-1}, \hat{\mathbf{a}}_{t-1}), \quad \hat{\mathbf{a}}_t \sim \pi_\theta(\cdot | \hat{\mathbf{s}}_t), \quad 1 \leq t \leq h+1,$$

where $\hat{R}_h(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ is obtained by the h -step MB return plus the terminal value at $h+1$ ($h=0$ reduces back to MF). In reality, errors in the learned model \hat{f} compound if rolled out for a large h . Thus, it is standard to set h to a small number.

3 CONSERVATIVE BAYESIAN MVE FOR OFFLINE POLICY OPTIMIZATION

The major limitations of MVE when applied to offline RL are as follows:

1. The model predictions \hat{s}_t and \hat{r}_t in (3) become increasingly less accurate as t increases because model errors can compound, leading to largely biased target values. This issue is exacerbated in the offline setup because we cannot obtain additional experiences to reduce the model error.
2. The most common sidestep to avoid the issue above is to use short-horizon rollouts only. However, rolling out the model for only a short horizon even when the model can be trusted could severely restrict the benefit of being model-based.
3. Finally, when the model rollouts go outside the support of \mathcal{D} , \hat{R}_h in (3) can have a large overestimation bias, which will eventually be propagated into the learned Q_ϕ function.

Ideally, we want to **control the reliance on the model \hat{f} and the bootstrapped $Q_{\phi'}$ according to their respective epistemic uncertainty, while also preventing Q_ϕ from accumulating large overestimation errors**. That is, when we can trust \hat{f} , we can safely roll out the model for more steps to get a better value estimation. On the contrary, if the model is uncertain about the future it predicts, we should shorten the rollout horizon and bootstrap from $Q_{\phi'}$ early on. Indeed, Figure 1 (right) exemplifies that CBOP relies much more on the MB rollouts at the beginning of training because the value function is just initialized. As $Q_{\phi'}$ becomes more accurate over time, CBOP automatically reduces the weights assigned to longer MB rollouts.

Below, we present CBOP, a Bayesian take on achieving the aforementioned two goals: trading off the MF and MB value estimates based on their uncertainty while obtaining a conservative estimation of the target $\hat{y}(s, \mathbf{a}, s')$. To this end, we first let $\hat{Q}^\pi(s_t, \mathbf{a}_t)$ denote the value of the policy π at (s_t, \mathbf{a}_t) in the learned MDP defined by its dynamics \hat{f} ; that is,

$$\hat{Q}^\pi(s_t, \mathbf{a}_t) = \mathbb{E}_{\hat{f}, \pi} \left[\sum_{k=0}^{\infty} \gamma^k \hat{r}(\hat{s}_{t+k}, \hat{\mathbf{a}}_{t+k}) \right], \quad (\hat{s}_t, \hat{\mathbf{a}}_t) = (s_t, \mathbf{a}_t), \quad \hat{\mathbf{a}}_{t+k} \sim \pi(\cdot | \hat{s}_{t+k}). \quad (4)$$

Note that in the offline MBRL setting, we typically cannot learn Q^π due to having only an approximation \hat{f} of the model, and thus we focus instead on learning \hat{Q}^π .

Although there exists a unique $\hat{Q}^\pi(s, \mathbf{a})$ at each (s, \mathbf{a}) given a fixed model \hat{f} , we cannot directly observe the value unless we infinitely roll out the model from (s, \mathbf{a}) until termination, which is computationally infeasible. Instead, we view each $\hat{R}_h \forall h$ defined in (3) as a conditionally independent

(biased) noisy observation of the true underlying parameter \hat{Q}^π .¹ From this assumption, we can construct the Bayesian posterior over \hat{Q}^π given the observations $\hat{R}_h \forall h$. With the closed-form posterior distribution at hand, we can take various conservative estimates from the distribution; we use the lower confidence bound (LCB) in this work. Algorithm 1 summarizes the procedure at a high-level. Please see Algorithm 2 in Appendix B.1 for the full description of CBOP.

Algorithm 1 Conservative Bayesian MVE

Input: $(s_t, \mathbf{a}_t, r_t, s_{t+1}), \hat{f}, Q_{\phi'}$

1. Sample $\hat{R}_h \forall h \leq H$ using \hat{f} and $Q_{\phi'}$ as in (3)

2. Estimate μ_h, σ_h according to (8), (9)

3. Compute the posterior $\mathcal{N}(\mu, \sigma)$ using (7)

return conservative value target (e.g., LCB $\mu - \psi\sigma$)

3.1 CONSERVATIVE VALUE ESTIMATION VIA BAYESIAN INFERENCE

In this part, we formally discuss the conservative value estimation of CBOP based on Bayesian posterior inference. Specifically, the parameter of interest is \hat{Q}^π , and we seek its posterior estimation:

$$\mathbb{P}(\hat{Q}^\pi | \hat{R}_0, \dots, \hat{R}_H) \propto \mathbb{P}(\hat{R}_0, \dots, \hat{R}_H | \hat{Q}^\pi) \mathbb{P}(\hat{Q}^\pi) = \mathbb{P}(\hat{Q}^\pi) \prod_{h=0}^H \mathbb{P}(\hat{R}_h | \hat{Q}^\pi), \quad (5)$$

where we assume that \hat{R}_h ($h = 0, \dots, H$) are conditionally independent given \hat{Q}^π (see Appendix A where we discuss in detail about the assumptions present in CBOP).

In this work, we model the likelihood of observations $\mathbb{P}(\hat{R}_h | \hat{Q}^\pi)$ as normally distributed with the mean μ_h and the standard deviation σ_h :

$$\hat{R}_h | \hat{Q}^\pi \sim \mathcal{N}(\mu_h, \sigma_h^2), \quad (6)$$

¹We will omit (s, \mathbf{a}, s') henceforth if it is clear from the context.

since it leads to a closed-form posterior update. Furthermore, since \hat{R}_h can be seen as a sum of future immediate rewards, when the MDP is ergodic and γ is close to 1, the Gaussian assumption (approximately) holds according to the central limit theorem (Dearden et al., 1998). Also, note that our Bayesian framework is not restricted to the Gaussian assumptions, and other surrogate probability distributions such as the Student-t distribution could be used instead.

For the prior, we use the improper (or uninformative) prior, $\mathbb{P}(\hat{Q}^\pi) = 1$, since it is natural to assume that we lack generally applicable prior information over the target value across different environments and tasks (Christensen et al., 2011). The use of the improper prior is well justified in the Bayesian literature (Wasserman, 2010; Berger, 1985), and the particular prior we use in CBOP corresponds to the Jeffreys prior, which has the invariant property under a change of coordinates. The Gaussian likelihood and the improper prior lead to a ‘proper’ Gaussian posterior density that integrates to 1, from which we can make various probabilistic inferences (Wasserman, 2010).

The posterior (5) is a Gaussian with mean μ and variance σ^2 , defined as follows:

$$\rho = \sum_{h=0}^H \rho_h, \quad \mu = \sum_{h=0}^H \left(\frac{\rho_h}{\sum_{h=0}^H \rho_h} \right) \mu_h, \quad (7)$$

where $\rho = 1/\sigma^2$ and $\rho_h = 1/\sigma_h^2$ are the precisions of the posterior and the likelihood of \hat{R}_h , respectively. The posterior mean μ corresponds to the MAP estimation of \hat{Q}^π . Note that μ has the form of a weighted sum, $\sum_h w_h \mu_h$, with $w_h = \rho_h / \sum_{h=0}^H \rho_h \in (0, 1)$ being the weight allocated to \hat{R}_h . If the variance of \hat{R}_h for some h is relatively large, we give a smaller weight to that observation. If, on the other hand, \hat{R}_h all have the same variance (e.g. $\rho_0 = \dots = \rho_H$), we recover the usual H -step return estimate. Recall that the quality of \hat{R}_h is determined by that of the model rollout return and the bootstrapped terminal value. Thus intuitively speaking, the adaptive weight w_h given by the Bayesian posterior allows the trade-off between the epistemic uncertainty of the model with that of the Q function.

Figure 2 illustrates the overall posterior estimation procedure. Given a transition tuple (s, \mathbf{a}, r, s') , we start the model rollout from $s_1 = s'$. At each rollout horizon h , the cumulative discounted reward $\sum_{t=0}^h \gamma^t \hat{r}_t$ is sampled by the dynamics model and the terminal value \hat{Q}_h is sampled by the Q function (the sampling procedure is described in Section 3.2). We then get \hat{R}_h by adding the h -step MB return samples and the terminal values $\gamma^{h+1} \hat{Q}_{h+1}$, which we deem as sampled from the distribution $\mathbb{P}(\hat{R}_h | \hat{Q}^\pi)$ parameterized by μ_h, σ_h^2 (we use the sample mean and variance). These individual h -step observations are then combined through the Bayesian inference to give us the posterior distribution over \hat{Q}^π .

It is worth noting that the MAP estimator can also be derived from the perspective of variance optimization (Buckman et al., 2018) over the target values. However, we have provided much evidence in Section 4 and Appendix D.3 that the point estimate does not work in the offline setting due to value overestimation. Hence, it is imperative that we should have the full posterior distribution over the target value, such that we can make a conservative estimation rather than the MAP estimation.

To further understand the impact of using the MAP estimator for the value estimation, consider an estimator \tilde{Q} of \hat{Q}^π and its squared loss: $L(\hat{Q}^\pi, \tilde{Q}) = (\hat{Q}^\pi - \tilde{Q})^2$. It is known that the posterior mean of \hat{Q}^π minimizes the *Bayes risk* w.r.t. $L(\hat{Q}^\pi, \tilde{Q})$ (Wasserman, 2010), meaning that the posterior risk $\int L(\hat{Q}^\pi, \tilde{Q}) \mathbb{P}(\hat{Q}^\pi | \hat{R}_0, \dots, \hat{R}_H) d\hat{Q}^\pi$ is minimized at $\tilde{Q} = \mu$. In this context, μ is also called the (generalized) *Bayes estimator* of \hat{Q}^π , which is an admissible estimator (Robert, 2007). Despite seemingly advantageous, this result has a negative implication in offline RL. That is, the MAP estimator minimizes the squared loss from $\hat{Q}^\pi(s, \mathbf{a})$ over the entire support of the posterior, weighted by the posterior distribution. Now, the distribution shift of π from π_β can lead to significantly biased \hat{Q}^π compared to the true Q^π . In this case, the quality of the MAP estimator when

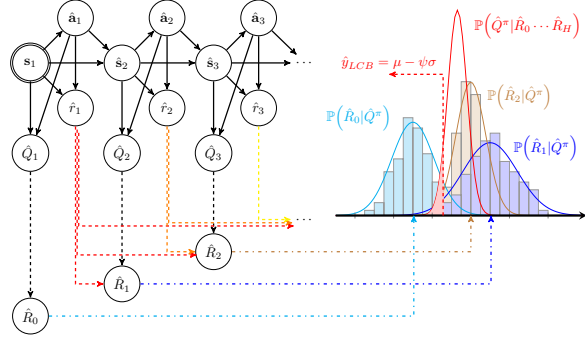


Figure 2: The graphical model representation of CBOP

evaluated in the real MDP would be poor. Especially, the overestimation bias in the MAP estimation can quickly propagate to the Q_ϕ function and thereby exacerbate the distribution shift.

3.2 ENSEMBLES OF DYNAMICS AND Q FUNCTIONS FOR SAMPLING H-STEP MVE TARGETS

In this section, we discuss how we estimate the parameters μ_h, σ_h^2 of $\mathbb{P}(\hat{R}_h | \hat{Q}^\pi)$ from the ensemble of dynamics models and that of Q functions.

Assume we have a bootstrapped dynamics ensemble model \hat{f} consisting of K different models $(\hat{f}_1, \dots, \hat{f}_K)$ trained with different sequences of mini-batches of \mathcal{D} (Chua et al., 2018; Janner et al., 2019). Similarly, we assume a Q ensemble of size M . Given a state \hat{s}_t and an action \hat{a}_t , we can construct the probability over the next state \hat{s}_{t+1} and reward \hat{r}_t by the ensemble as follows:

$$\mathbb{P}(\hat{s}_{t+1}, \hat{r}_t | \hat{s}_t, \mathbf{a}_t) = \sum_{k=1}^K \mathbb{P}(\hat{f}_k) \cdot \mathbb{P}(\hat{s}_{t+1}, \hat{r}_t | \hat{s}_t, \mathbf{a}_t, \hat{f}_k)$$

where $\mathbb{P}(\hat{f}_k)$ is the probability of selecting the k th model from the ensemble, which is $1/K$ when all models are weighted equally. Now, the sampling method that exactly follows the probabilistic graphical model shown in Figure 2 would first sample a model from the ensemble at each time step, followed by sampling the next state transition (and reward) from the model, which should then be repeated K times per state to generate a single sample. Then, we evaluate the resulting state \hat{s}_{t+1} and action $\hat{a}_{t+1} \sim \pi_\theta(\hat{s}_{t+1})$ with the Q ensemble to obtain M samples. To obtain N trajectories from a single initial state to estimate μ_h and σ_h^2 for $h = 1, \dots, H$, the overall procedure requires $\mathcal{O}(NKH)$ computation, which can quickly become infeasible for moderately large K and N values.

To reduce the computational complexity, we follow Chua et al. (2018) where each *particle* is propagated by a single model of the ensemble for H steps. With this, we can obtain N trajectories of length H from one state with $\mathcal{O}(NH)$ instead of $\mathcal{O}(NKH)$ (below we use $N = K$, i.e., we generate one particle per model). Concretely, given a single transition $\tau = (\mathbf{s}_0, \mathbf{a}_0, r_0, \mathbf{s}_1)$, we create K numbers of *particles* by replicating \mathbf{s}_1 K times, denoted as $\hat{s}_1^{(k)} \forall k$. The k th particle is propagated by a fixed model \hat{f}_k and the policy π_θ for H steps, where $(\hat{s}_t^{(k)}, \hat{r}_{t-1}^{(k)}) = \hat{f}_k(\hat{s}_{t-1}^{(k)}, \hat{a}_{t-1}^{(k)})$ and $\hat{a}_t^{(k)} \sim \pi_\theta(\hat{s}_t^{(k)})$. At each imagined timestep $t \in [0, H + 1]$, M number of terminal values are sampled by the $Q_{\phi'}$ ensemble at $(\hat{s}_t^{(k)}, \hat{a}_t^{(k)})$.

Despite the computational benefit, an implication of this sampling method is that it no longer directly follows the graphical model representation in Figure 2. However, we can still correctly estimate μ_h and σ_h^2 by turning to the law of total expectation and the law of total variance. That is,

$$\mu_h = \mathbb{E}_{\pi_\theta} [\hat{R}_h | \tau] = \mathbb{E}_{\hat{f}_k} [\mathbb{E}_{\pi_\theta} [\hat{R}_h | \tau, \hat{f}_k]] \quad (8)$$

where the outer expectation is w.r.t. the dynamics ensemble sampling probability $\mathbb{P}(\hat{f}_k) = 1/K$. Hence, given a fixed dynamics model \hat{f}_k , we sample \hat{R}_h by following π_θ and compute the average of the h -step return, which is then averaged across different ensemble models. In fact, the resulting μ_h is the mean of all aggregated $M \times K$ samples of \hat{R}_h .

The h -step return variance $\text{Var}_{\pi_\theta}(\hat{R}_h | \tau)$ decomposes via the law of total variance as following:

$$\sigma_h^2 = \text{Var}_{\pi_\theta} [\hat{R}_h | \tau] = \underbrace{\mathbb{E}_{\hat{f}_k} [\text{Var}_{\pi_\theta} [\hat{R}_h | \tau, \hat{f}_k]]}_A + \underbrace{\text{Var}_{\hat{f}_k} [\mathbb{E}_{\pi_\theta} [\hat{R}_h | \tau, \hat{f}_k]]}_B. \quad (9)$$

Here, A is related to the epistemic uncertainty of the $Q_{\phi'}$ ensemble; while B is associated with the epistemic uncertainty of the dynamics ensemble. The total variance $\text{Var}_{\pi_\theta}(\hat{R}_h | \tau)$ captures both uncertainties. This way, even though we use a different sampling scheme than presented in the graphical model of Figure 2, we can compute the unbiased estimators of the Gaussian parameters.

Once we obtain μ_h and σ_h^2 , we plug them into (7) to compute the posterior mean and the variance. A conservative value estimation can be made by $\hat{y}_{LCB} = \mu - \psi\sigma$ with some coefficient $\psi > 0$ (Jin et al., 2021; Rashidinejad et al., 2021). Under the Gaussian assumption, this corresponds to the worst-case return estimate in a Bayesian credible interval for \hat{Q}^π . We summarize CBOP in Algorithm 2 in Appendix B.1.

Table 1: Normalized scores on D4RL MuJoCo Gym environments. Experiments ran with 5 seeds.

		MOPO	MOReL	COMBO	CQL	TD3+BC	EDAC	IQL	CBOP
random	halfcheetah	35.4 ± 2.5	25.6	38.8	35.4	10.2 ± 1.3	28.4 ± 1.0	-	32.8 ± 0.4
	hopper	11.7 ± 0.4	53.6	17.9	10.8	11.0 ± 0.1	31.3 ± 0.0	-	31.4 ± 0.0
	walker2d	13.6 ± 2.6	37.3	7.0	7.0	1.4 ± 1.6	21.7 ± 0.0	-	17.8 ± 0.4
medium	halfcheetah	42.3 ± 1.6	42.1	54.2	44.4	42.8 ± 0.3	67.5 ± 1.2	47.4	74.3 ± 0.2
	hopper	28.0 ± 12.4	95.4	94.9	79.2	99.5 ± 1.0	101.6 ± 0.6	66.2	102.6 ± 0.1
	walker2d	17.8 ± 19.3	77.8	75.5	58.0	79.7 ± 1.8	92.5 ± 0.8	78.3	95.5 ± 0.4
medium replay	halfcheetah	53.1 ± 2.0	40.2	55.1	46.2	43.3 ± 0.5	63.9 ± 0.8	44.2	66.4 ± 0.3
	hopper	67.5 ± 24.7	93.6	73.1	48.6	31.4 ± 3.0	101.8 ± 0.5	94.7	104.3 ± 0.4
	walker2d	39.0 ± 9.6	49.8	56.0	26.7	25.2 ± 5.1	87.1 ± 2.3	73.8	92.7 ± 0.9
medium expert	halfcheetah	63.3 ± 38.0	53.3	90.0	62.4	97.9 ± 4.4	107.1 ± 2.0	86.7	105.4 ± 1.6
	hopper	23.7 ± 6.0	108.7	111.1	98.7	112.2 ± 0.2	110.7 ± 0.1	91.5	111.6 ± 0.2
	walker2d	44.6 ± 12.9	95.6	96.1	111.0	101.1 ± 9.3	114.7 ± 0.9	109.6	117.2 ± 0.5
expert	halfcheetah	-	-	-	-	105.7 ± 1.9	106.8 ± 3.4	-	100.4 ± 0.9
	hopper	-	-	-	-	112.2 ± 0.2	110.3 ± 0.3	-	111.4 ± 0.2
	walker2d	-	-	-	-	105.7 ± 2.7	115.1 ± 1.9	-	122.7 ± 0.8
full replay	halfcheetah	-	-	-	-	-	84.6 ± 0.9	-	85.5 ± 0.3
	hopper	-	-	-	-	-	105.4 ± 0.7	-	108.1 ± 0.3
	walker2d	-	-	-	-	-	99.8 ± 0.7	-	107.8 ± 0.2

4 EXPERIMENTS

We have designed the experiments to answer the following research questions: **(RQ1)** Is CBOP able to adaptively determine the weights assigned to different h -step returns according to the relative uncertainty of the learned model and that of the Q function? **(RQ2)** How does CBOP perform in the offline RL benchmark? **(RQ3)** Does CBOP with LCB provide conservative target Q estimation? **(RQ4)** How does having the full posterior over the target values compare against using the MAP estimation in performance? **(RQ5)** How much better is it to adaptively control the weights to h -step returns during training as opposed to using a fixed set of weights throughout training?

We evaluate these RQs on the standard *D4RL* offline RL benchmark (Fu et al., 2020). In particular, we use the D4RL MuJoCo Gym dataset that contains three environments: *halfcheetah*, *hopper*, and *walker2d*. For each environment, there are six different behavior policy configurations: *random* (r), *medium* (m), *medium-replay* (mr), *medium-expert* (me), *expert* (e), and *full-replay* (fr). We release our code at <https://github.com/jihwan-jeong/CBOP>.

4.1 CBOP CAN AUTOMATICALLY ADJUST RELIANCE ON THE LEARNED MODEL

To investigate RQ1, we use the notion of the *expected rollout horizon*, which we define as $\mathbb{E}[h] = \sum_{h=0}^H w_h \cdot h$. Here, w_h is the weight given to the mean of \hat{R}_h as defined in (7), which sums to 1. A larger $\mathbb{E}[h]$ indicates that more weights are assigned to longer-horizon model-based rollouts.

Figure 1 already shows that $\mathbb{E}[h]$ decreases as the Q function becomes better over time. On the other hand, Figure 3 shows how the quality of the learned model affects $\mathbb{E}[h]$. Specifically, we trained the dynamics model on *halfcheetah-m* for different numbers of epochs (10, ..., 100); then, we trained the policy with CBOP for 150 epochs.

4.2 PERFORMANCE COMPARISON

To investigate RQ2, we select baselines covering both model-based and model-free approaches: (*model-free*) **CQL** (Kumar et al., 2020), **IQL** (Kostrikov et al., 2022), **TD3+BC** (Fujimoto & Gu, 2021), **EDAC** (An et al., 2021); (*model-based*) **MOPO** (Yu et al., 2020), **MOReL** (Kidambi et al., 2020), and **COMBO** (Yu et al., 2021). Details of experiments are provided in Appendix C.1.

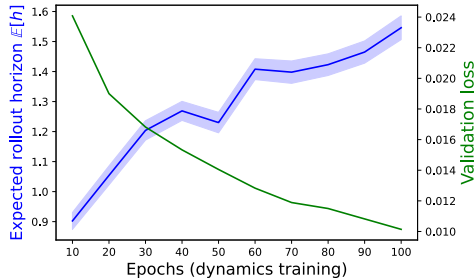


Figure 3: $\mathbb{E}[h]$ during CBOP training with the dynamics model trained for different numbers of epochs. CBOP can place larger weights to longer-horizon rollouts as the dynamics model becomes more accurate.

Table 1 shows the experimental results. Comparing across all baselines, CBOP presents new state-of-the-art performance in **11 tasks out of 18** while performing similar in the remaining configurations. Notably, CBOP outperforms prior works in *medium*, *medium-replay*, and *full-replay* configurations with large margins. We maintain that these are the datasets of greater interest than, e.g., *random* or *expert* datasets because the learned policy needs to be much different than the behavior policy in order to perform well. Furthermore, the improvement compared to previous model-based arts is substantial: CBOP outperforms MOPO, MOREL, and COMBO by 116.4%, 23.2% and 23.7% (respectively) on average across four behavior policy configurations.

4.3 CBOP LEARNS CONSERVATIVE VALUES

To answer RQ3, we have selected 3 configurations (*m*, *me*, and *mr*) from the *hopper* environment and evaluated the value function at the states randomly sampled from the datasets, i.e., $\mathbb{E}_{\mathbf{s} \sim \mathcal{D}}[\hat{V}^\pi(\mathbf{s})]$ (nb. a similar analysis is given in CQL). Then, we compared these estimates with the Monte Carlo estimations from the true environment by rolling out the learned policy until termination.

Table 2 reports how large are the value predictions compared to the true returns. Notice that not only the mean predictions are negative but also the maximum values are, which affirms that CBOP indeed has learned conservative value functions. Despite the predictions by CBOP being smaller than those of CQL in *hopper-mr* and *me*, we can see that CBOP significantly outperforms CQL in these settings. See Appendix D.1 for more details.

Table 2: Difference between the values predicted by the learned Q functions and the true discounted returns from the environment.

Task name	CQL		CBOP	
	Mean	Max	Mean	Max
hopper-m	-61.84	-3.20	-55.83	-16.21
hopper-mr	-142.89	-28.73	-172.45	-39.45
hopper-me	-79.67	-5.16	-114.39	-11.24

4.4 ABLATION STUDIES

LCB vs. MAP in the offline setting To answer RQ4, we compare CBOP with STEVE (Buckman et al., 2018) which is equivalent to using the MAP estimation for target Q predictions. Figure 1 (left) shows the case where the value function learned by STEVE blows up (orange). Further, we include the performance of STEVE in all configurations in Appendix D.3. To summarize the results, STEVE fails to learn useful policies for 11 out of 18 tasks. Especially, except for the *fr* datasets, using the MAP estimation has led to considerable drops in the performances in the *hopper* and *walker2d* environments, which reaffirms that it is critical to have the full posterior distribution over the target values such that we can make conservative target predictions.

Adaptive weighting For RQ5, we also considered an alternative way of combining $\hat{R}_h \forall h$ by explicitly assigning a fixed set of weights: uniform or geometric. We call the latter λ -weighting, in reference to the idea of TD(λ) (Sutton, 1988). We evaluated the performance of the fixed weighting scheme with various $\lambda \in (0, 1)$ values, and report the full results in Appendix D.3. In summary, there are some λ values that work well in a specific task. However, it is hard to pick a single λ that works across all environments, and thus λ should be tuned as a hyperparameter. In contrast, CBOP can avoid this problem by automatically adapting the rollout horizon.

Benefits of full posterior estimation To ablate the benefits of using the full posterior distribution in conservative policy optimization, we have compared CBOP to a quantile-based approach that calculates the conservative estimate through the α -quantile of the sampled returns $\hat{y}(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ (3) from the ensemble. The experimental details and results are reported in Appendix D.3. In summary, we have found that CBOP consistently outperformed this baseline on all tasks considered, and CBOP was more stable during training, showing the effectiveness of the Bayesian formulation.

5 RELATED WORK

In the pure offline RL setting, it is known that the direct application of off-policy algorithms fails due to value overestimation and the resulting policy distribution shift (Kumar et al., 2019; 2020; Fujimoto & Gu, 2021; Yu et al., 2021). Hence, it is critical to strike the balance between *conservatism* and *generalization* such that we mitigate the extent of policy distribution shift while ensuring

that the learned policy π_θ performs better than behavior policy π_β . Below, we discuss how existing model-free and model-based methods address these problems in practice.

Model-free offline RL *Policy constraint* methods directly constrain the deviation of the learned policy from the behavior policy. For example, BRAC (Wu et al., 2019) and BEAR (Kumar et al., 2019) regularize the policy by minimizing some divergence measure between these policies (e.g., MMD or KL divergence). Alternatively, BCQ (Fujimoto et al., 2019) learns a generative model of the behavior policy and uses it to sample perturbed actions during policy optimization. On the other hand, *value regularization* methods such as CQL (Kumar et al., 2020) add regularization terms to the value loss in order to implicitly regulate the distribution shift (Kostrikov et al., 2021; Wang et al., 2020). Recently, some simple yet effective methods have been proposed. For example, TD3+BC (Fujimoto & Gu, 2021) adds a behavioral cloning regularization term to the policy objective of an off-policy algorithm (TD3) (Fujimoto et al., 2018) and achieves SOTA performances across a variety of tasks. Also, by extending Clipped Double Q-learning (Fujimoto et al., 2018) to an ensemble of N Q functions, EDAC (An et al., 2021) achieves good benchmark performances.

Model-based offline RL Arguably, the learning paradigm of offline RL strongly advocates the use of a dynamics model, trained in a supervised way with a fixed offline dataset. Although a learned model can help generalize to unseen states or new tasks, model bias poses a significant challenge. Hence, it is critical to know when to trust the model and when not to. MOPO (Yu et al., 2020) and MOREL (Kidambi et al., 2020) address this issue by constructing and learning from a pessimistic MDP whose reward is penalized by the uncertainty of the state prediction. On the other hand, COMBO (Yu et al., 2021) extends CQL within the model-based regime by regularizing the value function on OOD samples generated via model rollouts. Rigter et al. (2022) also takes an adversarial approach by optimizing the policy with respect to a worst-case dynamics model. In contrast to these, CBOP estimates a full Bayesian posterior over values by using ensembles of models and value functions during policy evaluation of an actor-critic algorithm. In principle, having the full distribution that CBOP provides could also facilitate the use of other risk-informed statistics and epistemic risk measures to address value overestimation (see, e.g., Eriksson & Dimitrakakis (2020)).

Model-based value expansion Unlike Dyna-style methods that augment the dataset with model-generated rollouts (Sutton, 1990; Janner et al., 2019), MVE (Feinberg et al., 2018) uses them for better estimating TD targets during policy evaluation. While equally weighted h -step model returns were used in MVE, STEVE (Buckman et al., 2018) introduced an adaptive weighting scheme from the optimization perspective by approximately minimizing the variance of the MSBE loss, while ignoring the bias. Interestingly, the Bayesian posterior mean (i.e., the MAP estimator) we derive in (7) matches the weighting scheme proposed in STEVE. However as we show in Figure 1 and 10, using the MAP estimator as value prediction in the offline setting often results in largely overestimated Q values, which immensely hampers policy learning. See Section 3.1 for the related discussion.

6 CONCLUSION

In this paper, we present CBOP: conservative Bayesian model-based value expansion (MVE) for offline policy optimization. CBOP is a model-based offline RL algorithm that trades off model-free and model-based value estimates according to their respective epistemic uncertainty during policy evaluation while facilitating conservatism by taking a lower bound on the Bayesian posterior value estimate. Viewing each h -step MVE target as a conditionally independent noisy observation of the *true* target value under the learned MDP, we derive the Bayesian posterior distribution over the target value. For a practical implementation of CBOP, we use the ensemble of dynamics and that of Q function to sample MVE targets to estimate the Gaussian parameters, which in turn are used to compute the posterior distribution. Through empirical and analytical analysis, we find that the MAP estimator of the posterior distribution could easily lead to value overestimation when the learned MDP is not accurate under the current policy. In contrast, CBOP constructs the LCB from the Bayesian posterior as a conservative estimation of the target value to successfully mitigate the issue while achieving state-of-the-art performance on several benchmark datasets.

REFERENCES

- Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.
- 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>.
- James O Berger. *Statistical decision theory and Bayesian analysis; 2nd ed.* Springer series in statistics. Springer, New York, 1985. doi: 10.1007/978-1-4757-4286-2. URL <https://cds.cern.ch/record/1327974>.
- 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>.
- Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
- Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS’18*, pp. 8234–8244, 2018.
- Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.
- R. Christensen, W. Johnson, A. Branscum, and T.E. Hanson. *Bayesian Ideas and Data Analysis: An Introduction for Scientists and Statisticians*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2011. ISBN 9781439803554. URL <https://books.google.ca/books?id=qPERhCbePNcC>.
- 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>.
- Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian q-learning. *Aai/iaai*, 1998:761–768, 1998.
- Bradley Efron. *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.
- Hannes Eriksson and Christos Dimitrakakis. Epistemic risk-sensitive reinforcement learning. In *ESANN*, 2020.
- Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning, 2018.
- 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, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591. PMLR, 2018. URL <http://proceedings.mlr.press/v80/fujimoto18a.html>.
- 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>.
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018. URL <https://proceedings.mlr.press/v80/haarnoja18b.html>.
- 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.
- 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>.
- 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*, 2020.
- Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. 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. 5774–5783. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/kostrikov21a.html>.
- 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.
- Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pp. 3703–3712. PMLR, 2019.
- Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharmashan Kumaran,

- Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL <http://dx.doi.org/10.1038/nature14236>.
- Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*, 2020.
- Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 11702–11716. Curran Associates, Inc., 2021. URL <https://proceedings.neurips.cc/paper/2021/file/60ce36723c17bbac504f2ef4c8a46995-Paper.pdf>.
- Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *arXiv preprint arXiv:2204.12581*, 2022.
- C. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer Texts in Statistics. Springer New York, 2007. ISBN 9780387715988. URL <https://books.google.ca/books?id=6oQ4s8Pq9pYC>.
- Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, August 1988. URL <http://www.cs.ualberta.ca/~sutton/papers/sutton-88.pdf>.
- Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Bruce Porter and Raymond Mooney (eds.), *Machine Learning Proceedings 1990*, pp. 216–224. Morgan Kaufmann, San Francisco (CA), 1990.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.
- Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34:29420–29432, 2021.
- Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7768–7778, 2020. URL <https://proceedings.neurips.cc/paper/2020/file/588cb956d6bbe67078f29f8de420a13d-Paper.pdf>.
- Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, 2010. ISBN 1441923225.
- 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.

A ASSUMPTIONS

In this part, we discuss and analyze the core assumptions that we have made in the derivation and implementation of CBOP. First, recall that we view different h -step MVE returns \hat{R}_h for all $h = 0, \dots, H$ as *conditionally independent* observations of the true underlying parameter \hat{Q}^π . Second, we have modeled the likelihood of the observations with the Gaussian distribution with mean μ_h and standard deviation σ_h , which we estimate via sampling from the ensemble of dynamics and that of Q function. Third, we use the improper prior, which still provides us a proper posterior distribution that is also Gaussian. Below, we describe in more detail about each of these assumptions.

A.1 THE CONDITIONAL INDEPENDENCE ASSUMPTION

In order to meet the conditional independence assumption between \hat{R}_h , we need to estimate each \hat{R}_h with samples that are independently sampled. One way of achieving this is to generate samples per each h , resulting in an algorithm that requires $\mathcal{O}(NH^2)$ samples (and computation). However, we have found that *there is no specific benefit in this computational intensive sampling procedure in terms of the final performance*. Hence, our practical implementation only performs the forward sampling once, reducing the computational cost down to $\mathcal{O}(NH)$.

A.2 THE BAYESIAN POSTERIOR ESTIMATION

The improper prior assumption We have used the improper (or uninformative) prior in deriving CBOP in Section 3.1. Not to mention that the improper priors have been widely used in literature (Wasserman, 2010; Berger, 1985; Christensen et al., 2011), we further argue that it is quite natural (and sometimes necessary) not to assume any prior information if we are to apply our algorithm to general environments/tasks that have different dynamics. When some prior information is available, however, it is possible to incorporate it as long as we can use a conjugate prior that leads to a closed-form posterior update. It is critical to keep the posterior in closed-form since otherwise we have to resort to, e.g., posterior sampling, which will substantially (and unnecessarily) increase the computational footprint.

Empirical evidence supporting the Gaussian assumption over $\mathbb{P}(\hat{R}_h | \hat{Q}^\pi)$ First, note that the true return distribution should have a single peak in the locomotion environments we consider due to their deterministic nature, as long as the policy is deterministic. However, model-generated returns can have bimodality in their distributions since different models in the dynamics ensemble can lead to different trajectories, some of which can early terminate with low returns, while others continue to receive larger returns. Hence, it is interesting to examine whether it is reasonable to assume the Gaussian distribution over the h -step returns.

To answer this question, we have plotted the histograms of h -step returns for different h values in three tasks: *halfcheetah-mr*, *hopper-mr*, and *walker-mr*. Figure 4 (a)-(c) show that it is reasonable to assume \hat{R}_h are normally distributed. We have also observed that the empirical distribution of \hat{R}_h sampled from certain states can have bimodality (Figure 4d). Notice that the histograms are more spread out as h increases, which is due to compounded model errors. However, we note that the Gaussian distribution can still capture the support of the return distribution reasonably well.

The Gaussian likelihood assumption As discussed above and shown in Figure 4, the Gaussian assumption captures the actual return distributions reasonably well. Although it is possible to derive a closed-form posterior update in Student t distribution by making an additional assumption in the variance of \hat{R}_h likelihood (nb. we omit the actual derivation as it is not the contribution of this paper), we have observed that this does not lead to meaningful performance improvements compared to the much simpler Gaussian posterior that we derive in Section 3.1.

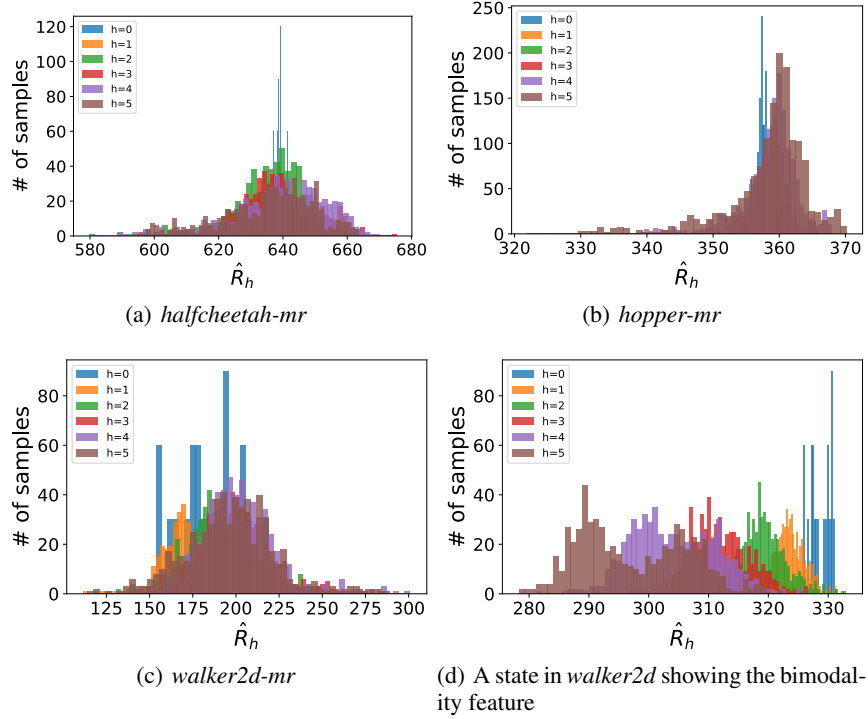


Figure 4: The histogram of $\hat{R}_h \forall h \in [0, 5]$ of a randomly selected state during training, evaluated across three locomotion environments with the *medium-replay-v2* configuration.

Algorithm 2 CBOP: Conservative Bayesian MVE for Offline Policy Optimization

- 1: **Input:** Data \mathcal{D} , discount factor γ , rollout horizon H , LCB coefficient ψ
 - 2: Initialize actor π_θ , Q ensemble Q_ϕ and target $Q_{\phi'}$, dynamics ensemble $\hat{f}_k = (\hat{T}_k, \hat{r}_k) \forall k$
 - 3: Pretrain \hat{f}_ξ on \mathcal{D} till convergence
 - 4: Pretrain π_θ and Q_ϕ on \mathcal{D} with BC and FQE respectively (Appendix B.3)
 - 5: **while** π_θ not converged **do**
 - 6: Sample a batch of transitions $B = \{\tau_i : \tau_i = (\mathbf{s}, \mathbf{a}, r, \mathbf{s}')\}_{i=1}^{|B|} \subset \mathcal{D}$
 - 7: **for** $\tau_i \in B$ **do** ▷ this step happens in parallel for all $\tau_i \in B$
 - 8: $\hat{\mathbf{s}}_0^k \leftarrow \mathbf{s}, \hat{\mathbf{s}}_1^k \leftarrow \mathbf{s}', \hat{\mathbf{a}}_0^k \leftarrow \mathbf{a}, \hat{r}_0^k \leftarrow r, \quad \forall k \in [1, K]$
 - 9: **for** $h = 0$ to H **do**
 - 10: **if** $h \geq 1$ **then**
 - 11: Sample an action $\hat{\mathbf{a}}_h^k \sim \pi_\theta(\hat{\mathbf{s}}_h^k) \forall k$
 - 12: Sample next state transition and reward $(\hat{\mathbf{s}}_{h+1}^k, \hat{r}_h^k) \leftarrow \hat{f}_k(\hat{\mathbf{s}}_h^k, \hat{\mathbf{a}}_h^k) \forall k$
 - 13: **end if**
 - 14: $\hat{R}_h^{k,m} \leftarrow \sum_{t=0}^h \gamma^t \hat{r}_t^k + \gamma^{h+1} \hat{Q}_{\phi'}^m(\hat{\mathbf{s}}_{h+1}^k, \hat{\mathbf{a}}_{h+1}^k) \quad \forall m$
 - 15: **end for**
 - 16: Compute μ_h and σ_h by (8) and (9), respectively
 - 17: Estimate μ, σ^2 of $\mathbb{P}(\hat{Q} | \hat{R}_0, \dots, \hat{R}_H) \sim \mathcal{N}(\mu, \sigma^2)$ by (7)
 - 18: Compute target Q value: $y_i(\mathbf{s}, \mathbf{a}, \mathbf{s}') \leftarrow \mu - \psi\sigma$
 - 19: **end for**
 - 20: Update π_θ and Q_ϕ following an off-policy actor-critic algorithm (e.g., SAC Haarnoja et al. (2018))
 - 21: Update the target network $Q_{\phi'}$
 - 22: **end while**
-

Algorithm 3 FQE: Fitted Q-Evaluation (Le et al., 2019)

```

1: Input: Dataset  $\mathcal{D} = \{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}_{i=1}^n$ , policy  $\pi$  to be evaluated
2: Initialize the parameters of  $Q_{\phi^{(0)}}$  randomly
3: for  $t = 1, \dots, T$  do
4:   Compute the targets  $y_i = r_i + \gamma Q_{\phi^{(t-1)}}(\mathbf{s}'_i, \pi(\mathbf{s}'_i)) \forall i$ 
5:   Build the training set  $\mathcal{D}^{(t)} = \{(\mathbf{s}_i, \mathbf{a}_i), y_i\}_{i=1}^n$ 
6:   Solve a supervised learning problem:
7:      $\phi^{(t)} = \arg \min_{\phi} \mathbb{E}_{\{(\mathbf{s}_i, \mathbf{a}_i), y_i\} \sim \mathcal{D}^{(t)}} \left[ (Q_{\phi}(\mathbf{s}_i, \mathbf{a}_i) - y_i)^2 \right]$ 
8: end for
9:  $\phi \leftarrow \phi^{(T)}$ 
10: return  $Q_{\phi}$ 

```

B ALGORITHM DETAILS

B.1 ALGORITHM SUMMARY

Algorithm 2 summarizes CBOP. In lines 20-21, we can use any off-policy actor-critic algorithm as the backbone of our approach, since the only part that changes is the computation of the target value $y(\mathbf{s}, \mathbf{a}, \mathbf{s}')$. In this work, we follow EDAC (An et al., 2021) — which builds on SAC (Haarnoja et al., 2018) — because it also employs Q ensembles. As discussed in Appendix B.3, a large discrepancy in the scale of the terminal $Q_{\phi'}$ predictions and that of the model-based rollout returns $\sum \gamma^t \hat{r}_t$ in the initial iterations greatly hampers policy learning. Hence, we pretrain the policy π_{θ} and Q_{ϕ} with with behavioral cloning (BC) and policy evaluation (PE) as elaborated in Appendix B.3.

B.2 DYNAMICS MODEL ARCHITECTURE

In this work, we approximate the true dynamics with a probabilistic ensemble model introduced by PETS (Chua et al., 2018). We follow the common configurations used in the literature, e.g., MBPO (Janner et al., 2019) and MOPO (Yu et al., 2020). Each model in the ensemble has 4 fully-connected layers with 200 neurons. Specifically, we train the ensemble of 30 models, from which we select 20 models (often called ‘elite’) with smaller validation errors. For next state predictions, we train the ensemble model to predict the *delta* states, or $\Delta = \mathbf{s}' - \mathbf{s}$ for $(\mathbf{s}, \mathbf{s}') \in \mathcal{D}$. We normalize the inputs and outputs of the model for training and evaluation.

The approach for training the dynamics ensemble closely follows previous work on Bayesian ensemble estimation (Chua et al., 2018; Janner et al., 2019). To reduce the effect of correlation, we follow the existing work by using independent initialization for each ensemble member and by training each of them using different mini-batches sampled from the dataset. Although in practice some correlation may be inevitable, there are several key advantages to estimating uncertainty in this way. Firstly, bootstrapped uncertainty estimates have been shown to have strong theoretical properties — see, e.g. Efron (1982) or Breiman (1996). Secondly, bootstrapping avoids the computational challenges associated with estimating the uncertainty of model predictions directly, and our experiments have shown that the uncertainty we obtained was indeed well-calibrated. For further details, please see the expected horizon analysis shown in Figure 3 and Section 4.1, which demonstrates the effectiveness of CBOP subject to different qualities of the learned dynamics ensemble.

B.3 PRETRAINING

In some environments, we notice that training Q_{ϕ} and π_{θ} from scratch could be challenging, and Figure 5 illustrates the reason. Remember that we pretrain the dynamics ensemble with the offline data \mathcal{D} before starting the policy optimization. This means that the reward predictions made by the learned model would have the proper scale. On the other hand, the $Q_{\phi'}$ ensemble is initialized with small random values. Hence, in the early iterations of policy learning, even though the Q_{ϕ} ensemble has not been trained yet, its predictions have a very small variance compared to the model-based rollout returns given by the learned dynamics ensemble (Figure 5(a)). This will then lead to all weights being concentrated on \hat{R}_0 , effectively MF; the MB rollouts would only slow down

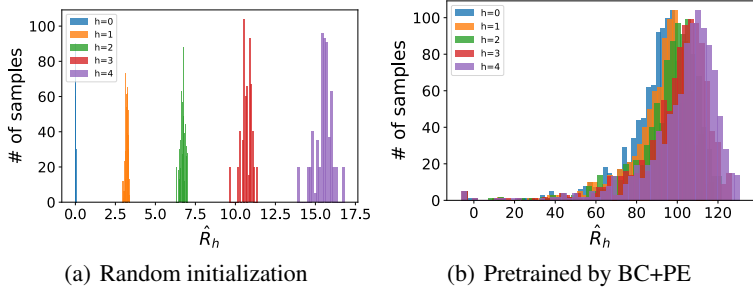


Figure 5: The histogram of $\hat{R}_h \forall h \in [0, 4]$ evaluated on *halfcheetah-medium-v2*

learning without contributing anything in this case. Besides, the variance of $Q_{\phi'}$ ensemble would be negligible, suggesting that taking the LCB would not introduce a sufficient level of conservatism into learning, which can hurt the performance.

Therefore in the experiments, we pretrain Q_{ϕ} and π_{θ} with the offline data. Specifically, we use behavior cloning (BC) for the policy network π_{θ} . In BC, we minimize the mean squared loss $\mathcal{L}_{\text{BC}}(\theta) = \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \mathcal{D}}[(\mathbf{a} - \pi_{\theta}(\mathbf{s}))^2]$. For the value network Q_{ϕ} , we perform policy evaluation (PE) using Fitted Q-Evaluation (FQE) (Le et al., 2019), which is schematically explained in the pseudocode in Algorithm 3. In line 4, when the policy to be evaluated is the behavior policy π_{β} , we can take the recorded next action \mathbf{a}_{i+1} from \mathcal{D} in place of $\pi(\mathbf{s}'_i)$.

More concretely, at each iteration t of FQE, a supervised learning dataset $\mathcal{D}^{(t)} = \{(\mathbf{s}_i, \mathbf{a}_i), y_i\}_{i=1}^n$ is constructed by estimating the target value y_i for each $(\mathbf{s}_i, \mathbf{a}_i) \sim \mathcal{D}$ with the current Q approximation $Q_{\phi^{(t-1)}}$ and the associated transition tuple $(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)$ via $y_i = r_i + \gamma Q_{\phi^{(t-1)}}(\mathbf{s}'_i, \pi(\mathbf{s}'_i))$. We then update the Q function parameters ϕ by minimizing the MSE loss. That is, $\phi^{(t)} \leftarrow \arg \min_{\phi} \frac{1}{n} \sum_{i=1}^n [Q_{\phi^{(t-1)}}(\mathbf{s}_i, \mathbf{a}_i) - y_i]^2$. FQE repeats the two steps (i.e., constructing the dataset and minimizing the MSE loss) to learn the Q_{ϕ} ensemble model.

C EXPERIMENT DETAILS

C.1 EXPERIMENTAL SETTINGS

D4RL MuJoCo Gym We use the v2 version for each dataset as provided by the D4RL library (Fu et al., 2020). Following Algorithm 2, we pretrain π_{θ} and Q_{ϕ} with BC and FQE, respectively. The resulting policy and the Q ensemble are trained for 1,000 more epochs using CBOP. Table 1 reports the mean and standard deviation obtained from 5 random seeds.

Comparison of target Q values of MAP and CBOP (Figure 1(a)) In Figure 1(a), we compare the MAP estimation with the LCB in the *hopper-random* dataset. We have plotted the mean and \pm one standard error over the course of training. The MAP estimation simply uses the mean μ in (7) as the target $y(\mathbf{s}, \mathbf{a}, \mathbf{s}')$, where as the LCB utilizes the variance of the posterior distribution to compute $y(\mathbf{s}, \mathbf{a}, \mathbf{s}') = \mu - \psi \cdot \sigma$. Note that we can also use other conservative estimate of the target using the posterior distribution; for example, we can use value-at-risk (VaR), conditional value-at-risk (CVaR) or other quantiles.

Expected rollout horizon of CBOP (Figure 1(b) and Figure 3) In Figure 1 and 3, we report the expected rollout horizon values. The expected rollout horizon can be computed per each sample in the batch during policy training, and we have reported the average value across all samples in a batch.

C.2 HYPERPARAMETERS

Table 3 summarizes the CBOP hyperparameters we use in the experiments presented in Section 4. The only hyperparameter that we have tuned is the LCB coefficient ψ through the grid search over

Table 3: The LCB coefficient ψ used in the D4RL MuJoCo Gym experiments.

Task Name	ψ		
	halfcheetah	hopper	walker2d
random	3.0	5.0	5.0
medium	0.5	3.0	3.0
medium-replay	0.5	2.0	2.0
medium-expert	3.0	3.0	3.0
expert	5.0	3.0	3.0
full-replay	2.0	3.0	2.0

the set $\{0.5, 2.0, 3.0, 5.0\}$. We have used $H = 10$, $K = 20$, $M = 20$, and $lr = 3 \times 10^{-4}$ for all experiments, except for the *hopper* environment where we used $M = 50$.² The LCB parameters reported in Table 3 are tuned based on the final online evaluation performance from corresponding environments.

Offline Hyperparameter Selection via FQE (Paine et al., 2020) When strictly adhering to the offline paradigm of policy learning, it is crucial to restrict access to online interactions at all stages of learning including the hyperparameter selection. However, many existing works still use the online evaluation for hyperparameter selection (An et al., 2021; Wang et al., 2021; Fujimoto & Gu, 2021; Chen et al., 2021) and we followed the same evaluation protocol for tuning the hyperparameters of our method. We believe there is a dire need for standardizing the evaluation protocol in the offline RL, but this work should be addressed by the offline RL research community as a whole, which is beyond the scope of our paper. One important way to reduce the amount of online interactions used for hyperparameter selection is to minimize the number of hyperparameters to tune. In this regard, CBOP is particularly advantageous since we need only to tune the LCB coefficient ψ .

To further validate the choice of ψ values in Table 3, we performed a post hoc analysis following the hyperparameter selection work proposed in Paine et al. (2020). To this end, we considered three data configurations (m , mr , fr) and two environments (*halfcheetah*, *walker2d*), and we retrieved the model checkpoints of the learned policy networks for all seeds. Then, we evaluated each policy π_θ with the following metric:

$$\mathbb{E}_{\mathbf{s}_0 \sim \mathcal{D}}[Q_\zeta(\mathbf{s}_0, \pi_\theta(\mathbf{s}_0))] \quad (10)$$

Here, \mathbf{s}_0 are the initial states stored in the offline dataset and Q_ζ is the value function associated with the policy π_θ , which is obtained by running FQE (Algorithm 3). This Q_ζ is different from the learned value function Q_ϕ , and Paine et al. (2020) noted that using Q_ζ is better than using Q_ϕ for the purpose of hyperparameter selection. The candidate ψ values are sorted based on the scores from (10), and we can use ψ with the highest score.

Table 4 compares the rankings of the four ψ values we considered in the experiments from FQE and the online evaluation. The rightmost column shows the Spearman’s rank correlation coefficient (ρ) which is the correlation coefficient between the two sets of rankings. Notably, the ψ values selected via FQE match the values we obtained from the online evaluation for 4 out of 6 tasks. In *halfcheetah-m*, $\psi = 0.5$ has the online performance of 74.3 (as reported in Table 1), while the performance from $\psi = 2$ is 72.4 which is only slightly worse. For *walker2d-fr*, $\psi = 2$ is at 107.8 (reported in Table 1) and $\psi = 3$ gives 89.3 when evaluated in the true environment. Even if $\psi = 3$ was chosen based on FQE, we can easily see that this is still a substantial improvement compared to the data-logging policy which has the average normalized score of 39.8.

²In the early stage of algorithm development, we selected the *medium* configuration from the three environments in the D4RL benchmark and used $M = 20$ for all experiments when testing the performance of CBOP. It turned out that CBOP works well in the HalfCheetah and Walker2d environments without tuning, but we found that we needed to have a larger value ensemble to get reasonable performance in the Hopper environment. We chose $M = 50$ since it worked well and this choice is also supported by previous work (An et al., 2021). Accordingly during hyperparameter tuning, we used $M = 50$ for Hopper and $M = 20$ for the other two environments.

Table 4: Comparing the rankings of the LCB coefficient ψ based on the online evaluation and FQE (Paine et al., 2020)

Task Name	Ranking	ψ				Rank correlation (ρ)
		0.5	2.0	3.0	5.0	
halfcheetah-m	FQE	2	1	3	4	0.8
	Online	1	2	3	4	
halfcheetah-mr	FQE	1	2	4	3	0.8
	Online	1	2	3	4	
halfcheetah-fr	FQE	3	1	2	4	0.8
	Online	2	1	3	4	
walker2d-m	FQE	4	2	1	3	1.0
	Online	4	2	1	3	
walker2d-mr	FQE	4	1	2	3	1.0
	Online	4	1	2	3	
walker2d-fr	FQE	3	2	1	4	0.8
	Online	3	1	2	4	

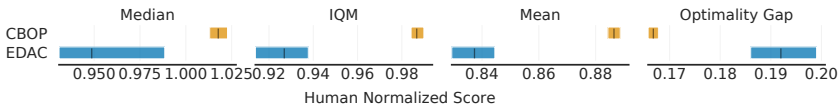


Figure 6: RLiabale results across all 18 locomotion tasks. Shaded regions show 95% CIs. We refer readers to (Agarwal et al., 2021) for detailed explanation of the metrics considered.

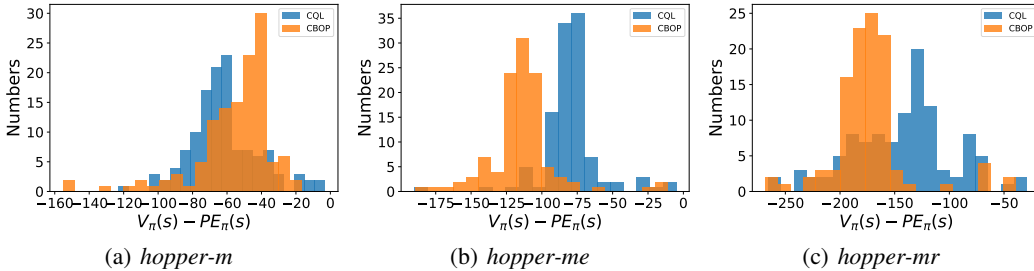
Overall, the Spearman’s rank correlation values are always greater than or equal to 0.8, suggesting that the rankings from FQE align very well with those from the online evaluation. This suggests that (1) CBOP can be reliably tuned solely with an offline dataset via FQE and that (2), with the benefit of hindsight, our selection of ψ values in Table 3 is a valid one.

Other considerations CBOP trades off the uncertainty of the learned dynamics model with that of the learned Q ensemble. In practice, we use the ensemble models to implicitly capture the respective epistemic uncertainty. Hence, it is critical that the models we use indeed exhibit well-calibrated uncertainty in their predictions. In this regard, we found that it is useful to incorporate the gradient diversification loss for the Q ensemble as introduced in An et al. (2021), which helps prevent the uncertainty in predictions from collapsing. Instead of tuning the hyperparameter η that controls the level of gradient diversification loss, we use a fixed number $\eta = 1$ across all experiments.

Please note that the use of the ensemble diversification trick is orthogonal to our contributions in this work. Furthermore, we provide a reliable performance comparison between CBOP and EDAC to validate that CBOP outperforms EDAC. To this end, we use RLiabale (Agarwal et al., 2021) which provides various metrics other than the simple average to more reliably determine the relative performances of compared methods. Specifically, we have reproduced EDAC and compared its performance against CBOP using the Median, IQM (interquartile mean), Mean, and Optimality Gap (Figure 6). In all metrics considered, CBOP exhibits substantially better performance without overlapping 95% confidence intervals (CI). In fact, another important performance metric, called the probability of improvement, of CBOP against EDAC is 88.27%, which strongly indicates the superiority of CBOP.

Table 5: A full comparison across three environments showing the difference between the values predicted by the learned Q functions and the true discounted returns from the environment.

Task name	CQL		CBOP	
	Mean	Max	Mean	Max
hopper-m	-61.84	-3.20	-55.83	-16.21
hopper-mr	-142.89	-28.73	-172.45	-39.45
hopper-me	-79.67	-5.16	-114.39	-11.24
halfcheetah-m	-222.43	-180.97	-106.24	-66.97
halfcheetah-mr	-363.00	-198.42	-84.42	-8.48
halfcheetah-me	-310.95	-23.74	-210.51	-54.58
walker2d-m	-167.36	-8.88	-84.70	-15.00
walker2d-mr	-285.02	-25.44	-80.31	-14.06
walker2d-me	-156.71	-64.64	-75.89	-42.30

Figure 7: The distribution of difference between policy values predicted by algorithms and Monte Carlo policy evaluation results in the true environment. Here, $s \sim \mathcal{D}$, $a = \pi(s)$.

D ADDITIONAL EXPERIMENTS

D.1 CONSERVATISM ANALYSIS

In Section 4.3, we have empirically verified that CBOP indeed learns a conservative value function. Specifically, given the offline dataset \mathcal{D} , we compute the following value difference:

$$\mathbb{E}_{s \sim \mathcal{D}} [\hat{V}^\pi(s) - \mathbb{E}[V^\pi(s)]] \quad (11)$$

where we compute the true value $\mathbb{E}[V^\pi]$ via the Monte Carlo estimation in the true environment. We have provided the comparison of CQL and CBOP evaluated in the *hopper* environment in Table 2, and Figure 7 shows the full histograms of (11) for in this environment. Furthermore, Table 5 includes the results from all three MuJoCo locomotion environments. We can clearly see that CBOP has learned a conservative value function in these tasks.

D.2 DECOMPOSITION OF h -STEP RETURN VARIANCE

In Section 3.2, we have shown that the variance of h -step returns can be decomposed into A and B terms according to the law of total variance, which we restate here for ease of exposition:

$$\sigma_h^2 = \text{Var}_{\pi_\theta} [\hat{R}_h | \tau] = \underbrace{\mathbb{E}_{\hat{f}_k} [\text{Var}_{\pi_\theta} [\hat{R}_h | \tau, \hat{f}_k]]}_A + \underbrace{\text{Var}_{\hat{f}_k} [\mathbb{E}_{\pi_\theta} [\hat{R}_h | \tau, \hat{f}_k]]}_B. \quad (12)$$

Here, A reflects the epistemic uncertainty from the $Q_{\phi'}$ ensemble, while B accounts for the uncertainty derived from the learned dynamics ensemble. The beauty of CBOP is that it can capture both uncertainties by sampling through the dynamics and value ensembles and subsequently compute the value target in a conservative way through the Bayesian posterior formulation. A natural question may be whether A would vanish and become unnecessary when the policy and value function have converged?

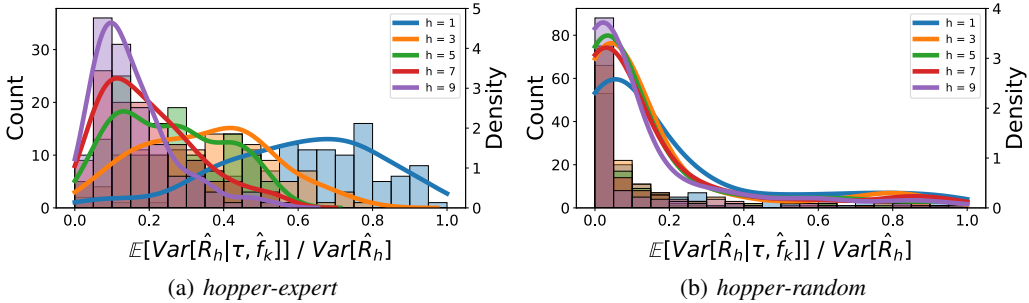


Figure 8: The distribution of the ratio, $\frac{A}{A+B} = \mathbb{E}_{\hat{f}_k} \left[\text{Var}_{\pi_\theta} \left[\hat{R}_h | \tau, \hat{f}_k \right] \right] / \sigma_h^2$, from (12) when π_θ and Q_ϕ are trained with the *hopper-r* dataset. (a) evaluates π_θ and Q_ϕ with (s, \mathbf{a}) sampled from the *hopper-e* dataset; (b) is the result from evaluating with the *hopper-r* dataset. The histogram shows the empirical distribution based on a batch of samples. Probability density functions are the kernel density estimation results corresponding to each histogram with the same color.

To answer this question, recall that in the offline setting, the logged data will typically only cover a subset of the state-action space. Hence, when we use the learned dynamics ensemble to forward sample rollout trajectories during the target value estimation procedure in CBOP, some of the trajectories will inevitably visit unseen states. Even after the policy and the value have sufficiently converged, the rolled out trajectories will still visit OOD states (in fact, as the learned policy has shifted from the behavior policy, it is more likely that it visits more OOD states during the rollouts). Thus, we can say that the A term will not (and should not) vanish at these OOD state/actions such that CBOP can account for the epistemic uncertainty in the value and act conservatively against it.

We have further empirically verified the relative contributions of the A and B terms, respectively, after the policy/value have converged. Firstly, we considered the case when a policy and value ensemble learned with the *hopper-r* dataset is used for sampling the h -step returns \hat{R}_h starting from a set of initial states randomly selected from the *hopper-e* dataset. Roughly speaking, this setup would ensure that we evaluate the total variance at states and actions that the policy/value have not been trained with. Thus, we expect a relatively large amount of epistemic uncertainty still left in the A term. On the other hand, we also evaluated the learned policy/value from the states sampled from the same dataset they were trained with (i.e., *hopper-r*). In this case, we would like to see relatively little epistemic uncertainty left in A since the policy and value were repeatedly trained with those states and actions.

To this end, we retrieved the policy and value ensemble checkpoints trained with the *hopper-r* dataset. Then, we calculated the proportion of A with respect to the total variance, $\frac{A}{A+B}$, per each h -step return per each (s, \mathbf{a}) sample, which was sampled randomly from either the *hopper-e* or *hopper-r* dataset.

As expected, Figure 8(a) shows that there is a significant amount of variance left in the A term even though we have evaluated the converged policy and value function since they were evaluated with OOD states/actions. Especially when h is small, the A term contributes more to the total variance than when h is large. As h increases, we can see that the weight shifts gradually towards B , which indicates there is more uncertainty in the model-based estimates of the returns for longer horizon rollouts. In contrast, Figure 8(b) shows much less contributions from A compared to B even for smaller h .

We studied the trends from other tasks as well. Specifically, we picked the *m* and *fr* D4RL configurations from the three MuJoCo environments and performed the same evaluations as discussed above. This time, the policy/value function trained with a certain dataset were evaluated with the same dataset to see if there is still a meaningful epistemic uncertainty left in A term after convergence. Figure 9 clearly shows that, in most of the cases, the contribution from A to the total variance is not negligible, despite the policy/value being already converged. Similar to the *hopper-r* case, A generally contributes more than B does for small h values. As discussed, this is an intuitive result

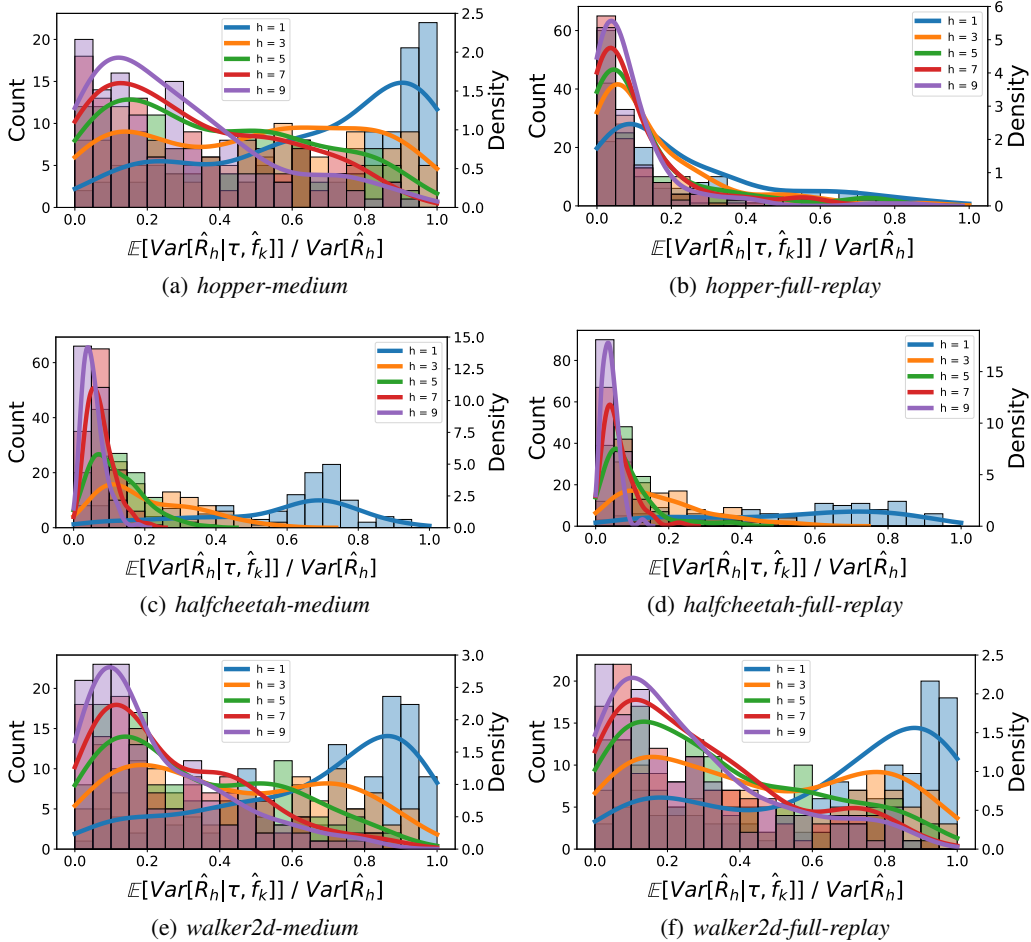


Figure 9: The distribution of the ratio, $\frac{A}{A+B} = \mathbb{E}_{\hat{f}_k} \left[\text{Var}_{\pi_\theta} \left[\hat{R}_h | \tau, \hat{f}_k \right] \right] / \sigma_h^2$, from (12). The histogram shows the empirical distribution based on a batch of samples. Probability density functions are the kernel density estimation results corresponding to each histogram with the same color.

since the learned model would typically be quite accurate for single-step predictions, hence smaller B compared to A .

It is also notable that in the *fr* tasks of the *hopper* and *halfcheetah* environments shown in Figure 9(d) and 9(b), much more contribution is coming from B even for small h (however, A still has noticeable contribution). Note that (1) the *fr* (full-replay) dataset was curated such that it covers all transition samples encountered by various policies, starting from a random policy all the way to an expert policy. Now, also note that (2) since we pre-train the dynamics model and fix it during policy training, the epistemic uncertainty baked in the dynamics ensemble is kept fixed, whereas the uncertainty in the value ensemble can diminish as training continues. These two factors combined can explain why we would see more contributions in the total variance from B rather than A in the *fr* datasets.

D.3 ABLATIONS

In this part, we provide additional ablations that complement the results presented in the main text.

The effectiveness of conservatism via LCB compared to MAP STEVE (Buckman et al., 2018) introduced an adaptive weighting scheme for MVE, which corresponds to the MAP estimation of

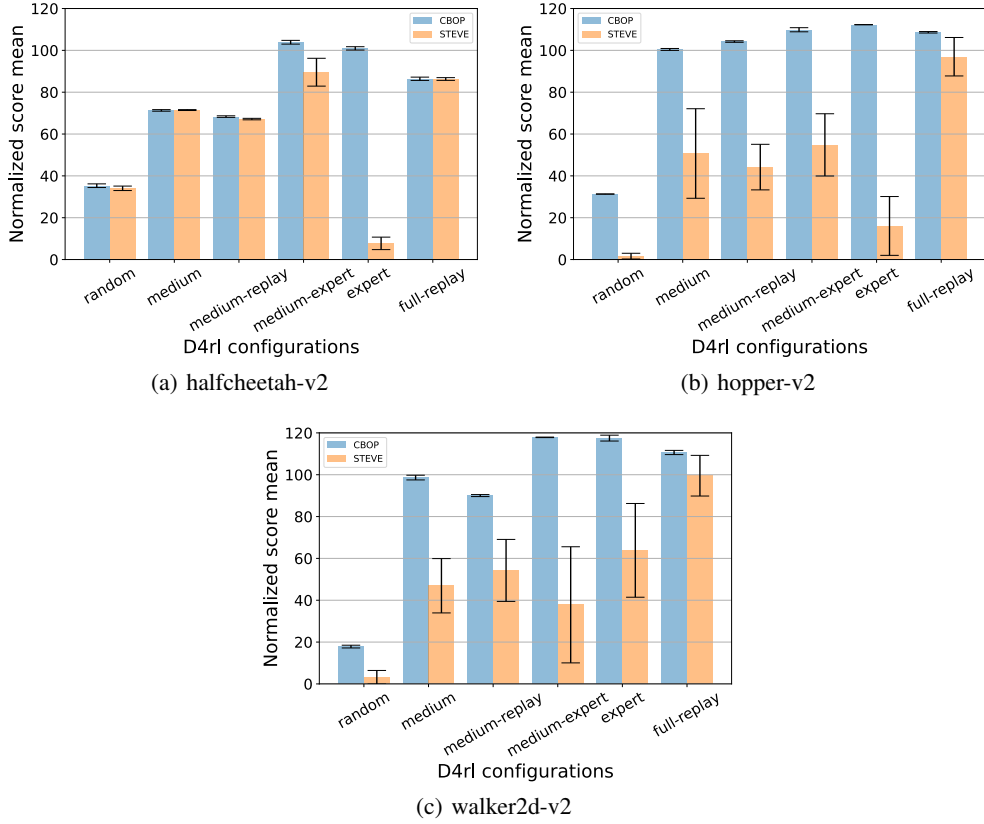


Figure 10: Comparison of the MAP estimation and the LCB estimation in the D4RL MuJoCo benchmark tasks. Experiments are run with 3 random seeds.

the posterior we get in (7). In this part, we provide the complete ablations comparing CBOP and STEVE in all tasks.

In Figure 10(a), we see that STEVE performs comparably to CBOP in 4 of the 6 tasks, where small ψ have been used in CBOP (Table 3). However, for the *medium-expert* and *expert* tasks — where we have used $\psi = 3$ and 5, respectively — CBOP outperforms STEVE.

The differences in the performances are even more striking in the other two environments. Figure 10(b) and 10(c) show that CBOP significantly outperforms STEVE, suggesting that conservatism plays a crucial role. It is worth reasserting that the original adaptive weighting scheme derived in STEVE does not lend itself to a conservative value estimation as we can do with CBOP.

The effectiveness of the Bayesian weighting scheme In Section 4, we have presented a part of the ablations comparing the adaptive weighting scheme of CBOP with the fixed weighting scheme, i.e., uniform and λ weighting. The weights in the uniform weighting correspond to $w_h = \frac{1}{H+1}$, while those in the λ -weighting are $w_h = \frac{1-\lambda}{1-\lambda^{H+1}} \lambda^h$. In the latter, the larger the λ parameter, the more weight is allocated to longer-horizon model-based rollouts; $\lambda = 1$ corresponds to solely using the H -step MVE target, whereas $\lambda = 0$ bootstraps immediately at s' as in the model-free case.

In order to better isolate the impact of the different weighting schemes, we have used the conservative value estimation for these two fixed weighting schemes as well. More concretely, we have sampled $M \times K \hat{R}_h$ samples for $h = 0, \dots, H$ and computed the weighted sums ($\sum_{h=0}^H w_h \hat{R}_h$) to get MK samples of target values. With these samples, we have computed the empirical mean and the variance, from which we have taken the LCB $\mu - \psi \cdot \sigma$ as the target values.

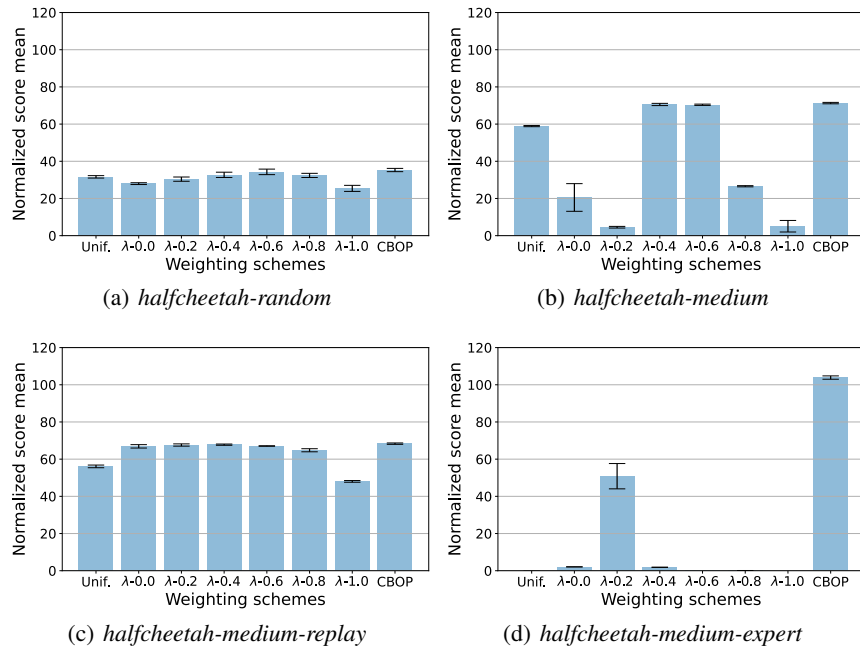


Figure 11: Comparing the fixed weighting schemes and CBOP on the *halfcheetah* environment. Experiments are run with 3 seeds.

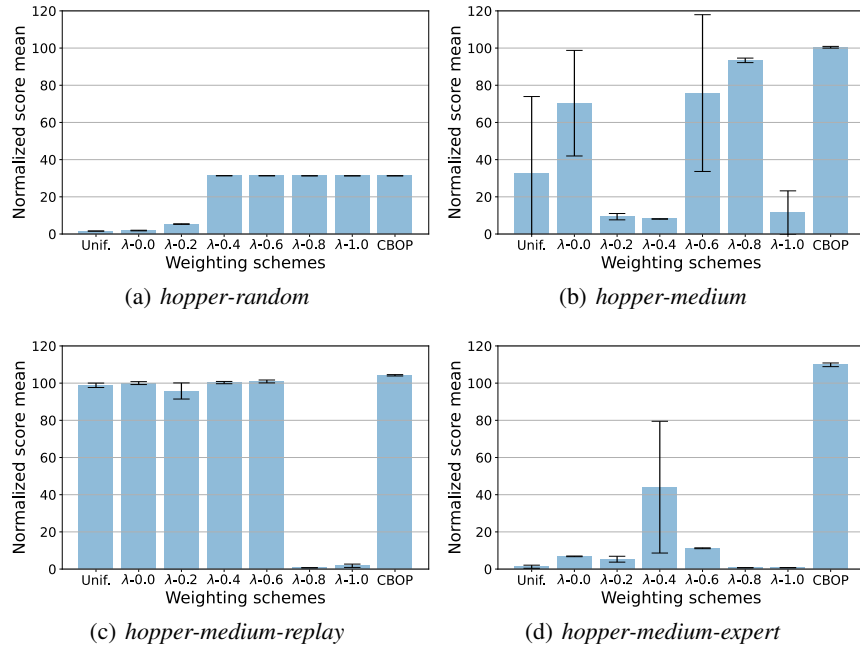


Figure 12: Comparing the fixed weighting schemes and CBOP on the *hopper* environment. Experiments are run with 3 seeds.

Figure 11 - 13 show the results on the *halfcheetah*, *hopper*, and *walker2d* environments, respectively. We have found that the fixed weighting does not work in the *walker2d* tasks, regardless of the λ values. Also, CBOP has significantly outperformed the fixed weighting schemes in narrow datasets (i.e., *medium-expert*) across all environments.

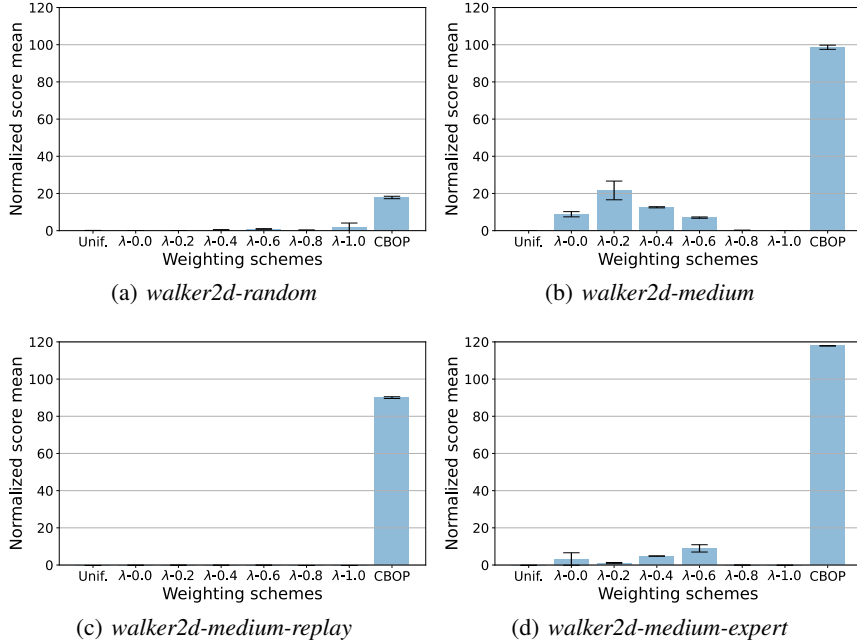


Figure 13: Comparing the fixed weighting schemes and CBOP on the *walker2d* environment. Experiments are run with 3 seeds.

In some tasks — such as *medium* and *medium-replay* tasks in *hopper* and *halfcheetah* environments, there are some λ values that can show similar performances as CBOP. However, large fluctuations across different λ values as exhibited in *halfcheetah-medium* and *hopper-medium* suggest that finding λ that works robustly across all tasks may be impossible. On the contrary, the adaptive Bayesian weighting scheme of CBOP can work reliably across all tasks considered.

Additional Baseline: quantile-based conservative MVE We have seen that CBOP is able to adaptively regulate the reliance on model-based and model-free value estimates while acting conservatively with respect to both. The uncertainties in the learned dynamics model and the value function are captured through the sampling procedure we detailed in Section 3.2. The ablation studies presented in Section 4.4 show the strong merits that the Bayesian interpretation provides us through the adaptive control of the roll-out horizon and the conservative value estimates from the Bayesian posterior. Here, we further strengthen the case and ablate the benefits of being Bayesian by comparing CBOP against another baseline that we dub *Distributional MVE (DiMVE)*.

Instead of forming a Bayesian posterior over \hat{Q}^π , DiMVE simply aggregates all *MKH* return samples that we collect from a single pass of forward sampling. Then, it performs a quantile-based conservative value estimation. Formally, let

$$\hat{R}_h^{m,k}(\mathbf{s}, \mathbf{a}, \mathbf{s}') := \sum_{t=0}^h \gamma^t \hat{r}_t(\hat{\mathbf{s}}_t^{(k)}, \hat{\mathbf{a}}_t^{(k)}) + \gamma^{h+1} Q_{\phi'}^m(\hat{\mathbf{s}}_{h+1}^{(k)}, \hat{\mathbf{a}}_{h+1}^{(k)})$$

be the roll-out collected using the k th particle from the model ensemble and the m th particle from the value ensemble. The goal of DiMVE is to empirically estimate the left α -quantile of the posterior return distribution induced by the model ensemble for $\alpha \in (0, 1]$:

$$\hat{y}_{DiMVE}(\alpha) = \inf \{y \in \mathbb{R} : \mathbb{P}(\hat{y}(\mathbf{s}, \mathbf{a}, \mathbf{s}') \leq y) > \alpha\}. \tag{13}$$

Let $\hat{R}_1 \leq \hat{R}_2 \leq \dots \hat{R}_{M \times K \times H}$ be the ordering of the $\hat{R}_h^{m,k}$, in the case where the samples are unique the DiMVE estimate can be written simply as

$$\hat{y}_{DiMVE}(\alpha) \approx \hat{R}_{\lfloor \alpha \times M \times K \times H \rfloor}.$$

Table 6: Comparison of CBOP and DiMVE

Task name	CBOP	DiMVE (best α)
halfcheetah-m	74.3 \pm 0.2	70.9 \pm 0.6 (0.3085)
halfcheetah-mr	66.4 \pm 0.3	65.0 \pm 0.3 (0.3085)
halfcheetah-me	100.4 \pm 0.9	84.4 \pm 6.6 (0.3085)
halfcheetah-fr	85.5 \pm 0.3	83.4 \pm 0.8 (0.4)
walker2d-m	95.5 \pm 0.4	65.1 \pm 3.4 (0.3085)
walker2d-mr	92.7 \pm 0.9	88.5 \pm 0.2 (0.3085)
walker2d-me	117.2 \pm 0.5	113.0 \pm 9.8 (0.3085)
walker2d-fr	107.8 \pm 0.2	104.6 \pm 1.0 (0.3085)

Table 6 compares the performance of CBOP and DiMVE for the *walker2d* and *halfcheetah* environments with the *m*, *mr*, *me*, and *fr* dataset configurations, where α was tuned among $\{0.4, 0.3085, 0.0228, 0.0013, 2.87 \times 10^{-7}\}$. Here, the last four α values correspond to $\psi = 0.5, 2.0, 3.0, 5.0$, respectively, if assuming the $\hat{R}_h^{m,k}$ samples are normally distributed. We noted that α value smaller than 0.3085 resulted in value divergence towards negative infinity, and so we report the performance with the best α values in Table 6. Clearly, CBOP outperforms the baseline in all tasks, showing the effectiveness of our Bayesian formulation. Furthermore, we found DiMVE to be more unstable during training and it consistently showed larger variances in the performance.