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 $\hat{Q}^\pi$. 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 $\mathcal{O}(NH)$.

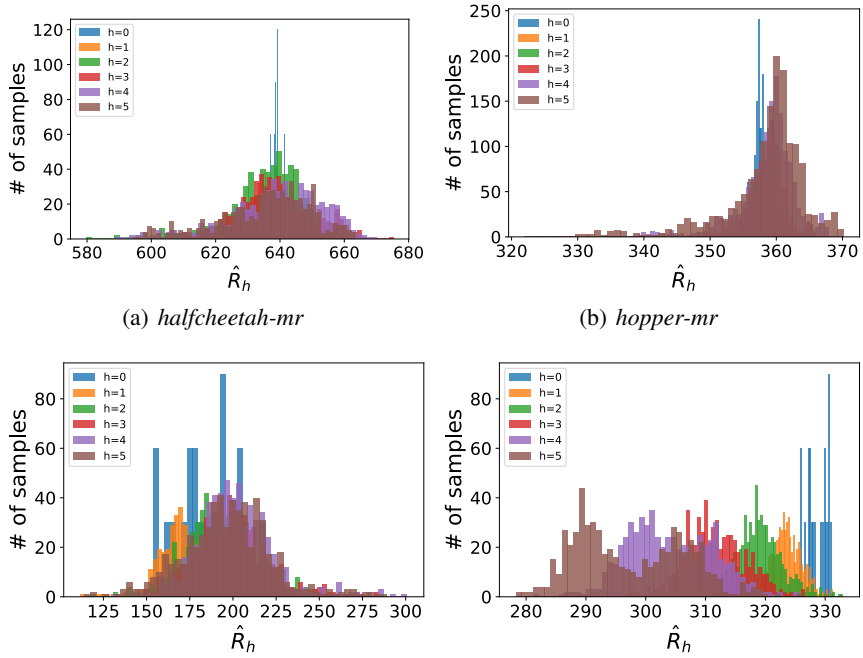
A.2 THE BAYESIAN POSTERIOR ESTIMATION

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 $\mathbb{P}\left(\hat{R}_h \mid \hat{Q}^\pi\right)$ 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.



(a) *halfcheetah-mr* (b) *hopper-mr*

(c) *walker2d-mr* (d) A state in *walker2d* showing the bimodality feature

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

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

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

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

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.

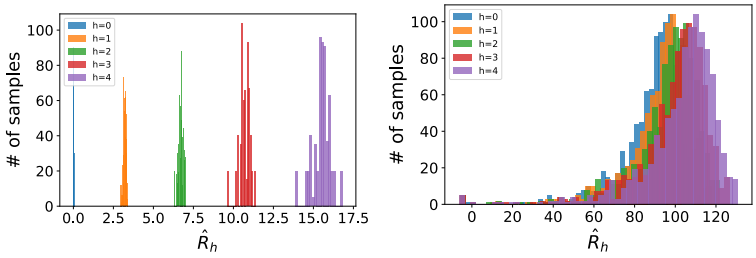
B.2 Dynamics Model Architecture

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 ensmeble 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



(a) Random initialization (b) Pretrained by BC+PE

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 $\mathcal{L}_{\mathrm{BC}}(\theta) = \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}[(\mathbf{a} - \pi_\theta(\mathbf{s}))^2]$. 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.

Comparision 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.

	ψ		
Task Name	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:

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

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.

[2] In 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 0.5	2.0	3.0	5.0	Rank correlation (ρ)
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	

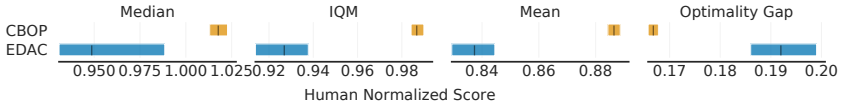


Figure 6: RLiable results across all 18 locomotion tasks. Shaded regions show 95% CIs. We refer readers to (Agarwal et al., 2021) for detailed explanation of the metrics considered.

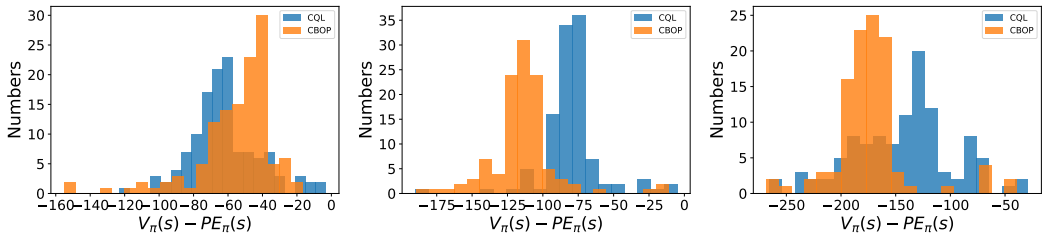
Overall, the Spearman's rank correlation values are always greater than or equal to 0.8, suggesting that the rankings from FQE align very well with those from the online evaluation. This suggests that (1) CBOP can be reliably tuned solely with an offline dataset via FQE and that (2), with the benefit 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 RLiable (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.

	CQL		CBOP	
Task name	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



(a) *hopper-m* (b) *hopper-me* (c) *hopper-mr*

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:

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

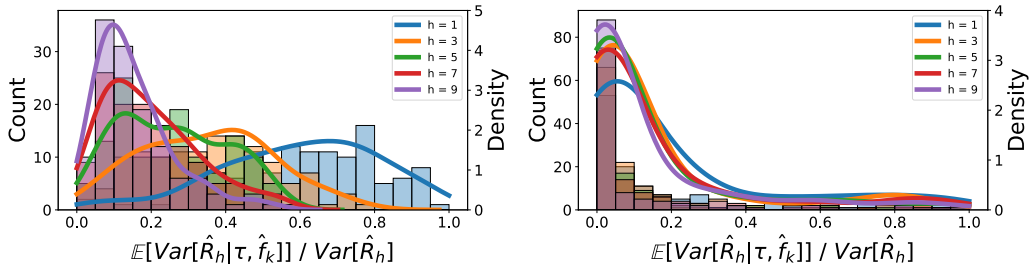
where we compute the true value $\mathbb{E}[V^\pi]$ via the Monte Carlo estimation in the true environment. We have provided the comparison of CQL and CBOP evaluated in the *hopper* environment in Table 2, and Figure 7 shows the full histograms of (11) for in this environment. Furthermore, Table 5 includes 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:

$$\sigma_h^2 = \mathrm{Var}_{\pi_\theta}\left[\hat{R}_h|\tau\right] = \underbrace{\mathbb{E}_{\hat{f}_k}\left[\mathrm{Var}_{\pi_\theta}\left[\hat{R}_h|\tau,\hat{f}_k\right]\right]}_{A} + \underbrace{\mathrm{Var}_{\hat{f}_k}\left[\mathbb{E}_{\pi_\theta}\left[\hat{R}_h\,\middle|\,\tau,\hat{f}_k\right]\right]}_{B}. \tag{12}$$

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?



(a) *hopper-expert* (b) *hopper-random*

Figure 8: The distribution of the ratio, $\frac{A}{A+B} = \mathbb{E}_{\hat{f}_k}\left[\mathrm{Var}_{\pi_\theta}\left[\hat{R}_h|\tau, \hat{f}_k\right]\right] / \sigma_h^2$, from (12) when π_θ and Q_ϕ are trained with the *hopper-r* dataset. (a) evaluates π_θ and Q_ϕ with $(\mathbf{s}, \mathbf{a})$ 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.

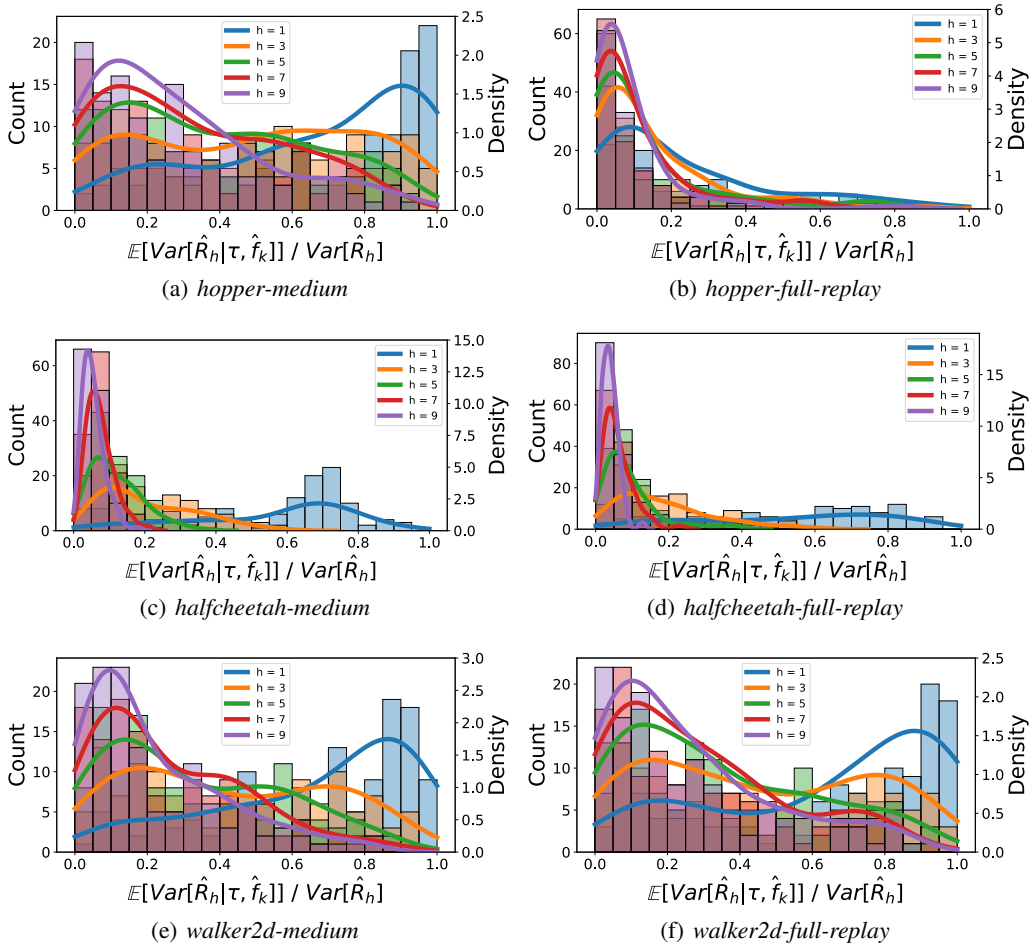
To answer this question, recall that in the offline setting, the logged data will typically only cover a subset of the state-action space. Hence, when we use the learned dynamics ensemble to forward sample rollout trajectories during the target value estimation procedure in CBOP, some of the trajectories 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 $(\mathbf{s}, \mathbf{a})$ 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



(a) *hopper-medium* (b) *hopper-full-replay*

(c) *halfcheetah-medium* (d) *halfcheetah-full-replay*

(e) *walker2d-medium* (f) *walker2d-full-replay*

Figure 9: The distribution of the ratio, $\frac{A}{A+B} = \mathbb{E}_{\hat{f}_k}\left[\mathrm{Var}_{\pi_\theta}\left[\hat{R}_h|\tau, \hat{f}_k\right]\right] / \sigma_h^2$, 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.

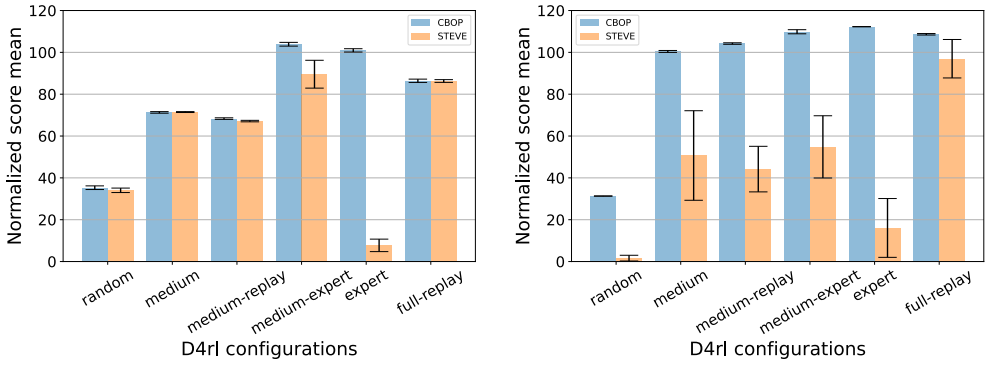
since the learned model would typically be quite accurate for single-step predictions, hence smaller B compared to A.

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.

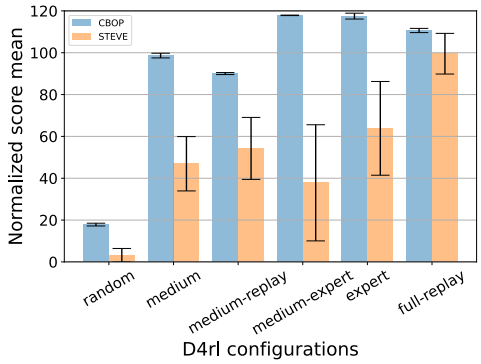
D.3 Ablations

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



(a) halfcheetah-v2 (b) hopper-v2



(c) walker2d-v2

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

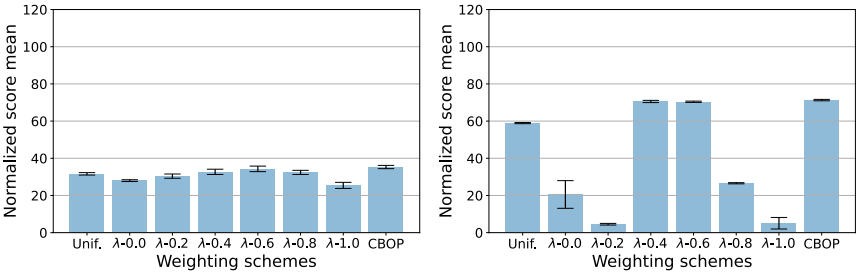
the posterior we get in (7). In this part, we provide the complete ablations comparing CBOP and STEVE in all tasks.

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.

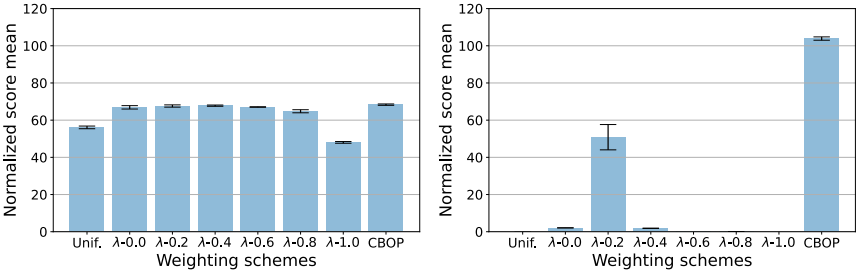
The differences in the performances are even more striking in the other two environments. Figure 10(b) and 10(c) show that CBOP significantly outperforms STEVE, suggesting that conservatism plays a crucial role. It is worth reasserting that the original adaptive weighting scheme derived in STEVE does 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 $\mathbf{s}'$ 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.

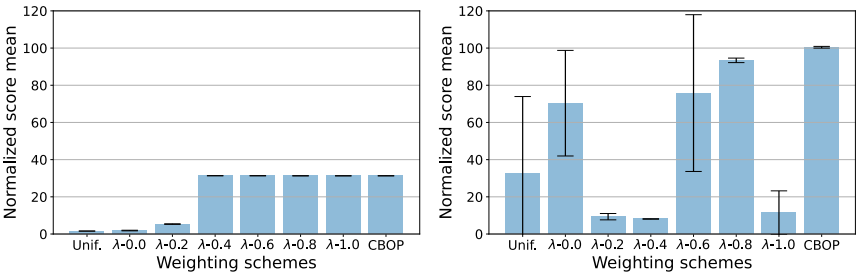


(a) *halfcheetah-random* (b) *halfcheetah-medium*

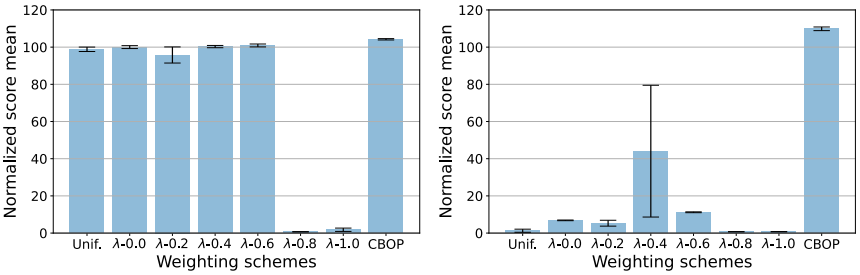


(c) *halfcheetah-medium-replay* (d) *halfcheetah-medium-expert*

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



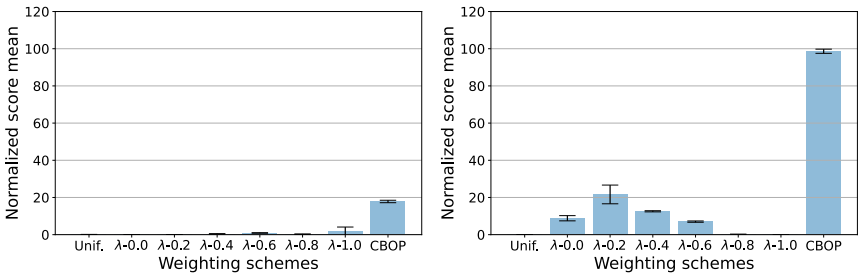
(a) *hopper-random* (b) *hopper-medium*



(c) *hopper-medium-replay* (d) *hopper-medium-expert*

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.



(a) *walker2d-random* (b) *walker2d-medium*

(c) *walker2d-medium-replay* (d) *walker2d-medium-expert*

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 $\hat{Q}^\pi$, 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 kth particle from the model ensemble and the mth 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 $\alpha \in (0, 1]$:

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

Let $\hat{R}_1 \leq \hat{R}_2 \leq \ldots \hat{R}_{M \times K \times H}$ 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	$\mathbf{74.3} \pm 0.2$	70.9 ± 0.6 (0.3085)
halfcheetah-mr	$\mathbf{66.4} \pm 0.3$	65.0 ± 0.3 (0.3085)
halfcheetah-me	$\mathbf{100.4} \pm 0.9$	84.4 ± 6.6 (0.3085)
halfcheetah-fr	$\mathbf{85.5} \pm 0.3$	83.4 ± 0.8 (0.4)
walker2d-m	$\mathbf{95.5} \pm 0.4$	65.1 ± 3.4 (0.3085)
walker2d-mr	$\mathbf{92.7} \pm 0.9$	88.5 ± 0.2 (0.3085)
walker2d-me	$\mathbf{117.2} \pm 0.5$	113.0 ± 9.8 (0.3085)
walker2d-fr	$\mathbf{107.8} \pm 0.2$	104.6 ± 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.