Lightweight Uncertainty for Offline Reinforcement Learning via Bayesian Posterior

Anonymous authors
Paper under double-blind review

Abstract

Offline Reinforcement Learning (RL) aims to learn optimal policies from fixed datasets. Directly applying off-policy RL algorithms to offline datasets typically suffers from the distributional shift issue and fails to obtain a reliable value estimation for out-of-distribution (OOD) actions. To this end, several methods penalize the value function with uncertainty quantification and achieve tremendous success from both theoretical and empirical perspectives. However, such uncertainty-based methods typically require estimating the lower confidence bound (LCB) of the Q-function based on a large number of ensemble networks, which is computationally expensive. In this paper, we propose a lightweight uncertainty quantifier based on approximate Bayesian inference in the last layer of the Q-network, which estimates the Bayesian posterior with fewer ensembles. We then obtain the uncertainty quantification by the disagreement of the Q-posterior. Moreover, to avoid mode collapse in OOD samples and improve diversity in the Q-posterior, we introduce a repulsive force for OOD predictions in training. We show that our method recovers the provably efficient LCB-penalty under linear MDP assumptions. We further compare our method with other baselines on the D4RL benchmark. The experimental results show that our proposed method achieves state-of-the-art performance on most tasks with more lightweight uncertainty quantifiers.

1 Introduction

Offline Reinforcement Learning aims to learn policies from a fixed dataset without interacting with the environment (Levine et al., 2020), which is appealing for applications that are costly to perform online exploration (Levine et al., 2018; Kalashnikov et al., 2021). Directly applying the off-policy RL algorithms in offline settings often suffers from distributional shift problems since the learned policy is different from the behavior policy. Previous offline RL methods address this problem by restricting the learned policy to stay close to the behavior policy (Fujimoto et al., 2019; Kumar et al., 2019) or penalizing OOD actions to perform value regularization (Kumar et al., 2020). Nevertheless, these methods rely heavily on behavior policies or ignore potentially good OOD actions, leading to an overly conservative policy.

Uncertainty-based methods provide an alternative and promising way to address the distributional shift by characterizing the uncertainty of Q-values and pessimistically updating the estimation. In this case, it is important to obtain an accurate and reliable uncertainty quantification, especially for OOD data points. To this end, PBRL (Bai et al., 2022) measures the uncertainty by the disagreement of bootstrapped Q-functions and performs OOD sampling to handle the extrapolation error; SAC-N (An et al., 2021) extends the critic with a large number of ensembles and achieves strong performance in offline environments. In spite of such progress, these methods require lots of ensembles, which are computationally expensive and intractable in practice. Therefore, the following question emerges: *Can we obtain a reliable uncertainty quantification with more efficient methods?*

In this paper, we propose a Bayesian uncertainty algorithm for offline RL (BURL), an uncertainty-based method with a lightweight approximate posterior. We propose two components for reliable uncertainty: approximate Bayesian learning and repulsive regularization. Specifically, we adopt a Bayesian method to learn the posterior distribution of value function, then the low-confidence-bound (LCB) of the Q-posterior is used for pessimistic value update. Theoretically, the Bayesian uncertainty defined by the disagreement among different Q-samples from the posterior recovers the provably efficient LCB-penalty in the Bayesian linear regression case. Nevertheless, exact Bayesian inference is unachievable in large-scale problems. In BURL, we adopt a Bayesian neural network (BNN) in the last layer of the Q-network to learn the approximate Bayesian posterior, which brings few parameters compared to an ordinary Q-network. In addition, to prevent different Q-samples from collapsing into the same function, we propose a repulsive regularization term for the OOD data sampled following the learned policy. Since the uncertainty hinges on the quality of the posterior, the repulsive force enforces diversity among different posterior samples to improve the uncertainty quantification. We obtain the uncertainty estimation by sampling several Q-functions from the posterior distribution and learn the policy by following the LCB of the value function.

Compared to SAC-N, which is heavily dependent on the number of networks in the ensemble, BURL quantifies the uncertainty by sampling multiple Q-values from the posterior. Although performing Bayesian inference is approximately equivalent to sampling from infinite ensemble networks, in practice, we find that purely relying on the Bayesian posterior is still inferior to ensemble-based methods in large-scale tasks. To further boost the empirical performance, we combine BNN and ensemble in a unified architecture by using several BURL networks as an ensemble and enforcing repulsive force on posterior samples from both the ensemble and BNN. Experiments show that this approach achieves state-of-the-art performance with a few ensembles on D4RL benchmarks (Fu et al., 2020). Our contributions can be concluded as follows:

- We propose BURL, a lightweight uncertainty-based offline RL algorithm that uses last-layer BNN to approximate the Bayesian posterior with the help of a few ensembles. We relate the Bayesian uncertainty to the LCB penalty in linear MDP settings.
- We introduce a repulsive regularization term to improve the diversity among different posterior samples, leading to better parametric efficiency.
- We conduct a large number of experiments and ablation studies to validate our algorithm, which shows competitive performance and reasonable uncertainty quantification.

2 Preliminaries

Episodic MDP. We consider an episodic Markov Decision Process (MDP) formulated as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, T)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in (0, 1]$ is the discount factor, P is the transition distribution, and T is the episode length. In RL, we aim to learn an optimal policy $\pi(a|s)$ that maximizes the cumulative discounted reward as $\mathbb{E}[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)]$. This objective can be optimized by estimating a state or state-action value function, which gives the expected cumulative reward, and then recovering a near-optimal policy. We present the recursive definitions for these value functions as $V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)]$ and $Q^\pi(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\pi(s')]$. These two equations can be combined to express the $Q^\pi(s,a)$ in terms of $Q^\pi(s',a')$ Levine et al. (2020): $Q^\pi(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim P(\cdot|s,a), a'\sim\pi(\cdot|s)}[Q^\pi(s',a')]$.

In this work, we use a parameterized policy and a parameterized state-action value function $Q^\pi_w(s,a)$, where w is the parameter of the Q-network. The Q-function is learned corresponding to the current policy $\pi(\cdot|s)$ and obeys the following equation by omitting the superscript:

$$Q_w(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim P(\cdot|s,a), a'\sim\pi(\cdot|s)}[Q_w(s',a')].$$

This equation can be expressed as the Bellman operator, which indicates the learning target for the Q-network. By employing the target network that is softly updated for stability Mnih et al. (2015), the Bellman operator is given by:

$$\widehat{\mathcal{T}}Q_w(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim P(\cdot|s,a), a'\sim\pi(\cdot|s)}[Q_{w^-}(s',a')], \tag{1}$$

where w^- is the parameter of the target Q-network.The Q-function is learned by minimizing the Bellman residual error as $\mathbb{E}_{s'\sim P(\cdot|s,a), a\sim\pi(\cdot|s)}[Q(s,a) - \mathcal{T}Q(s,a)]^2$, where the transitions are obtained by interacting with the environment. We refer to Sutton & Barto (2018) for more details.

Offline RL. In offline RL, the agent needs to learn a policy from a fixed dataset $\mathcal{D}_m = \{s^i_t, a^i_t, r^i_t, s^i_{t+1}\}_{i\in[m]}$ that contains m episodes. Then the Bellman operator $\widehat{\mathcal{T}}Q$ is computed by estimating the expectation based on the offline datasets $\mathcal{D}_m$, and the Bellman error becomes

$\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_m}[Q(s,a)-\widehat{\mathcal{T}}Q(s,a)]^2$ by sampling experiences from the dataset. The policy $\pi(\cdot|s)$ is learned by maximizing the value function $Q(s,a)$. The problem of offline RL is the greedy action a' chosen by the policy $\pi(\cdot|s)$ in Eq. (1) can be OOD actions, since a' is often scarcely covered in $\mathcal{D}_m$. These OOD actions cause high extrapolation errors in value function estimation, which is also called distributional shift. Since the errors cannot be corrected by online interactions, the distributional shift often results in sub-optimal policies.

To address such a problem, several offline RL algorithms (e.g. BCQ Fujimoto et al. (2019) and CQL Kumar et al. (2020)) pessimistically update the value functions by constraining the learned policy or penalizing the value function of OOD actions. Nevertheless, these methods lack specific uncertainty estimation and lead to overly conservative behavior. Alternatively, uncertainty-based algorithms like SAC-N combine hundreds of Q-networks as an ensemble and pessimistically update the policy to maximize the LCB of the Q-value, which obtains strong performance while being computationally expensive. Other methods reduce the computation by performing additional OOD sampling (Bai et al., 2022) or diversifying the gradient of value functions (An et al., 2021), respectively. Nevertheless, they still need 10$\sim$50 independent networks for uncertainty quantification.

3 Method

In this section, we first introduce the Bayesian posterior for uncertainty quantification and relate it to the LCB penalty. Since the LCB penalty cannot be directly applied to environments with high-dimensional observations, we approximate the LCB-term via the Bayesian posterior of the non-linear neural networks. To further improve the computational efficiency, we propose the repulsive force for regularization and improve the diversity in different posterior samples. Finally, we give the concrete optimization objective and algorithmic descriptions.

3.1 Bayesian Posterior for Uncertainty Quantification

We present the motivation and foundation of the proposed uncertainty in linear MDPs (Jin et al., 2021), where the transition and reward function are assumed to be linear in the state-action feature $\phi(s,a)$. Considering that we have a learned representation $\phi(s,a)$, the value function $Q(s,a)=\phi(s,a)^\top w$ is also linear in $\phi(s,a)$, where w is the underlying true parameter of the linear Q-function. We refer to related work Jin et al. (2021; 2020); Agarwal et al. (2022) for more details about linear MDPs. In BURL, our objective is to approximate the posterior $\mathbb{P}(Q\,|\,s,a,\mathcal{D}_m)$ of the value function.

Assumption 1. *There exists a known feature mapping $\phi:\mathcal{S}\times\mathcal{A}\to\mathbb{R}^d$ and an unknown vector $w\sim\mathcal{N}(0,I_d/\lambda)$ such that $Q(s,a)=\phi(s,a)^\top w+\epsilon$, where $\epsilon\sim\mathcal{N}(0,1)$ and λ is a constant.*

We focus on the linear MDP setting where the transition dynamics and the reward function are assumed to be linear. On the basis of previous work Jin et al. (2020), Assumption 1 is reasonable and simplifies the LCB derivation by hypothesizing that w is a non-informative Gaussian prior, and noise ϵ in the least-square problem follows the standard Gaussian. We refer to §A for more details.

Theorem 1. *Under Assumption 1, it holds for the standard deviation of the estimated posterior distribution $\mathbb{P}(\tilde{Q}\,|\,s,a,\mathcal{D}_m)$ that*

$$\mathrm{Std}(\tilde{Q}\,|\,s_t,a_t,\mathcal{D}_m)=\big[\phi(s_t,a_t)^\top\Lambda_t^{-1}\phi(s_t,a_t)\big]^{1/2}:=\Gamma_t^{\mathrm{lcb}}(s_t,a_t),\tag{2}$$

where $\Lambda_t=\sum_{i\in[m]}\phi(s_t^i,a_t^i)\phi(s_t^i,a_t^i)^\top+\lambda\cdot\mathbf{I}$, and $\Gamma_t^{\mathrm{lcb}}(s_t,a_t)$ is defined as the LCB-term.

The proof is given in §A. The feature covariance matrix Λ_t accumulates the state-action features in learning the Q-function, and the LCB of the Q-function given in Eq. (2) measures the confidence interval of the Q-functions with the offline data $\mathcal{D}_m$. Intuitively, the LCB-term $\Gamma_t^{\mathrm{lcb}}(s_t,a_t)$ can also be considered as a reciprocal pseudo-count of the state-action pair in the representation space of ϕ.

Approximate the Q-posterior $\mathbb{P}(\tilde{Q}\,|\,s,a,\mathcal{D}_m)$. Theorem 1 shows that the standard deviation of the Q-posterior is equivalent to the LCB-term $\Gamma_{\mathrm{lcb}}(s_t,a_t)$ in linear MDPs. Nevertheless, such a term is specifically designed for linear MDPs and cannot be directly applied to environments with high-dimensional observations. For BURL in deep offline RL, we approximate the posterior $\mathbb{P}(\tilde{Q}\,|\,s,a,\mathcal{D}_m)$ via non-linear neural networks. Specifically, we interpret Assumption 1 by considering $\phi(s,a)$ as the output of *hidden layers* of the Q-network, and w as the weight of the *last layer*.

Then we still have the relationship of $Q(s, a) = \phi(s, a)^\top w + \epsilon$, where the noise term ϵ captures the aleatoric uncertainty in the Q-value.

With Assumption 1, it suffices to approximate $\mathbb{P}(w \,|\, \mathcal{D}_m)$ in order to approximate the posterior. To that end, we adopt a variational inference approach (Blundell et al., 2015). Note that

$$\begin{aligned}\mathbb{P}(\mathcal{D}_m) &= \mathbb{E}_{\mathbb{P}(w \,|\, \mathcal{D}_m)}[\mathbb{P}(\mathcal{D}_m \,|\, w)] \\ &= \underbrace{\mathbb{E}_{q_\theta(w \,|\, \mathcal{D}_m)}[\log \mathbb{P}(\mathcal{D}_m \,|\, w)] - \mathrm{KL}(q_\theta(w \,|\, \mathcal{D}_m) \,\|\, \mathbb{P}(w))}_{\text{ELBO}} + \mathrm{KL}(q_\theta(w \,|\, \mathcal{D}_m) \,\|\, \mathbb{P}(w \,|\, \mathcal{D}_m)), \end{aligned} \tag{3}$$

where we use q_θ as the variational distribution of the posterior of w. We denote by q^* the optimal solution to maximizing the evidence lower bound (ELBO). According to Eq. (3), maximizing the ELBO term is equivalent to minimizing the Kullback-Leibler divergence between our variational distribution $q(w \,|\, \mathcal{D}_m)$ and the posterior distribution $\mathbb{P}(w \,|\, \mathcal{D}_m)$, which leads to the desired result $q^*(w \,|\, \mathcal{D}_m) = \mathbb{P}(w \,|\, \mathcal{D}_m)$. We denote by $\tilde{Q}_w$ the random variable for the estimated state-action value with a corresponding weight $w \sim q^*(w \,|\, \mathcal{D}_m)$.

The ELBO learning objective. To derive the practical learning objective of BURL based on Eq. (3), we construct a new offline dataset $\widehat{\mathcal{D}}_m = \{\phi_t^i, y_t^i\}_{i \in [m]}$, where the elements can be obtained based on dataset $\mathcal{D}_m$ by calculating the state-action feature $\phi(s_t^i, a_t^i)$ and the target-value function $y_t^i = \widehat{\mathcal{T}} Q(s_t^i, a_t^i)$ by Eq. (1). In our analysis, we consider y_t^i as a fixed learning target based on the current Q-function. As a result, the optimal solution of q_θ^* in the ELBO becomes

$$\begin{aligned} q_\theta^* &= \arg\max \mathbb{E}_{q_\theta(w \,|\, \widehat{\mathcal{D}}_m)}\big[\log \mathbb{P}\big(\widehat{\mathcal{D}}_m \,|\, w\big)\big] - \mathrm{KL}\big(q_\theta(w \,|\, \widehat{\mathcal{D}}_m) \,\|\, \mathbb{P}(w)\big) \\ &= \arg\max \mathbb{E}_{q_\theta(w \,|\, \widehat{\mathcal{D}}_m)}\Big[\sum\nolimits_{i \in [m], t \in [T]} \log \mathbb{P}\big(y_t^i \,|\, \phi_t^i, w\big)\Big] - \mathrm{KL}\big(q_\theta(w \,|\, \widehat{\mathcal{D}}_m) \,\|\, \mathbb{P}(w)\big) \\ &\approx \arg\max \sum\nolimits_{k \in [n]} \sum\nolimits_{i \in [m], t \in [T]} \log \mathbb{P}\big(y_t^i \,|\, \phi_t^i, w^{(k)}\big) - \mathrm{KL}\big(q_\theta(w \,|\, \widehat{\mathcal{D}}_m) \,\|\, \mathbb{P}(w)\big), \end{aligned} \tag{4}$$

where the last step uses Monte Carlo sampling to approximate the variational posterior, and $w^{(k)}$ denotes the k-th sample drawn from $q_\theta(w \,|\, \widehat{\mathcal{D}}_m)$. To allow gradient backpropagation to optimize the objective, we suppose that the posterior is a diagonal Gaussian distribution and use the re-parameterization trick to sample several random vectors $\{z^{(k)} \sim p(z)\}_{k \in [n]}$ from a standard Gaussian. Then we obtain the posterior weight as $\{w^{(k)} = \sigma_w^\top z^{(k)} + \mu_w\}_{k \in [n]}$, where we denote by $q_\theta(w) = \mathcal{N}(\mu_w, \Sigma_w)$. The sampled Q-value is obtained by $\{Q^{(k)} = \phi^\top w^{(k)}\}_{k \in [n]}$.

- For the first term in Eq. (4), we simplify the target by using a deterministic response, then the log-likelihood reduces to Bellman residual error between the target $\widehat{\mathcal{T}} Q^{(k)}$ and $Q^{(k)}$. Such an error is calculated by taking the average among state-action pairs and posterior samples.
- For the second term in Eq. (4), the prior function is also assumed to be a Gaussian distribution $\mathbb{P}(w) \sim \mathcal{N}(0, \sigma_p)$ and the KL term is easy to optimize.

Practical considerations. We have two practical considerations in approximating the Bayesian posterior. (i) In offline RL, the learning target y_t^i is calculated following Eq. (1) based on the target Q-network, which will be updated in training. Such a target is different from an ordinary regression task in which the target is fixed and given in the dataset. Empirically, we find that a changing target does not cause instability since it is generated by the target Q-network that changes infrequently. (ii) The representation $\phi(s, a)$ is assumed to be fixed in Assumption 1, while in practice, we use $\phi(s, a)$ as part of the Q-network to extract the feature of state-action pairs. We learn the representation $\phi(s, a)$ and the posterior distribution simultaneously by following the gradients of Eq. (4).

Pessimistic Q-learning. Based on the posterior distribution $q_\theta(w) = \mathcal{N}(\mu_w, \Sigma_w)$, there are two methods to obtain the pessimistic value function. (i) We can calculate the standard deviation of the posterior as $\Gamma_t(s_t, a_t) = \phi(s_t, a_t)^\top \Sigma_w \phi(s_t, a_t)$ to serve as an estimation of Γ_t^{lcb} in Eq. (2). Then the pessimistic value function is obtained using the LCB as a penalty, i.e., $Q_{\mathrm{lcb}}^1(s_t, a_t) = \bar{Q}(s_t, a_t) - \alpha \Gamma_t(s_t, a_t)$, where $\bar{Q}(s_t, a_t) = \phi(s_t, a_t)^\top \mu_w$ is the mean Q-value. (ii) Alternatively, we can sample several Q-values from the posterior distribution, and obtain the pessimistic Q-value by taking the minimal value of the Q-samples as $Q_{\mathrm{lcb}}^2(s_t, a_t) = \min_{k \in [n]} Q^{(k)}(s_t, a_t)$. We remark

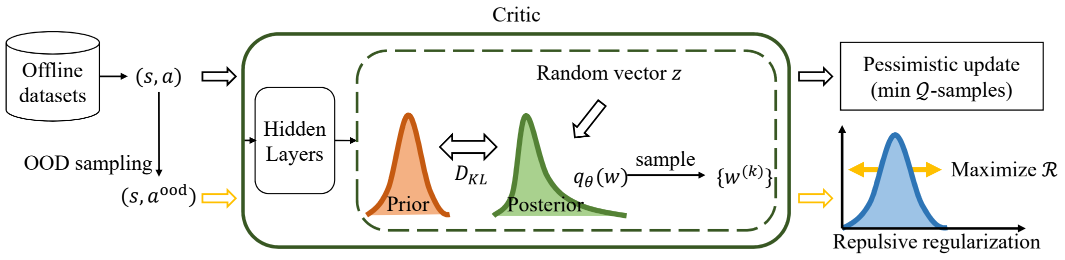

Figure 1: The architecture of BURL, where we utilize approximate Bayesian inference in the last layer of the critic network. We perform OOD sampling to obtain (s, a^{ood}) pairs based on $(s, a) \sim \mathcal{D}_m$. We input (s, a) and (s, a^{ood}) pairs to obtain the Q-samples and the repulsive term, respectively. Then (s, a) is used for the pessimistic update of the Q-values, and (s, a^{ood}) provides regularization.

that Q^1_{lcb} and Q^2_{lcb} are approximately equivalent with an appropriate α. Since the realization $Q^{(k)}$ has a linear mapping to $w^{(k)}$, $Q^{(k)}$ also follows a Gaussian distribution. According to the previous work (Royston, 1982; An et al., 2021), we have

$$\mathbb{E}\big[\min_{k\in[n]} Q^{(k)}(s, a)\big] \approx \bar{Q}(s, a) - \alpha \cdot \text{Std}\big(\{Q^{(k)}(s, a)\}_{k\in[n]}\big) \approx \bar{Q}(s, a) - \alpha\Gamma_t(s, a), \tag{5}$$

where $\alpha = \Phi^{-1}(\frac{N-\pi/8}{N-\pi/4+1})$, and $\Phi(\cdot)$ is the cumulative distribution function of the standard Gaussian. In practice, we use Q^2_{lcb} as the pessimistic value function since (i) it is easy to combine with ensemble methods by sampling multiple values from both the ensemble and BNNs; and (ii) the sampling method can be calculated effectively via matrix manipulation by using multiple noise vectors.

3.2 Repulsive Regularization

Based on the estimated Bayesian posterior, we can sample infinite Q-values from the posterior for a single (s, a) pair, which is approximately equivalent to infinite Q-ensembles (Pearce et al., 2020). Previous studies show that the major challenge for ensemble methods is the quality of the ensemble hinges on the diversity among ensemble members (D'Angelo & Fortuin, 2021; Hiraoka et al., 2022). To address this problem, several works propose to improve the diversity of the ensemble without compromising the individual accuracy. For example, Random Prior (Osband et al., 2018) provides independent prior networks for ensemble members, and Stein variational gradient descent (SVGD) (Liu & Wang, 2016) introduces a kernelized repulsive force in the parameter space. Inspired by these methods, we propose a simple repulsive term to regularize the Bayesian posterior.

In BURL, we explicitly increase the diversity of different Q-samples to prevent the posterior from collapsing into the same solution. The repulsive regularization $\mathcal{R}(\cdot)$ can either be applied in the weight space (i.e., $\mathcal{R}\big(\{w^{(k)}\}\big)$) or the function space (i.e., $\mathcal{R}\big(\{Q^{(k)}\}\big)$). Since the weight space has much higher dimensions, calculating the regularization on the function space is more efficient. Empirically, we find that these two methods have similar performance. We use the repulsive term as

$$\mathcal{R}\big(Q(s, a^{\text{ood}})\big) = \text{Std}\big(\{Q^{(k)}(s, a^{\text{ood}})\}_{k\in[n]}\big), \tag{6}$$

where the OOD action a^{ood} is sampled from the learned policy $\pi(\cdot|s)$, s is sampled from the dataset, and $Q^{(k)}$ is sampled from the posterior. By maximizing $\mathbb{E}_{s,a^{\text{ood}}\sim\pi}\big[\mathcal{R}\big(Q(s, a^{\text{ood}})\big)\big]$, we explicitly maximize the disagreement of Q-samples of OOD actions, which helps improve the reliability of the uncertainty quantification. We remark that we do not enforce regularization for (s, a) pairs sampled from the offline dataset since the uncertainty tends to be small for in-distribution samples. The whole process of Bayesian inference and repulsive regularization is shown in Fig. 1. The offline data and OOD data are used to perform pessimistic Bellman update and repulsive regularization, respectively.

Although performing Bayesian inference and repulsive regularization are sufficient to obtain the posterior function, we find it still inferior to ensemble-based methods empirically in large-scale offline tasks. To further improve BURL, we combine the proposed Bayesian posterior and the ensemble in a unified architecture using M BNNs as an ensemble $\{Q^{(k)}_j\}_{k\in[n],j\in[M]}$. Then, the repulsive regularization is calculated by sampling from both the ensemble (i.e., $j \in [M]$) and the Bayesian posterior (i.e., $k \in [n]$). As a result, BURL combines ensemble, BNN, and repulsion to obtain a posterior. Compared to the previous uncertainty-based methods (An et al., 2021; Bai et al., 2022), our method is more efficient as we require much fewer ensemble members (i.e., 2∼5).

Algorithm 1 Bayesian Uncertainty for offline RL (BURL)

```
Require: Offline dataset $\mathcal{D}_m$, policy network $\pi_\psi$, $Q$-networks $\{Q_{w_j}\}_{j\in[M]}$, and target $Q$-networks
    $\{Q_{w_j^-}\}_{j\in[M]}$. Initialize the parameters, including $\psi$, $\theta$, $\phi$, $\{w_j\}_{j\in[M]}$, and $\{w_j^-\}_{j\in[M]}$.
1: while not coverage do
2:     Sample transitions $\{(s,a,r,s')\}$ from $\mathcal{D}_m$ and construct the OOD data as $\{(s,a^{\rm ood})\}$.
3:     Calculate the pessimistic target $Q$-values by sampling from the posterior via Eq. (7).
4:     Minimize the critic loss $\mathcal{L}_{\rm critic}$ in Eq. (8) and update the parameters.
5:     Train the pessimistic policy by minimizing the actor loss $\mathcal{L}_{\rm actor}$ in Eq. (9).
6:     Update target $Q$-networks by $\{w_j^-\}' \leftarrow \rho\{w_j^-\}' + (1-\rho)\{w_j\}$
7: end while
```

3.3 Learning with Bayesian Uncertainty

We now introduce the practical learning algorithm of BURL. We use M ensemble BNNs as a critic; then the pessimistic learning target for the j-th Q-network is given as,

$$\widehat{\mathcal{T}}^{\rm lcb}Q_{w_j}(s,a)=r(s,a)+\gamma\hat{\mathbb{E}}_{\substack{s'\sim P(\cdot|s,a),a'\sim\pi_\psi(\cdot|s')\\ w_j^{(k-)}\sim q_\theta(w_j^-|\mathcal{D}_m)}}\Big[\min_{\substack{j\in[M]\\k\in[n]}}Q^{(k)}_{w_j^-}(s',a')-\beta\log\pi_\psi(a'|s')\Big],\quad(7)$$

where w_j and w_j^- are the Bayesian parameters of the main Q-network and the target Q-network, respectively, π_ϕ is the policy network. In Eq. (7), we sample n Q-values from each BNN and take the minimum value among all the $n\times M$ sampled Q-values of the M BNNs to compute the target. Given the prior function and repulsive term, the loss function for training the j-th Q-network is

$$\begin{aligned}\mathcal{L}^j_{\rm critic}&=\eta_q\cdot\mathbb{E}_{s,a,r,s'\sim\mathcal{D}_m}\Big[\Big(Q_{w_j}(s,a)-\widehat{\mathcal{T}}^{\rm lcb}Q_{w_j}(s,a)\Big)^2+{\rm KL}\big(q_\theta(w_j)\,\|\,\mathbb{P}(w)\big)\Big]\\&\quad-\eta_{\rm ood}\cdot\mathbb{E}_{s\sim\mathcal{D}_m,a^{\rm ood}\sim\pi}\left[\mathcal{R}\big(Q_{w_j}(s,a^{\rm ood})\big)\right],\end{aligned}\quad(8)$$

where $\widehat{\mathcal{T}}^{\rm lcb}Q_{w_j}$ is defined in Eq. (7), η_q and $\eta_{\rm ood}$ are constant factors, and $\mathcal{R}(\cdot)$ is the repulsive term defined in Eq. (6). The first two terms in $\mathcal{L}^j_{\rm critic}$ recover the variational ELBO defined in Eq. (4), where the first term maximizes the log-likelihood of the target, and the second term regularizes the posterior distribution. The last repulsive term maximizes the diversity of samples for OOD actions. The loss function is used to train the parameter θ of the Gaussian distribution $q_\theta(w_j)$ and also the representation $\phi(s,a)$.

Based on the value function, the policy network is updated by maximizing the minimal sampled value from the Bayesian posterior and ensemble, as

$$\mathcal{L}_{\rm actor}=-\mathbb{E}_{s\sim\mathcal{D}_m,a\sim\pi_\psi(\cdot|s),w_j^{(k)}\sim q_\theta(w_j|\mathcal{D}_m)}\left[\min_{k\in[n],j\in[M]}Q^{(k)}_{w_j}(s,a)-\beta\log\pi_\psi(a|s)\right],\quad(9)$$

which includes an SAC-style entropy regularization. We summarize the training process in Alg. 1.

4 Related Work

Offline RL Most offline RL works aim to address the distributional shift problem and avoid overestimation of the Q-values for OOD actions. Specifically, previous works (i) restrict the learned policy to the static dataset by explicitly estimating the behavior policy (Fujimoto et al., 2019; Wu et al., 2019) or implicitly regularizing policy learning (Kumar et al., 2019; Peng et al., 2019; Nair et al., 2020; Fujimoto & Gu, 2021); (ii) utilize value regularization (Kumar et al., 2020; Nachum et al., 2019) and pessimistically update the Q-values for OOD actions; (iii) perform one-step policy improvement to avoid overestimation (Brandfonbrener et al., 2021; Kostrikov et al., 2022). However, these methods could be overly conservative to ignore potentially good OOD actions. In contrast, our method obtains better value estimation for OOD data via reliable uncertainty quantification. There are also model-based offline approaches that aim to learn a pessimistic MDP (Yu et al., 2020; Kidambi et al., 2020), use ensembles to characterize the uncertainty of dynamics models (Yu et al., 2020; Kidambi et al., 2020; Matsushima et al., 2021), combine conservative value estimation with dynamic models (Yu et al., 2021), or model the trajectory distribution using a high-capacity Transformer (Chen et al., 2021; Janner et al., 2021; Wang et al., 2022). Nevertheless, these methods rely on the prediction accuracy of the dynamics model or need additional modules such as Transformers. On the contrary, BURL follows the model-free manner and is computationally efficient. Recently, an adversarial trained actor-critic algorithm (Cheng et al., 2022) based on a two-player Stackelberg game presents promising results, while still being computationally expensive.

Uncertainty Quantification The uncertainty quantification is pervasively used for exploration in the online RL setting (Bellemare et al., 2016; Lee et al., 2021), i.e., *optimism in the face of uncertainty*; its offline counterpart tends to be *pessimism in the face of uncertainty* (Jin et al., 2021). Because of the limited coverage of offline datasets and the extrapolation error from OOD actions, it is more challenging to obtain accurate uncertainty quantification in an offline setting. The ensemble of neural networks is a widely used method to estimate predictive uncertainty via independent initializations (Agarwal et al., 2020; Bai et al., 2022; An et al., 2021), different hyperparameters (Wenzel et al., 2020), the depth of networks (Antorán et al., 2020), or early exits (Qendro et al., 2021). However, Large numbers of neural networks are required to approximate the posterior and may suffer from model collapse. Bayesian neural network provides an alternative way to learn the posterior over the parameters of the model. Due to that the exact posterior is intractable to compute, previous methods choose to approximate it by Markov chain Monte Carlo (Welling & Teh, 2011), variational inference (Blundell et al., 2015), or Monte-Carlo dropout (Gal & Ghahramani, 2016; Hiraoka et al., 2022; Wu et al., 2021). Noisy-Net (Fortunato et al., 2018) injects noise to the parameter while is not ensured to approximate the posterior distribution of parameters. Our method is related to Hyper-DQN (Li et al., 2021), which adopts Thompson sampling and hyper-network for exploration, while we use variational inference to estimate the posterior distribution for pessimism.

5 EXPERIMENTS

In this section, we conduct multiple experiments and comparisons in different setups to answer the following questions: 1) Does our method exhibit better performance than other baseline methods in different setups, including continuous control tasks and navigation tasks? 2) Can our method obtain a reasonable uncertainty estimation of the value function? 3) What about the computational or parametric efficiency of BURL? 4) Which parts or factors are critical in our methods? We evaluate our algorithm using the standard D4RL benchmarks, including Gym Mujoco tasks and challenging AntMaze navigation tasks. We release our codes at https://anonymous.4open.science/r/BURL-4158.

Baselines. To verify the feasibility and superiority of our algorithm, we compare BURL with different types of offline RL methods: 1) model-free value-based methods, including CQL (Kumar et al., 2020), TD3-BC (Fujimoto & Gu, 2021), and IQL (Kostrikov et al., 2022); 2) model-based MOPO algorithm (Yu et al., 2020), which uses ensembles to learn transition dynamics; 3) ATAC (Cheng et al., 2022), which exhibits promising results by framing offline RL as a two-player game; 4) uncertainty-based methods, including PBRL (Bai et al., 2022), EDAC (An et al., 2021), and SAC-N. In addition, we compare BURL with uncertainty-based methods to show that BURL obtains reliable uncertainty estimation and achieves better performance with fewer ensemble networks.

Results in Gym-Mujoco tasks. We compare the performance of all methods in Gym Mujoco tasks, which include three environments: halfcheetah, walker2d, and hopper. Each environment consists of five datasets (random, medium, medium-replay, medium-expert, and expert) generated by different behavior policies. We report the normalized scores for each task and the average scores in Gym Mujoco tasks in Table 1. We refer to §B for more experimental details.

The experimental results in Table 1 show that BURL outperforms CQL, TD3-BC, IQL, and MOPO by a large margin in most cases and exhibits better performance than ATAC in halfcheetah and walker2d environments. Compared to uncertainty-based methods such as PBRL or SAC-N, we find BURL performs better in most tasks with reliable uncertainty quantification. Although BURL is marginally worse than EDAC in halfcheetah/walker2d-medium-expert and walker2d-expert tasks, it obtains reasonable scores with much fewer ensembles. We illustrate the number of ensembles required to achieve the strong performance of EDAC and BURL in Fig. 2.

Results in AntMaze tasks. We also evaluate our method in challenging AntMaze navigation tasks, which require an 8-DoF ant robot to reach a target position in a maze scenario. In this task, the

Table 1: Performance of baseline algorithms and our proposed method on Gym Mujoco tasks. The two highest scores are in bold. The abbreviations ‘r’, ‘m’, ‘m-r’, ‘m-e’, ‘e’ correspond to random, medium, medium-replay, medium-expert, and expert datasets.

Environments	CQL	TD3-BC	IQL	MOPO	ATAC	PBRL	EDAC	SAC-N	BURL
halfcheetah-r	17.5±1.7	10.8±1.6	12.6±2.1	**35.9±2.9**	4.8	13.1±1.2	27.7±1.0	28.0±0.9	**30.9±1.1**
halfcheetah-m	47.0±0.5	48.3±0.3	47.3±0.3	**73.1±2.4**	54.3	58.2±1.5	66.0±1.5	67.5±1.2	**69.2±2.7**
halfcheetah-m-r	45.5±0.8	44.6±0.3	44.4±0.5	**69.2±1.1**	49.5	49.5±0.8	62.3±1.4	63.9±0.8	**66.7±2.0**
halfcheetah-m-e	75.6±28.7	90.1±4.4	89.3±3.0	70.3±21.9	95.5	93.6±2.3	**107.7±2.1**	**107.1±2.0**	104.2±2.4
halfcheetah-e	96.3±1.4	96.7±0.9	95.1±0.3	81.3±21.8	94.0	96.2±2.3	**106.6±1.2**	105.2±2.6	**107.1±2.6**
walker2d-r	5.1±1.5	0.8±0.7	6.0±1.1	4.2±5.7	8.0	8.8±6.3	**17.6±9.7**	**21.7±0.0**	7.4±5.7
walker2d-m	73.3±20.0	85.0±1.0	75.6±2.3	41.2±30.8	91.0	90.3±1.2	**93.7±1.3**	87.9±0.2	**95.1±2.7**
walker2d-m-r	81.8±3.0	80.6±9.1	67.0±12.2	73.7±9.4	**94.1**	86.2±3.4	85.7±2.2	78.7±0.7	**96.1±2.3**
walker2d-m-e	107.9±1.8	110.4±0.6	109.4±0.7	77.4±27.9	**116.3**	109.8±0.2	113.5±1.1	**116.7±0.4**	112.4±1.1
walker2d-e	108.6±0.6	110.2±0.5	109.8±0.1	62.4±3.2	108.2	108.8±0.2	**115.4±1.5**	107.4±2.4	**112.5±0.5**
hopper-r	7.9±0.4	9.0±0.3	8.1±0.9	16.7±12.2	**31.8**	**31.6±0.3**	12.6±10.5	31.3±0.0	27.9±7.8
hopper-m	53.0±31.8	59.0±4.0	67.9±6.4	38.3±34.9	**102.8**	81.6±14.5	101.9±0.4	100.3±0.3	**102.0±1.2**
hopper-m-r	88.7±14.4	52.3±23.5	94.8±4.9	32.7±9.4	**102.8**	100.7±0.4	100.8±0.1	101.8±0.5	**102.9±1.1**
hopper-m-e	105.6±14.4	102.9±7.0	90.8±14.8	60.6±32.5	**112.6**	111.2±0.7	110.6±0.2	110.1±0.3	**112.3±1.8**
hopper-e	96.5±31.4	108.9±2.5	109.1±3.2	62.5±29.0	**111.5**	110.4±0.3	109.8±0.1	110.3±0.3	**111.4±1.2**
Average	67.4±10.2	67.3±3.8	68.5±3.5	53.3±16.3	78.5	76.7±2.4	**82.9±2.2**	82.5±0.8	**83.9±2.4**

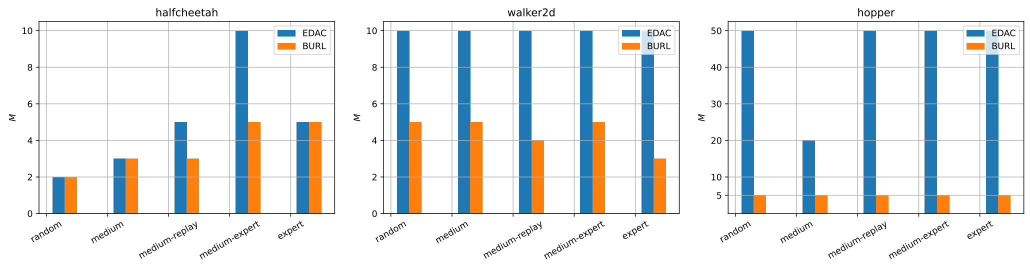

Figure 2: Minimum number of Q-ensembles (M) required to obtain the performance in Table 1. BURL needs much fewer ensembles than EDAC in most cases and reduces the required parameters.

key challenges are dealing with sparse rewards and performing sub-trajectory stitching to reach the goal. We conduct experiments on two environments (umaze and medium) with two types of datasets. Table 2 summarizes the comparison of our method with behavior cloning (BC), Filtered (‘Filt.’) BC which learns from the best 10% trajectories, and value-based methods, including CQL, TD3-BC, and IQL. We observe that our method can achieve better performance than BC and Filt. BC, as well as CQL and TD3-BC. Meanwhile, BURL outperforms IQL in the umaze task with only 5 ensembles, and obtain competitive performance in the other environments with 10 ensembles.

Table 2: Performance of the baseline algorithms and the proposed method in AntMaze tasks. We do not report the performance of other baselines like PBRL and EDAC since their original implementations cannot obtain reasonable results in AntMaze domains.

Environments	BC	Filt. BC	CQL	TD3-BC	IQL	BURL
umaze	54.6	60.0	23.4	78.6	88.6±0.5	94.2±3.1
umaze-diverse	45.6	46.5	44.8	71.4	63.6±10.3	59.4±9.9
medium-diverse	0.0	37.2	0.0	3.0	73.0±3.4	56.2±8.7
medium-play	0.0	42.1	0.0	10.6	74.0±3.6	65.2±14.4
Average	25.1	46.5	17.5	40.9	74.8±8.0	68.8±9.0

Uncertainty Quantification. To validate whether BURL provides a reasonable uncertainty quantification, we visualize the uncertainty estimation of Q-values for the in-distribution samples and the OOD samples in Fig. 3. Specifically, we train the Bayesian posterior in the medium-replay dataset and evaluate the learned uncertainty for both the in-distribution data and OOD data. Because expert trajectories are not contained in the medium-replay dataset, the state-action pairs induced by the expert policy can be regarded as OOD data points. Therefore, we sample in-distribution data points from the medium-replay dataset and OOD data points from the expert dataset. The original states and actions are projected into two-dimensional space using t-SNE. The uncertainty quantification of BURL for different state-action pairs is shown in the contour plot. As shown in Fig. 3, the in-distribution samples (white) mainly distribute in darker areas and get lower uncertainty. In contrast,

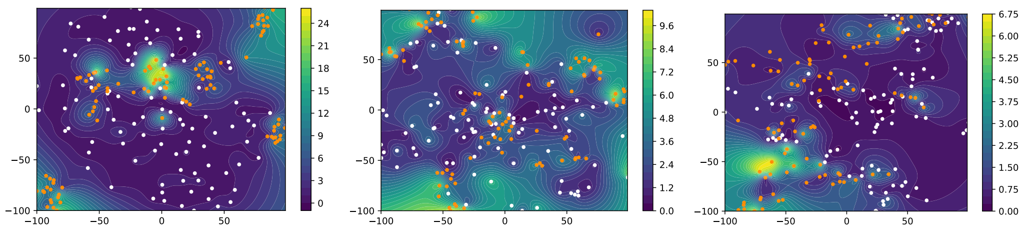
(a) halfcheetah (b) walker2d (c) hopper

Figure 3: Uncertainty estimation of the in-distribution samples (white) and OOD samples (orange). The brighter area in contour indicates higher uncertainty, while the darker area indicates lower uncertainty.

the OOD samples (orange) locate in brighter areas and are assigned with higher uncertainty. We find the uncertainty estimation generalizes well from in-distribution areas to OOD areas, which provides reasonable penalties for OOD actions and does not lead to overly conservative policies.

Computational Efficiency. We evaluate the computational efficiency using two criteria: the run-time per epoch and the number of parameters. We compare our method with other uncertainty-based methods, including PBRL, SAC-N, and EDAC. Each method is tested on the same physical machine with Tesla P40 GPU. As shown in Table 3, PBRL needs a much longer time since the implementation does not apply parallel processing to the ensemble. In addition, EDAC takes more time than SAC-N to calculate the diversified gradient regularization, and BURL spends time sampling from the posterior distribution and OOD data. We remark that although BURL takes a bit longer time than EDAC or SAC-N (with N=10 or 20), it offers a significant improvement in parametric efficiency. In halfcheetah and walker2d tasks, BURL needs half of the parameters of EDAC or PBRL. In hopper environments, EDAC or SAC-N rely on a large number of networks, while BURL only needs about one-tenth or one-percent of their parameters.

Table 3: Computational costs of uncertainty-based methods.

Environments	Methods	Runtime (s/epoch)	Number of parameters
halfcheetah-m	PBRL	147.3	2.9M
	SAC-N (N=10)	19.4	2.9M
	EDAC	24.9	2.9M
	BURL	33.4	**1.5M**
walker2d-m	PBRL	163.7	2.9M
	SAC-N (N=20)	17.5	5.7M
	EDAC	24.5	2.9M
	BURL	33.8	**1.5M**
hopper-m	PBRL	150.7	2.8M
	SAC-N (N=500)	117.1	135.8M
	EDAC	25.5	13.7M
	BURL	33.1	**1.5M**

Ablation study. We perform ablation studies to discuss the effect of i) the number of ensembles, ii) the components of loss functions, iii) the number of Bayesian samples, iv) the number of OOD actions. In general, we find that more ensembles can lead to more stable performance and that BURL can obtain favorable performance with 5 or fewer ensembles. In terms of the loss functions, we observe that the repulsive regularization term plays a vital role in policy learning, while the effect of KL divergence which regularizes the posterior distribution is relatively minor. Meanwhile, BURL can obtain reliable uncertainty quantification with a few samples from the posterior distribution, and the repulsive diversification term can regularize the value function without many OOD actions. A detailed analysis is presented in §C.

6 Conclusion

In this paper, we propose a lightweight yet efficient uncertainty-based offline RL algorithm, named BURL. We leverage Bayesian neural networks to estimate the Bayesian posterior and uncertainties for Q-values by sampling from the posterior. Under the linear MDP assumptions, BURL recovers the provably efficient LCB penalty. By explicitly utilizing repulsive regularization, BURL obtains significant performance benefits with fewer ensembles. The performance and parametric efficiency have been validated on the D4RL benchmarks, including challenging AntMaze tasks. Future works will extend this idea to learn environmental dynamics models and explore the online fine-tuning capability, as well as unifying BNN, ensemble, and repulsive term from the theoretical perspective.

REFERENCES

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, volume 24, pp. 2312–2320, 2011.

Alekh Agarwal, Nan Jiang, Sham M. Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms, 2022.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 104–114. PMLR, 2020.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *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 *Advances in Neural Information Processing Systems*, 2021.

Javier Antorán, James Urquhart Allingham, and José Miguel Hernández-Lobato. Depth uncertainty in neural networks. In *NeurIPS*, 2020.

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhi-Hong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *International Conference on Learning Representations*, 2022.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pp. 1613–1622. PMLR, 2015.

David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Advances in Neural Information Processing Systems*, 34:4933–4946, 2021.

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.

Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning*, 2022.

Francesco D'Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. *Advances in Neural Information Processing Systems*, 34:3451–3465, 2021.

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline RL via supervised learning? In *ICLR*. OpenReview.net, 2022.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059. PMLR, 2016.

Seyed Kamyar Seyed Ghasemipour, Shixiang Shane Gu, and Ofir Nachum. Why so pessimistic? estimating uncertainties for offline rl through ensemb les, and why their independence matters. In *Advances in Neural Information Processing Systems*, 2022.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *NeurIPS*, pp. 1273–1286, 2021.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pp. 2137–2143. PMLR, 2020.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pp. 5084–5096. PMLR, 2021.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21810–21823, 2020.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *ICLR*. OpenReview.net, 2022.

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, pp. 11784–11794, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191, 2020.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 6131–6141. PMLR, 2021.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

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

Ziniu Li, Yingru Li, Yushun Zhang, Tong Zhang, and Zhi-Quan Luo. Hyperdqn: A randomized exploration method for deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in Neural Information Processing Systems*, 29, 2016.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. In *ICLR*. OpenReview.net, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Tim Pearce, Felix Leibfried, and Alexandra Brintrup. Uncertainty in neural networks: Approximately bayesian ensembling. In *International conference on artificial intelligence and statistics*, pp. 234–244. PMLR, 2020.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Lorena Qendro, Alexander Campbell, Pietro Liò, and Cecilia Mascolo. Early exit ensembles for uncertainty quantification. In *ML4H@NeurIPS*, volume 158 of *Proceedings of Machine Learning Research*, pp. 181–195. PMLR, 2021.

JP Royston. Expected normal order statistics(exact and approximate): algorithm as 177. *Applied Statistics*, 31(2):161–5, 1982.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Kerong Wang, Hanye Zhao, Xufang Luo, Kan Ren, Weinan Zhang, and Dongsheng Li. Bootstrapped transformer for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688. Citeseer, 2011.

Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems*, 33:6514–6527, 2020.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua M. Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. In *International Conference on Machine Learning*, volume 139, pp. 11319–11328, 2021.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 14129–14142, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2021.

A Theoretical Proof

A.1 Background of linear MDPs

Our theoretical derivations have close connections with linear MDPs and LSVI. The former is a widely used assumption Jin et al. (2021; 2020); Agarwal et al. (2022), while the latter is a classic method frequently used in the linear MDPs to compute the closed-form solution. In the following, we introduce the pessimistic value estimation in linear MDPs (Jin et al., 2020; Abbasi-Yadkori et al., 2011; Jin et al., 2021), where the transition dynamics and reward function take the following form for $\forall(s_{t+1}, a_t, s_t) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, as

$$\mathbb{P}_t(s_{t+1} \,|\, s_t, a_t) = \langle \varphi(s_{t+1}), \phi(s_t, a_t) \rangle, \quad r(s_t, a_t) = v^\top \phi(s_t, a_t), \tag{10}$$

where the feature embedding $\phi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^d$ is known. We further assume that the reward function $r : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ is bounded and the feature is bounded by $\|\phi\|_2 \leq 1$. We consider the settings of $\gamma = 1$ in the following. Then for any policy π, the state-action value function is also linear to ϕ, as

$$Q^\pi(s_t, a_t) = w^\top \phi(s_t, a_t). \tag{11}$$

In offline RL settings with a fixed dataset $\mathcal{D}_m = \{s_t^i, a_t^i, r_t^i, s_{t+1}^i\}_{i \in [m]}$, the parameter of the w can be solved via least-squares value iteration (LSVI) algorithm, as

$$\widehat{w}_t = \min_{w \in \mathbb{R}^d} \sum_{i=1}^m \big(\phi(s_t^i, a_t^i)^\top w - r(s_t^i, a_t^i) - V_{t+1}(s_{t+1}^i) \big)^2 \tag{12}$$

where V_{t+1} is the estimated value function in the $(t+1)$-th step. Following LSVI, the explicit solution to (12) takes the form of

$$\widehat{w}_t = \Lambda_t^{-1} \sum_{i=1}^m \phi(s_t^i, a_t^i) y_t^i, \quad \text{where } \Lambda_t = \sum_{i=1}^m \phi(s_t^i, a_t^i) \phi(s_t^i, a_t^i)^\top \tag{13}$$

is the feature covariance matrix of the state-action pairs in the offline dataset, and $y_t^i = r(s_t^i, a_t^i) + V_{t+1}(s_{t+1}^i)$ is the Bellman target in regression.

Lemma 1. (Proposition 2.3 in Jin et al. (2020)) For a linear MDP, for any policy π, there exist weights $\{w_h^\pi\}_{h \in [H]}$ such that for any $(s, a, h) \in \mathcal{S} \times \mathcal{A} \times [H]$, we have $Q_h^\pi(s, a) =< \phi(s, a), w_h^\pi >$.

We cite the proposition from Jin et al. (2020) here to demonstrate that Q-functions can be equivalent to $\phi(s, a)^\top w$. In our method, we use the hidden layers of the critic network to output the representation $\phi(s, a)$ and express w as the parameter of the last layer.

A.2 Proof of the LCB-term

To learn the posterior distribution of Q-function, we consider a Bayesian perspective of LSVI and use Bayesian linear regression to learn the posterior of w. Under Assumption 1, the following theorem establishes the method to approximate the LCB-penalty $\Gamma_t^{\rm lcb}(s_t, a_t)$.

Theorem (Theorem 1 restate). *Under Assumption 1, it holds for the standard deviation of the estimated posterior distribution $\mathbb{P}(\tilde{Q} \,|\, s, a, \mathcal{D}_m)$ follows*

$$\mathrm{Std}(\tilde{Q} \,|\, s_t, a_t, \mathcal{D}_m) = \big[\phi(s_t, a_t)^\top \Lambda_t^{-1} \phi(s_t, a_t) \big]^{1/2} := \Gamma_t^{\rm lcb}(s_t, a_t), \tag{14}$$

where $\Lambda_t = \sum_{i \in [m]} \phi(s_t^i, a_t^i) \phi(s_t^i, a_t^i)^\top + \lambda \cdot \mathbf{I}$, and $\Gamma_t^{\rm lcb}(s_t, a_t)$ is defined as the LCB-term.

Proof. It holds from Bayes rule that

$$\log q^*(w \,|\, \mathcal{D}_m) = \log \mathbb{P}(w \,|\, \mathcal{D}_m) = \log \mathbb{P}(w) + \log \mathbb{P}(\mathcal{D}_m \,|\, w) + \text{Const.}$$

By Assumption 1, the prior of w is the standard normal distribution, and we have

$$Q \,|\, s, a, w \sim \mathcal{N}(\phi(s, a)^\top w, 1).$$

Thus, it holds that

$$\log q^*(w \mid \mathcal{D}_m) = -\frac{\lambda}{2}\|w\|^2 - \sum_{i\in[m]} \frac{1}{2}\|\phi(s_t^i, a_t^i)^\top - y_t^i\|^2 + \text{Const}$$
$$= -\frac{1}{2}(w-\mu_t)^\top \Lambda_t^{-1}(w-\mu_t) + \text{Const}.$$

where we have

$$\mu_t = \Lambda_t^{-1} \sum_{\tau\in[m]} \phi(s_t^i, a_t^i) y_t^i. \quad \Lambda_t = \sum\nolimits_{i\in[m]} \phi(s_t^i, a_t^i)\phi(s_t^i, a_t^i)^\top + \lambda \cdot \mathbf{I}$$

Thus, we obtain that $w \mid \mathcal{D}_m \sim \mathcal{N}(\mu_t, \Lambda_t^{-1})$. It then holds for any $(s, a) \in \mathcal{S} \times \mathcal{A}$ that

$$\text{Var}(\tilde{Q} \mid S = s, A = a) = \text{Var}(\phi(s,a)^\top w) = \phi(s,a)^\top \Lambda_t^{-1} \phi(s,a),$$

which concludes the proof of Theorem 1. $\square$

Based on the LCB term $\Gamma_t^{\text{lcb}}(s,a)$, the pessimistic value function $Q_{\text{lcb}}(s,a) = Q(s,a) - \Gamma_t^{\text{lcb}}(s,a)$ with uncertainty penalty is known to be information-theoretically optimal in linear MDPs with finite horizon (Jin et al., 2021). For deep offline RL, we approximate such an LCB term Γ_t^{lcb} by estimating the posterior distribution of the Q-function via Bayesian neural networks.

B Implementation Detail

Experimental setups and baselines. All the experiments are conducted on D4RL with the 'v2' datasets and 5 random seeds. We run EDAC and BURL for 3M gradient steps and report the normalized scores of each policy. In Gym Mujoco tasks, the policy is evaluated every 1000 steps, and the evaluation contains 1 episode. In AntMaze tasks, the policy is evaluated every 20000 steps, and the return averages over 100 episodes.

For baselines, we cite the reported results of TD3-BC, PBRL, and SAC-N from their own publications (Fujimoto & Gu, 2021; Bai et al., 2022; An et al., 2021). For IQL[1] and EDAC[2], we reproduce the results with their official implementations. For ATAC[3], we run the official codes for the results of the 'expert' datasets and cite other results from the original paper Cheng et al. (2022). For CQL and MOPO, we cite the reported scores in (Bai et al., 2022).

In Table 2, we adopt the reported performance of several baselines from (Emmons et al., 2022). For IQL, we run its official implementations and report its performance on 'v2' dataset. Since the official codes of EDAC or SAC-N cannot work in AntMaze tasks, we do not include them in comparison. Inspired by (Ghasemipour et al., 2022), BURL utilizes independent targets in AntMaze tasks and achieves competitive performance with a few networks. Considering the sparse rewards in Antaze tasks, we adopt the same way as prior works (Kumar et al., 2020; Ghasemipour et al., 2022) and transform the rewards in the offline datasets by $4(r - 0.5)$.

Hyper-parameters. Most hyper-parameters of BURL are the same as those of EDAC. In our experiments, we use different loss weights for the Gym Mujoco tasks and AntMaze navigation tasks. In addition, we add weight decay and layer norm in some tasks to prevent overfitting or divergence. The hyper-parameter settings are shown in Table 4. 'T/F' for Layernorm means whether using Layernorm layers.

C Ablation Study

In this section, we perform the ablation study on walker2d-medium and hopper-medium-replay environments. Each experiment runs with 3 random seeds.

[1]https://github.com/ikostrikov/implicit_q_learning
[2]https://github.com/snu-mllab/EDAC
[3]https://github.com/microsoft/ATAC

Table 4: Hyper-paramters of BURL

Environments	number of ensembles M	η_q	η_{ood}	L_2 decay	Layernorm
halfcheetah-random	2	50	1	0.01	F
halfcheetah-medium	3	50	1	0.05	F
halfcheetah-medium-replay	3	50	1	0.05	F
halfcheetah-medium-expert	5	50	5	0	T
halfcheetah-expert	5	10	1	0	T
walker2d-random	5	50	5	0.5	F
walker2d-medium	5	5	3	0.003	T
walker2d-medium-replay	4	10	5	0.01	T
walker2d-medium-expert	5	20	5	0.001	T
walker2d-expert	3	10	5	0	T
hopper-random	5	10	1	0	F
hopper-medium	5	5	3	0	T
hopper-medium-replay	5	50	3	0	T
hopper-medium-expert	5	1	3	0	T
hopper-expert	5	1	3	0	T
antmaze-umaze	5	1	5	0	T
antmaze-umaze-diverse	10	1	10	0.001	F
antmaze-medium-diverse	10	1	10	0	T
antmaze-medium-play	10	1	10	0	T

Ensemble number. We evaluate our algorithm with different numbers of Q-networks $M \in \{2, 3, 4, 5, 6\}$ and present the experimental results in Fig. 4. We find that by using approximate Bayesian posterior and repulsive regularization, BURL can learn effective policies with $M \approx 5$. We also present the minimum number of ensembles required for other tasks in Fig. 2. It is obvious that BURL needs fewer ensembles than EDAC, thus improving the parametric efficiency. Another suggestion from Fig. 4 is that more ensembles can provide more stable performance, which is consistent with our intuition that more ensembles can generate more samples and better approximation.

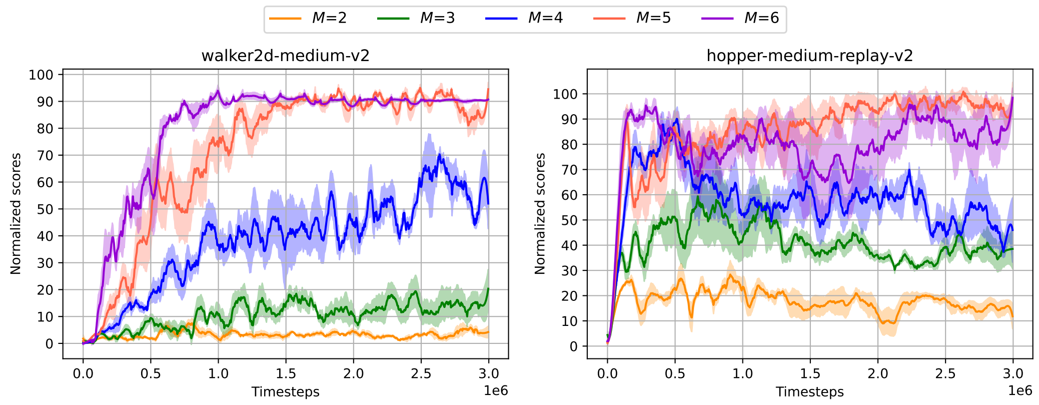

Figure 4: Ablation on the number of ensembles. The shaded area indicates the variance. BURL can exhibit superior and stable performance with 5 or 6 ensembles.

Loss function components. As shown in Eq. (8), the total loss consists of the TD error, the KL divergence, and the repulsive regularization term that maximizes the diversity of Q-samples for the OOD data. We name these components 'qfs loss', 'ood loss', and 'kl loss'. To illustrate the effects of these components on the final results, we perform an ablation study with three settings: only 'qfs' loss, 'qfs + ood' loss, 'qfs + ood + kl' loss. As illustrated in Fig. 5, 'kl loss' is less important than the other two components. The repulsive term ('ood' loss) is crucial for the walker2d-medium task.

To further explore the role of OOD diversity loss on our performance, we add comparisons on 9 tasks in Gym-Mujoco environments and present the results in Table 5. In hopper-medium-replay, halfcheetah-medium, and halfcheetah-medium-replay tasks, optimizing ensemble BNNs without OOD diversity regularization is sufficient to obtain competitive performance. This result is consistent with Fig. 5, where OOD loss and KL divergence have only a small impact on performance in the hopper-medium-replay task. In the other tasks, the OOD diversity regularization term is necessary for favorable performance. Therefore, we suggest that the OOD diversity term is useful for our method.

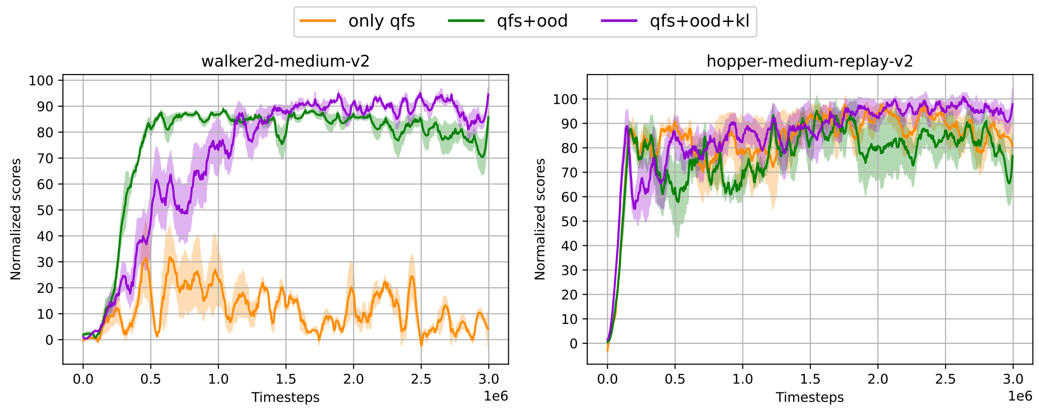

Figure 5: Ablation study on the loss functions components. The shaded area indicates the variance. Only optimizing the TD error cannot learn effective policies in the walker2d-medium task, while showing inferior performance to optimizing the total loss in the hopper-medium-replay task.

Table 5: Ablations on optimizing the OOD diversity loss.

Environments	w/o OOD term	with OOD term
halfcheetah-m	70.6	69.2 ± 2.7
halfcheetah-m-r	66.6	66.7 ± 2.0
halfcheetah-m-e	11.8	104.2 ± 2.4
walker2d-m	-0.1	95.1 ± 2.7
walker2d-m-r	17.7	96.1 ± 2.3
walker2d-m-e	2.1	112.4 ± 1.1
hopper-m	1.0	102.0 ± 1.2
hopper-m-r	103.2	102.9 ± 1.1
hopper-m-e	15.5	112.3 ± 1.8

Bayesian sampling. Our method samples multiple Q-values from the Bayesian posterior and ensemble members to estimate the LCB of the Q-function. Intuitively, more samples can provide a better estimation of the LCB. We use n to denote the number of samples from each ensemble member and compare the performance in two tasks. The results are displayed in Fig. 6. In general, we find that BURL obtains reasonable performance even with a small number of samples (e.g., $n = 2$). We speculate that variational methods can quickly obtain a posterior estimate. Meanwhile, the repulsive regularization diversifies the different samples, which drives the posterior samples widely distributed in the posterior and makes the LCB estimation easier.

Number of OOD actions. The diversification of OOD data is implemented by OOD sampling and explicitly maximizing the uncertainty of Q-values for OOD actions. In the OOD sampling process, K actions are sampled from the learned policy. We perform experiments with $K \in \{1, 3, 5, 7, 10\}$ to analyze the sensitivity of our algorithm to the number of OOD actions. The results in Fig. 7 illustrate that the number of OOD samples does not significantly influence the performance in the walker-medium task. Nevertheless, in hopper-medium-replay task, the value distribution requires more OOD actions to diversify the posterior samples.

Comparison with dropout ensembles. We also compare BURL with dropout ensembles and present the results in Fig. 8. We evaluate dropout ensembles with the same number of ensembles as BURL and equip them with gradient diversification from EDAC or repulsive regularization from BURL. However, we find that dropout ensembles cannot learn effective policies in all settings. We

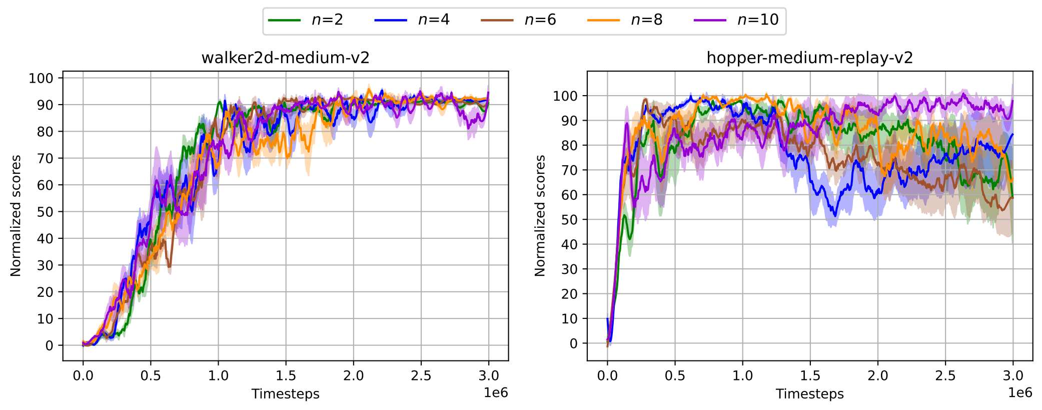

Figure 6: Ablation on the number of Bayesian samples. The shaded area indicates the variance. In the walker2d-medium environment, a small number of samples are enough, while more samples are needed to obtain stable performance in the hopper-medium-replay environment.

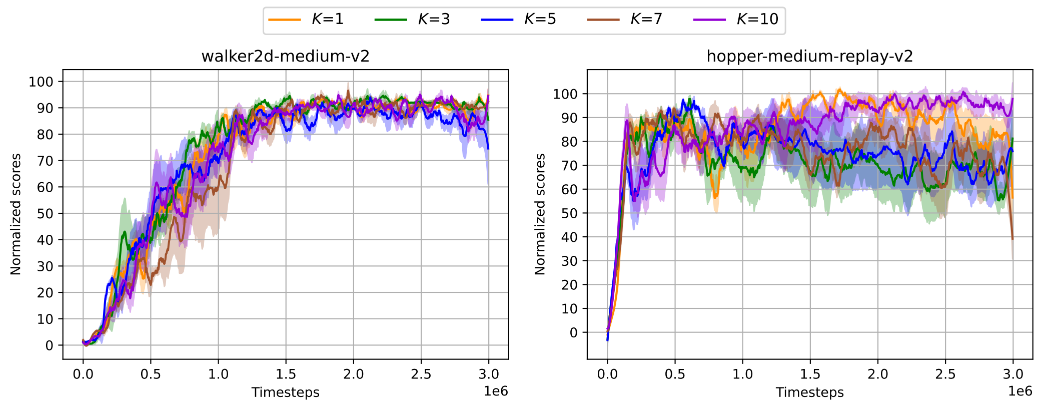

Figure 7: Ablation on the number of OOD actions. The shaded area indicates the variance. We tend to sample $K = 10$ OOD actions in both environments to keep strong performance.

speculate that it is difficult for dropout ensembles to provide reliable uncertainty quantification and value estimation with a small number of ensembles (Wu et al., 2021).

D Additional Experiments

D.1 Comparisons with more offline RL methods

We additionally compare our methods with more offline RL methods and consider two algorithms: model-free algorithm UWAC Wu et al. (2021) and conservative model-based algorithm COMBO Yu et al. (2021). UWAC adopts Monte Carlo dropout to estimate the uncertainty for the detection of OOD state-action pairs, which are then down-weighted in the training objectives, and requires policy constraints Kumar et al. (2019) for regularization. COMBO obtains a conservative estimate of the value function for OOD state-action tuples by combining offline model-based methods and value regularization, without explicit uncertainty estimation. Since the reported results of UWAC rely on the 'v0' datasets, we choose the reimplementation results on the 'v2' datasets from PBRL to evaluate these methods in the same setting. We cite the results of COMBO from the original paper. The normalized scores are presented in Table 6. Compared with UWAC and COMBO, BURL is mainly an uncertainty-based offline RL algorithm and achieves better performance.

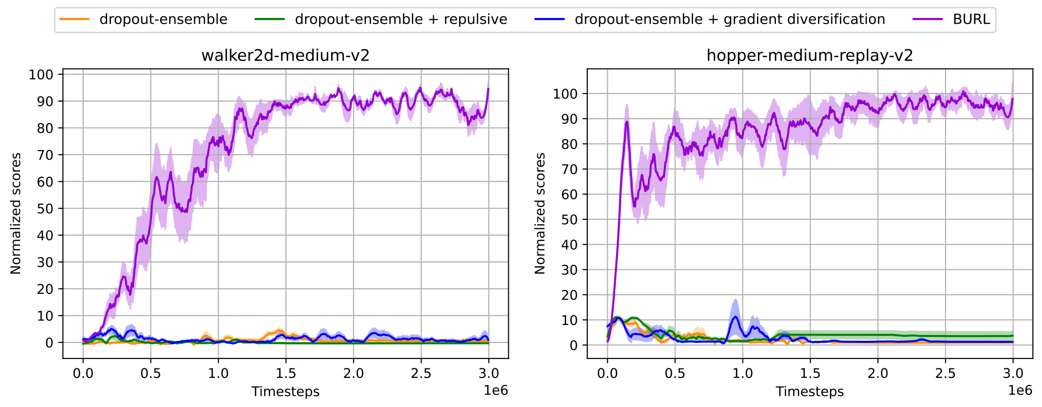

Figure 8: Comparisons with dropout ensembles. We evaluate dropout ensembles and BURL with the same number of ensembles ($M = 5$).

Table 6: Comparions with UWAC and COMBO.

Environments	UWAC	COMBO	BURL
halfcheetah-r	2.3 ± 0.0	**38.8** $\pm$ **3.7**	30.9 ± 1.1
halfcheetah-m	42.2 ± 0.4	54.2 ± 1.5	**69.2** $\pm$ **2.7**
halfcheetah-m-r	35.9 ± 3.7	55.1 ± 1.0	**66.7** $\pm$ **2.0**
halfcheetah-m-e	42.7 ± 0.3	90.0 ± 5.6	**104.2** $\pm$ **2.4**
halfcheetah-e	92.9 ± 0.6	-	**107.1** $\pm$ **2.6**
walker2d-r	2.0 ± 0.4	7.0 ± 3.6	**7.4** $\pm$ **5.7**
walker2d-m	75.4 ± 3.0	81.9 ± 2.8	**95.1** $\pm$ **2.7**
walker2d-m-r	23.6 ± 6.9	56.0 ± 8.6	**96.1** $\pm$ **2.3**
walker2d-m-e	96.5 ± 9.1	103.3 ± 5.6	**112.4** $\pm$ **1.1**
walker2d-e	108.4 ± 0.4	-	**112.5** $\pm$ **0.5**
hopper-r	2.7 ± 0.3	17.9 ± 1.4	**27.9** $\pm$ **7.8**
hopper-m	50.9 ± 4.4	97.2 ± 2.2	**102.0** $\pm$ **1.2**
hopper-m-r	25.3 ± 1.7	89.5 ± 1.8	**102.9** $\pm$ **1.1**
hopper-m-e	44.9 ± 8.1	111.1 ± 2.9	**112.3** $\pm$ **1.8**
hopper-e	110.5 ± 0.5	-	**111.4** $\pm$ **1.2**

D.2 Different ways to obtain OOD actions

In BURL, we introduce a repulsive term to improve the diversity among samples from the posterior and ensembles. Specifically, we sample OOD actions from the learned policy, which is the same as previous work PBRL. In general, we can roughly regard any actions that are far from behavior policies as OOD actions. Since the behavior policy is unknown, we can only generate OOD actions by sampling from some distributions that are different from the behavior policy. We consider two ways to generate OOD actions: randomly sampling from the action space or from the learned policy.

In our work, we sample OOD actions from the learned policy. As shown in Eq. 1, the Q-value of the state-action pair $Q(s, a)$ is updated by regressing the target $\mathbb{E}[r + \gamma Q(s', a')]$, where $Q(s', a')$ is often inaccurate in the offline setting since a' can be OOD actions chosen by the learned policy. As a result, we suggest that sampling OOD actions from the learned policy leads to better pessimism for the neighbor field of the learned policy. Instead, randomly sampling from the action space is insufficient to enforce pessimism.

In this part, we compare the performance of these two ways on Gym-Mujoco datasets, and the normalized scores are shown in Table 7. The empirical results indicate that sampling from the current policy is reasonable and more effective, while random sampling OOD actions causes overestimation and final divergence of the value function in halfcheetah and walker tasks. Moreover, we remark that randomly sampling from the action space exhibits a large distance to sampling from the learned

Table 7: Performance of different ways to obtain OOD actions.

Environments	**randomly sampling from the action space**	**sampling from the learned policy**
halfcheetah-m	0.5	69.2 $\pm$ 2.7
halfcheetah-m-r	1.1	66.7 $\pm$ 2.0
halfcheetah-m-e	-1.9	104.2 $\pm$ 2.4
walker2d-m	-0.3	95.1 $\pm$ 2.7
walker2d-m-r	6.9	96.1 $\pm$ 2.3
walker2d-m-e	-0.3	112.4 $\pm$ 1.1
hopper-m	30.5	102.0 $\pm$ 1.2
hopper-m-r	99.9	102.9 $\pm$ 1.1
hopper-m-e	14.1	112.3 $\pm$ 1.8

policies in most tasks, while showing similar performance in the hopper-medium-replay task. We speculate that our OOD repulsive term plays an important role in most tasks, while only optimizing loss function for BNNs is enough for competitive performance in the hopper-medium-replay task. This result is consistent with Fig. 5.

D.3 Is Bayesian inference module necessary?

As presented in Eq. (8), the critic network is optimized by minimizing the TD error and KL divergence and maximizing the OOD repulsive term. In Appendices C and D.2, we have demonstrated the importance of the OOD repulsive term and sampling OOD actions from the learned policy. In this section, we discuss the necessity of Bayesian neural networks and whether the OOD repulsive term is useful for other methods. The difference between ensemble BNNs and other ensemble models (e.g. PBRL or EDAC) lies in the last layer of the critic network.

We first illustrate that other methods cannot perform well with fewer ensembles, but may benefit from the OOD repulsive term. In practice, we choose PBRL as the basis and add comparisons with 1) PBRL with fewer ensembles and 2) PBRL with fewer ensembles and OOD diversity loss. Experimental results are shown in Table 8. The first comparison illustrates that PBRL with fewer ensembles cannot achieve the same performance as BURL. And the second comparison shows that PBRL may benefit from the OOD repulsive term in several tasks, but is still inferior to BURL. This result shows that the Bayesian inference module is necessary, and the OOD repulsive term may be useful for other ensemble-based methods.

Table 8: Extended comparisons with PBRL with fewer ensembles and its combination with the OOD repulsive term. Each method uses the same number of ensembles M, which is presented in brackets.

Environments	**PBRL with fewer ensembles**	**PBRL (fewer ensembles) + OOD**	**BURL**
halfcheetah-m	59.8 (M=3)	60.3 (M=3)	**69.2** $\pm$ **2.7** (M=3)
halfcheetah-m-r	46.7 (M=3)	52.3 (M=3)	**66.7** $\pm$ **2.0** (M=3)
halfcheetah-m-e	93.3 (M=5)	93.6 (M=5)	**104.2** $\pm$ **2.4** (M=5)
walker2d-m	89.2 (M=5)	91.1 (M=5)	**95.1** $\pm$ **2.7** (M=5)
walker2d-m-r	90.9 (M=4)	84.5 (M=4)	**96.1** $\pm$ **2.3** (M=4)
walker2d-m-e	109.6 (M=5)	105.6 (M=5)	**112.4** $\pm$ **1.1** (M=5)
hopper-m	25.2 (M=5)	102.2 (M=5)	**102.0** $\pm$ **1.2** (M=5)
hopper-m-r	98.7 (M=5)	101.1 (M=5)	**102.9** $\pm$ **1.1** (M=5)
hopper-m-e	110.3 (M=5)	110.8 (M=5)	**112.3** $\pm$ **1.8** (M=5)

To further explore the importance of BNNs, we remove the OOD repulsive term from BURL and only use ensemble BNNs. We compare BURL (only ensemble BNNs) with PBRL and EDAC, whose results are directly taken from their papers. We use the same number of ensembles as PBRL or EDAC in most tasks, and use more ensembles in several walker2d or hopper tasks. We remark that although more ensembles are required in several tasks, our method does not need pessimistic updates on OOD data in PBRL or gradient diversification in EDAC. The results in Table 9 validate that the Bayesian posterior for uncertainty quantification, which is one of our main contributions, is sufficient for favorable performance. It also implies that the OOD repulsive term, which is another contribution of our method, can decrease the number of ensembles required and lead to better computational efficiency.

Table 9: Extended comparisons for BURL (only ensemble BNNs) with the results reported in PBRL and EDAC. The numbers of ensembles required M are presented in brackets.

Environments	**PBRL**	**EDAC**	**BURL (only ensemble BNNs)**
halfcheetah-m	58.2 $\pm$ 1.5 (M=10)	65.9$\pm$0.6 (M=10)	68.1 (M=10)
halfcheetah-m-r	49.5$\pm$0.8 (M=10)	61.3$\pm$1.9 (M=10)	62.8 (M=10)
halfcheetah-m-e	93.6$\pm$2.3 (M=10)	106.3$\pm$1.9 (M=10)	105.6 (M=10)
walker2d-m	90.3$\pm$1.2 (M=10)	92.5$\pm$0.8 (M=10)	93.5 (M=20)
walker2d-m-r	86.2$\pm$3.4 (M=10)	87.0$\pm$2.3 (M=10)	96.6 (M=10)
walker2d-m-e	109.8$\pm$0.2 (M=10)	114.7$\pm$0.9 (M=10)	113.7 (M=10)
hopper-m	81.6$\pm$14.5 (M=10)	101.6$\pm$0.6 (M=50)	100.9 (M=50)
hopper-m-r	100.7$\pm$0.4 (M=10)	101.0$\pm$0.5 (M=50)	102.9 (M=10)
hopper-m-e	111.2$\pm$0.7 (M=10)	110.7$\pm$0.1 (M=50)	111.4 (M=50)

E Statistical Uncertainty

As pointed out in (Agarwal et al., 2021), aggregate measures such as *mean* are not sufficiently reliable due to the effect of outlier scores. We need more efficient and robust evaluation principles to characterize the statistical uncertainty. In this section, we adopt two evaluation methods from (Agarwal et al., 2021): i) aggregate metrics, which contain median, interquartile mean (IQM), optimality gap, and mean. IQM computes the mean score of the middle 50% runs with 95% confidence intervals (CIs), and the optimality gap measures the number of runs where the algorithms fail to achieve a minimum score beyond which improvements are not very important. ii) performance profiles which use score distributions to express performance variability. We compare our algorithm with TD3-BC, IQL, PBRL, and EDAC and conduct experiments on Gym Mujoco tasks with 5 random seeds.

The comparisons are presented in Fig. 9 and Fig. 10. In Fig. 9, higher mean, median, and IQM scores, and lower optimality gap are expected, and our algorithm outperforms other baselines. In Fig. 10, our algorithm also exhibits more efficient and robust performance.

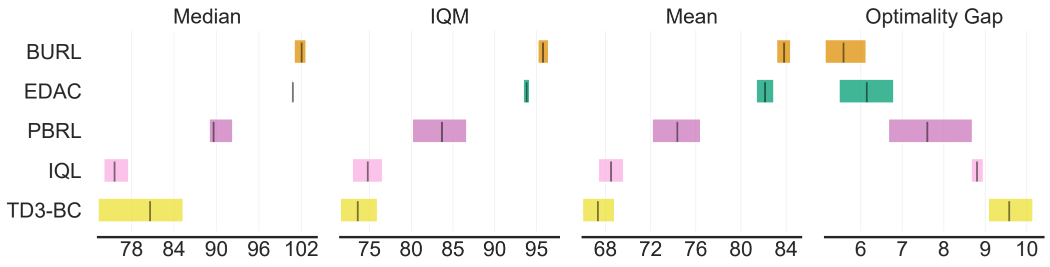

Figure 9: Aggregate metrics on D4RL with 95% CIs based on Gym Mujoco tasks and 5 random seeds for each task. BURL shows higher median, mean, and IQM, and lower optimality gap than other methods.

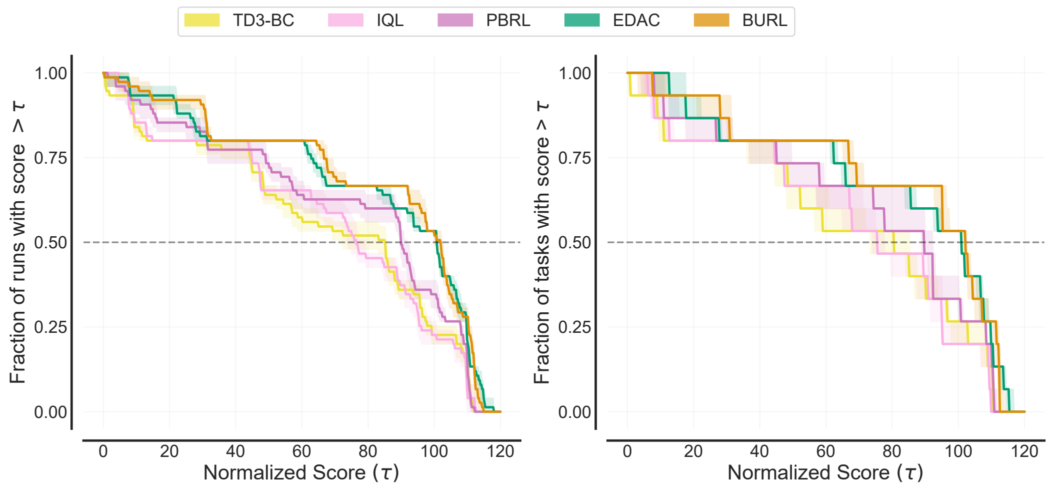

Figure 10: Comparisons on performance profiles based on score distributions (left) and average score distributions (right). The shaded area indicates pointwise 95% confidence bands. BURL achieves the best performance.