Model-based Offline Reinforcement Learning with Count-based Conservatism

Byeongchan Kim [1] **Min-hwan Oh** [1]

Abstract

In this paper, we propose a model-based offline reinforcement learning method that integrates count-based conservatism, named `Count-MORL`. Our method utilizes the count estimates of state-action pairs to quantify model estimation error, marking the first algorithm of demonstrating the efficacy of count-based conservatism in model-based offline deep RL to the best of our knowledge. For our proposed method, we first show that the estimation error is inversely proportional to the frequency of state-action pairs. Secondly, we demonstrate that the learned policy under the count-based conservative model offers near-optimality performance guarantees. Through extensive numerical experiments, we validate that `Count-MORL` with hash code implementation significantly outperforms existing offline RL algorithms on the D4RL benchmark datasets. The code is accessible at https://github.com/oh-lab/Count-MORL.

1. Introduction

Reinforcement Learning (RL) provides a paradigm in which an agent learns sequential actions within uncertain environments, all while aiming to maximize cumulative rewards. The empirical effectiveness of RL has been validated across diverse domains, underlining its capability to learn complex tasks (Mnih et al., 2015; Silver et al., 2016; 2017; Fawzi et al., 2022). A defining feature of RL, which distinguishes it from other machine learning paradigms, is its inherent necessity for active information acquisition, in which the agent collects data through firsthand experimentation with the environment. This subfield of RL, characterized by direct interaction, is commonly referred to as *online* RL.

Nevertheless, this approach with direct experimentation is often infeasible or potentially unsafe in many real-world applications, including but not limited to robotics (Gu et al., 2017), autonomous driving (Kiran et al., 2021), and healthcare (Yu et al., 2021a). Furthermore, even when direct intervention is viable, each interaction between the agent and the environment often incurs cost. Thus, if conditions permit, it would be advantageous for the agent to learn from readily available offline datasets, thereby negating the necessity for direct experimentation.

Offline RL, also referred to as batch RL, is a subfield of the RL framework that relies solely on previously acquired static datasets. In this framework, the agent is given a collection of experiences that a behavior policy gathered and subsequently utilizes for policy learning without any additional interactions with the environment. Despite its advantages of learning from an offline dataset, offline RL also presents unique challenges (Levine et al., 2020). Among these, striking a balance between enhancing generalization capabilities and circumventing undesired out-of-distribution (OOD) behaviors remains prominent. The crux of finding such a balance pivots around managing *distributional shift*. During policy evaluation, the application of Bellman updates to value functions involves querying the values of OOD state-action pairs. This can potentially lead to an accumulation of extrapolation errors. The issue becomes even more intricate due to the widespread use of high-capacity function approximators, such as neural networks, which further complicates its complexity.

To address this issue with OOD actions, various offline RL algorithms adapt conservatism in estimated values by inducing some form of pessimism. These methods include both model-free (Kumar et al., 2020; Shi et al., 2022) and model-based (Yu et al., 2020; Kidambi et al., 2020; Rafailov et al., 2021; Lu et al., 2022; Rigter et al., 2022) approaches. Recent model-based offline RL methods aim to harness conservative value estimates by using analytical performance bounds (Yu et al., 2020; Kidambi et al., 2020; Rafailov et al., 2021). These methods adjust the estimated MDP model learned from the offline dataset to elicit conservative behavior by imposing a penalty on the policy or rewards when the policy visits states where the estimated model's accuracy is likely low. If the learned policy takes actions in states where the accuracy of model prediction is high, it is plausible that the estimated value of the policy is likely to retain a high level of accuracy. Specifically, many existing model-based offline RL algorithms incorporate conservatism by estimating

[1]Seoul National University, Seoul, South Korea. Correspondence to: Min-hwan Oh <minoh@snu.ac.kr>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

the model uncertainty and penalizing the reward function proportional to the estimated uncertainty (Yu et al., 2020; Kidambi et al., 2020; Rafailov et al., 2021).

A recent work (Lu et al., 2022) investigates a variety of estimated model uncertainty, showing that a different choice of model uncertainty can significantly affect performances. However, existing algorithms (Yu et al., 2020; Rafailov et al., 2021) use the total variation distance between the estimated and true models as a penalty in theory, but such a distance is difficult to compute in practice. Because of unreliable estimates of the model uncertainty, other approaches incorporate conservatism by regularizing the value function (Yu et al., 2021b) or modifying the estimated transition dynamics in an adversarial manner (Rigter et al., 2022) without model uncertainty quantification. Hence, the questions of which choice of model uncertainty should be used and what conservatism is more suitable in practice remain open.

In this work, we propose a new method for model-based offline RL, Count-based Conservatism for Model-based Offline RL (`Count-MORL`). Our proposed method utilizes the estimated frequency (counts) of state-action pairs in the offline dataset to quantify the model estimation error. The reward function is penalized by the model estimation error inversely proportional to the frequency of state-action pairs. Using this count-based penalized reward, we construct the count-based conservative MDP model and learn a policy using this model. However, when state and action spaces are not discrete, i.e., when state and action spaces are represented by high-dimensional features or features that take continuous values, it can be difficult to enumerate and quantify the exact counts of state-action pairs. To efficiently implement our proposed method and address the aforementioned issue, we utilize hash code heuristics to approximate the frequency of state-action pairs (Tang et al., 2017), previously used in online RL but has not been utilized for offline RL. While we present our method with this hash code version of count estimation for clear and easy exposition, other count estimations (Bellemare et al., 2016; Ostrovski et al., 2017; Fu et al., 2017; Martin et al., 2017; Machado et al., 2020) can be utilized in our proposed method. Another salient feature of our proposed method is the incorporation of uncertainty in count estimation, which is shown to affect the performance of the proposed method. Hence, a suitable consideration of the count uncertainty can further enhance performance. We show in our extensive numerical evaluations that our proposed count-based conservatism derives superior performances, consistently outperforming the existing methods in offline RL.

Our main contributions are summarized as follows:

- We propose a new model-based offline RL method that incorporates count-based conservatism, named `Count-MORL`, where we utilize the count estimates of state-action pairs to quantify the model estimation error. To our best knowledge, this work is the first to show the efficacy of count-based conservatism in model-based offline deep RL.
- We provide two theoretical analyses established for our proposed method. First, the estimation error is bounded, inversely proportional to the frequency of state-action pairs, which can be extended to consider count approximation, not just exact counts. Second, the learned policy under the count-based conservative model has a performance guarantee on near-optimality that depends on this estimation error and the count approximation.
- The numerical experiments show that `Count-MORL` with hash code implementation significantly outperforms the existing offline RL algorithms in the D4RL benchmark datasets. The results support that count-based conservatism is both efficient and practical for model-based offline RL.

2. Related Work

Offline RL addresses the problem of learning policies from a logged static dataset. Model-free offline algorithms do not require an estimated model and mainly belong to one of the three categories: regularizing the learned policy to be close to the behavior policy (Fujimoto et al., 2019; Wu et al., 2019; Liu et al., 2020; Siegel et al., 2020; Fujimoto & Gu, 2021; Kostrikov et al., 2022), quantifying the uncertainty with ensemble techniques to obtain a robust value function (Agarwal et al., 2020b; An et al., 2021; Kumar et al., 2019), and adapting conservatism to a value function (Kumar et al., 2020; Xie et al., 2021). In contrast, model-based offline algorithms use an estimated model based on a fixed dataset to query model outputs for state transition and rewards so that algorithms can use them for planning to improve a policy. However, training a policy using an inaccurately estimated model can be harmful (Janner et al., 2019). Such potential risk can be further exacerbated by distributional shifts. Hence, it is important to *correct* the insufficiently learned portion of estimated models. Many existing model-based approaches (Yu et al., 2020; Kidambi et al., 2020; Matsushima et al., 2021; Swazinna et al., 2021; Yu et al., 2021b; Argenson & Dulac-Arnold, 2021; Lee et al., 2021; Hishinuma & Senda, 2021; Rafailov et al., 2021; Rigter et al., 2022) address this challenge with various techniques, which regularizes the value function without uncertainty estimation (Yu et al., 2021b; Rigter et al., 2022).

A line of the model-based offline RL literature, most related to our paper, includes the works that utilize penalization on the estimated model (Yu et al., 2020; Kidambi et al., 2020; Rafailov et al., 2021). MOPO (Yu et al., 2020) pe-

nalizes the reward function with the estimation error between the true and estimated models based on the maximum aleatoric uncertainty. MOReL (Kidambi et al., 2020) constructs a pessimistic model using the unknown state-action detector based on a threshold of model uncertainty. LOMPO (Rafailov et al., 2021) handles image data with latent dynamics models and model uncertainty quantification based on the variance of log-likelihoods. While new methods are being proposed for model-based offline RL, it is still unclear as to concretely what and how estimation error is used for practice. Often, there appears to be a gap between what is proposed in theory and what is used in practice. Our work aims to address this issue with a concrete uncertainty quantification using count estimation.

Count-based uncertainty quantification is widely used in online RL. Count-based exploration methods (Bellemare et al., 2016; Ostrovski et al., 2017; Fu et al., 2017; Martin et al., 2017; Tang et al., 2017; Machado et al., 2020) are based on estimating the trial counts of state-action pairs and incorporating this count estimate into a bonus reward. However, in offline RL, a count-based approach has been relatively under-utilized. To our knowledge, the only work (Hong et al., 2023), which uses count-based conservatism for model-free offline RL, has focused on updating Q-functions by incorporating confidence levels derived from inverse state visitations. Compared to Hong et al. (2023), our proposed method has several distinct features. First, in model-based offline deep RL, we utilize the count estimation of state-action pairs to quantify the model estimation error and construct the count-based conservatism model using this count estimation. Second, unlike Hong et al. (2023) which requires the last layer of the neural network of Q-function to be linear for approximating state visitations, our method can utilize various count-based methods used in online RL. Additionally, for the first time, we address the issue of uncertainty in count estimation, which has been shown to enhance performance.

3. Preliminaries

3.1. Markov Decision Process

We consider a Markov decision process (MDP) specified by the tuples $M = (\mathcal{S}, \mathcal{A}, P, r, d_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to [-R_{\max}, R_{\max}]$ is the reward function, $d_0 \in \Delta(\mathcal{S})$ is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is a mapping from states to a probability distribution over actions. The value function $V^\pi_{P,r}(s) := \mathbb{E}_{\pi,P}\left[\sum_{t=0}^\infty \gamma^t r(s_t, a_t) | s_0 = s\right]$ represents the expected cumulative discounted reward of π under P and r when starting from state s. We denote the discounted state visitation distribution of π under P using $d^\pi_P(s) := (1-\gamma)\sum_{t=0}^\infty \gamma^t d^\pi_P(s_t = s)$, where $d^\pi_P(s_t = s)$ is the probability of reaching state s at a time-step t under π and P. Similarly, we denote the discounted state-action visitation distribution as $d^\pi_P(s, a) := d^\pi_P(s)\pi(a|s)$.

3.2. Offline RL

The goal of the RL agent in general is to obtain an optimal policy π^* that maximizes the expected cumulative discounted reward under d_0:

$$\max_\pi V^\pi_M := \mathbb{E}_{s\sim d_0}\left[V^\pi_{P,r}(s)\right] = \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim d^\pi_P}\left[r(s,a)\right]$$

In offline RL, a policy is learned by utilizing a static dataset without additional interactions with the environment. The agent is given an offline dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^n$ that consists of transition tuples from trajectories gathered in advance by some potentially unknown behavior policy π_β. Since $\mathcal{D}$ is typically a (possibly very limited) subset of the entire tuple space, finding the optimal policy using only this fixed dataset is not only challenging but also sometimes impossible especially when actions that belong to the optimal policy do not even exist in the given dataset. Therefore, the goal of offline RL is to construct an algorithm that can learn a policy $\hat{\pi}$ minimizing the sub-optimality gap $V^{\pi^*}_M - V^{\hat{\pi}}_M$ computed based on $\mathcal{D}$ while hoping that $\hat{\pi}$ is close to π^*.

3.3. Model-based Offline RL

A model-based approach to offline RL utilizes an estimated transition dynamics model that approximates the true transition dynamics model. Without loss of generality, we assume that the reward function $r(s, a)$ is known.[1] Let $\widehat{P}(\cdot|s, a)$ denote the maximum likelihood estimator (MLE) of the true state transition model $P^\star(\cdot|s, a)$, computed based on the dataset $\mathcal{D}_{s,a} := \{(s_i, a_i, s'_i)\}_{s_i=s,a_i=a}$ which is a subset of the entire offline dataset $\mathcal{D}$. We denote the number of samples in $\mathcal{D}_{s,a}$ as $n(s, a) := |\mathcal{D}_{s,a}|$. Then, we construct the estimated MDP $\widehat{M} := (\mathcal{S}, \mathcal{A}, \widehat{P}, r, d_0, \gamma)$ with the estimated transition model that approximates the true MDP $M^\star = (\mathcal{S}, \mathcal{A}, P^*, r, d_0, \gamma)$. The existing methods in model-based offline RL use the estimated model $\widehat{M}$ to query synthetic trajectories by simulating H-step rollouts starting from a given state observed in the offline dataset (Kidambi et al., 2020; Yu et al., 2020; 2021b; Rafailov et al., 2021; Rigter et al., 2022). Similar to a replay buffer utilized in off-policy RL, the synthetic trajectories are stored in a replay buffer $\mathcal{D}_{\text{model}}$. However, an inaccurately estimated model for unobserved state-action pairs in $\mathcal{D}$ can lead to poor performance of the learned policy (Janner et al., 2019).

[1] This is commonly assumed in the model-based RL literature (Yang & Wang, 2019; 2020; Zhou et al., 2021; Hwang & Oh, 2023) without loss of generality since learning $r(s, a)$ is considered much easier than learning $P(\cdot|s, a)$. If r is unknown, it can be replaced with estimated reward $\hat{r}$ learned together with $\widehat{P}$.

To address this challenge, one of the most widely used approaches in model-based offline RL is to incorporate conservatism by estimating the model uncertainty and constructing an uncertainty-penalized MDP with a penalized reward where a penalty is proportional to the estimated uncertainty (Yu et al., 2020; Kidambi et al., 2020; Rafailov et al., 2021; Lee et al., 2021). For example, an uncertainty-penalized MDP introduced in Yu et al. (2020) utilizes a penalized reward $\tilde{r}(s,a) := r(s,a) - \lambda u(s,a)$, where $u(s,a)$ denotes an admissible error estimator for the state-action pair (s,a). Based on this uncertainty-penalized reward $\tilde{r}$, a learned policy maximizes the expected cumulative discounted rewards:

$$\mathbb{E}_{s\sim d_0}\left[V^{\pi}_{\widehat{P},\tilde{r}}(s)\right] = \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim d^{\pi}_{\widehat{P}}}\left[r(s,a) - \lambda u(s,a)\right].$$

However, one of the main issues in this approach (Yu et al., 2020) as well as many other existing methods (Kidambi et al., 2020; Rafailov et al., 2021) is the unavailability of a readily usable uncertainty estimate $u(s,a)$, which quantifies the estimation error between an estimated MDP and the unknown true MDP.

4. Count-based Conservatism for Model-based Offline RL

We propose a model-based algorithm for offline RL as described in Section 4.2. The main idea of our proposed method is to compute the estimation error between the learned and true transition dynamics by quantifying the frequency of state-action pairs (or the frequency of the features of state-actions) in the offline dataset. We use the estimated frequency of the observed data to construct a *count-based* conservative MDP.

For our proposed method, we provide an estimation error bound based on the estimated frequency of state-action pairs in Theorem 1 (in Section 4.1). For each state-action pair, we penalize the reward function, where the penalty is inversely proportional to the estimated frequency. With this penalized reward function, we construct a count-based conservative MDP. In Theorem 2, we provide a guarantee of the sub-optimality gap between the optimal policy under the true MDP and the learned policy under the conservative MDP. In Section 4.2, we describe our algorithm that incorporates the aforementioned theoretical results to quantify the suitable conservatism and derives efficient learning in offline RL.

4.1. Theoretical Analysis

Suppose that we are given a hypothesis class of transition models $\mathcal{M} = \{P : \mathcal{S}\times\mathcal{A} \to \Delta(\mathcal{S})\}$. We assume realizability $P^\star \in \mathcal{M}$, that is, the true transition model exists in the hypothesis class (Agarwal et al., 2020a; Uehara & Sun, 2022). We denote a total variation distance between two distributions P_1 and P_2 as $\mathrm{TV}(P_1, P_2)$. Now, we begin by presenting the following theorem that establishes an upper bound of the total variation distance between the learned and true transition dynamics. The theorem is adapted from Theorem 21 in Agarwal et al. (2020a).

Theorem 1 (Estimation error of transition dynamics)**.** *Fix $\delta \in (0,1)$, assume $|\mathcal{M}| < \infty$ and $P^\star \in \mathcal{M}$. Given a state-action pair (s,a) is observed in $\mathcal{D}$ with $\mathcal{D}_{s,a} = \{(s_i, a_i, s_i')\}_{s_i=s, a_i=a}$ and $n(s,a) = |\mathcal{D}_{s,a}|$. Define the MLE of transition dynamics as*

$$\widehat{P}(\cdot \mid s,a) \in \operatorname*{arg\,max}_{P\in\mathcal{M}} \sum_{(s,a,s')\in\mathcal{D}_{s,a}} \log P(s' \mid s,a)$$

for a given (s,a). Then with probability at least $1-\delta$,

$$\mathrm{TV}\Big(\widehat{P}(\cdot \mid s,a), P^\star(\cdot \mid s,a)\Big) \le \sqrt{\frac{2\log(|\mathcal{M}|/\delta)}{n(s,a)}}.$$

Discussion of Theorem 1. Theorem 1 states that the estimation error between the estimated and true transition dynamics is smaller for the state-action pairs that are more frequently observed in the offline dataset $\mathcal{D}$, and provides the upper bound on the rate of the convergence in terms of the frequency, i.e., $O(1/\sqrt{n(s,a)})$. Such quantification of the model estimation error is critical for offline RL algorithms since the penalization on reward functions needed for learning conservative policies depends on the specification of the error bound. The key of this theorem is that the upper bound of the model estimation error can be determined based on the number of observations in the offline dataset, which we aim to estimate in the proposed algorithm (particularly when the environment is not tabular).

As an immediate corollary, we expand the estimation error for individual state-action pairs from the offline dataset $\mathcal{D}$ to the entire state-action space.

Corollary 1. *Given a state-action pair $(s,a) \in \mathcal{S}\times\mathcal{A}$, with probability at least $1-\delta$, the estimated transition dynamics $\widehat{P}$ satisfies the following inequality by Theorem 1:*

$$\mathrm{TV}\Big(\widehat{P}(\cdot \mid s,a), P^\star(\cdot \mid s,a)\Big) \le C^{\delta}_{\widehat{P}}(s,a),$$

where $C^{\delta}_{\widehat{P}}(s,a) := \min\left(1, \sqrt{\frac{2\log(|\mathcal{M}|/\delta)}{n(s,a)}}\right)$.

We define the estimation error bound based on the true count for the estimated transition dynamics $\widehat{P}$ as $C^{\delta}_{\widehat{P}} : \mathcal{S}\times\mathcal{A} \to [0,1]$. By the definition of total variation distance, the estimation error for all state-action pairs is bounded by 1. There are two cases in which the estimation error takes the value of 1. First, for unobserved state-action pairs in $\mathcal{D}$, we cannot estimate the transition dynamics or compute the estimation error. Second, when $n(s,a)$ is less than $\sqrt{2\log(|\mathcal{M}|/\delta)}$,

the estimation error becomes greater than 1. Excluding these cases, all estimation errors are inversely proportional to observation counts $n(s,a)$.

So far, we have assumed that we can compute exact counts for each observation $n(s,a)$ which is only applicable to tabular settings. However, for large state and action spaces, or continuous state-action pairs, computing the exact frequency of state-action pairs may be intractable. To this end, we introduce the notion of *approximate counts* $\hat{n}$ to approximate the true counts.

Definition 1 (Estimation error with approximate count)**.** *Define the approximate count as $\hat{n}: \mathcal{S}\times\mathcal{A}\to\mathbb{R}$, which approximates the true count n, and the estimation error bound based on the approximate count as $\widehat{C}^{\delta}_{\widehat{P}}: \mathcal{S}\times\mathcal{A}\to[0,1]$:*

$$\widehat{C}^{\delta}_{\widehat{P}}(s,a) := \min\left(1, \sqrt{\frac{2\log(|\mathcal{M}|/\delta)}{\hat{n}(s,a)}}\right).$$

We allow an approximation error incurred by $\hat{n}$. We define the maximal approximation error as follows.

Definition 2 (Approximation error of counts)**.** *Define the maximal approximation error between estimation error bounds based on the true count and approximate count over all state-action pairs as*

$$\epsilon := \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left| C^{\delta}_{\widehat{P}}(s,a) - \widehat{C}^{\delta}_{\widehat{P}}(s,a) \right|.$$

The following lemma shows the gap of returns between the estimated MDP $\widehat{M}$ and the true MDP $M^\star$ of any given policy π. This result is an adaptation of Lemma 4.1 based on the telescoping lemma in Yu et al. (2020) with the approximate count.

Lemma 1 (Value gap of estimated model)**.** *Suppose $M^\star$ and $\widehat{M}$ be two MDPs with the true transition dynamics $P^\star$ and estimated transition dynamics $\widehat{P}$, respectively. Given the estimation error bound based on the approximate count $\widehat{C}^{\delta}_{\widehat{P}}$ and the maximal approximation error ϵ. Then, with probability at least $1-\delta$, for any policy π,*

$$V^{\pi}_{\widehat{M}} - V^{\pi}_{M^\star} \le \frac{\gamma R_{\max}}{(1-\gamma)^2}\mathbb{E}_{(s,a)\sim d^{\pi}_{\widehat{P}}}\left[\widehat{C}^{\delta}_{\widehat{P}}(s,a)\right] + \frac{\gamma R_{\max}}{(1-\gamma)^2}\epsilon\,.$$

Discussion of Lemma 1. Lemma 1 states that the value gap between the estimated and true model under any policy π is bounded by the model estimation error under the visitation distribution $d^{\pi}_{\widehat{P}}$ and the count approximation error. Our approach of quantifying the model estimation error using the approximate count $\hat{n}(s,a)$ for all state-action pairs, instead of an admissible error estimator $u(s,a)$, provides a practical way of constructing a reward penalty concretely defined once an approximate count is provided.

Note that $\widehat{C}^{\delta}_{\widehat{P}}$ is a function of $\widehat{P}$ and π. In order to understand the implication of Lemma 1, first, consider a special case when $\pi=\pi_\beta$. Then, the policy takes actions frequently observed in $\mathcal{D}$ and visits states frequently observed in $\mathcal{D}$, so that the estimation error bound for state-action pairs will be generally small by Theorem 1. Hence, the expectation of $\widehat{C}^{\delta}_{\widehat{P}}$ under $d^{\pi_\beta}_{\widehat{P}}$ will be small. Conversely, if a policy takes actions (or visits states) less observed or unobserved in $\mathcal{D}$, then the estimation error bound for unobserved state-action pairs is likely to be large. Thus, the expectation of $\widehat{C}^{\delta}_{\widehat{P}}$ under $d^{\pi}_{\widehat{P}}$ tends to be large. Note that the maximal approximation error ϵ does not depend on $\widehat{P}$ or π, that is, even if the estimation of the model is accurate the error incurred by the approximation still exists. Hence, when the approximation error is sufficiently small, the value function will be efficiently learnable.

The key insight of Lemma 1 is that the value gap for the learned policy, which maximizes the value function with respect to $\widehat{M}$, can be large when the learned policy takes actions (or visits states) beyond the offline dataset $\mathcal{D}$, or when the count approximation is inaccurate. Using this insight, our proposed method augments count-based conservatism by incorporating the reward penalty based on count-based estimation error. Also, our method even addresses the uncertainty in count approximation.

Now, we define a count-based conservative MDP as follows.

Definition 3 (Count-based conservative MDP)**.** *Define the count-based conservative MDP $\widetilde{M} := (\mathcal{S},\mathcal{A},\widehat{P},\tilde{r},d_0,\gamma)$ with the estimated transition dynamics $\widehat{P}$ and count-based penalized reward $\tilde{r}(s,a) := r(s,a) - \frac{\gamma R_{\max}}{1-\gamma}\widehat{C}^{\delta}_{\widehat{P}}(s,a)$.*

We invoke the following corollary to demonstrate that $\widetilde{M}$ is conservative in that the value of a policy under $\widetilde{M}$ is generally not higher than its value under $M^\star$.

Corollary 2. *Given the true MDP $M^\star$ and the count-based conservative MDP $\widetilde{M} = (\mathcal{S},\mathcal{A},\widehat{P},\tilde{r},d_0,\gamma)$ as defined in Definition 3. Then, with probability at least $1-\delta$, for any policy π,*

$$V^{\pi}_{M^\star} \ge V^{\pi}_{\widetilde{M}} - \frac{\gamma R_{\max}}{(1-\gamma)^2}\epsilon\,.$$

Here, the conservatism error is given by a multiple of the maximal approximation error ϵ. That is, if ϵ is 0, then the value under $\widetilde{M}$ is pessimistic with high probability.

We define the learned policy $\hat{\pi}$ that maximizes the value function with respect to $\widetilde{M}$, that is, $\hat{\pi} := \arg\max_\pi V^{\pi}_{\widetilde{M}}$. Then, our main theorem (Theorem 2) provides a performance guarantee on the learned policy $\hat{\pi}$ under $M^\star$.

Theorem 2 (Sub-optimality gap)**.** *Given the estimation error bound $\widehat{C}^{\delta}_{\widehat{P}}$ based on the approximate count $\hat{n}$ and the maximal count approximation error ϵ, with probability at*

least $1-\delta$*, the learned policy* $\hat{\pi}$ *under the count-based conservative MDP* $\widetilde{M}$ *satisfies*

$$V^{\hat{\pi}}_{M^\star} \geq \sup_{\pi} \left\{ V^{\pi}_{M^\star} - \frac{2\gamma R_{\max}}{(1-\gamma)^2} \mathbb{E}_{(s,a)\sim d^{\pi}_{\widehat{P}}} \left[\widehat{C}^{\delta}_{\widehat{P}}(s,a) \right] \right\} - \frac{2\gamma R_{\max}}{(1-\gamma)^2}\epsilon\,.$$

In particular, for the optimal policy π^**,*

$$V^{\pi^*}_{M^\star} - V^{\hat{\pi}}_{M^\star} \leq \frac{2\gamma R_{\max}}{(1-\gamma)^2} \mathbb{E}_{(s,a)\sim d^{\pi^*}_{\widehat{P}}} \left[\widehat{C}^{\delta}_{\widehat{P}}(s,a) \right] + \frac{2\gamma R_{\max}}{(1-\gamma)^2}\epsilon\,.$$

Discussion of Theorem 2. Theorem 2 establishes that the value of the learned policy $\hat{\pi}$ under the true MDP $M^\star$ can be controlled in terms of the trade-off between the value under $M^\star$ and the expected estimation error under $\widehat{M}$ and the count approximation error. Theorem 2 has two important interpretations. First, the learned policy $\hat{\pi}$ will perform as least as well as the behavior policy π_β under $M^\star$ since the expectation of $\widehat{C}^{\delta}_{\widehat{P}}$ under $d^{\pi_\beta}_{\widehat{P}}$ is small. Second, $\hat{\pi}$ aims to find the optimal balance between the value under $M^\star$ and the expectation of $\widehat{C}^{\delta}_{\widehat{P}}$ under $d^{\pi}_{\widehat{P}}$. There exists a tradeoff between the two quantities since a policy that increases $V^{\pi}_{M^\star}$ may also increase the expectation of $\widehat{C}^{\delta}_{\widehat{P}}$ under $d^{\pi}_{\widehat{P}}$. A learned policy different from π_β may take actions not observed in the dataset or visit unobserved states with high-value functions. Then, the resulting $\widehat{C}^{\delta}_{\widehat{P}}(s,a)$ for those state-action pairs will be large since $\hat{n}(s,a)$ is small. Hence, this conservative MDP encourages the agent to find a near-optimal policy but simultaneously discourages the agent to take actions that are unobserved in $\mathcal{D}$. The second inequality in Theorem 2 shows that the sub-optimality gap between the optimal policy π^* and the learned policy $\hat{\pi}$ under $M^\star$ depends on the expectation of $\widehat{C}^{\delta}_{\widehat{P}}$ under $d^{\pi^*}_{\widehat{P}}$. If the quality of the offline dataset $\mathcal{D}$ is high — that is, if the behavior policy π_β is similar to π^*, then the expectation of $\widehat{C}^{\delta}_{\widehat{P}}$ becomes small, allowing $\hat{\pi}$ to be closer to π^*.

4.2. Proposed Algorithm: `Count-MORL`

We present our algorithm, *Count-based Conservatism for Model-based Offline RL* (`Count-MORL`). The algorithm is summarized in Algorithm 1. Our algorithm learns the estimated transition model and computes the approximate counts of state-action pairs in the offline dataset to penalize the reward function, so as to induce count-based conservatism. Our proposed algorithm is generic in terms of incorporating count estimation. Hence, one can utilize any type of count approximation studied in the online RL literature (Bellemare et al., 2016; Tang et al., 2017; Martin et al., 2017; Ostrovski et al., 2017; Machado et al., 2020).

Algorithm 1 : Count-based Conservatism for Model-based Offline RL (`Count-MORL`)

Require: counting functions $\{n_i : \mathbb{R}^d \to \mathbb{R}^+ \cup \{0\}\}_{i=1}^N$, reward penalty coefficient β, rollout horizon H, rollout batch size B, offline dataset $\mathcal{D}$

Train an ensemble of N dynamics models $\{\widehat{P}_i\}_{i=1}^N$ and feature mappings $\{\phi_i : \mathcal{S}\times\mathcal{A} \to \mathbb{R}^d\}_{i=1}^N$ on $\mathcal{D}$.
Initialize policy π and the replay buffer $\mathcal{D}_{\text{model}} \leftarrow \varnothing$.
for `epoch` $= 1, 2, \cdots$ **do**
  for `rollout` $= 1, 2, \cdots, B$ **do**
    Sample initial rollout state s_1 from $\mathcal{D}$.
    for $t = 1, 2, \cdots, H$ **do**
      Sample an action $\hat{a}_t \sim \pi(\hat{s}_t)$
      Sample a dynamics model $\widehat{P}$ from $\{\widehat{P}_i\}_{i=1}^N$
      Sample $\hat{s}_{t+1}, \hat{r}_t \sim \widehat{P}(\hat{s}_t, \hat{a}_t)$
      Compute $\hat{n}(\hat{s}_t, \hat{a}_t)$ using Algorithm 2
      Compute

$$\tilde{r}_t = \begin{cases} \hat{r}_t - \frac{\beta}{\sqrt{\hat{n}(\hat{s}_t,a_t)}} & \texttt{if } \hat{n}(\hat{s}_t,\hat{a}_t) > 0 \\ \hat{r}_t - \beta & \texttt{otherwise}. \end{cases}$$

      Add sample $(\hat{s}_t, \hat{a}_t, \tilde{r}_t, \hat{s}_{t+1})$ to $\mathcal{D}_{\text{model}}$.
    end for
  end for
  Draw samples from $\mathcal{D} \cup \mathcal{D}_{\text{model}}$ to update π.
end for

Another salient feature of our method is incorporation of the uncertainty of the count estimation, shown in Algorithm 2 which is a sub-routine of `Count-MORL`. Algorithm 2 allows three possible types of count estimation: lower confidence (`LC`), average (`AVG`), and upper confidence (`UC`). As shown in the numerical experiments of Section 5, a certain type of count estimation performs superior to others depending on how samples in the offline dataset were collected (see Table 2 for more details). Hence, the consideration of uncertainty in count estimation is crucial.

Algorithm 1 presents a general version of our proposed method with feature mappings.[2] The first step of our algorithm is to train an ensemble of N dynamics models $\{\widehat{P}_i(s', r|s,a)\}_{i=1}^N$ and feature mappings $\{\phi_i(s,a)\}_{i=1}^N$. Given a state-action pair, each composition function of the feature mapping and counting function, $\{n_i(\phi_i(s,a))\}_{i=1}^N$, are obtained. A mean and a standard deviation are calculated by using these composition functions as $\bar{n}(s,a)$ and $\sigma(s,a)$. When we have prior information available about the offline dataset, the count estimation (`LC`, `AVG`, or `UC`) can be chosen appropriately. For example, if the offline dataset is given by a clearly suboptimal behavior policy (in other words, the observed trajectories significantly differ from trajectories possibly given by an optimal policy), then

[2]For tabular settings, feature representation for each state-action pair can be given by one-hot encoding.

Algorithm 2 : Count Estimation

Require: standard deviation coefficient α, feature mappings $\{\phi_i\}_{i=1}^N$, counting functions $\{n_i\}_{i=1}^N$

1: **for** $i = 1, 2, \cdots, N$ **do**
2: Compute $\phi_i(s, a)$
3: Compute $\hat{n}_i(s, a) = n_i(\phi_i(s, a))$
4: **end for**
5: Compute $\bar{n}(s, a) = \frac{1}{N}\sum_{i=1}^N \hat{n}_i(s, a)$
6: Compute $\sigma(s, a) = \sqrt{\frac{\sum_{i=1}^N \{\hat{n}_i(s,a) - \bar{n}(s,a)\}^2}{N-1}}$
7: Compute

$$\hat{n}(s,a) = \begin{cases} \bar{n}(s,a) - \alpha\sigma(s,a) & \texttt{if LC count} \\ \bar{n}(s,a) & \texttt{if AVG count} \\ \bar{n}(s,a) + \alpha\sigma(s,a) & \texttt{if UC count.} \end{cases}$$

8: **return** an approximate count $\hat{n}(s, a)$

it is sensible to consider a higher level of pessimism on the observed samples, therefore utilizing LC counts in this case. Then, a learned policy is more inclined to take actions (and visit states) outside the offline dataset — since the penalization on observed samples is increased while the penalization on unobserved data is fixed. Conversely, if the offline dataset is given by a behavior policy which is near-optimal, then it is appropriate to utilize UC counts to penalize with a lower level of pessimism on the observed samples. In that case, a learned policy is encouraged to take actions within the offline dataset.

4.3. Architecture and Efficient Implementation

Although Count-MORL is a comprehensive method not limited to a specific architecture, we provide a concrete architectural example (used for the experiments in Section 5) that details its efficient implementation using *hash-code* count estimation, adapted from a technique used in the online RL literature (Tang et al., 2017). The overall architecture is depicted in Figure 1.

The estimated dynamics model for continuous state transitions is constructed using a neural network that predicts a Gaussian distribution over the estimated next state and reward, similar to the approach of Yu et al. (2020). This model is linked to an autoencoder at the hidden layer. The autoencoder's input is the output of this hidden layer, which includes a bottleneck layer comprised of d sigmoid functions. By rounding the sigmoid activations of the bottleneck layer to their nearest binary values, we convert all state-action pairs into binarized d-dimensional binary vectors. As the learning process advances, the latent representation of these state-action pairs, along with their corresponding binarized d-dimensional vectors, become more stable.

To compute the frequency of potentially high-dimensional or continuous state-action pairs, we count each unique binary vector corresponding to each state-action pair. However, since the binary vector corresponding to each state-action pair can vary based on different estimated models (given that we use an ensemble of transition models), we apply an approximate count computed from the count obtained for each learned model, as outlined in Algorithm 2.

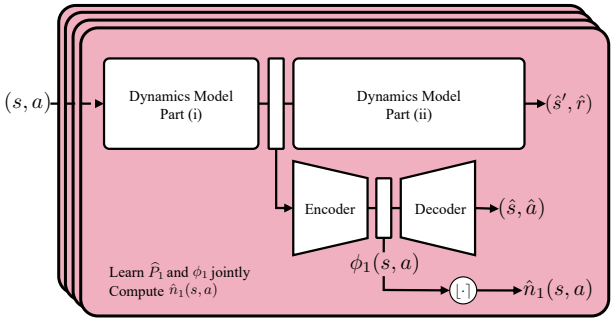


Figure 1. Efficient implementation of Count-MORL with hash codes: given a state action pair as input, for each model, taking the feature mapping ϕ and computing the counting function $\hat{n}$ using ϕ.

5. Experiments

In this section, we present our experimental procedures and their respective outcomes, seeking to answer the following pertinent questions:

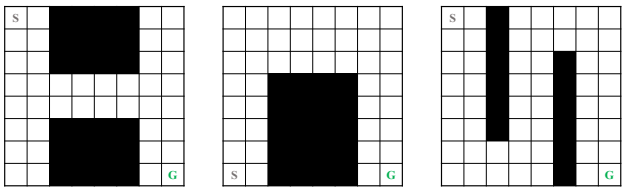
(i) Can the proposed method produce approximate counts that accurately estimate the true counts for each state-action pair?

(ii) How does the performance of Count-MORL measure up against the previous state-of-the-art (SOTA) benchmarks in offline RL?

(iii) How do count estimation methods, which account for uncertainty in count estimation, perform when exposed to different types of datasets?

5.1. Experimental Setup

Grid-World. To address the question (i), we start by precisely counting the number of samples for each state-action pair in the Grid-World setting. As we have the true count for each state-action pair at hand, we can evaluate the discrepancy between the approximate counts and the authentic counts. We employ four types of Grid-World environments —*Empty*, *Bridge*, *Cliff*, and *ZigZag* — as depicted in Figure 2 (Biedenkapp et al., 2021). In each of these environments, we amass transition samples from a replay buffer of the policy trained via Q-learning. The *Empty* dataset contains 10^6 samples and covers all state-action pairs. In contrast, the *Bridge*, *Cliff*, and *ZigZag* datasets encompass 1.6×10^5,



(a) Bridge (b) Cliff (c) Zigzag

Figure 2. 3 types of 8×8 Grid World, Bridge, Cliff, and Zigzag. The objective is to reach a fixed goal state (G) from a fixed start state (S) while avoiding the lava area colored in black.

1.4×10^5 and 3×10^5 transition samples, respectively, but they do not include samples for all actions taken within the lava zones, represented by black grids.

MuJoCo. We evaluate `Count-MORL` on datasets in the D4RL benchmark (Fu et al., 2020), which comprises a total of 12 datasets from 3 different environments (*HalfCheetah*, *Hopper*, and *Walker2d*), each with 4 dataset types (*Random, Medium, Medium-Replay, Medium-Expert*). *Random* dataset contains 10^6 transition samples collected by a random policy. *Medium* dataset contains 10^6 transition samples collected by a partially trained SAC policy. *Medium-Replay* dataset contains 10^5 (2×10^5 for *Walker2d*) transition samples, which are from the replay buffer that accumulates samples obtained by interacting with the environment until the policy trained using SAC reaches the performance of *Medium* agent. *Medium-Expert* dataset contains 10^6 transition samples, a mixture of sub-optimal and expert samples obtained by the partial and fully-trained policy.

Counting Method. As explained earlier, there are three count estimation methods: `LC`, `AVG`, and `UC` counts. When using `LC` and `UC` count methods, for each dataset, we choose the standard deviation coefficient α as 0.5. By changing the value of α, the approximate count can be adjusted to be either large or small (Line 7 in Algorithm 2).

Hyperparameter Details. Our algorithm adopts its foundational hyperparameters from the MOPO framework (Yu et al., 2020). While MOPO typically utilizes relatively short rollout lengths, such as 2 or 5 steps, recent research (Lu et al., 2022) emphasizes the crucial role that the rollout length parameter, denoted by h, plays in the performance of model-based offline RL algorithms. Therefore, we extend the rollout length to accommodate up to 20 steps. We select an optimal rollout length and a reward penalty coefficient from the following potential values: $h \in \{5, 20\}$ and $\beta \in \{0.5, 1, 3, 5\}$.

Hyperparameter Details of Autoencoder. We employ a d-dimensional binary vector to count the samples within our offline dataset. Our experiments explore the performance with different dimensions of this binary vector, specifically considering five distinct values: $d \in \{16, 32, 50, 64, 80\}$.

Baselines. We compare our algorithm against the state-of-the-art model-based algorithms, RepBSDE (Lee et al., 2021), RAMBO (Rigter et al., 2022), COMBO (Yu et al., 2021b), MOPO (Yu et al., 2020) and MOReL (Kidambi et al., 2020), and model-free algorithms, CQL (Kumar et al., 2020) and TD3+BC (Fujimoto & Gu, 2021), for offline RL. COMBO, MOPO, and CQL evaluate the performance of their algorithms on the MuJoCo-v0 datasets. However, RepB-SDE, RAMBO, MOReL, and TD3+BC papers include their performance on the MuJoCo-v2 datasets. We provide normalized scores for `Count-MORL` and reproduce the performance of COMBO, MOPO, and CQL algorithms on the MuJoCo-v2 datasets. This allows for a fair comparison with all SOTA offline RL algorithms, which have also been evaluated on the MuJoCo-v2 datasets.

5.2. Results

5.2.1. Results on Grid-World

We assess the accuracy of our approach by comparing the approximate counts with the true counts in the 8×8 Grid-World environments. The autoencoder, which is connected to the dynamics model, uses the output from the dynamics model's hidden layer as its input, as detailed in our neural network architecture (see Figure 1). Despite the intricacies of this setup, we find that the approximate count, derived from a hash code, matches the true count for each state-action pair across all the Grid-World environments that we tested, *Bridge*, *Cliff*, and *Zigzag*. These findings underscore the accuracy and robustness of our count estimation method. More detailed results are provided in Appendix C.1.

5.2.2. Results on D4RL tasks

Our experimental results, summarized in Tables 1 and 2, are based on evaluations carried out on the D4RL benchmark datasets. To address the question (ii), our method, `Count-MORL`, achieves the best or competitive performance in 10 out of the 12 settings. As seen in Table 1, `Count-MORL` outperforms others across datasets with narrower (*Medium*, *Medium-Expert*) and more diverse state-action distributions (*Random*, *Medium-Replay*). However, the *Random* datasets of *Hopper* and *Walker2d* are exceptions, where MOReL performs better. Excluding MOReL from comparison, our method leads in performance for *Hopper* and *Walker2d* on the *Random* dataset. Additionally, our method surpasses all other algorithms for the *Random* dataset of *Halfcheetah*. These results demonstrate the superior performance of `Count-MORL` against state-of-the-art offline RL algorithms across various dataset types.

Table 2 exhibits the performance of different count estimation methods (`LC`, `AVG`, and `UC`) on the D4RL benchmark

Table 1. Results for D4RL datasets. Each number is the normalized score proposed in Fu et al. 2020 of the policy during the last 5 iterations averaged over 5 seeds, where ± denotes the standard deviation over seeds. We take the results of RepB-SDE, RAMBO, MOReL, and TD3+BC from their papers. We reproduce the results of COMBO, MOPO, and CQL with MuJoCo-v2 datasets. We include the score of behavior cloning (BC) for comparison. Bold numbers are the scores within 2% of the highest score in each environment.

			Model-based baselines					Model-free baselines		
Dataset type	Environment	Count-MORL	RepB-SDE	RAMBO	COMBO	MOPO	MOReL	CQL	TD3+BC	BC
Random	Halfcheetah	**41.0 ± 0.9**	32.9	40.0	36.7 ± 2.1	34.0 ± 2.8	25.6	26.6 ± 0.8	11.0	2.1
	Hopper	30.7 ± 1.3	8.6	21.6	7.8 ± 0.9	7.0 ± 1.9	**53.6**	9.4 ± 0.6	8.5	9.8
	Walker2d	21.9 ± 0.2	21.1	11.5	5.9 ± 0.3	5.6 ± 5.7	**37.3**	−0.4 ± 0.8	1.6	1.6
Medium	Halfcheetah	**76.5 ± 1.7**	49.1	**77.6**	61.6 ± 1.5	67.8 ± 2.3	42.1	47.2 ± 0.6	48.3	36.1
	Hopper	**103.6 ± 3.7**	34.0	92.8	63.3 ± 2.4	20.9 ± 13.9	95.4	62.2 ± 4.4	59.3	29.0
	Walker2d	**87.6 ± 3.7**	72.1	**86.9**	70.1 ± 5.0	−0.1 ± 0.1	77.8	76.1 ± 1.6	83.7	6.6
Medium-Replay	Halfcheetah	**71.5 ± 1.8**	57.5	68.9	57.0 ± 1.2	66.2 ± 3.0	40.2	44.6 ± 0.7	44.6	38.4
	Hopper	**101.7 ± 0.8**	62.2	96.6	71.5 ± 8.0	64.2 ± 28.9	93.6	98.3 ± 1.3	60.9	11.8
	Walker2d	**87.7 ± 3.0**	49.8	85.0	52.6 ± 4.5	67.9 ± 15.7	49.8	82.1 ± 2.6	81.8	11.3
Medium-Expert	Halfcheetah	**100.0 ± 4.9**	55.4	93.7	65.3 ± 9.7	91.7 ± 9.9	53.3	90.6 ± 4.4	90.7	35.8
	Hopper	**111.4 ± 0.5**	82.6	83.3	105.4 ± 4.5	21.9 ± 20.9	108.7	98.2 ± 11.0	98.0	**111.9**
	Walker2d	**112.3 ± 1.8**	88.8	68.3	73.7 ± 12.7	4.0 ± 5.4	95.6	109.3 ± 0.6	**110.1**	6.4
MuJoCo-v2 Average:		**78.8 ± 2.0**	51.2	68.9	55.9 ± 4.4	37.6 ± 9.2	64.4	62.0 ± 2.5	58.2	25.1

Table 2. Performance of Count-MORL using each count estimation method (LC, AVG, UC). We bold the highest score.

		Count Estimation		
Dataset type	Environment	LC count	AVG count	UC count
Random	Halfcheetah	**41.0 ± 0.9**	38.7 ± 0.5	39.1 ± 1.0
	Hopper	**30.7 ± 1.3**	27.7 ± 6.2	26.5 ± 6.5
	Walker2d	**21.9 ± 0.1**	**21.9 ± 0.1**	**21.9 ± 0.2**
Medium	Halfcheetah	74.2 ± 2.5	**76.5 ± 1.7**	74.6 ± 1.6
	Hopper	99.7 ± 7.2	101.8 ± 4.7	**103.6 ± 3.7**
	Walker2d	84.2 ± 2.9	**87.6 ± 3.7**	85.2 ± 2.6
Medium-Replay	Halfcheetah	71.2 ± 2.9	71.1 ± 0.8	**71.5 ± 1.8**
	Hopper	98.9 ± 3.9	100.2 ± 0.9	**101.7 ± 0.8**
	Walker2d	84.3 ± 3.1	85.8 ± 2.8	**87.7 ± 3.0**
Medium-Expert	Halfcheetah	94.8 ± 5.5	98.1 ± 2.5	**100.0 ± 4.9**
	Hopper	107.2 ± 4.7	109.4 ± 1.2	**111.4 ± 0.5**
	Walker2d	109.7 ± 1.4	110.7 ± 0.5	**112.3 ± 1.8**

datasets. LC count performs best for *Random* datasets, while UC count outperforms others for *Medium-Expert* datasets. These results suggest that we can tailor the count estimation method based on the nature of the offline dataset. Unobserved state-action pairs in the offline dataset are assigned a constant penalty of 1 (see Corollary 1). For the *Random* dataset, LC count provides a relatively larger penalty to observed state-action pairs by under-estimating counts, allowing the agent to potentially take actions or visit states beyond the dataset's boundaries. Conversely, for the *Medium-Expert* dataset, UC count computes over-approximate counts incurring smaller penalties and encouraging the agent to exploit the states within the dataset. If the replay buffer consistently contains high-quality data, as observed in the *Medium-Replay* datasets, UC count tends to perform better, as indicated in Table 2. However, if the behavior policy generates low-quality data significantly different from data from an optimal policy, LC count can be more beneficial. Thus, the appropriate choice of count estimation is likely to achieve improved performance.

6. Conclusion

Count-MORL introduces a novel and tractable approach to model-based offline RL that incorporates count-based conservatism, effectively bridging the theoretical and practical divide in model-based offline deep RL. Consequently, our method outperforms existing state-of-the-art offline RL algorithms, further solidifying our approach's efficacy.

Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2021M3E5D2A01024795, No. 2022R1C1C1006859, and No. 2022R1A4A103057912).

References

Agarwal, A., Kakade, S., Krishnamurthy, A., and Sun, W. Flambe: Structural complexity and representation learning of low rank mdps. *Advances in neural information processing systems*, 33:20095–20107, 2020a.

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020b.

An, G., Moon, S., Kim, J.-H., and Song, H. O. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

Argenson, A. and Dulac-Arnold, G. Model-based offline planning. In *International Conference on Learning Representations*, 2021.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Biedenkapp, A., Rajan, R., Hutter, F., and Lindauer, M. Temporl: Learning when to act. In *International Conference on Machine Learning*, pp. 914–924. PMLR, 2021.

Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., R Ruiz, F. J., Schrittwieser, J., Swirszcz, G., et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

Fu, J., Co-Reyes, J., and Levine, S. Ex2: Exploration with exemplar models for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hishinuma, T. and Senda, K. Weighted model estimation for offline model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 34:17789–17800, 2021.

Hong, J., Kumar, A., and Levine, S. Confidence-conditioned value functions for offline reinforcement learning. In *International Conference on Learning Representations*, 2023.

Hwang, T. and Oh, M.-h. Model-based reinforcement learning with multinomial logistic function approximation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 2023.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., and Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Lee, B.-J., Lee, J., and Kim, K.-E. Representation balancing offline model-based reinforcement learning. In *International Conference on Learning Representations*, 2021.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. Provably good batch off-policy reinforcement learning without great exploration. *Advances in neural information processing systems*, 33:1264–1274, 2020.

Lu, C., Ball, P., Parker-Holder, J., Osborne, M., and Roberts, S. J. Revisiting design choices in offline model based reinforcement learning. In *International Conference on Learning Representations*, 2022.

Machado, M. C., Bellemare, M. G., and Bowling, M. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5125–5133, 2020.

Martin, J., Narayanan, S. S., Everitt, T., and Hutter, M. Count-based exploration in feature space for reinforcement learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 2471–2478, 2017.

Matsushima, T., Furuta, H., Matsuo, Y., Nachum, O., and Gu, S. Deployment-efficient reinforcement learning via model-based offline optimization. In *International Conference on Learning Representations*, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. Count-based exploration with neural density models. In *International conference on machine learning*, pp. 2721–2730. PMLR, 2017.

Rafailov, R., Yu, T., Rajeswaran, A., and Finn, C. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pp. 1154–1168. PMLR, 2021.

Rigter, M., Lacerda, B., and Hawes, N. Rambo-rl: Robust adversarial model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Shi, L., Li, G., Wei, Y., Chen, Y., and Chi, Y. Pessimistic q-learning for offline reinforcement learning: Towards optimal sample complexity. In *International Conference on Machine Learning*, pp. 19967–20025. PMLR, 2022.

Siegel, N., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., Heess, N., and Riedmiller, M. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Swazinna, P., Udluft, S., and Runkler, T. Overcoming model bias for robust offline deep reinforcement learning. *Engineering Applications of Artificial Intelligence*, 104: 104366, 2021.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Xi Chen, O., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Uehara, M. and Sun, W. Pessimistic model-based offline reinforcement learning under partial coverage. In *International Conference on Learning Representations*, 2022.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Xie, T., Cheng, C.-A., Jiang, N., Mineiro, P., and Agarwal, A. Bellman-consistent pessimism for offline reinforcement learning. *Advances in neural information processing systems*, 34:6683–6694, 2021.

Yang, L. and Wang, M. Sample-optimal parametric q-learning using linearly additive features. In *International Conference on Machine Learning*, pp. 6995–7004. PMLR, 2019.

Yang, L. and Wang, M. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In *International Conference on Machine Learning*, pp. 10746–10756. PMLR, 2020.

Yu, C., Liu, J., Nemati, S., and Yin, G. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021a.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021b.

Zhou, D., Gu, Q., and Szepesvari, C. Nearly minimax optimal reinforcement learning for linear mixture markov decision processes. In *Conference on Learning Theory*, pp. 4532–4576. PMLR, 2021.

A. Proosfs

A.1. Proof of Theorem 1

Proof. Given a state-action (s, a) pair observed in $\mathcal{D}$ with $\mathcal{D}_{s,a} = \{(s_i, a_i, s_i')\}_{s_i=s, a_i=a}$ and $n(s, a) = |\mathcal{D}_{s,a}|$ such that the MLE of transition dynamics is

$$\widehat{P}(\cdot \mid s, a) \in \arg\max_{P \in \mathcal{M}} \sum_{(s,a,s') \in \mathcal{D}_{s,a}} \log P(s' \mid s, a) \tag{1}$$

for given (s, a). By Theorem 21 in Agarwal et al. (2020a), with probability at least $1 - \delta$,

$$\mathbb{E}_{(s_i,a_i)\sim\mathcal{D}_{s,a}} \left[\mathrm{TV}\Big(\widehat{P}(\cdot \mid s_i, a_i), P^\star(\cdot \mid s_i, a_i)\Big)^2 \right] \leq \frac{2\log(|\mathcal{M}|/\delta)}{n(s, a)}.$$

We can directly bound the total variation distance between the estimated transition dynamics $\widehat{P}$ and the true transition dynamics $P^\star$ due to the subset $\mathcal{D}_{s,a} = \{(s, a, s_i')\}_{i=1}^{n(s,a)}$. Thus,

$$\mathrm{TV}\Big(\widehat{P}(\cdot \mid s, a), P^\star(\cdot \mid s, a)\Big) \leq \sqrt{\frac{2\log(|\mathcal{M}|/\delta)}{n(s, a)}}.$$

□

A.1.1. Proof of Corollary 1

Proof. By Theorem 1, given a state-action (s, a) pair observed in $\mathcal{D}$, the estimated transition dynamics is satisfied by Equation (1) such that we have

$$\mathrm{TV}\Big(\widehat{P}(\cdot \mid s, a), P^\star(\cdot \mid s, a)\Big) \leq \sqrt{\frac{2\log(|\mathcal{M}|/\delta)}{n(s, a)}}.$$

If given an unobserved state-action pair in $\mathcal{D}$, we cannot estimate the transition dynamics or compute the estimation error. And, when $n(s, a)$ is less than $\sqrt{2\log(|\mathcal{M}|/\delta)}$, the estimation error becomes greater than 1. Thus, by the definition of the total variation distance between probability distributions P and Q as

$$\mathrm{TV}(P, Q) = \sup_A |P(A) - Q(A)|,$$

we bound the estimation error as 1 for these cases. Therefore, with probability at least $1 - \delta$, for any state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$,

$$\mathrm{TV}\Big(\widehat{P}(\cdot \mid s, a), P^\star(\cdot \mid s, a)\Big) \leq \min\left(1, \sqrt{\frac{2\log(|\mathcal{M}|/\delta)}{n(s, a)}}\right).$$

□

A.2. Proof of Theorem 2

Before we prove Theorem 2, we prove Lemma 1 and Corollary 2 that will be used in the latter proof.

A.2.1. Proof of Lemma 1

The proof of Lemma 1 follows from modifying the proofs of Lemma 4.1 in MOPO (Yu et al., 2020) based on count-based estimation error.

Proof. We denote the expectation of cumulative discounted reward under π and $\widehat{P}$ for j steps and then switching to $P^\star$ for the rest steps:

$$W_j = \mathop{\mathbb{E}}_{\substack{s\sim d_0 \\ \forall t\geq 0, a_t\sim\pi(\cdot|s_t) \\ \forall j>t\geq 0, s_{t+1}\sim\widehat{P}(\cdot|s_t,a_t) \\ \forall t\geq j, s_{t+1}\sim P^\star(\cdot|s_t,a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s \right]$$

Note that $W_0 = V^{\pi}_{M^\star}$ and $W_\infty = V^{\pi}_{\widehat{M}}$. By the telescoping lemma, we have that

$$V^{\pi}_{\widehat{M}} - V^{\pi}_{M^\star} = \sum_{j=0}^{\infty} (W_{j+1} - W_j)\,.$$

Rewriting W_j and W_{j+1} for the trajectory distribution as

$$W_j = R_j + \mathop{\mathbb{E}}_{s_j,a_j\sim\pi,\widehat{P}} \left[\mathop{\mathbb{E}}_{s_{j+1}\sim\widehat{P}(\cdot|s_j,a_j)} \left[\gamma^{j+1} V^{\pi}_{P^\star,r}(s_{j+1})\right]\right]$$
$$W_{j+1} = R_j + \mathop{\mathbb{E}}_{s_j,a_j\sim\pi,\widehat{P}} \left[\mathop{\mathbb{E}}_{s_{j+1}\sim P^\star(\cdot|s_j,a_j)} \left[\gamma^{j+1} V^{\pi}_{P^\star,r}(s_{j+1})\right]\right],$$

where R_j denotes the sum of the discounted rewards obtained from the first step to j step under π and $\widehat{P}$.

Then,

$$\begin{aligned}
W_{j+1} - W_j &= \gamma^{j+1} \mathop{\mathbb{E}}_{s_j,a_j\sim\pi,\widehat{P}} \left[\mathop{\mathbb{E}}_{s_{j+1}\sim\widehat{P}(\cdot|s_j,a_j)} \left[V^{\pi}_{P^\star,r}(s_{j+1})\right] - \mathop{\mathbb{E}}_{s_{j+1}\sim P^\star(\cdot|s_j,a_j)} \left[V^{\pi}_{P^\star,r}(s_{j+1})\right]\right] \\
&\le \frac{\gamma^{j+1} R_{\max}}{1-\gamma} \mathop{\mathbb{E}}_{s_j,a_j\sim\pi,\widehat{P}} \left[\mathrm{TV}\Big(\widehat{P}(\cdot|s_j,a_j), P^\star(\cdot|s_j,a_j)\Big)\right] \\
&\le \frac{\gamma^{j+1} R_{\max}}{1-\gamma} \mathop{\mathbb{E}}_{s_j,a_j\sim\pi,\widehat{P}} \left[C^{\delta}_{\widehat{P}}(s_j,a_j)\right] \\
&\le \frac{\gamma^{j+1} R_{\max}}{1-\gamma} \mathop{\mathbb{E}}_{s_j,a_j\sim\pi,\widehat{P}} \left[\widehat{C}^{\delta}_{\widehat{P}}(s_j,a_j) + \epsilon(s_j,a_j)\right], \qquad (2)
\end{aligned}$$

where first inequality step uses the fact that $\left\|V^{\pi}_{P^\star,r}(s)\right\|_\infty \le \frac{R_{\max}}{1-\gamma}$; second inequality step uses the estimation error bound based on the true count $C^{\delta}_{\widehat{P}}$ in Corollary 1; and the last inequality step uses the estimated error bound based on the approximate count $\widehat{C}^{\delta}_{\widehat{P}}$ and denote the approximation error for each state-action pair as $\epsilon(s,a) := \left|C^{\delta}_{\widehat{P}}(s,a) - \widehat{C}^{\delta}_{\widehat{P}}(s,a)\right|$.

Therefore, with probability at least $1-\delta$, for any policy π,

$$\begin{aligned}
V^{\pi}_{\widehat{M}} - V^{\pi}_{M^\star} &= \sum_{j=0}^{\infty} (W_{j+1} - W_j) \\
&\le \frac{R_{\max}}{1-\gamma} \sum_{j=0}^{\infty} \gamma^{j+1} \mathop{\mathbb{E}}_{s_j,a_j\sim\pi,\widehat{P}} \left[\widehat{C}^{\delta}_{\widehat{P}}(s_j,a_j) + \epsilon(s_j,a_j)\right] \\
&= \frac{\gamma R_{\max}}{(1-\gamma)^2} \mathbb{E}_{(s,a)\sim d^{\pi}_{\widehat{P}}} \left[\widehat{C}^{\delta}_{\widehat{P}}(s,a) + \epsilon(s_j,a_j)\right] \\
&\le \frac{\gamma R_{\max}}{(1-\gamma)^2} \mathbb{E}_{(s,a)\sim d^{\pi}_{\widehat{P}}} \left[\widehat{C}^{\delta}_{\widehat{P}}(s,a)\right] + \frac{\gamma R_{\max}}{(1-\gamma)^2}\epsilon\,,
\end{aligned}$$

where the first inequality step uses Equation (2); and the last inequality uses the definition of the maximal approximation error as $\epsilon = \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} \epsilon(s,a)$. □

A.2.2. Proof of Corollary 2

Proof. In Definition 3, we define the count-based conservatism MDP $\widetilde{M} := (\mathcal{S}, \mathcal{A}, \widehat{P}, \tilde{r}, d_0, \gamma)$ with the estimated transition dynamics $\widehat{P}$ and the count-based penalized reward $\tilde{r}(s,a) = r(s,a) - \frac{\gamma R_{\max}}{1-\gamma}\widehat{C}^{\delta}_{\widehat{P}}(s,a)$. Then, we represent the result of

Lemma 1 based on the value gap between the count-based conservatism MDP $\widetilde{M}$ and the true MDP $M^\star$.

$$\begin{aligned} V^\pi_{M^\star} &\geq V^\pi_{\widetilde{M}} - \frac{\gamma R_{\max}}{(1-\gamma)^2}\mathbb{E}_{(s,a)\sim d^\pi_{\widehat{P}}}\left[\widehat{C}^\delta_{\widehat{P}}(s,a)\right] - \frac{\gamma R_{\max}}{(1-\gamma)^2}\epsilon \\ &= \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim d^\pi_{\widehat{P}}}\left[r(s,a) - \frac{\gamma R_{\max}}{1-\gamma}\widehat{C}^\delta_{\widehat{P}}(s,a)\right] - \frac{\gamma R_{\max}}{(1-\gamma)^2}\epsilon \\ &= \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim d^\pi_{\widehat{P}}}\left[\tilde{r}(s,a)\right] - \frac{\gamma R_{\max}}{(1-\gamma)^2}\epsilon \\ &= V^\pi_{\widetilde{M}} - \frac{\gamma R_{\max}}{(1-\gamma)^2}\epsilon\,, \end{aligned} \tag{3}$$

where the first inequality step uses the value gap of the estimated model in Lemma 1; and the second equality step uses the definition of the count-based penalized reward. □

Now, we prove our main theorem (Theorem 2) that provides a performance guarantee under $M^\star$.

Proof. For any policy π, we have that

$$\begin{aligned} V^{\hat{\pi}}_{M^\star} &\geq V^{\hat{\pi}}_{\widetilde{M}} - \frac{\gamma R_{\max}}{(1-\gamma)^2}\epsilon \\ &\geq V^{\pi}_{\widetilde{M}} - \frac{\gamma R_{\max}}{(1-\gamma)^2}\epsilon \\ &= V^{\pi}_{\widehat{M}} - \frac{\gamma R_{\max}}{(1-\gamma)^2}\mathbb{E}_{(s,a)\sim d^\pi_{\widehat{P}}}\left[\widehat{C}^\delta_{\widehat{P}}(s,a)\right] - \frac{\gamma R_{\max}}{(1-\gamma)^2}\epsilon \\ &\geq V^{\pi}_{M^\star} - \frac{2\gamma R_{\max}}{(1-\gamma)^2}\mathbb{E}_{(s,a)\sim d^\pi_{\widehat{P}}}\left[\widehat{C}^\delta_{\widehat{P}}(s,a)\right] - \frac{2\gamma R_{\max}}{(1-\gamma)^2}\epsilon\,, \end{aligned}$$

where the first inequality step uses the result of Corollary 2; the second inequality step uses the definition of $\hat{\pi}$ as $\hat{\pi} = \arg\max_\pi V^\pi_{\widetilde{M}}$; and the last inequality step uses the result of Theorem 1. □

B. Experimental Details.

This section contains all details of `Count-MORL` (github.com/oh-lab/Count-MORL) with hash codes based on the official MOPO code from github.com/tianheyu927/mopo.

B.1. Model and Policy Training

We represent the dynamics model as an ensemble of probabilistic neural networks that outputs a Gaussian distribution over the next state and reward given the current state and action:

$$\widehat{P}_\theta(s', r|s, a) = \mathcal{N}(\mu_\theta(s,a), \Sigma_\theta(s,a)).$$

Each dynamics model consists of a 4-layer neural network with 200 hidden units per layer and after the last hidden layer, the dynamics model outputs the mean and variance using a two-head architecture. We connect an autoencoder to the hidden layer of the dynamics model, which takes the output of this hidden layer as an input. We determine the architecture of the autoencoder with the output dimension d of the bottleneck layer. For $d < 50$, we use 6-layer neural network with $[100, 50, d, 50, 100, 100]$ units. For $50 \leq d < 100$, we use 5-layer neural network with $[100, 100, d, 100, 100]$ units. We learn the dynamics model and the autoencoder using the log-likelihood objective function and mean square error, respectively. Following previous works (Janner et al., 2019; Yu et al., 2020; 2021b; Rigter et al., 2022), we train an ensemble of 7 such models that each contain the dynamics model and autoencoder and pick the best 5 models based on the validation prediction error on a held-out test set of 1000 transitions from the offline dataset $\mathcal{D}$. For the soft actor-critic (SAC) (Haarnoja et al., 2018) updates, we sample a batch of 256 transitions, 5% of them from $\mathcal{D}$ and the rest of them from $\mathcal{D}_{\text{model}}$.

B.2. Hyperparameters

We found that the hyperparameters that have a significant influence on the performance of `Count-MORL`. We take a rollout length (H), a standard deviation coefficient (α) for the count estimation, a reward penalty coefficient (β) for each count estimation method, and a dimension of hash codes (d). For a rollout length and a reward penalty coefficient, we found that a length $H \in \{5, 20\}$ and a coefficient $\beta \in \{0.5, 1, 3, 5\}$ performed well across all datasets. This is a slight modification to the values of $H \in \{1, 5\}$ and $\beta \in \{0.5, 1, 5\}$ in previous model-based offline RL algorithms (Yu et al., 2020). And, in Lu et al. (2022), the authors show that a rollout length and a reward penalty coefficient play key parameters in determining the performance of model-based offline RL algorithms. Thus, we utilize the longer rollout length as 20 steps. We fix a standard deviation coefficient α as 0.5.

Depending on the dataset, we take a dimension d of hash code over $\{16, 32, 50, 64, 80\}$. We experimentally confirmed that the number of hash codes used for counting samples in the offline dataset does not exponentially increase when the binary vector has a high dimension. For each dimension of the hash code, the count of samples ranged from 5 to 10 (the total number of samples in the offline dataset divided by the number of hash codes used for counting) in the highest reported score for each data type on average. In some other cases, the sample count for each hash code ranged from 1 to 2, or even up to 100. Details of implementation are in the Table 3.

Table 3. Hyperparameters used in the D4RL datasets.

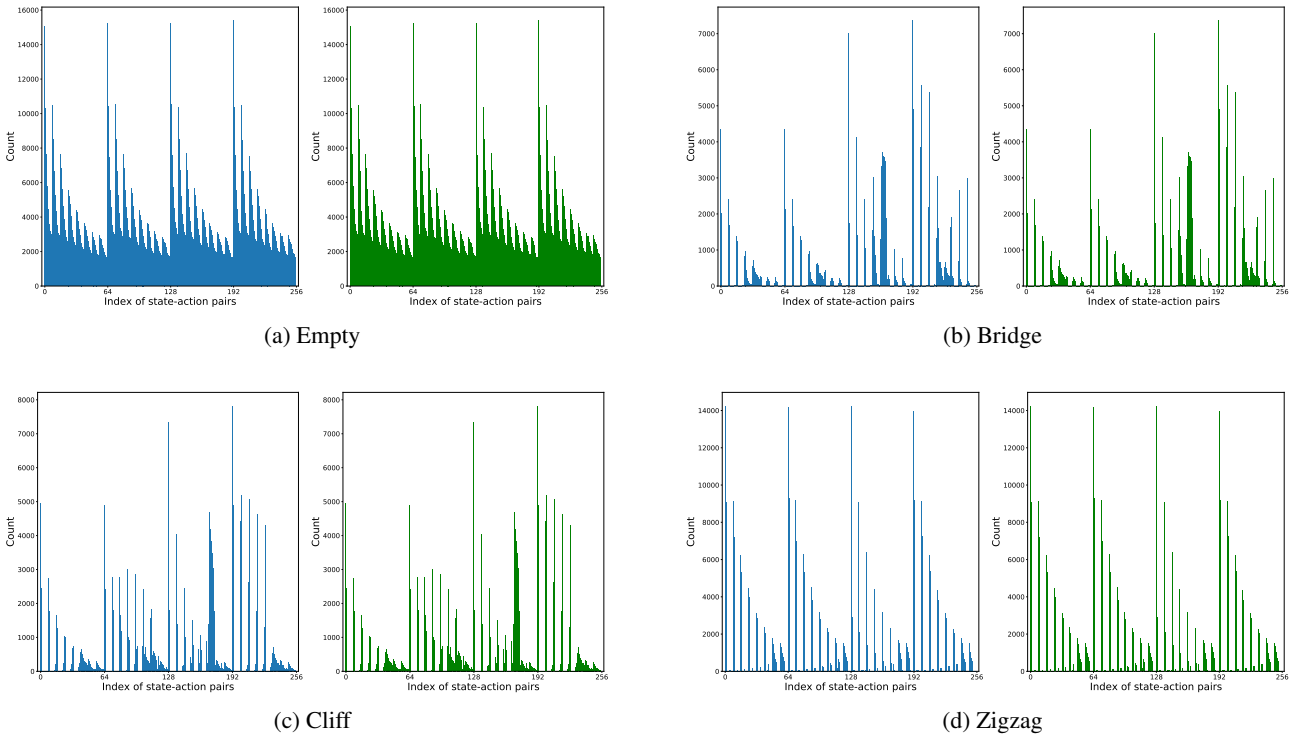
Dataset type	Environment	H	β `LC`	β `AVG`	β `UC`	d
Random	Halfcheetah	5	1	0.5	1	50
	Hopper	20	1	1	3	80
	Walker2d	20	1	1	1	64
Medium	Halfcheetah	5	1	1	3	32
	Hopper	20	1	1	1	50
	Walker2d	20	3	3	3	32
Medium-Replay	Halfcheetah	5	1	3	3	32
	Hopper	5	3	3	1	50
	Walker2d	5	1	3	1	32
Medium-Expert	Halfcheetah	5	3	1	3	32
	Hopper	20	3	3	3	50
	Walker2d	20	3	3	3	50

C. Additional Results

C.1. Results on Grid-World

We compute the approximate count and the known true count on the 8×8 Grid-World environments. Each environment comprises 256 state-action pairs from 64 states, each with 4 actions (up, down, left, right). We convert all state-action pairs into the multi-hot encoder to input. Before counting the number of samples, we train the dynamics model and the autoencoder with the 20-dimension of the bottleneck layer on the replay buffer of the policy trained by Q-learning. When we use the bottleneck layer's dimension less than 20, the hash code cannot divide samples into 256 clusters since the rounding of the bottleneck layer's output turns to the value either 0 or 1.

Empty, *Bridge*, *Cliff*, and *ZigZag* datasets contain 10^6, 1.6×10^5, 1.4×10^5 and 3×10^5 transition samples for each. *Empty* dataset has samples for all state-action pairs, but *Bridge*, *Cliff* and *Zigzag* datasets have no samples on all actions taken in the lava (black grids). In Figure 3, the blue and green histogram presents the known true count and the approximate count, respectively. We observe that the count error between the true and approximate count is zero for all state-action pairs on the four environments. Therefore, Figure 3 shows that our implementation model structure is able to exactly estimate the true count in Grid-World.



(a) Empty (b) Bridge

(c) Cliff (d) Zigzag

Figure 3. Comparison between the true count (Blue) and the approximate count (Green) on Grid-World environments.

C.2. Dimension of hash codes

In Table 4, we investigate the impact of hash code dimension on performance. We perform a rough grid search over a range of the dimension of hash codes. In fact, without much intensive the dimension of hash codes tuning to derive the reported results, `Count-MORL` is shown to perform significantly superior to the existing offline deep RL algorithms, which we believe was another strength of our method. We implement experiments on a total of five dimensions for each dataset, evaluating count estimation methods that showed good performance in Table 2.

Table 4. Performance of `Count-MORL` for each dimension of hash codes. We bold the highest score.

Dataset type	Environment	Count	Dimension of hash codes $d = 16$	$d = 32$	$d = 50$	$d = 64$	$d = 80$
Random	Halfcheetah	LC	36.7 ± 0.5	38.4 ± 0.9	$\mathbf{41.0 \pm 0.9}$	36.2 ± 1.0	35.6 ± 0.8
	Hopper	LC	22.9 ± 6.2	21.5 ± 8.4	24.7 ± 6.8	27.6 ± 6.4	$\mathbf{30.7 \pm 1.3}$
	Walker2d	LC	21.8 ± 0.1	21.8 ± 0.1	$\mathbf{21.9 \pm 0.1}$	$\mathbf{21.9 \pm 0.1}$	21.8 ± 0.1
Medium	Halfcheetah	AVG	74.4 ± 1.5	$\mathbf{76.5 \pm 1.7}$	75.7 ± 1.4	75.2 ± 1.9	73.9 ± 1.6
	Hopper	UC	90.5 ± 3.1	93.2 ± 4.6	$\mathbf{103.6 \pm 3.7}$	98.3 ± 5.8	95.1 ± 3.9
	Walker2d	AVG	82.2 ± 0.9	$\mathbf{87.6 \pm 3.7}$	81.4 ± 1.8	80.1 ± 1.5	80.5 ± 0.7
Medium-Replay	Halfcheetah	UC	68.7 ± 1.9	$\mathbf{71.5 \pm 1.8}$	68.2 ± 1.5	67.3 ± 1.7	65.8 ± 1.7
	Hopper	UC	94.4 ± 3.7	97.1 ± 4.1	$\mathbf{101.7 \pm 0.8}$	94.6 ± 4.9	92.3 ± 3.2
	Walker2d	UC	85.8 ± 1.7	$\mathbf{87.7 \pm 3.0}$	82.4 ± 3.8	80.7 ± 1.3	81.0 ± 2.5
Medium-Expert	Halfcheetah	UC	98.1 ± 3.4	$\mathbf{100.0 \pm 4.9}$	99.1 ± 2.8	98.6 ± 3.2	93.4 ± 2.7
	Hopper	UC	95.2 ± 7.3	90.3 ± 16.6	$\mathbf{111.4 \pm 0.5}$	108.3 ± 1.8	102.8 ± 3.2
	Walker2d	UC	103.4 ± 2.6	106.1 ± 3.4	$\mathbf{112.3 \pm 1.8}$	109.6 ± 2.0	105.8 ± 2.9

C.3. Performance on MuJoCo-v2 datasets

We shows the performance of `Count-MORL` (`LC`, `AVG`, `UC`) and MOPO on MuJoCo-v2 datasets. We confirm that our algorithm performs better than MOPO just by applying the count-based conservatism instead of the uncertainty heuristics of the model in Figure 4. The x-axis of the graph represents episodes, while the y-axis represents the cumulative reward. All count estimation methods are able to train the estimated policy with fewer episodes compared to MOPO to approximate an optimal policy.

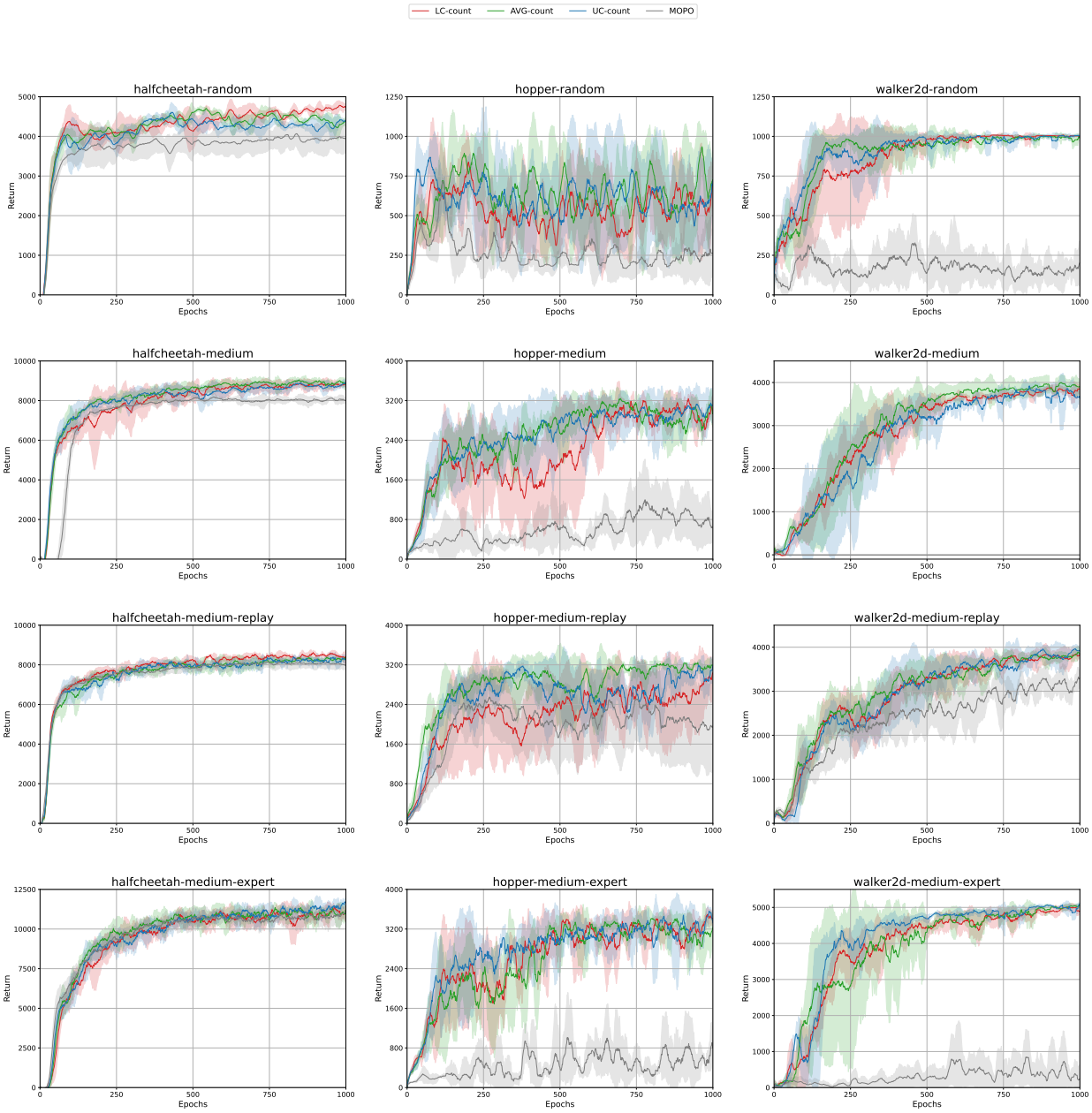


Figure 4. Performance of MuJoCo-v2 datasets

C.4. Results on MuJoCo-v0 datasets

In Table 1, we show the performance of `Count-MORL` and offline RL baselines on MuJoCo-v2 datasets. When reproducing the performance in MuJoCo-v2 datasets, we used the implementation of COMBO from github.com/takuseno/d3rlpy, MOPO from github.com/tianheyu927/mopo, and CQL from github.com/yihaosun1124/OfflineRL-Kit. However, the reproduced scores for MOPO, COMBO, and CQL are lower than those from their paper. Therefore, we implement `Count-MORL` with hash codes on MuJoCo-v0 datasets in Table 5. `Count-MORL` achieves the best or comparable performance on 10 out of 12 settings in MuJoCo-v0 datasets.

Table 5. Results for D4RL datasets. Each number is the normalized score proposed in Fu et al. 2020 of the policy during the last 5 iterations of training averaged over 5 seeds, where $\pm$ denotes the standard deviation over seeds. We take the results of COMBO, MOPO and CQL from their original papers. We bold the scores within 2% of the highest score across all algorithms.

Dataset type	Environment	`Count-MORL`	COMBO	MOPO	CQL
Random	Halfcheetah	$\mathbf{40.5 \pm 0.4}$	38.8	35.4	35.4
	Hopper	11.9 ± 0.2	**17.9**	11.7	10.8
	Walker2d	$\mathbf{21.5 \pm 0.2}$	7.0	13.6	7.0
Medium	Halfcheetah	$\mathbf{56.2 \pm 0.8}$	54.2	42.3	44.4
	Hopper	82.3 ± 2.4	**97.2**	28.0	86.6
	Walker2d	$\mathbf{80.5 \pm 0.7}$	**81.9**	17.8	74.5
Medium-Replay	Halfcheetah	$\mathbf{59.4 \pm 0.5}$	55.1	53.1	46.2
	Hopper	$\mathbf{91.1 \pm 0.7}$	**89.5**	67.5	48.6
	Walker2d	$\mathbf{76.1 \pm 4.0}$	56.0	39.0	32.6
Medium-Expert	Halfcheetah	$\mathbf{102.9 \pm 0.1}$	90.0	63.3	62.4
	Hopper	$\mathbf{112.1 \pm 0.3}$	**111.1**	23.7	**111.0**
	Walker2d	$\mathbf{102.3 \pm 0.5}$	**103.3**	44.6	98.7
MuJoCo-v0 Average:		69.7 ± 0.9	66.8	36.7	54.9