SUMO: Search-Based Uncertainty Estimation for Model-Based Offline Reinforcement Learning

Zhongjian Qiao[1], **Jiafei Lyu**[1], **Kechen Jiao**[1], **Qi Liu** [2], **Xiu Li** [1]

[1]Tsinghua Shenzhen International Graduate School, Tsinghua University
[2]Harbin Institute of Technology, Shenzhen
{qzj22, lvjf20, jkc22}@mails.tsinghua.edu.cn, qiliu8827@stu.hit.edu.cn, li.xiu@sz.tsinghua.edu.cn

Abstract

The performance of offline reinforcement learning (RL) suffers from the limited size and quality of static datasets. Model-based offline RL addresses this issue by generating synthetic samples through a dynamics model to enhance overall performance. To evaluate the reliability of the generated samples, uncertainty estimation methods are often employed. However, model ensemble, the most commonly used uncertainty estimation method, is not always the best choice. In this paper, we propose a **S**earch-based **U**ncertainty estimation method for **M**odel-based **O**ffline RL (SUMO) as an alternative. SUMO characterizes the uncertainty of synthetic samples by measuring their cross entropy against the in-distribution dataset samples, and uses an efficient search-based method for implementation. In this way, SUMO can achieve trustworthy uncertainty estimation. We integrate SUMO into several model-based offline RL algorithms including MOPO and Adapted MOReL (AMOReL), and provide theoretical analysis for them. Extensive experimental results on D4RL datasets demonstrate that SUMO can provide accurate uncertainty estimation and boost the performance of base algorithms. These indicate that SUMO could be a better uncertainty estimator for model-based offline RL when used in either reward penalty or trajectory truncation.

Code — https://github.com/qzj-debug/SUMO.git

Introduction

Offline reinforcement learning (RL) (Levine et al. 2020; Prudencio, Maximo, and Colombini 2023) aims to learn the optimal policy from a static dataset collected in advance, avoiding the risks and costs associated with environmental interaction in typical RL (Sutton and Barto 2018). Nevertheless, since the datasets cannot cover the entire state-action space, the offline agent cannot accurately estimate the Q-value for out-of-distribution (OOD) samples, ultimately leading to a degradation in the agent's performance.

Model-based offline RL (Yu et al. 2020, 2021; Kidambi et al. 2020) employs a promising idea to address OOD issues. By leveraging an environmental dynamics model obtained by supervised learning, the agent can collect samples within the dynamics model and train the policy using these samples. This significantly enhances the performance and generalization of the agent. However, synthetic samples generated by the dynamics model may not be reliable (Lyu, Li, and Lu 2022), as the dynamics model's reliability in OOD regions is not guaranteed. Training the policy with unreliable samples can lead to a performance decline. Therefore, evaluating the reliability of generated samples is critical. It is common to assess the reliability with *uncertainty estimation* (Lockwood and Si 2022; An et al. 2021). Current model-based offline RL methods often employ model ensemble-based techniques for uncertainty estimation, such as max aleatoric (Yu et al. 2020) and max pairwise diff (Kidambi et al. 2020). Subsequently, the estimated uncertainty can be used for trajectory truncation (Kidambi et al. 2020; Zhang et al. 2023b) or reward penalties (Yu et al. 2020) to mitigate OOD issues. However, model ensemble-based uncertainty estimation methods may be unreliable (Yu et al. 2021) because the models can be poorly learned, making their uncertainty estimation questionable. We wonder: *Can we design a better uncertainty estimation method for model-based offline RL?*

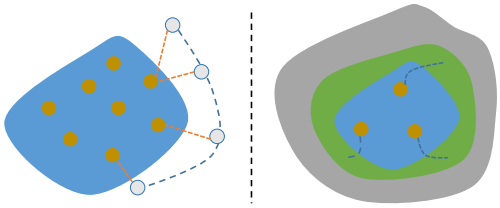

Figure 1: **Left:** The key idea of SUMO. For a synthetic sample, we calculate its KNN distance within the dataset as the uncertainty estimation. **Right:** An example of combining SUMO with AMOReL for trajectory truncation. Thanks to the accurate uncertainty estimation provided by SUMO, the agent can explore ID regions while avoiding OOD regions.

In this paper, we propose a **S**earch-based **U**ncertainty estimation method for **M**odel-based **O**ffline RL (SUMO). SUMO characterizes the uncertainty of synthetic samples as the cross entropy between model dynamics and true dynamics, which can be shown as a more reasonable uncertainty

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

estimation than model ensemble-based estimation. Moreover, the estimated uncertainty for a given sample does not involve extra training of neural networks. In contrast, the uncertainty estimated by model ensemble methods is correlated with the training process and training data distribution since they need to train dynamics models parameterized by neural networks. To estimate the cross entropy practically, we employ a particle-based entropy estimator (Singh et al. 2003), transforming the problem into a k-nearest neighbor (KNN) search problem, as shown in Figure 1 (left), that is the reason we call SUMO a search-based method. Furthermore, given the large search space and high data dimensionality, we employ FAISS (Johnson, Douze, and Jégou 2019) to ensure efficient KNN search. We note that SUMO is algorithm-agnostic, allowing us to integrate it with any model-based offline RL algorithm which needs uncertainty estimation. For instance, it can be combined with MOPO (Yu et al. 2020) or with Adapted MOReL (Kidambi et al. 2020) (AMOReL, which we discuss in later sections), as shown in Figure 1 (right).

We integrate SUMO with several off-the-shelf model-based offline RL algorithms like MOPO and AMOReL and theoretically analyze their performance bounds after introducing SUMO. Empirically, we conduct extensive experiments on the D4RL (Fu et al. 2020) benchmark, and the experimental results indicate that SUMO can significantly enhance the performance of base algorithms. We also show that SUMO can provide more accurate uncertainty estimation than commonly used model ensemble-based methods. Our contributions can be summarized as follows:

- We propose a novel search-based uncertainty estimation method for model-based offline RL, SUMO.
- We combine SUMO with AMOReL and MOPO, and provide theoretical performance bounds.
- We empirically demonstrate that SUMO incurs accurate uncertainty estimation and can bring significant performance improvement over the base algorithms on numerous D4RL datasets.

Background

Reinforcement Learning (RL). We consider a Markov Decision Process (MDP) (Garcia and Rachelson 2013) modeled by $\mathcal{M} = \langle S, A, r, P, \rho, \gamma \rangle$, where S is the state space, A is the action space, r is the reward function: $S \times A \to \mathbb{R}$, P is the transition dynamics: $S \times A \times S \to [0, 1]$, ρ is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. RL aims to get a policy $\pi_\theta(a|s)$ that maximizes the cumulative expected discounted return: $J_\rho(\pi, \mathcal{M}) = \max_{\pi_\theta} \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 \sim \rho\right]$. We specify $\mathcal{M}$ to stress that $J_\rho(\pi, \mathcal{M})$ is obtained in the MDP $\mathcal{M}$.

Offline RL. In offline RL, the agent aims to learn the optimal batch-constraint policy based on a static dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^{N}$, where the samples are collected by a (unknown) behavior policy μ, without accessing the environment. Since the dataset can not cover the entire state-action space, the performance of offline RL is often limited.

Model-based Offline RL. Model-based offline RL leverages the learned dynamics model $P_{\widehat{\mathcal{M}}}(\cdot|s, a)$ to generate synthetic samples. This allows us to train the policy using both dataset transitions and samples generated by the dynamics model. We denote the model MDP as $\widehat{\mathcal{M}}$, the state distribution probability at timestep t following policy π and dynamics $P_{\widehat{\mathcal{M}}}$ as $\mathbb{P}^{\pi}_{\widehat{\mathcal{M}},t}(s)$, and discounted occupancy measure of π under dynamics $P_{\widehat{\mathcal{M}}}$ as $\hat{\rho}^{\pi}(s, a) := \pi(a|s) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^{\pi}_{\widehat{\mathcal{M}},t}(s)$.

Related Work

Model-free Offline RL. Model-free offline RL aims to train an agent merely on a static dataset without a dynamics model. The major challenge for model-free offline RL is extrapolation error (Fujimoto, Meger, and Precup 2019; Kumar et al. 2020; Lyu et al. 2022) of the Q-function due to the inability to explore. Existing methods introduce conservatism to the policy (Fujimoto and Gu 2021; Kumar et al. 2019; Fujimoto, Meger, and Precup 2019) or Q-function (Kumar et al. 2020; Lyu et al. 2022; An et al. 2021; Yang et al. 2024) to tackle the issue. However, the involvement of conservatism restricts the generalization ability of the policy.

Model-based Offline RL. Model-based offline RL leverages a dynamics model to extend the dataset and enhance the generalization ability. Since the dynamics model may not be accurate on all transitions, conservatism is still necessary for learning a good policy. Some works (Yu et al. 2020; Kidambi et al. 2020) incorporate conservatism into the generated samples, while other works (Yu et al. 2021; Rigter, Lacerda, and Hawes 2022; Liu et al. 2023) introduce conservatism to Q-function.

Uncertainty Estimation in Offline RL. In model-free offline RL, uncertainty measures the distance of samples from the dataset distribution. EDAC (An et al. 2021) leverages an ensemble of Q-networks to estimate the uncertainty. DARL (Zhang et al. 2023a) computes the KNN distances between the samples and the dataset as uncertainty estimation. In model-based offline RL, uncertainty measures the discrepancy between the simulated dynamics and the true dynamics. MOPO (Yu et al. 2020) and MOReL (Kidambi et al. 2020) estimate the uncertainty by model ensemble. MOBILE (Sun et al. 2023) quantifies the uncertainty through model-bellman inconsistency. All of these uncertainty estimation methods rely on model ensemble, while our method does not. Some other uncertainty estimation methods (Kim and Oh 2023; Tennenholtz and Mannor 2022) suffer from heavy computational burden and complex implementation, while our method is both efficient and easy to implement.

Methodology

Search-Based Method for Uncertainty Estimation

In this part, we formally present our search-based uncertainty estimation method for model-based offline RL. Firstly, we outline how we model the uncertainty estimation problem as an estimation of cross entropy. Subsequently, we employ a particle-based entropy estimator, which approximates the calculation of cross entropy as a search problem.

We characterize the uncertainty of the model dynamics as its cross entropy against the true dynamics:

$\mathcal{H}(P_{\widehat{\mathcal{M}}}(\cdot|s,a), P(\cdot|s,a)) = -\sum P_{\widehat{\mathcal{M}}}(\cdot|s,a) \log P(\cdot|s,a)$, where $P_{\widehat{\mathcal{M}}}(\cdot|s,a)$ and $P(\cdot|s,a)$ are the model dynamics and the true dynamics, respectively. It is hard to calculate $\mathcal{H}(P_{\widehat{\mathcal{M}}}(\cdot|s,a), P(\cdot|s,a))$ since we have no knowledge about the true dynamics $P(\cdot|s,a)$. We therefore compute the *cross entropy* between the model dynamics and the dataset dynamics: $\mathcal{H}(P_{\widehat{\mathcal{M}}}(\cdot|s,a), P_d(\cdot|s,a)) = -\sum P_{\widehat{\mathcal{M}}}(\cdot|s,a) \log P_d(\cdot|s,a)$, where $P_d(\cdot|s,a)$ denotes the dataset transition dynamics of dataset MDP, which is formally defined below.

Definition 1 (dataset MDP)**.** *The dataset MDP is defined by the tuple $\hat{\mathcal{M}}_d = (S, A, r, P_d, \rho_d, \gamma)$, where S, A, r and γ are the same as the original MDP. The dataset transition dynamics P_d is defined as:*

$$P_d(\cdot|s,a) = \begin{cases} P(\cdot|s,a), & \text{if } (s,a) \in \mathcal{D}, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

and ρ_d is the state distribution of the dataset.

Remark: Intuitively, P_d only accounts for transitions that lie in the dataset; we set the probability for transitions not in the dataset to be zero. Note that the summation of P_d for a given state-action pair (s,a) may be not 1, we can scale P_d accordingly to make it a valid probability density function.

Then based on the definition of cross entropy, we have $\mathcal{H}(P_{\widehat{\mathcal{M}}}, P_d) = -\frac{1}{n}\sum_{i=1}^{n} \log P_d(x_i)$, where $\{x_i\}_{i=1}^n \sim P_{\widehat{\mathcal{M}}}$. According to the particle-based entropy estimator (Singh et al. 2003), given a dataset $\{y_j\}_{j=1}^m \sim P_d$, we have $P_d(x_i) = \frac{k\Gamma(d/2+1)}{m\pi^{d/2}R_{i,k,m}^d}$, where d is the dimension of x_i, Γ is the Gamma function, and $R_{i,k,m}^d = \|x_i - x_i^{k,m}\|_2^d$ represents the distance between x_i and the k-nearest neighbor of x_i in $\{y_j\}_{j=1}^m$. Then, we can compute the cross entropy as:

$$\begin{aligned} \mathcal{H}(P_{\widehat{\mathcal{M}}}, P_d) &= -\frac{1}{n}\sum_{i=1}^{n} \log \frac{k\Gamma(d/2+1)}{m\pi^{d/2}R_{i,k,m}^d} \\ &\propto \frac{1}{n}\sum_{i=1}^{n} \log \|x_i - x_i^{k,m}\|_2 \end{aligned} \tag{2}$$

Notice that Equation (2) is an uncertainty estimation of the model dynamics. To estimate the uncertainty of a sample x_i generated by the dynamics model, we can assume $P_{\widehat{\mathcal{M}}}$ is a delta distribution, where $P_{\widehat{\mathcal{M}}}(x_i) = \infty$, $P_{\widehat{\mathcal{M}}}(x) = 0$ for any $x \neq x_i$. Thus, we transform the problem of estimating the uncertainty of synthetic samples into a KNN search problem. To be specific, given an offline dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^N$ and a synthetic sample $(\hat{s}, \hat{a}, \hat{r}, \hat{s}')$, we can estimate the uncertainty of the synthetic sample as:

$$u(\hat{s}, \hat{a}, \hat{s}') = \log\left(\|(\hat{s} \oplus \hat{a} \oplus \hat{s}') - (\hat{s} \oplus \hat{a} \oplus \hat{s}')^{k,N}\|_2\right) \tag{3}$$

where $\oplus$ is the vector concatenation operator, $(\hat{s} \oplus \hat{a} \oplus \hat{s}')^{k,N}$ returns the nearest neighbor of the query sample $(\hat{s}, \hat{a}, \hat{r}, \hat{s}')$ in the offline dataset, k is the number of neighbors for KNN search, and N is the number of samples in the dataset. To ensure the estimated uncertainty is non-negative, we practically compute the uncertainty as:

$$u(\hat{s}, \hat{a}, \hat{s}') = \log\left(\|(\hat{s} \oplus \hat{a} \oplus \hat{s}') - (\hat{s} \oplus \hat{a} \oplus \hat{s}')^{k,N}\|_2 + 1\right) \tag{4}$$

Note that one can also involve the rewards in Equation (4) for computing the uncertainty, which we empirically do not find obvious performance difference in Appendix D.3. To ensure time efficiency, we employ FAISS (Johnson, Douze, and Jégou 2019), an efficient GPU-based KNN search method to estimate the uncertainty. Note that using KNN search to calculate distances between given samples and datasets is a widely used approach in model-free offline RL (Zhang et al. 2023a; Ran et al. 2023; Lyu et al. 2024). However, they often rely on constraining the distance between the learned policy and the behavior policy, and the concatenated vector only includes dimensions of s and a. In contrast, for model-based offline RL, what matters is the discrepancy between the model dynamics and the true dynamics. Therefore, we need to consider the influence of s'.

We then discuss the advantage of SUMO compared with model ensemble-based methods. First, SUMO estimates the uncertainty in an unsupervised learning manner, meaning that the estimated uncertainty is stable and not correlated with the training process. Second, model ensemble-based uncertainty estimation is an empirical approximation. A mismatch can be observed in theoretical analysis of prior works, e.g., $D_{TV}(P_{\widehat{\mathcal{M}}}(\cdot|s,a), P(\cdot|s,a)) \leq \alpha$ is required in MOReL, where D_{TV} is the total variation distance and α is the threshold. Model ensemble-based uncertainty estimation can only practically approximate the bound of D_{TV} with u. In contrast, SUMO provides an unbiased estimate of cross entropy, relationships between u and D_{TV} can be established using Pinsker's inequality (Csiszár and Körner 2011), as shown in Appendix A.2.

Integrating SUMO into Existing Methods

As an uncertainty estimation method, SUMO is compatible with any model-based offline RL algorithm that requires an estimation of uncertainty. There are two typical ways of using uncertainty in model-based offline RL, using the uncertainty as the reward penalty, e.g., MOPO (Yu et al. 2020), and using the uncertainty to truncate synthetic trajectories from the learned model, e.g., MOReL (Kidambi et al. 2020). In this part, we introduce the procedure of combining SUMO with MOPO and Adapted MOReL, respectively.

MOPO with SUMO MOPO adopts the uncertainty as the reward penalty, and we can simply replace the uncertainty estimator as the SUMO estimator. The detailed pseudocode of MOPO+SUMO can be found in Appendix B, and the detailed process can summarized as:

Step1: Learning the Environmental Dynamics Model: To generate synthetic samples for training, we construct an environmental dynamics model to simulate the true dynamics. Following previous works (Yu et al. 2020; Kidambi et al. 2020), we model the dynamics model as a multi-layer neural network which outputs a Gaussian distribution over the next state and reward: $\hat{P}_\psi(s', r|s,a) = \mathcal{N}(\mu_\psi(s,a), \Sigma_\psi(s,a))$. We use an ensemble dynamics model which consists of N ensemble members: $\hat{P}_\psi = \{\hat{P}_{\psi_i}\}_{i=1}^N$. It is worth mentioning that we use the ensemble dynamics model solely to reduce the error in model predictions, not for estimating the uncertainty. Then we can train each ensemble member using the

offline dataset via maximum log-likelihood:

$$\mathcal{L}_{\psi_i} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[-\log \hat{P}_{\psi_i}(s', r|s, a)\right] \tag{5}$$

Step2: Reward Penalty Calculation with SUMO: We utilize the current policy π and the dynamics model $\hat{P}_\psi$ to generate synthetic samples. For each generated sample (s, a, s'), we first calculate its unnormalized uncertainty with SUMO via Equation (4). We assume that the reward function satisfies: $0 \leq r(s, a) \leq r_{max}, \forall(s, a)$. This is easy to achieve in practice. Then we normalize $u(s, a, s')$ to the range $[0, r_{max}]$:

$$\hat{u}(s, a, s') = \frac{u(s, a, s')}{\max_{(s,a,s')} u(s, a, s')} \times r_{max} \tag{6}$$

We adopt $\hat{u}$ to penalize the rewards, $\hat{r} = r - \lambda\hat{u}$, where λ is a hyperparameter that controls the magnitude of the penalty. We can then add the synthetic samples with reward penalty to the synthetic dataset $\mathcal{D}_{model}$.

Step 3: Model-based Policy Optimization: We then follow MOPO to train SAC (Haarnoja et al. 2018) agent with samples drawn from the offline dataset $\mathcal{D}$ and synthetic dataset $\mathcal{D}_{model}$. Typically, we set a sampling coefficient $\eta \in [0, 1]$, sampling a proportion of $\eta\mathcal{B}$ from the offline dataset $\mathcal{D}$, and a proportion of $(1 - \eta)\mathcal{B}$ from the synthetic dataset $\mathcal{D}_{model}$ given the batch size $\mathcal{B}$.

Adapted MOReL with SUMO MOReL calculates model ensemble discrepancy as an uncertainty estimation to determine whether a state-action pair (s, a) is within a known region and penalizes samples from unknown regions. We can simply replace the uncertainty estimation method with SUMO to determine whether a transition (s, a, s') is reliable. In our practical implementation, we make slight modifications to MOReL. Specifically, when encountering unreliable samples, we directly truncate the trajectory instead of applying a constant penalty. This ensures the reliability of the samples used for training. And we update the policy via SAC (Haarnoja et al. 2018) instead of planning. We refer to this modified version as Adapted MOReL (AMOReL), and divide the process of SUMO with AMOReL into three steps:

Step 1: Constructing the Environmental Dynamics Model: Following the same procedure of MOPO+SUMO, we train an ensemble dynamics model $\hat{P}_\psi = \{\hat{P}_{\psi_i}\}_{i=1}^N$.

Step 2: Trajectory Truncation with SUMO: We then utilize the current policy π and dynamics model $\hat{P}_\psi$ to generate synthetic samples. We use SUMO to measure the uncertainty of the samples and truncate the synthetic trajectory accordingly. In specific, in the process of generating synthetic trajectories, we apply Equation (4) to estimate the uncertainty of each generated sample $(\hat{s}, \hat{a}, \hat{s}')$ and set a truncating threshold ϵ. If the uncertainty of any sample exceeds this threshold, we consider the sample unreliable and stop generating the trajectory, adding the generated trajectory to the synthetic dataset $\mathcal{D}_{model}$. In this way, we ensure using only reliable samples for training. It is important to decide how to choose the truncating threshold ϵ, we can choose to set the threshold as a constant; however, since data distributions and dimensions may vary across different datasets, setting it as a constant might not be the optimal choice. Therefore, **we set the threshold to be the maximum uncertainty among all the dataset samples**. In specific, for a dataset $\mathcal{D}$, we compute the KNN distances for each sample in $\mathcal{D}$ to other samples. We take the maximum uncertainty value as the truncating threshold. For flexibility, we can also multiply the threshold ϵ by a coefficient α for adjustment:

$$\epsilon = \alpha \cdot \max_{\mathcal{D}} \log\left(\|(s \oplus a \oplus s') - (s \oplus a \oplus s')^{k,N}\|_2 + 1\right) \tag{7}$$

where α is a hyperparameter, a larger α makes the algorithm more optimistic, using more synthetic samples for training.

Step 3: Model-based Policy Optimization: We then draw samples from $\mathcal{D} \cup \mathcal{D}_{model}$ with a sampling coefficient η to train an SAC agent following **Step 3** of MOPO+SUMO.

We summarize the full pseudocode of AMOReL+SUMO in Appendix B.

Theoretical Analysis

In this part, we provide theoretical analysis for SUMO. Due to space limit, missing proofs are deferred to Appendix A.

We first provide theoretical analysis when leveraging SUMO for penalizing rewards in MOPO. We show that the policy learned by the MOPO+SUMO has a performance guarantee as follows:

Theorem 1 (Informal)**.** *Denote the behavior policy of the dataset as μ, the uncertainty estimator $u(s, a, s')$ defined in Equation (4), then under some mild assumptions, MOPO+SUMO incurs the policy π that satisfies:*

$$J_\rho(\pi) \geq J_\rho(\mu) - 2\lambda\mathbb{E}_{\hat{\rho}^\pi}[u(s, a, s')],$$

where $\lambda \in \mathbb{R}$ is the penalty coefficient term in MOPO, $\hat{\rho}^\pi$ is the discounted occupancy measure of policy π. This theorem states that in the original MDP $\mathcal{M}$, the policy induced by MOPO+SUMO can be at least as good as the behavior policy if the uncertainty is small.

We then present theoretical guarantees when using SUMO for trajectory truncation in AMOReL. We first define the ϵ-Model MDP:

Definition 2 (ϵ-Model MDP)**.** *ϵ-Model MDP is defined by the tuple $\widehat{\mathcal{M}}_\epsilon = \langle S, A, r, \hat{P}_\epsilon, \hat{\rho}_\epsilon, \gamma\rangle$, where S, A, r and γ are the same as the original MDP, $\hat{\rho}_\epsilon$ is the state distribution of the learned dynamics model. The transition dynamics $\hat{P}_\epsilon$ is defined as:*

$$\hat{P}_\epsilon(s'|s, a) = \begin{cases} P_{\widehat{\mathcal{M}}}(s'|s, a), & \text{if } u(s, a, s') < \epsilon, \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

Intuitively, $\widehat{\mathcal{M}}_\epsilon$ only contains transitions satisfying $u(s, a, s') < \epsilon$. In this way, we ensure the reliability of synthetic samples for training. We note that the summation of $\hat{P}_\epsilon$ may be not 1, and we can scale it to make it a valid probability density function. Then we present our theoretical results of performance bounds for ϵ-Model MDP.

Theorem 2 (Performance bounds)**.** *Denote ρ_d as the state distribution of the dataset. For any policy π, its return in*

the ϵ-Model MDP $\widehat{\mathcal{M}}_\epsilon$ and the original MDP $\mathcal{M}$ (with state distribution ρ) satisfies:

$$J_{\hat{\rho}_\epsilon}(\pi, \widehat{\mathcal{M}}_\epsilon) \geq J_\rho(\pi, \mathcal{M}) - \frac{r_{max}}{1-\gamma}(1 + \sqrt{\frac{\epsilon}{2}} + 2D_{TV}(\rho, \rho_d) + 2D_{TV}(\hat{\rho}_\epsilon, \rho_d)),$$

$$J_{\hat{\rho}_\epsilon}(\pi, \widehat{\mathcal{M}}_\epsilon) \leq J_\rho(\pi, \mathcal{M}) + \frac{r_{max}}{1-\gamma}(\sqrt{\frac{\epsilon}{2}} + 2D_{TV}(\rho, \rho_d) + 2D_{TV}(\hat{\rho}_\epsilon, \rho_d))$$

Remark: Theorem 2 states that the performance difference for any policy π in ϵ-Model MDP and the original MDP is related to three factors: (1) the truncation threshold ϵ, which determines the synthetic samples for training. (2) the total variance distance between the true and dataset state distribution $D_{TV}(\rho, \rho_d)$. (3) the total variance distance between ϵ-Model and dataset state distribution $D_{TV}(\hat{\rho}_\epsilon, \rho_d)$.

Experiment

In our experimental part, we empirically evaluate our method, SUMO. We aim to answer the following questions: (1) How much performance gain can SUMO bring to off-the-shelf model-based offline RL algorithms? (2) Can SUMO provide more accurate uncertainty estimation compared to model ensemble-based methods? (3) How different design choices affect the performance of SUMO? (4) How sensitive is SUMO to the introduced hyperparameters?

For evaluation, we use the D4RL MuJoCo datasets which includes three tasks: *halfcheetah*, *hopper* and *walker2d*. Each task provides five types of offline datasets: *random*, *medium*, *medium-replay*, *medium-expert* and *expert*. We conduct experiments on 15 D4RL MuJoCo datasets to comprehensively evaluate the performance of SUMO across various tasks and datasets of different qualities.

Experimental Results on the D4RL Benchmark

SUMO is an uncertainty estimation method designed for model-based offline RL, it can be seamlessly combined with any model-based offline RL algorithm which needs uncertainty estimation. In this part, we combine SUMO with several model-based offline RL algorithms, including MOPO, AMOReL, MOReL and MOBILE (Sun et al. 2023). We conduct extensive experiments on widely used D4RL MuJoCo datasets and examine whether SUMO can bring performance improvement to these base algorithms. We run all experiments with 5 different random seeds.

We summarize the experimental results in Table 1, where we observe that SUMO significantly boosts the performance of base algorithms. Notably, MOPO+SUMO outperforms vanilla MOPO in **11** out of 15 tasks, AMOReL+SUMO and MOReL+SUMO surpasses the original AMOReL and MOReL in **10** out of 15 tasks, MOBILE+SUMO achieves a higher score than MOBILE in **9** out of 15 tasks. Regarding the average score on all 15 tasks, SUMO brings an overall performance improvement for all four base algorithms, indicating the versatility of SUMO. We include more comparison with model-based methods without uncertainty estimation such as COMBO (Yu et al. 2021) and RAMBO (Rigter, Lacerda, and Hawes 2022) in Appendix D.5.

We also evaluate our methods on D4RL Antmaze tasks, which are challenging for model-based offline RL methods (Wang et al. 2021). Due to space limit, we present the results in Appendix D.1. We find that base model-based RL algorithms all struggle to perform well in the Antmaze domains. When combined with SUMO, the performance of base algorithms has been improved. We believe these further demonstrate the strengths of SUMO.

Comparison with other Uncertainty Estimators

In this section, we aim to demonstrate the superiority of SUMO in a more intuitive manner, i.e., whether SUMO can provide more accurate uncertainty estimation compared to widely used model ensemble-based uncertainty estimation methods. This is crucial and necessary for validating our central claim in this paper.

For model ensemble-based uncertainty estimation methods, the core idea is to measure the uncertainty of generated samples by leveraging the diverse predictions of ensemble members regarding the environmental dynamics. We choose the following model ensemble-based uncertainty estimation methods for comparison, which come from recent literature in both offline and online model-based RL:

Max Aleatoric: It measures the sample uncertainty as: $u(s,a) = \max_{i=1,\dots,N} \|\Sigma_{\psi_i}(s,a)\|_F$, where Σ_{ψ_i} is the covariance matrix predicted by the i-th ensemble member, and $\|\cdot\|_F$ denotes the Frobenius norm. This uncertainty heuristic is used in MOPO and is related to the aleatoric uncertainty.

Max Pairwise Diff: It measures the sample uncertainty as: $u(s,a) = \max_{i,j} \left\|\mu_{\psi_i}(s,a) - \mu_{\psi_j}(s,a)\right\|_2$, μ_{ψ_i} is the mean vector predicted by the i-th ensemble member. This uncertainty heuristic is used in MOReL and measures the maximum pairwise difference between ensemble predictions.

LOO (Leave-One-Out) KL Divergence: It measures the uncertainty as: $u(s,a) = D_{KL}(\hat{P}_{\psi_i}(\cdot|s,a)||\hat{P}_{\psi_{-i}}(\cdot|s,a))$ where $\hat{P}_{\psi_{-i}}(\cdot|s,a)$ represents the aggregated Gaussian distribution of all ensemble members except the i-th member. It is used in M2AC (Pan et al. 2020).

We then compare the ability of these four methods in detecting out-of-distribution (OOD) samples. In offline RL, OOD samples refer to state-action pairs that lie out of the sample distribution of the dataset. It is difficult to decide whether a transition is truly OOD. Nevertheless, things are easier in model-based offline RL, mainly due to the fact that it involves learning an environmental dynamics model. The difference between the simulated environmental dynamics and the real environmental dynamics can be a signal for detecting OOD samples. In practice, we start from the tuple (s,a) sampled from the offline dataset and generate its next state $\hat{s}'$ via the learned dynamics model. We deem that the transition $(s,a,\hat{s}')$ is OOD if $\hat{s}'$ significantly differs from the dynamics of the real environment. A good uncertainty estimation method should be sensitive to unrealistic environmental dynamics. In other words, when there is a large error in the predicted environmental dynamics, it should provide a higher uncertainty estimate, and when the error is small, it should have a lower uncertainty estimate.

Empirically, we conduct experiments on the 9 datasets

Task Name	MOPO		AMOReL		MOReL		MOBILE	
	SUMO	Base	SUMO	Base	SUMO	Base	SUMO	Base
half-r	**37.2±1.9**	34.9±1.4	**44.2±2.1**	31.8±2.4	**37.3±2.1**	29.8±1.2	34.9±2.1	**37.8±2.9**
hopper-r	**24.5±0.9**	19.4±0.7	29.7±0.6	**32.4**±1.2	**33.2±0.7**	30.1±1.0	30.8±0.9	**32.6 ± 1.2**
walker2d-r	11.4±1.3	**13.1±1.1**	20.3±0.2	**21.0±0.3**	17.8±0.6	**19.4±0.3**	**27.9±2.0**	16.3± 4.6
half-m-r	**73.1±2.1**	65.0±3.3	**76.8±2.7**	49.6±2.3	**67.9±2.5**	51.2±1.9	**76.2±1.3**	67.9±2.0
hopper-m-r	**65.4±3.2**	38.8±2.4	**88.7±1.3**	80.5±1.0	**83.9±1.3**	76.3±1.0	**109.9±1.4**	104.9±0.9
walker2d-m-r	70.3±0.6	**74.8±1.5**	**65.3±2.7**	46.0±1.9	**61.3±3.1**	48.1±4.2	78.2±1.5	**83.9±1.3**
half-m	68.9±2.3	**73.1±2.7**	**82.1±2.8**	69.2±1.2	57.9±1.2	**62.4±1.3**	**84.3±2.4**	75.1±1.5
hopper-m	**74.6±1.9**	45.6±2.5	**95.0±2.1**	87.2±3.4	82.1±1.4	**84.7±3.1**	**104.8±2.1**	102.9±1.9
walker2d-m	**57.3±1.6**	42.3±0.8	67.4±0.9	**71.2±1.3**	**77.1±3.5**	67.6±2.2	**94.1±2.5**	89.1±1.0
half-m-e	**84.1±1.4**	76.6±1.0	**99.4±3.6**	90.6±2.1	**98.6±3.5**	92.3±4.6	106.6±2.4	**109.2±3.8**
hopper-m-e	**88.1±1.9**	69.1±1.2	101.5±0.4	**106.2±1.5**	**105.8±1.4**	102.4±0.9	107.8±0.7	**110.1±1.3**
walker2d-m-e	**81.9±1.6**	75.4±1.1	**109.6±0.7**	92.3±0.9	86.1±1.8	**90.4±1.4**	**122.8±0.4**	115.9±0.8
half-e	87.1±1.2	**88.7±1.6**	**112.3±2.5**	103.2±1.9	**109.9±2.1**	105.8±1.6	111.5±1.5	**113.1±2.1**
hopper-e	**101.2± 1.8**	83.9±0.7	**105.4±0.6**	94.5±0.3	**101.8±1.9**	92.5±1.0	**115.9 ± 2.9**	112.4±3.5
walker2d-e	**114.4±1.1**	95.3±3.4	106.3±1.3	**107.2±1.0**	106.2±1.5	**108.3±2.1**	**116.3±1.5**	113.7±1.1
Average score	**69.3**	59.7	**80.3**	72.2	**75.1**	70.7	**88.2**	85.6

Table 1: Normalized average score comparison between vanilla base algorithms and the version with SUMO, on top of MOPO, AMOReL, MOReL and MOBILE on 15 D4RL MuJoCo datasets, and the version of datasets we use is "-v2". We abbreviate "halfcheetah" as "half", "random" as "r", "medium" as "m", "medium-replay" as "m-r", "medium-expert" as "m-e" and "expert" as "e". We run each algorithm for 1M gradient steps with 5 random seeds. We report the final average performance and ± captures the standard deviation. Bold numbers represent the best average scores within each group.

Task Name	Max Aleatoric		Max Pairwise Diff		LOO KL Divergence		SUMO(Ours)	
	ρ	r	ρ	r	ρ	r	ρ	r
halfcheetah-random-v2	0.63	0.58	0.58	0.47	0.10	0.06	**0.84**	**0.83**
hopper-random-v2	0.76	0.71	**0.85**	**0.77**	0.13	0.09	0.82	0.74
walker2d-random-v2	**0.75**	0.62	0.71	0.58	0.14	0.10	0.73	**0.70**
halfcheetah-medium-replay-v2	0.70	0.65	0.73	0.67	0.21	0.16	**0.82**	**0.68**
hopper-medium-replay-v2	0.73	0.70	0.79	0.67	0.18	0.08	**0.87**	**0.80**
walker2d-medium-replay-v2	0.66	0.54	0.65	0.59	0.09	0.06	**0.88**	**0.86**
halfcheetah-medium-v2	0.69	0.59	0.65	0.57	0.15	0.11	**0.84**	**0.77**
hopper-medium-v2	0.74	0.66	0.70	0.63	0.11	0.05	**0.86**	**0.79**
walker2d-medium-v2	0.68	0.56	0.74	**0.73**	0.11	0.08	**0.82**	0.71
halfcheetah-medium-expert-v2	0.67	0.63	0.70	0.68	0.13	0.07	**0.88**	**0.81**
hopper-medium-expert-v2	0.71	0.68	**0.78**	**0.72**	0.22	0.14	0.75	0.68
walker2d-medium-expert-v2	0.62	0.59	0.65	0.60	0.08	0.06	**0.84**	**0.80**
halfcheetah-expert-v2	**0.79**	**0.75**	0.69	0.67	0.12	0.10	0.76	0.68
hopper-expert-v2	0.73	0.64	0.76	0.67	0.17	0.14	**0.86**	**0.80**
walker2d-expert-v2	0.63	0.57	0.65	0.58	0.09	0.05	**0.83**	**0.76**

Table 2: Spearman rank (ρ) and Pearson bivariate (r) correlations comparison of SUMO against Max Aleatoric, Max Pairwise Diff and LOO KL Divergence. We choose these two metrics following (Lu et al. 2021). The closer ρ and r are to 1, the better the uncertainty estimation. Each method is evaluated by 5 runs and we report the average ρ and r. Bolded numbers represent the method with the highest average ρ and r on each dataset.

from D4RL MuJoCo tasks. On each dataset, we first train an ensemble dynamics model via supervised learning. Then we train a policy inside the model following the process of MOPO, without adding any penalty. To generate synthetic samples, we randomly select 100 states from the dataset as initial states. For each initial state, we use the trained policy to perform rollouts in the dynamics model. To ensure the generation of OOD samples, we set the rollout horizon to be 100 (the dynamics model tends to output bad transitions under larger rollout horizon due to compounding error). In total, we obtain 10,000 synthetic transitions for each dataset. For any synthetic transition $(s, a, \hat{s}')$, we can replay the state-action pair (s, a) in the real environment to obtain the true next state s'. Then we can calculate the L2-norm error between s' and $\hat{s}'$: $\|s' - \hat{s}'\|_2$, and this can reflect the difference between the true environmental dynamics and the

Task Name	$s \oplus s'$		$s \oplus a$		$s \oplus a \oplus s'$	
	ρ	r	ρ	r	ρ	r
half-m	0.51	0.44	0.62	0.58	**0.84**	**0.77**
hopper-m	0.50	0.48	0.55	0.52	**0.86**	**0.79**
walker2d-m	0.64	0.49	0.59	0.51	**0.82**	**0.71**
half-m-e	0.57	0.54	0.63	0.61	**0.88**	**0.81**
hopper-m-e	0.60	0.57	0.69	0.63	**0.75**	**0.68**
walker2d-m-e	0.52	0.46	0.67	0.59	**0.84**	**0.80**
Mean	0.56	0.50	0.63	0.57	**0.83**	**0.76**

Table 3: Spearman rank (ρ) and Pearson bivariate (r) correlations comparison of SUMO with different search vectors. We run each experiment by 5 different random seeds and report the average ρ and r.

simulated dynamics. We can then use the aforementioned methods to estimate the uncertainty of the synthetic transition. For a good uncertainty estimation method, we expect a strong correlation between the estimated uncertainty and the L2-norm error of the next state. The remaining question is how to measure the correlation. Following previous work (Lu et al. 2021), we use Spearman rank (ρ) and Pearson bivariate (r) correlations, where ρ captures the rank correlations and r measures the linear correlations.

For each dataset, we calculate ρ and r on the synthesized 10,000 transitions and present the results in Table 2. It is evident that the advantages of SUMO are significant. SUMO achieves the best ρ or r metrics in **12** out of 15 datasets compared with other model ensemble-based methods. We then conclude that SUMO incurs a better uncertainty estimation.

Ablation Study & Parameter Study

In this part, we examine different design choices of SUMO and investigate the sensitivity of SUMO to the introduced hyperparameters. For the design choices of SUMO, we mainly focus on two aspects: (a) the components included in the search vector; and (b) the choice of distance measure.

The Choice of Search Vector: Recall that in Equation (4), we use $(s \oplus a \oplus s')$ as the search vector for KNN search because this reflects the difference between simulated dynamics and real dynamics. What if we replace the search vector with $(s \oplus s')$ or $(s \oplus a)$?

We design experiments to investigate the effectiveness of these three choices of search vectors. We follow the experimental settings in Section and conduct experiments with these search vectors on 6 D4RL MuJoCo datasets. We also use ρ and r as evaluation metrics. We present the experimental results in Table 3. It is evident that using $(s \oplus a \oplus s')$ as the search vector is superior to using $(s \oplus s')$, or $(s \oplus a)$. The scores in terms of ρ and r on all six datasets are the highest when using $(s \oplus a \oplus s')$ among the three choices. It turns out that it is vital to include the action a for measuring uncertainty. The inclusion of the next state is also necessary because the learned dynamics model is responsible for predicting the next state given (s, a).

The Choice of Distance Measure: Different distance measure for KNN search can induce different uncertainty estimation. To examine whether SUMO is sensitive to different distance measure, we change the default Euclidean distance we use to Manhattan distance and Cosine similarity, and conduct experiments on six MuJoCo datasets, based on MOPO+SUMO. The results are presented in Table 4. We can observe using Euclidean distance is slightly better than other two distance measure, but there is no obvious performance distinction, so we can simply use Euclidean distance as distance measure.

Task Name	Euclidean	Manhattan	Cosine
half-m	**68.9**	67.4	68.0
hopper-m	**74.6**	72.9	73.8
walker2d-m	57.3	**59.1**	56.2
half-m-e	**84.1**	83.9	82.1
hopper-m-e	88.1	**90.4**	88.6
walker2d-m-e	**81.9**	79.4	77.2
Average Score	**75.8**	75.5	74.3

Table 4: Performance comparison of MOPO+SUMO between difference distance measure. Each experiment is run with 5 seeds.

Task Name	$k = 1$		$k = 5$		$k = 10$	
	ρ	r	ρ	r	ρ	r
half-m	0.84	0.77	0.81	0.75	**0.86**	**0.78**
hopper-m	**0.86**	0.79	0.84	**0.81**	0.84	0.80
walker2d-m	0.82	0.71	**0.85**	**0.73**	0.81	0.70
half-m-e	**0.88**	**0.81**	0.86	0.77	0.84	0.79
hopper-m-e	0.75	0.68	**0.77**	0.71	0.74	**0.73**
walker2d-m-e	**0.84**	**0.80**	0.82	0.76	0.80	0.77
Mean	**0.83**	0.76	0.82	0.76	0.81	**0.77**

Table 5: Spearman rank (ρ) and Pearson bivariate (r) correlations comparison of SUMO with different values of k. We run each experiment by 5 different seeds and report the average ρ and r.

Number of Neighbors k: The main hyperparameter in SUMO is the number of nearest neighbors k in KNN search. To examine its influence on the uncertainty estimation, we sweep k across $\{1, 5, 10\}$ and run experiments on selected MuJoCo datasets. We use metrics of ρ and r. The results are shown in Table 5. We find that SUMO is robust to k.

Conclusion

In this paper, we propose SUMO, a novel search-based uncertainty estimation method for model-based offline RL. SUMO characterizes the uncertainty as the cross entropy between simulated dynamics and true dynamics, and employ a practical KNN search method for implementation. We show that SUMO can provide a better uncertainty estimation than model ensemble-based methods and boost the performance of a variety of base algorithms.

Acknowledgements

This work was supported by the STI 2030-Major Projects under Grant 2021ZD0201404. The authors also thank the anonymous reviewers for valuable comments.

References

An, G.; Moon, S.; Kim, J.-H.; and Song, H. O. 2021. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in Neural Information Processing Systems*, 34: 7436–7447.

Csiszár, I.; and Körner, J. 2011. *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press.

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

Fujimoto, S.; and Gu, S. S. 2021. A minimalist approach to offline reinforcement learning.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2052–2062. PMLR.

Garcia, F.; and Rachelson, E. 2013. Markov decision processes. *Markov Decision Processes in Artificial Intelligence*, 1–38.

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

Johnson, J.; Douze, M.; and Jégou, H. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3): 535–547.

Kidambi, R.; Rajeswaran, A.; Netrapalli, P.; and Joachims, T. 2020. Morel: Model-based offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 21810–21823.

Kim, B.; and Oh, M.-h. 2023. Model-based offline reinforcement learning with count-based conservatism. In *International Conference on Machine Learning*, 16728–16746. PMLR.

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

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

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

Liu, X.-Y.; Zhou, X.-H.; Xie, X.-L.; Liu, S.-Q.; Feng, Z.-Q.; Li, H.; Gui, M.-J.; Xiang, T.-Y.; Huang, D.-X.; and Hou, Z.-G. 2023. DOMAIN: MilDly COnservative Model-BAsed OfflINe Reinforcement Learning. *arXiv preprint arXiv:2309.08925*.

Lockwood, O.; and Si, M. 2022. A review of uncertainty for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Lu, C.; Ball, P. J.; Parker-Holder, J.; Osborne, M. A.; and Roberts, S. J. 2021. Revisiting Design Choices in Offline Model-Based Reinforcement Learning. *arXiv preprint arXiv:2110.04135*.

Lyu, J.; Li, X.; and Lu, Z. 2022. Double Check Your State Before Trusting It: Confidence-Aware Bidirectional Offline Model-Based Imagination. In *Neural Information Processing Systems*.

Lyu, J.; Ma, X.; Li, X.; and Lu, Z. 2022. Mildly conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 1711–1724.

Lyu, J.; Ma, X.; Wan, L.; Liu, R.; Li, X.; ; and Lu, Z. 2024. SEABO: A Simple Search-Based Method for Offline Imitation Learning. In *International Conference on Learning Representations*.

Pan, F.; He, J.; Tu, D.; and He, Q. 2020. Trust the model when it is confident: Masked model-based actor-critic. *Advances in Neural Information Processing Systems*, 33: 10537–10546.

Prudencio, R. F.; Maximo, M. R.; and Colombini, E. L. 2023. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*.

Ran, Y.; Li, Y.-C.; Zhang, F.; Zhang, Z.; and Yu, Y. 2023. Policy Regularization with Dataset Constraint for Offline Reinforcement Learning. *arXiv preprint arXiv:2306.06569*.

Rigter, M.; Lacerda, B.; and Hawes, N. 2022. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems*, 35: 16082–16097.

Singh, H.; Misra, N.; Hnizdo, V.; Fedorowicz, A.; and Demchuk, E. 2003. Nearest neighbor estimates of entropy. *American journal of mathematical and management sciences*, 23(3-4): 301–321.

Sun, Y.; Zhang, J.; Jia, C.; Lin, H.; Ye, J.; and Yu, Y. 2023. Model-Bellman inconsistency for model-based offline reinforcement learning. In *International Conference on Machine Learning*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tennenholtz, G.; and Mannor, S. 2022. Uncertainty estimation using riemannian model dynamics for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 19008–19021.

Wang, J.; Li, W.; Jiang, H.; Zhu, G.; Li, S.; and Zhang, C. 2021. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34: 29420–29432.

Yang, K.; Tao, J.; Lyu, J.; and Li, X. 2024. Exploration and Anti-Exploration with Distributional Random Network Distillation. *ArXiv*, abs/2401.09750.

Yu, T.; Kumar, A.; Rafailov, R.; Rajeswaran, A.; Levine, S.; and Finn, C. 2021. Combo: Conservative offline model-based policy optimization. *Neural Information Processing Systems*, 34: 28954–28967.

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

Zhang, H.; Shao, J.; He, S.; Jiang, Y.; and Ji, X. 2023a. DARL: distance-aware uncertainty estimation for offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhang, J.; Lyu, J.; Ma, X.; Yan, J.; Yang, J.; Wan, L.; and Li, X. 2023b. Uncertainty-Driven Trajectory Truncation for Data Augmentation in Offline Reinforcement Learning. In *ECAI 2023*, 3018–3025. IOS Press.