ECAI 2023
K. Gal et al. (Eds.)
© 2023 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA230618

Uncertainty-Driven Trajectory Truncation for Data Augmentation in Offline Reinforcement Learning

Junjie Zhang[1;*], **Jiafei Lyu**[1;*], **Xiaoteng Ma**[2], **Jiangpeng Yan**[2], **Jun Yang**[2], **Le Wan**[3] **and Xiu Li**[1;**]

[1]Tsinghua Shenzhen International Graduate School, Tsinghua University
[2]Department of Automation, Tsinghua University
[3]IEG, Tencent

Abstract. Equipped with the trained environmental dynamics, model-based offline reinforcement learning (RL) algorithms can often successfully learn good policies from fixed-sized datasets, even some datasets with poor quality. Unfortunately, however, it can not be guaranteed that the generated samples from the trained dynamics model are reliable (e.g., some synthetic samples may lie outside of the support region of the static dataset). To address this issue, we propose ***T**rajectory **T**runcation with **U**ncertainty* (TATU), which adaptively truncates the synthetic trajectory if the accumulated uncertainty along the trajectory is too large. We theoretically show the performance bound of TATU to justify its benefits. To empirically show the advantages of TATU, we first combine it with two classical model-based offline RL algorithms, MOPO and COMBO. Furthermore, we integrate TATU with several off-the-shelf model-free offline RL algorithms, e.g., BCQ. Experimental results on the D4RL benchmark show that TATU significantly improves their performance, often by a large margin. Code is available here.

1 Introduction

Offline reinforcement learning (RL) [23] defines the task of learning the optimal policy from a previously gathered fixed-sized dataset by some unknown process. Offline RL eliminates the need for online data collection, which is an advantage over online RL since interacting with the environment can be expensive, or even dangerous, especially in real-world applications. The advances in offline RL raise the opportunity of scaling RL algorithms in domains like autonomous driving [36], healthcare [34], robotics [12], etc.

In offline setting, the collected dataset often provides limited coverage in the state-action space. It is therefore hard for the offline agent to well-evaluate out-of-distribution (OOD) state-action pairs, due to the accumulation of extrapolation error [11] with bootstrapping. Many model-free offline RL algorithms have been proposed to avoid OOD actions, such as compelling the learned policy to stay close to the data-collecting policy (behavior policy) [39, 21, 11], learning conservative value function [22], leveraging uncertainty quantification [1, 40], etc. Nevertheless, these methods often suffer from loss of generalization due to limited size of datasets [38].

Model-based RL (MBRL) methods, instead, improve the generalization of the agent by generating synthetic imaginations with the learned dynamics model. It has been revealed that directly applying model-based online RL methods like MBPO [15] fails on offline datasets [44]. MBPO improves the sample efficiency for online model-based RL by introducing the branch rollout method that queries the environmental dynamics model for short rollouts. However, MBPO fails to resolve the issue of extrapolation error in the offline setting. Modern model-based offline RL methods usually leverage methods like uncertainty quantification [44, 17], penalizing value function to enforce conservatism [43], planning [45], for learning meaningful policies from static logged data. Utilizing the learned dynamics model for offline data augmentation is also explored recently [38, 27]. Nevertheless, there is no guarantee that the generated synthetic samples by the learned model are reliable and lie in the span of the offline dataset. [27] explores a double check mechanism for checking whether the generated samples are reliable, but it requires training bidirectional dynamics models.

In this paper, we propose trajectory truncation with uncertainty (TATU) for reliable imagined trajectory generation. With the estimation of accumulated uncertainty along the trajectory, the trajectory generation process becomes controllable and reliable and it's easy to adapt to different environments without much tuning of hyperparameters, e.g., rollout horizon. Our key idea is illustrated in Figure 1. As shown, we resort to uncertainty-based methods and truncate the trajectory (i.e., the dynamics model will stop producing imaginations) if the accumulated uncertainty along this trajectory is approaching our tolerance range. By doing so, we *enforce conservatism into syn-*

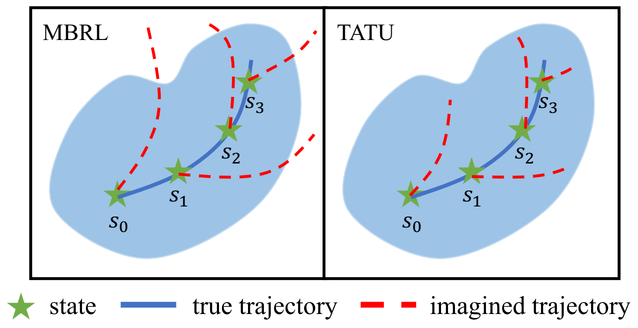

Figure 1. Comparison of TATU against common imagination generation in offline model-based RL (MBRL) methods. TATU truncates the imagined trajectory when the accumulated uncertainty is large and can therefore select good synthetic data in the trajectory. Blue region represents the support region of the static dataset.

* Equal contribution. Work done when Jiafei Lyu was an intern at Tencent.
** Corresponding Author. Email: li.xiu@sz.tsinghua.edu.cn.

thetic data generation by adaptively clipping the imagined trajectory and TATU can thus ensure that the dynamics model outputs reliable data. Generally, TATU involves three steps: (a) training the environmental dynamics model; (b) constructing an ϵ-Pessimistic MDP (see Definition 2); and (c) generating imagined trajectories with the ϵ-Pessimistic MDP. We theoretically show that for *any* policy, its performance in the true MDP is lower bounded by the performance in the ϵ-Pessimistic MDP, which sheds light on applying TATU in practice.

Intuitively, TATU can be combined with model-based offline RL methods like MOPO [44], COMBO [43], etc. Moreover, TATU can also benefit model-free offline RL methods by training an additional rollout policy and serving trustworthy offline data augmentation.

Our contributions can be summarized below:

- We propose a novel uncertainty-based synthetic trajectory truncation method, TATU, which can be easily integrated into existing offline model-based and model-free RL algorithms.
- We theoretically justify the advantages of TATU.
- We combine TATU with two model-based offline RL algorithms, MOPO and COMBO, and observe significant performance improvement upon them.
- We further incorporate TATU with three model-free offline RL algorithms, TD3_BC, CQL, and BCQ. Empirical results show that TATU markedly boosts their performance across a variety of D4RL benchmark tasks.

2 Related Work

Model-free Offline RL. Recent advances in model-free offline RL generally involve importance sampling [25, 32], injecting conservatism into value function [29, 8, 20, 22, 28], learning latent actions [46, 7], forcing the learned policy to be close to the behavior policy [10, 11, 39, 21, 19], adopting uncertainty measurement [1, 40, 3], using one-step method [4], etc. Despite these advances, the generalization capability of the model-free offline RL algorithms is often limited due to the partial coverage of the dataset in the entire state-action space [38, 27].

Model-based Offline RL. Model-based offline RL methods focus on policy learning under a learned dynamics model. To mitigate extrapolation error [11], several methods do not allow the learned policy to visit regions where the discrepancy between the learned dynamics and the true dynamics is large [2, 17, 44, 42]. Some researchers seek to constrain the learned policy to be similar to the behavior policy [5, 30, 37]. Also, sequential modeling is explored in offline RL [6, 16]. Some recent advances propose to leverage model-generated synthetic data for better training [43, 38, 27], or to enhance model training [26, 24, 41, 35, 13], etc. We, instead, aim at offering higher-quality synthetic data for offline training via an uncertainty-based trajectory truncation method. The most relevant to our work are M2AC [33], MOReL [17], and MOPP [45]. MOReL requires the generated single sample (instead of the trajectory) at each step to lie in a safe region. MOPP leverages offline planning and sorts the *entire* trajectory for high rewarding transitions based on uncertainty after planning is done. Both MOReL and MOPP optimize policy in a model-based manner. M2AC is an online MBRL algorithm that rejects the transitions (or trajectories) based on the instance uncertainty and modifies the way of measuring target value. TATU, instead, only serve data augmentation and its imagination generation process can be isolated from policy optimization. TATU truncates the synthetic trajectory based on the *accumulated* uncertainty, and adds penalty to the generated samples. TATU *early stops* the imagined trajectory and puts the whole truncated imagined trajectory into the model buffer for policy optimization.

3 Background

Reinforcement learning (RL) problems can be formulated by a Markov Decision Process (MDP), which consistes of $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, r, P, \rho_0, \gamma \rangle$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the scalar reward function, $P(s'|s,a)$ is the transition probability, ρ_0 is the initial state distribution, $\gamma \in [0,1)$ is the discount factor. The goal of RL is to find a policy $\pi : s \to \Delta(\mathcal{A})$ such that the expected discounted cumulative long-term rewards with states sampled according to ρ_0 is maximized:

$$\max_\pi J_{\rho_0}(\pi, \mathcal{M}) := \mathbb{E}_{s\sim\rho_0,\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t)\middle| s_0 = s\right], \quad (1)$$

where $\Delta(\cdot)$ denotes the probability simplex. We further denote the optimal policy $\pi^* := \arg\max_\pi J_{\rho_0}(\pi, \mathcal{M})$.

In offline RL, the agent can not interact with the environment, and can only get access to a previously logged dataset $\mathcal{D}_{\text{env}} = \{(s,a,r,s')\}$. The data can be collected by one or a mixture of behavior policy μ, which are often unknown. Offline RL aims at learning the optimal batch-constraint policy. Model-based offline RL methods learn the environmental dynamics model $\hat{P}(\cdot|s,a)$ solely from the dataset, and leverage the learned model to optimize the policy. Typically, the transition probability model $\hat{P}(\cdot|s,a)$ is trained using maximum likelihood estimation. A reward function $\hat{r}(s,a)$ can also be learned if it is unknown. We assume the reward function is known in this work. Then we can construct a model MDP $\hat{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, r, \hat{P}, \hat{\rho}_0, \gamma \rangle$. During training, the synthetic samples generated by the learned dynamics model will be put into the model buffer $\mathcal{D}_{\text{model}}$ and the policy is finally updated using data sampled from $\mathcal{D}_{\text{env}} \cup \mathcal{D}_{\text{model}}$.

4 Trajectory Truncation with Uncertainty

In this section, we first provide some theoretical insights of our proposed trajectory truncation with uncertainty (TATU). Then, we present the detailed practical algorithm for TATU.

4.1 Theoretical Analysis

To determine whether the generated synthetic trajectory is reliable, we define the accumulated uncertainty along the trajectory (AUT) and truncation indicator below.

Definition 1. *Given a state-action sequence $\{s_i, a_i\}_{i=0}^h$, define accumulated uncertainty along the trajectory at step t as:*

$$U_t(s_t,a_t) = \sum_{i=0}^{t} \gamma^t D_{\text{TV}}(\hat{P}(\cdot|s_i,a_i), P(\cdot|s_i,a_i)). \quad (2)$$

Furthermore, we define the truncation indicator as:

$$T_t^\epsilon(s,a) = \begin{cases} 0, & \text{if } U_t(s_t,a_t) \le \epsilon, \\ 1, & \text{otherwise}, \end{cases} \quad (3)$$

where $\hat{P}(\cdot|s,a)$ is the trained environmental transition probability, $P(\cdot|s,a)$ is the true dynamics, and $D_{\text{TV}}(p,q)$ is the total variation distance between two distributions p, q. Intuitively, $U_t(s,a)$ measures how much uncertainty is accumulated along the imagined trajectory and $T_t^\epsilon(s,a)$ illustrates whether the trajectory ought to be truncated or not. If the accumulated uncertainty is too large (e.g., exceeds the threshold ϵ), then the synthetic trajectory will be truncated and no more forward imagination will be included in the model buffer. We denote $u(s,a) = D_{\mathrm{TV}}(\hat{P}(\cdot|s,a), P(\cdot|s,a))$ as the uncertainty measurement for state-action pair (s,a). As a valid uncertainty measurement, $u(s,a)$ ought to satisfy $0 \le u(s,a) \le u_0 < \infty$ for some positive constant u_0. Obviously, our defined uncertainty measurement above meets this requirement since the total variation distance is bounded. We denote the maximum uncertainty is $u_{\max}$, then we have $|u(s,a)| \le u_{\max} \le 1 < \infty, \forall\, s, a$. It is easy to find that AUT $U_t(s_t,a_t) = \sum_{i=0}^t \gamma^t u(s_i,a_i)$.

We further define the ϵ-Pessimistic MDP as follows:

Definition 2 (ϵ-Pessimistic MDP)**.** *The ϵ-Pessimistic MDP is defined by the tuple $\hat{\mathcal{M}}_p = \langle \mathcal{S}, \mathcal{A}, r_p, \hat{P}_p, \hat{\rho}_0, \gamma\rangle$, where $\mathcal{S}, \mathcal{A}$ are state space and action space in the actual MDP $\mathcal{M}$, $\hat{\rho}_0$ is the initial state distribution learned from the dataset $\mathcal{D}$. The transition dynamics and reward function for $\hat{\mathcal{M}}_p$ gives:*

$$\hat{P}_p(\cdot|s_t,a_t) = \begin{cases} 0, & \text{if } T_t^\epsilon(s_t,a_t) = 1, \\ \hat{P}(\cdot|s_t,a_t), & \text{otherwise}, \end{cases} \tag{4}$$

$$r_p(s_t,a_t) = \begin{cases} r(s_t,a_t) - \lambda u(s_t,a_t) - \kappa, & \text{if } T_t^\epsilon(s_t,a_t) = 1, \\ r(s_t,a_t) - \lambda u(s_t,a_t), & \text{otherwise}, \end{cases} \tag{5}$$

where κ is the penalty for truncation state, and λ is the uncertainty penalty coefficient for each state-action pair (s,a). Note that the summation of $\hat{P}_p$ is not 1, while it is valid because for a fixed offline dataset, the cases of $T_t^\epsilon(s_t,a_t) = 1$ are almost sure. Hence, we can rearrange the probability distribution of $\hat{P}_p(\cdot|s_t,a_t)$ by weighting with a constant. By introducing the ϵ-Pessimistic MDP formulation, we involve conservatism into the reward function and the imagined state-action sequence. The trajectory imagination will be terminated if the truncation indicator $T_t^\epsilon = 1$ i.e., the accumulated uncertainty $U_t(s_t,a_t)$ exceeds ϵ at horizon t. An additional reward penalty κ will be allocated to the termination state in the synthetic trajectory.

Assume that the reward function is bounded almost surely, i.e., $|r(s,a)| \le r_{\max}, \forall\, s, a$. We now derive the performance bound for our ϵ-Pessimistic MDP. Due to the space limit, all proofs are omitted and deferred to Appendix A.

Theorem 1. *Denote $\bar{r} = r_{\max} + \lambda u_{\max} + \kappa$. Then the return of any policy π in the ϵ-Pessimistic MDP $\hat{\mathcal{M}}_p$ and its original MDP $\mathcal{M}$ satisfies:*

$$\begin{aligned} J_{\hat{\rho}_0}(\pi, \hat{\mathcal{M}}_p) \ge & J_{\rho_0}(\pi, \mathcal{M}) - \frac{2\bar{r}}{1-\gamma} \cdot D_{\mathrm{TV}}(\rho_0, \hat{\rho}_0) \\ & - \bar{r} \cdot \epsilon - \frac{\bar{r}}{1-\gamma}, \end{aligned} \tag{6}$$

$$J_{\hat{\rho}_0}(\pi, \hat{\mathcal{M}}_p) \le J_{\rho_0}(\pi, \mathcal{M}) + \frac{2\bar{r}}{1-\gamma} \cdot D_{\mathrm{TV}}(\rho_0, \hat{\rho}_0) + \bar{r} \cdot \epsilon. \tag{7}$$

Remark: Our method incurs a tighter performance bound compared to MOReL. To be specific, our methods guarantees $|J_{\hat{\rho}_0}(\pi, \hat{\mathcal{M}}_p) - J_{\rho_0}(\pi, \mathcal{M})| \le \mathcal{O}\left(\frac{r_{\max}}{1-\gamma}\right)$ thanks to the uncertainty-based trajectory truncation. However, MOReL only ensures that $|J_{\hat{\rho}_0}(\pi, \hat{\mathcal{M}}_p) - J_{\rho_0}(\pi, \mathcal{M})| \le \mathcal{O}\left(\frac{r_{\max}}{(1-\gamma)^2}\right)$.

Theorem 1 indicates that the return difference for any policy π in the true MDP and the ϵ-Pessimistic MDP relies on mainly three parts: (1) the total variation distance between the learned and true initial state distribution $D_{\mathrm{TV}}(\rho_0, \hat{\rho}_0)$; (2) the threshold for accumulated uncertainty ϵ; (3) the upper bound for the pessimistic reward function (since $|r_p| \le \bar{r}$ almost surely). As an immediate corollary, we have

Corollary 1. *If the policy in the ϵ-Pessimistic MDP is δ_π sub-optimal, i.e., $J_{\hat{\rho}_0}(\pi, \hat{\mathcal{M}}_p) \ge J_{\hat{\rho}_0}(\pi^*, \hat{\mathcal{M}}_p) - \delta_\pi$, then we have*

$$\begin{aligned} J_{\rho_0}(\pi^*, \mathcal{M}) - J_{\rho_0}(\pi, \mathcal{M}) \le & \delta_\pi + \frac{4\bar{r}}{1-\gamma} \cdot D_{\mathrm{TV}}(\rho_0, \hat{\rho}_0) \\ & + 2\bar{r}\epsilon + \frac{\bar{r}}{1-\gamma}. \end{aligned}$$

This corollary illustrates that the sub-optimality $(J(\pi^*, \mathcal{M}) - J(\pi, \mathcal{M}))$ of the policy in the true MDP is bounded by the sub-optimality of the policy trained with the ϵ-Pessimistic MDP. If the sub-optimality of the policy learned in the pessimistic MDP is small (i.e., δ_π is small), then the sub-optimality of the policy in the true MDP will also be small. Furthermore, if the dataset covers all possible transitions, i.e., the dataset is large enough, then it is easy to find both $D_{\mathrm{TV}}(\rho_0, \hat{\rho}_0)$ and ϵ approach 0 since all of the imagined samples will lie in the support region of the dataset with high probability. Naturally, the above upper bound thus can be further simplified.

Corollary 2. *If the dataset is large enough, then we have*

$$J_{\rho_0}(\pi^*, \mathcal{M}) - J_{\rho_0}(\pi, \mathcal{M}) \le \delta_\pi + \frac{\bar{r}}{1-\gamma}.$$

Corollary 2 also reveals that even when the dataset is large, the performance deviation between the optimal policy and the learnt policy is determined by the sub-optimality in the pessimistic MDP.

We then present the algorithmic details for TATU below.

4.2 Practical Algorithm

The practical implementation of TATU can be generally divided into three steps:

Step 1: Training Dynamics Models: Following prior work [15], we train the dynamics model $\hat{P}(\cdot|s,a)$ with a neural network $\hat{p}_\psi(s'|s,a)$ parameterized by ψ that produces a Gaussian distribution over the next state, i.e., $\hat{p}_\psi(s'|s,a) = \mathcal{N}(\mu_\theta(s,a), \Sigma_\phi(s,a)), \psi = \{\theta, \phi\}$. We train an ensemble of N dynamics models $\{\hat{p}_\psi^i = \mathcal{N}(\mu_\theta^i, \Sigma_\phi^i)\}_{i=1}^N$. We denote the loss function for training the forward dynamics model as $\mathcal{L}_\psi$. Then each model in the ensemble is trained independently via maximum log-likelihood:

$$\mathcal{L}_\psi = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[-\log \hat{p}_\psi(s'|s,a)\right]. \tag{8}$$

We model the difference in the current state and next state, i.e., $\mu_\theta(s,a) = s + \delta_\theta(s,a)$ to ensure local continuity.

Step 2: Constructing ϵ-Pessimistic MDP: After the environmental dynamics model is well-trained, we use it to construct the ϵ-Pessimistic MDP by following Equation (4) and (5) in Definition 2. It is then important to decide which uncertainty measurement to use in practice (we cannot use $D_{\mathrm{TV}}(\hat{P}(\cdot|s,a), P(\cdot|s,a))$ as true transition probability $P(s'|s,a)$ is often unknown and inaccessible), and how to decide the uncertainty threshold ϵ. As a valid and reasonable uncertainty measurement, we require $u(s,a)$ can capture how uncertain the state-action pair (s,a) is, and satisfy $|u(s,a)| \le u_{\max} < \infty$ almost surely for any (s,a). MOReL [17] adopts the ensemble discrepancy as the uncertainty quantifier, i.e., $u(s,a) = \max_{i,j} \|\mu_\theta^i(s,a) - \mu_\theta^j(s,a)\|_2$, where $\|\cdot\|_2$ is the L2-norm and $\mu_\theta^i(s,a), \mu_\theta^j(s,a), i,j \in \{1,\ldots,N\}$ are mean vectors of the Gaussian distributions in the ensemble. MOPO [44] utilizes the maximum standard deviation of the learned models in the ensemble as the uncertainty estimator, i.e., $u(s,a) = \max_{i=1}^N \|\Sigma_\phi^i(s,a)\|_F$, where $\|\cdot\|_F$ is the Frobenius-norm. $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|}$ for matrix $\mathbf{A}$ with size $m \times n$. Empirically, we do not observe much performance difference with these quantifiers in our experiments. We then use an MOPO-style uncertainty estimator in this work. Such uncertainty measurement generally reveals whether the state-action pair is reliable, and satisfies $|u(s,a)| \le u_{\max} < \infty$. One can also enforce this by clipping $u(s,a)$ to $[-u_{\max}, u_{\max}]$ with a manually set upper bound.

As for the uncertainty threshold ϵ, one naïve way is to set a constant threshold. However, this often lacks flexibility and the best threshold may be quite varied for different datasets. We instead propose to measure the uncertainty on *all of the samples* in the dataset $\mathcal{D}$. We then take the maximum transition uncertainty in the dataset and set the uncertainty threshold based on:

$$\epsilon = \frac{1}{\alpha} \max_{i\in[|\mathcal{D}|]} u(s_i,a_i) = \frac{1}{\alpha} \max_{i\in[|\mathcal{D}|]} \max_{j\in[N]} \|\Sigma_\phi^j(s_i,a_i)\|_F, \qquad (9)$$

where $[k] = \{1,\ldots,k\}$, $(s_i,a_i) \sim \mathcal{D}, i \in [|\mathcal{D}|]$, and $\alpha \in \mathbb{R}_+$ is the key hyperparameter that controls the strength of the threshold. By using a larger α, TATU becomes more conservative and includes fewer imagined samples into the model buffer $\mathcal{D}_{\text{model}}$. While if a small α is used, TATU exhibits more tolerance to generated data. The total uncertainty is controlled by the maximum uncertainty in the real dataset (i.e., $\max_{i=1}^{|\mathcal{D}|} u(s_i,a_i)$), which usually varies with different datasets. This allows a flexible and reasonable uncertainty threshold for adaptively selecting good imaginations.

Step 3: Conservative Trajectory Generation: Once the uncertainty threshold ϵ is ready, we can generate conservative trajectories. We first sample starting state s_0 from the static dataset $\mathcal{D}$ and generate an imagined trajectory $\hat{\tau} = \{\hat{s}_j, a_j, r_j, \hat{s}_{j+1}\}_{j=0}^{h-1}$ with the learned dynamics $\hat{p}_\psi$, where h is the horizon length. We simultaneously calculate the accumulated uncertainty of the imagined trajectory $U_t(s_t,a_t)$ at horizon t, $t \in [1,h]$. The generated trajectory is truncated at horizon t if $U_t(s_t,a_t) > \epsilon$. In deep RL, we usually sample a mini-batch of data from the real dataset $\mathcal{D}$ of size B with bootstrapping. By leveraging the learned dynamics model, we then can get B imagined trajectories, provably with different lengths (as the accumulated uncertainty varies with different starting states). We can incorporate TATU with existing popular model-based offline RL algorithms such as MOPO [44], COMBO [43], etc. Here, the actions in the imagined trajectory are generated with the learned policy π.

TATU can also be incorporated with off-the-shelf model-free offline RL methods, where TATU serves as a role of offline data augmentation to improve the generalization ability of model-free offline RL algorithms. At this time, we need to train an additional rollout policy such that the data augmentation process is isolated from the policy optimization process, i.e., the dataset is augmented beforehand. To avoid potential OOD actions, we model the rollout policy using the conditional variational auto-encoder (CVAE) [18]. We choose CVAE as it guarantees that the generated actions lie in the span of the dataset[1]. The CVAE is made up of an encoder $E_\xi(s,a)$ that produces a latent variable z under the Gaussian distribution and

[1] One can also use other generative models like GAN [31], diffusion model [14], etc.

Algorithm 1 Trajectory Truncation with Uncertainty (TATU)

```
1:  Require: Offline dataset $\mathcal{D}$, number of iterations $N$, horizon $h$,
    reward penalty coefficient $\lambda$, termination penalty $\kappa$
2:  Initialize model buffer $\mathcal{D}_{\text{model}} \leftarrow \emptyset$
3:  Train the ensemble dynamics models $\{\hat{p}_\psi^i(s'|s,a) = \mathcal{N}(\mu_\theta^i(s,a),\Sigma_\phi^i(s,a))\}_{i=1}^N$ on $\mathcal{D}$ using Equation (8)
4:  Calculate the truncation threshold $\epsilon$ using Equation (9)
5:  (Optional) Train a rollout policy via Equation (10)
6:  for epoch from 1 to $N$ do
7:      Sample state $s_0$ from dataset $\mathcal{D}$
8:      for $j$ in 1 to $h$ do
9:          Sample an action $a_j \sim \pi(\cdot|s_j)$ // learned policy
10:         (Optional) Draw an action $a_j$ from the rollout policy
11:         Randomly pick dynamics $\hat{p}$ from $\{\hat{p}_\psi^i\}_{i=1}^N$ and sample next
            state $s_{j+1} \sim \hat{p}$
12:         Calculate reward according to Equation (5)
13:         Calculate accumulated uncertainty along the trajectory
            $U_j = \sum_{k=1}^j \left[\max_{i=1}^N \|\Sigma_\phi^i(s_k,a_k)\|_F\right]$
14:         if $U_j \le \epsilon$ then
15:             Put the imagined transition $(s_j,a_j,r_j,s_{j+1})$ into the
                model buffer $\mathcal{D}_{\text{model}}$
16:         else
17:             break        // Truncate the synthetic trajectory
18:         end if
19:     end for
20:     Sample data from $\mathcal{D}\cup\mathcal{D}_{\text{model}}$ and use the base algorithm (e.g.,
        SAC, CQL, BCQ) to optimize policy $\pi$
21: end for
```

a decoder $D_\nu(s,z)$ that maps z to the desired space. The objective function for training CVAE is shown below.

$$\mathcal{L}_{\text{CVAE}} = \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mathcal{D}, z\sim E_\xi(s,a)} \Big[(a - D_\nu(s,z))^2 + D_{\text{KL}}\left(E_\xi(s,a)\|\mathcal{N}(0,\mathbf{I})\right)\Big], \qquad (10)$$

where the encoder $E_\xi(s,a)$ and decoder $D_\nu(s,z)$ are parameterized by ξ,ν, respectively. $D_{\text{KL}}(p\|q)$ denotes the KL-divergence between two distributions p,q, and $\mathbf{I}$ is an identity matrix. When drawing an action from the rollout policy, we first sample a latent variable z from the multivariate Gaussian distribution $\mathcal{N}(0,\mathbf{I})$ and process it along with the current state s with the decoder $D_\nu(s,z)$ to get the resulting action.

We summarize the detailed pseudo code for TATU in Algorithm 1. Note that when drawing samples from $\mathcal{D}\cup\mathcal{D}_{\text{model}}$, we sample a proportion of ηB real data from real dataset $\mathcal{D}$ and a proportion of $(1-\eta)B$ synthetic data from model buffer $\mathcal{D}$ given the batch size B and the real data ratio $\eta \in [0,1]$. We then use the underlying algorithms (e.g., CQL) to optimize the policy. To accommodate the reproducibility, we include our source code in the supplementary material and will open-source the code upon acceptance.

5 Experiments

In this section, we empirically examine how much can TATU benefit existing offline RL methods. Throughout our experimental evaluation, we aim at answering the following questions: (1) How much performance gain can model-based offline RL methods acquire when combined with TATU? (2) Can TATU also benefit model-free offline RL algorithms?

Table 1. Normalized average score comparison of TATU+MOPO and TATU+COMBO against their base algorithms and some recent baselines on the D4RL MuJoCo "-v2" dataset. half = halfcheetah, r = random, m = medium, m-r = medium-replay, m-e = medium-expert. Each algorithm is run for 1M gradient steps with 5 different random seeds. We report the final performance. ± captures the standard deviation. We bold the top 2 score of the left part and the best score of the right part.

Task Name	TATU+MOPO	MOPO	TATU+COMBO	COMBO	BC	CQL	IQL	DT	MOReL
half-r	33.3±2.6	**35.9**	29.3±2.7	**38.8**	2.2	17.5	13.1	-	**38.9**
hopper-r	**31.3**±0.6	16.7	**31.6**±0.6	17.9	3.7	7.9	7.9	-	**38.1**
walker2d-r	**10.4**±0.7	4.2	5.3±0.0	**7.0**	1.3	5.1	5.4	-	**16.0**
half-m-r	**67.2**±3.3	**69.2**	57.8±2.7	55.1	37.6	**45.5**	44.2	36.6	44.5
hopper-m-r	**104.4**±0.9	32.7	**100.7**±1.3	89.5	16.6	88.7	**94.7**	82.7	81.8
walker2d-m-r	**75.3**±0.2	73.7	**75.3**±1.7	56.0	20.3	**81.8**	73.8	66.6	40.8
half-m	61.9±2.9	**73.1**	**69.2**±2.8	54.2	43.2	47.0	47.4	42.6	**60.7**
hopper-m	**104.3**±1.3	38.3	**100.0**±1.3	97.2	54.1	53.0	66.2	67.6	**84.0**
walker2d-m	**77.9**±1.6	41.2	77.4±0.9	**81.9**	70.9	73.3	**78.3**	74.0	72.8
half-m-e	74.1±1.4	70.3	**96.4**±3.6	**90.0**	44.0	75.6	86.7	**86.8**	80.4
hopper-m-e	**107.0**±1.3	60.6	106.5±0.4	**111.1**	53.9	105.6	91.5	**107.6**	105.6
walker2d-m-e	**107.9**±0.9	77.4	**114.6**±0.7	103.3	90.1	107.9	**109.6**	108.1	107.5
Average score	**71.3**	49.4	**72.0**	66.8	36.5	59.1	59.9	-	**64.3**

To answer these questions, we first combine TATU with two popular model-based offline RL algorithms, MOPO [44] and COMBO [43] to examine whether TATU can boost their performance in Section 5.1. We also integrate TATU with three off-the-shelf model-free offline RL methods including CQL [22], TD3_BC [10], and BCQ [11], to see whether TATU can also benefit them in Section 5.2. We conduct experiments on D4RL [9] MuJoCo datasets for evaluation. In Section 5.3, We provide a detailed parameter study on some important hyperparameters in TATU, e.g., rollout horizon. We further compare TATU against other data selection methods in Section 5.4.

5.1 Combination with Model-based Offline RL

Since TATU is designed intrinsically for reliable imagination generation using a learnt dynamics model, we incorporate it with two widely used offline model-based RL algorithms, MOPO [44] and COMBO [43], giving rise to TATU+MOPO and TATU+COMBO. Since MOPO already penalizes the reward signal with uncertainty, we only add an additional penalty κ to the termination state. TATU modifies the way of imagination generation and concretely rejects bad transition tuples in these methods.

To examine whether TATU can achieve performance improvement upon MOPO and COMBO, we conduct experiments on 12 D4RL [9] MuJoCo datasets, which includes three tasks: *halfcheetah*, *hopper*, *walker2d*. We adopt four types of datasets for each task: *random*, *medium*, *medium-replay*, and *medium-expert*, as they are typically utilized for performance evaluation in model-based offline RL. We compare TATU+MOPO, TATU+COMBO against their base algorithms. We also compare them against some common baselines in offline RL, such as behavior cloning (BC), IQL [19], Decision Transformer (DT) [6]. We take the results of IQL on medium-level datasets from its original paper and run with its official implementation[2] on random and expert datasets. The results of BC are acquired by our own implementation. For methods that were originally evaluated on "-v0" datasets, we retrain them with the official implementations on "-v2" datasets, and take the results of other baselines from their original papers.

All of the algorithms are run for 1M gradient steps. Table 1 reports the normalized score over the final 10 evaluations for each task and their average performance over 5 different random seeds. It can be found that TATU markedly boosts the performance of both MOPO and COMBO on most of the datasets, often by a large and significant margin. For example, TATU+MOPO achieves a mean score of **104.4** on hopper-medium-replay-v2 and **77.9** on walker2d-medium-v2, while MOPO only attains 32.7 and 41.2 on them, respectively. It is worth noting that MOPO and COMBO gain **44%** and **7.8%** improvement in average score, respectively, with the aid of TATU. We also observe a competitive or better performance of TATU+MOPO and TATU+COMBO on the evaluated datasets against baseline methods like IQL, MOReL, etc. TATU+COMBO has the best average score (**72.0**) across all of the datasets, TATU+MOPO achieves an average score of **71.3**. Whereas, the best baseline only achieves 66.8. The empirical evaluations indicate that TATU can significantly benefit the existing model-based offline RL methods.

5.2 Combination with Model-free Offline RL

We now demonstrate that TATU can also benefit model-free offline RL algorithms. We first train the dynamics model and an additional CVAE rollout policy. During data generation, we truncate the synthetic trajectory if the accumulated uncertainty along it is large. Importantly, we leverage the rollout policy to produce actions in the imagined trajectory, thus isolating the data augmentation process from the policy optimization process.

To illustrate the generality and effectiveness of TATU, we combine it with three popular model-free offline RL algorithms, CQL [22], BCQ [11], and TD3_BC [10], yielding TATU+CQL, TATU+BCQ, and TATU+TD3_BC algorithms. We extensively compare them with their base algorithms and three model-free offline RL baselines (BC, IQL, DT) on 15 D4RL datasets, with additional *expert* datasets of three tasks compared to the experimental evaluation in Section 5.1.

We summarize the experimental results in Table 2. We find that TATU significantly improves the performance of the base algorithms on many datasets, often outperforming them by a remarkable margin, especially on many poor-quality datasets (e.g., random). As an example, TATU+TD3_BC has a mean normalized score of **31.6** on hopper-random-v2, while TD3_BC itself only achieves 8.5. The three base methods, TD3_BC, CQL, BCQ, gains **8.0%**, **8.9%** and **7.0%** performance improvement in average score, respectively. With the conser-

[2] https://github.com/ikostrikov/implicit_q_learning

Table 2. Normalized average score comparison of TATU+TD3_BC, TATU+CQL and TATU+BCQ against their base algorithms and some recent baselines on the D4RL MuJoCo "-v2" dataset. half = halfcheetah, r = random, m = medium, m-r = medium-replay, m-e = medium-expert, e = expert. Each algorithm is run for 1M gradient steps across 5 different random seeds and the final mean performance is reported. ± captures the standard deviation. We bold the top 3 score of the left part and the best score of the right part.

Task Name	TATU+TD3_BC	TD3_BC	TATU+CQL	CQL	TATU+BCQ	BCQ	BC	IQL	DT
half-r	**12.1**±2.3	11.0	**27.4**±2.6	**17.5**	2.3±2.3	2.2	2.2	**13.1**	-
hopper-r	**31.6**±0.6	8.5	**32.3**±0.7	7.9	**10.3**±0.8	7.8	3.7	**7.9**	-
walker2d-r	**21.4**±0.0	1.6	**23.0**±0.0	**5.1**	3.4±0.4	4.9	1.3	**5.4**	-
half-m-r	**45.9**±0.6	44.6	**48.0**±0.7	**45.5**	43.5±0.3	42.2	37.6	**44.2**	36.6
hopper-m-r	65.7±3.9	60.9	**96.8**±2.6	**88.7**	**72.9**±0.4	60.9	16.6	**94.7**	82.7
walker2d-m-r	**81.9**±2.7	**81.8**	**85.5**±1.2	**81.8**	77.7±1.0	57.0	20.3	**73.8**	66.6
half-m	**48.4**±2.7	**48.3**	44.9±0.3	47.0	**47.6**±0.2	46.6	43.2	**47.4**	42.6
hopper-m	**62.0**±1.0	59.3	**68.9**±0.9	53.0	**71.0**±0.3	59.4	54.1	66.2	**67.6**
walker2d-m	**84.3**±0.2	**83.7**	65.7±0.5	73.3	**80.5**±0.4	71.8	70.9	**78.3**	74.0
half-m-e	**97.1**±3.2	90.7	78.9±0.5	75.6	**96.1**±0.2	**95.4**	44.0	86.7	**86.8**
hopper-m-e	**113.0**±1.7	98.0	**111.5**±1.0	105.6	**108.2**±0.3	106.9	53.9	91.5	**107.6**
walker2d-m-e	**110.9**±0.5	110.1	**110.2**±0.1	107.9	**111.7**±0.3	107.7	90.1	**109.6**	108.1
half-e	**97.4**±0.4	**96.7**	90.8±3.4	**96.3**	**96.3**±1.4	89.9	91.8	**95.0**	-
hopper-e	**111.8**±1.1	**107.8**	106.8±0.9	96.5	103.9±0.6	**109.0**	107.7	**109.4**	-
walker2d-e	**110.0**±0.1	**110.2**	108.3±0.1	108.5	**109.7**±0.3	106.3	106.7	**109.9**	-
Average score	**72.9**	67.5	**73.3**	67.3	**69.0**	64.5	49.6	**68.9**	-

vative and reliable transition generation by our novel synthetic trajectory truncation method, model-free offline RL methods can benefit from better generalization capability, resulting in better performance. Based on these experiments, we conclude that TATU is general and can widely benefit both model-based and model-free offline RL algorithms.

Note that due to space limit, we omit the standard deviation for baseline methods, and the full comparison results of Table 1 and Table 2 can be found in Appendix D.

5.3 *Parameter Study*

There are three most critical hyperparameters in TATU, the rollout horizon h, the threshold parameter α, and the real data ratio $\eta \in [0, 1]$ in a sampled batch from $\mathcal{D} \cup \mathcal{D}_{\mathrm{model}}$.

Table 3. Comparison of TATU+MOPO and TATU+TD3_BC with different horizon lengths h on hopper-medium-replay-v2. The results are run for 1M gradient steps and averaged over the final 10 evaluations and 5 random seeds. Mean performance along with the standard deviation are reported.

horizon h	TATU+MOPO	TATU+TD3_BC
1	26.7±4.4	57.9±1.9
3	100.4±1.3	70.0±4.9
5	104.4±0.9	65.7±3.9
7	102.0±0.7	71.1±12.4
10	103.7±1.6	71.0±16.8

Rollout horizon h**.** This parameter measures how far that we allow the imagined trajectory to branch. With a larger horizon length for the dynamics model, more diverse data can be included in the model buffer, which, however, also increases the risk of involving bad transitions. This issue can be alleviated by TATU as it adaptively truncates the imagined trajectory. A large horizon length h can therefore be adopted. To see the influence of h, we conduct experiments on hopper-medium-replay-v2 with TATU+MOPO and TATU+TD3_BC. We keep other parameters fixed and sweep h across $\{1, 3, 5, 7, 10\}$. Figure 2 shows the corresponding learning curves. Results in Table 3 indicates that TATU enjoys better performance with a larger horizon and fails with horizon 1. This may be because the diversity of the generated samples is not enough with a too small horizon length. With additional experiments, we find the optimal horizon h is influenced by the quality of the dataset (please see Appendix for more details). The fact is that for expert datasets (with narrow span), generated samples are more likely to be OOD at a certain horizon h. On medium-quality datasets with a larger horizon, algorithms can benefit from conservative and diversified imaginations of TATU to improve performance. In our main experiments, we set the rollout horizon $h = 5$ by default.

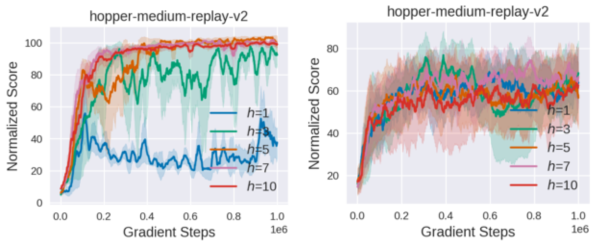

Figure 2. Performance of TATU+MOPO (left) and TATU+TD3_BC (right) with different rollout horizons on hopper-medium-replay-v2. The results are averaged over 5 different random seeds, and we report the mean performance and the standard deviation.

Threshold coefficient α**.** The threshold parameter α is probably the most critical parameter for TATU, which controls the strength that we reject the generated imaginations (larger α will reject more transitions). This parameter is highly related to the quality of the dataset, e.g., a large α is better for expert-level datasets while α can be small for dataset that is collected by a random policy. Generally, we require $\alpha \geq 1$. To show the effects of this parameter, we run TATU+MOPO and TATU+TD3_BC on hopper-medium-replay-v2 with $\alpha \in \{1.0, 2.0, 3.0, 4.0, 5.0\}$. We report the numerical comparison results in Table 4 and the learning curves in Figure 3. We see that larger α degrades the agent's performance of MOPO. This is harmful as the diversity in the imaginations is decreased and real data ratio η adopted in MOPO is small, few samples are admitted. Too large α also causes more unstable in learning as shown in Figure 3. For TD3_BC, however, a larger α is better as small α enables a large threshold ϵ, i.e., bad transitions may be included and are more destructive for TD3_BC (as it involves an imitation learning term) than for MOPO. We notice that we can luckily find a trade-off of α for different algorithms.

Table 4. Comparison of TATU+MOPO and TATU+TD3_BC with different threshold coefficient α on hopper-medium-replay-v2. Each algorithm is run for 1M steps with 5 random seeds. We report the mean performance over the final 10 evaluations, in conjunction with the standard deviation.

Coefficient α	TATU+MOPO	TATU+TD3_BC
1.0	101.4±0.8	32.1±11.1
2.0	104.4±0.9	50.2±4.4
3.0	98.7±3.3	52.2±10.1
4.0	96.4±2.7	61.9±10.7
5.0	52.8±31.7	58.6±13.4

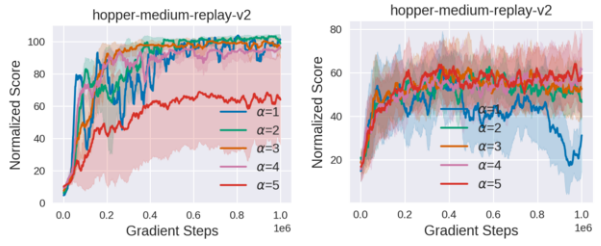

Figure 3. Normalized score comparison of TATU+MOPO (left) and TATU+TD3_BC (right) on hopper-medium-replay-v2 under different threshold coefficients α. The results are averaged over 5 different random seeds and the shaded region is the standard deviation.

Real data ratio η. The real data ratio controls the proportion of real data in a sampled mini-batch, i.e., it determines how many synthetic samples are used for offline policy learning. η also strongly depends on the specific dataset and setting. Model-based offline RL method usually uses a small η, and the η for model-free offline RL algorithm relies heavily on the quality of the dataset, e.g., a small η is preferred for datasets with poor quality, and a larger η is better for expert datasets. To see how η affects the performance of model-based and model-free offline RL algorithms with TATU, we run TATU+MOPO and TATU+TD3_BC with different real data ratio $\eta \in \{0.05, 0.25, 0.5, 0.7, 0.9\}$ on hopper-medium-replay-v2. We summarize the results in Table 5, where we find that TATU+MOPO achieves very good performance under a wide range of η, even a small $\eta = 0.05$ thanks to the conservative trajectory truncation by TATU. As the dataset is of medium quality, TATU+TD3_BC attains higher performance with larger η, as expected. The learning curves of TATU+MOPO and TATU+TD3_BC under different real data ratios are shown in Figure 4, which are consistent with our analysis above.

5.4 Comparison with Other Relevant Methods

In order to further show the advantages of our proposed TATU method, we compare it against other data selection methods, including CABI [27] and MOPP [45]. MOPP selects transitions by performing planning and then sorting the trajectory such that each sample in the left trajectory meets the uncertainty constraint. CABI trains bidirectional dynamics models and only admits transitions that the forward and backward model agree on. TATU ensures reliable data generation via adaptively truncating the imagined trajectory. We combine TATU and CABI with TD3_BC and conduct experiments on 6 *random*, *medium-expert* datasets from D4RL MuJoCo datasets. We follow the guidance in the CABI paper and implement it on our own. We take the results of MOPP from its original paper directly. We run each algorithm for 1M gradient steps over 5 different random seeds. The results are presented in Table 6, where we report the average final performance. We find that TATU+TD3_BC outperforms MOPP and CABI on most of the datasets, achieving the best performance on 4 out of 6 datasets. These we believe can well illustrate the superiority of TATU.

Table 5. Comparison of TATU+MOPO and TATU+TD3_BC under different real data ratio η on hopper-medium-replay-v2. The results are averaged over the final 10 evaluations and 5 random seeds. We report the mean performance and the standard deviation.

Real data ratio η	TATU+MOPO	TATU+TD3_BC
0.05	104.4±0.9	9.1±4.4
0.25	101.5±3.8	39.7±15.6
0.5	101.1±2.7	65.4±1.2
0.7	92.9±2.9	65.7±3.9
0.9	85.1±12.4	68.8±17.9

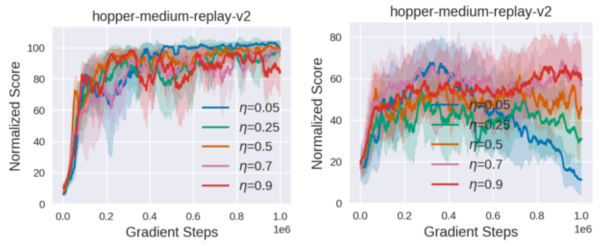

Figure 4. Performance of TATU+MOPO (left) and TATU+TD3_BC (right) on hopper-medium-replay-v2 using different real data ratios. All methods are run for 5 different random seeds, and the shaded region captures the standard deviation.

Table 6. Comparison of TATU against CABI (with TD3_BC as the base algorithm) and MOPP on six datasets from D4RL.

Task Name	TATU+TD3_BC	CABI+TD3_BC	MOPP
half-r	12.1±2.3	**14.3**±0.4	9.4±2.6
hopper-r	**31.6**±0.6	15.7±11.1	13.7±2.5
walker-r	**21.4**±0.0	6.0±0.3	6.3±0.1
half-m-e	97.1±3.2	94.8±1.0	**106.2**±5.1
hopper-m-e	**113.0**±1.7	111.4±0.3	95.4±28.0
walker2d-m-e	**110.9**±0.5	110.5±0.6	92.9±14.1

6 Conclusion

In this paper, we propose trajectory truncation with uncertainty (TATU) to facilitate both model-based and model-free offline RL algorithms. We adaptively truncate the imagined trajectory if the accumulated uncertainty of this trajectory is too large, which ensures the reliability of the synthetic samples. We theoretically demonstrate the advantages of our proposed truncation method. Empirical results on the D4RL benchmark show that TATU markedly improves the performance of model-based offline RL methods (e.g., MOPO) and model-free offline RL methods (e.g., CQL). These altogether illustrate the generality and effectiveness of TATU.

Acknowledgements

This work was supported in part by the Science and Technology Innovation 2030-Key Project under Grant 2021ZD0201404.

References

[1] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song, 'Uncertainty-based offline reinforcement learning with diversified q-ensemble', in *Neural Information Processing Systems*, (2021).

[2] Arthur Argenson and Gabriel Dulac-Arnold, 'Model-based offline planning', in *International Conference on Learning Representations*, (2020).

[3] Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhihong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang, 'Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning', in *International Conference on Learning Representation*, (2022).

[4] David Brandfonbrener, William F. Whitney, Rajesh Ranganath, and Joan Bruna, 'Offline rl without off-policy evaluation', in *Neural Information Processing Systems*, (2021).

[5] Catherine Cang, Aravind Rajeswaran, P. Abbeel, and Michael Laskin, 'Behavioral priors and dynamics models: Improving performance and domain transfer in offline rl', *ArXiv*, **abs/2106.09119**, (2021).

[6] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, P. Abbeel, A. Srinivas, and Igor Mordatch, 'Decision transformer: Reinforcement learning via sequence modeling', in *Neural Information Processing Systems*, (2021).

[7] Xi Chen, Ali Ghadirzadeh, Tianhe Yu, Jianhao Wang, Yuan Gao, Wenzhe Li, Liang Bin, Chelsea Finn, and Chongjie Zhang, 'LAPO: Latent-variable advantage-weighted policy optimization for offline reinforcement learning', in *Neural Information Processing Systems*, (2022).

[8] Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal, 'Adversarially trained actor critic for offline reinforcement learning', in *International Conference on Machine Learning*, (2022).

[9] Justin Fu, Aviral Kumar, Ofir Nachum, G. Tucker, and Sergey Levine, 'D4rl: Datasets for deep data-driven reinforcement learning', *ArXiv*, **abs/2004.07219**, (2020).

[10] Scott Fujimoto and Shixiang Gu, 'A minimalist approach to offline reinforcement learning', in *Neural Information Processing Systems*, (2021).

[11] Scott Fujimoto, David Meger, and Doina Precup, 'Off-policy deep reinforcement learning without exploration', in *International Conference on Machine Learning*, (2018).

[12] Shixiang Shane Gu, Ethan Holly, Timothy P. Lillicrap, and Sergey Levine, 'Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates', *International Conference on Robotics and Automation*, (2016).

[13] Kaiyang Guo, Yunfeng Shao, and Yanhui Geng, 'Model-based offline reinforcement learning with pessimism-modulated dynamics belief', in *Neural Information Processing Systems*, (2022).

[14] Jonathan Ho, Ajay Jain, and P. Abbeel, 'Denoising diffusion probabilistic models', in *Neural Information Processing Systems*, (2020).

[15] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine, 'When to trust your model: Model-based policy optimization', in *Neural Information Processing Systems*, (2019).

[16] Michael Janner, Qiyang Li, and Sergey Levine, 'Reinforcement learning as one big sequence modeling problem', in *Neural Information Processing Systems*, (2021).

[17] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims, 'Morel: Model-based offline reinforcement learning', in *Neural Information Processing Systems*, (2020).

[18] Diederik P. Kingma and Max Welling, 'Auto-encoding variational bayes', *ArXiv*, **abs/1312.6114**, (2013).

[19] Ilya Kostrikov, Ashvin Nair, and Sergey Levine, 'Offline reinforcement learning with implicit q-learning', in *International Conference on Learning Representation*, (2021).

[20] Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum, 'Offline reinforcement learning with fisher divergence critic regularization', in *International Conference on Machine Learning*, (2021).

[21] Aviral Kumar, Justin Fu, G. Tucker, and Sergey Levine, 'Stabilizing off-policy q-learning via bootstrapping error reduction', in *Neural Information Processing Systems*, (2019).

[22] Aviral Kumar, Aurick Zhou, G. Tucker, and Sergey Levine, 'Conservative q-learning for offline reinforcement learning', in *Neural Information Processing Systems*, (2020).

[23] Sascha Lange, Thomas Gabel, and Martin A. Riedmiller, 'Batch reinforcement learning', in *Reinforcement Learning*, (2012).

[24] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim, 'Representation balancing offline model-based reinforcement learning', in *International Conference on Learning Representations*, (2021).

[25] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill, 'Off-policy policy gradient with state distribution correction', *ArXiv*, **abs/1904.08473**, (2019).

[26] Cong Lu, Philip J. Ball, Jack Parker-Holder, Michael A. Osborne, and Stephen J. Roberts, 'Revisiting design choices in model-based offline reinforcement learning', in *International Conference on Learning Representations*, (2021).

[27] Jiafei Lyu, Xiu Li, and Zongqing Lu, 'Double check your state before trusting it: Confidence-aware bidirectional offline model-based imagination', in *Neural Information Processing Systems*, (2022).

[28] Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu, 'Mildly conservative q-learning for offline reinforcement learning', in *Neural Information Processing Systems*, (2022).

[29] Yecheng Jason Ma, Dinesh Jayaraman, and Osbert Bastani, 'Conservative offline distributional reinforcement learning', in *Neural Information Processing Systems*, (2021).

[30] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Shane Gu, 'Deployment-efficient reinforcement learning via model-based offline optimization', in *International Conference on Learning Representations*, (2020).

[31] Mehdi Mirza and Simon Osindero, 'Conditional generative adversarial nets', in *Neural Information Processing Systems*, (2014).

[32] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans, 'Algaedice: Policy gradient from arbitrary experience', *ArXiv*, **abs/1912.02074**, (2019).

[33] Feiyang Pan, Jia He, Dandan Tu, and Qing He, 'Trust the model when it is confident: Masked model-based actor-critic', in *Neural Information Processing Systems*, (2020).

[34] Elsa Riachi, Muhammad Mamdani, Michael Fralick, and Frank Rudzicz, 'Challenges for reinforcement learning in healthcare', *ArXiv*, **abs/2103.05612**, (2021).

[35] Marc Rigter, Bruno Lacerda, and Nick Hawes, 'Rambo-rl: Robust adversarial model-based offline reinforcement learning', in *Neural Information Processing Systems*, (2022).

[36] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Kumar Yogamani, 'Deep reinforcement learning framework for autonomous driving', *ArXiv*, **abs/1704.02532**, (2017).

[37] Phillip Swazinna, Steffen Udluft, and Thomas A. Runkler, 'Overcoming model bias for robust offline deep reinforcement learning', *Engineering Applications of Artificial Intelligence*, **104**, (2020).

[38] Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang, 'Offline reinforcement learning with reverse model-based imagination', in *Neural Information Processing Systems*, (2021).

[39] Yifan Wu, G. Tucker, and Ofir Nachum, 'Behavior regularized offline reinforcement learning', *ArXiv*, **abs/1911.11361**, (2019).

[40] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua M. Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh, 'Uncertainty weighted actor-critic for offline reinforcement learning', in *International Conference on Machine Learning*, (2021).

[41] Shentao Yang, Shujian Zhang, Yihao Feng, and Mi Zhou, 'A unified framework for alternating offline model training and policy learning', in *Neural Information Processing Systems*, (2022).

[42] Yijun Yang, Jing Ping Jiang, Tianyi Zhou, Jie Ma, and Yuhui Shi, 'Pareto policy pool for model-based offline reinforcement learning', in *International Conference on Learning Representations*, (2022).

[43] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn, 'Combo: Conservative offline model-based policy optimization', in *Neural Information Processing Systems*, (2021).

[44] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma, 'Mopo: Model-based offline policy optimization', in *Neural Information Processing Systems*, (2020).

[45] Xianyuan Zhan, Xiangyu Zhu, and Haoran Xu, 'Model-based offline planning with trajectory pruning', in *IJCAI-ECAI 22*, (2022).

[46] Wenxuan Zhou, Sujay Bajracharya, and David Held, 'Plas: Latent action space for offline reinforcement learning', in *Conference on Robot Learning*, (2020).