2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Improving Offline Reinforcement Learning with Inaccurate Simulators

Yiwen Hou, Haoyuan Sun, Jinming Ma, and Feng Wu*

***Abstract*— Offline reinforcement learning (RL) provides a promising approach to avoid costly online interaction with the real environment. However, the performance of offline RL highly depends on the quality of the datasets, which may cause extrapolation error in the learning process. In many robotic applications, an inaccurate simulator is often available. However, the data directly collected from the inaccurate simulator cannot be directly used in offline RL due to the well-known exploration-exploitation dilemma and the dynamic gap between inaccurate simulation and the real environment. To address these issues, we propose a novel approach to combine the offline dataset and the inaccurate simulation data in a better manner. Specifically, we pre-train a generative adversarial network (GAN) model to fit the state distribution of the offline dataset. Given this, we collect data from the inaccurate simulator starting from the distribution provided by the generator and reweight the simulated data using the discriminator. Our experimental results in the D4RL benchmark and a real-world manipulation task confirm that our method can benefit more from both inaccurate simulator and limited offline datasets to achieve better performance than the state-of-the-art methods.**

I. Introduction

Deep reinforcement learning (RL) has shown impressive success in many robotic applications [1]. However, applying RL to real-world scenarios is still very challenging because exploration and interaction with real-world environments are often costly or risky for physical robots, and RL methods often require millions of such data to learn a good policy. Most recently, offline RL emerges as a promising solution to address the dilemma above, allowing for learning efficient policies offline entirely from previously collected data [2].

However, Offline RL presents several significant challenges, such as the extrapolation error incurred by the mismatch between the experience distributions of the learned policy and the dataset [3]. To minimize this error, most prior work attempts to constrain the trained policy to the offline dataset's action space [3]–[6], value regularization [7], [8] on out-of-distribution (OOD) actions, or weighted [9]–[12] or conditioned [13]–[15] behavior cloning. Although these methods achieve considerable success in the offline setting, they are still heavily reliant on the quality of the offline dataset [16]–[18]. If a large portion of state-action space is not explored within the dataset, offline RL methods usually fail to learn good policies. In real-world scenarios, data collection is commonly expensive or limited, resulting in suboptimal and noisy offline datasets. This directly limits the potential application of offline RL in robotics.

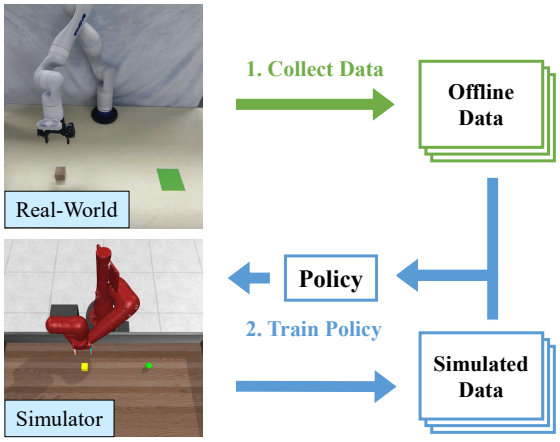


Fig. 1: Combination of offline datasets from real world and an inaccurate simulator (MuJoCo) for improving offline RL.

Fortunately, in many robotics tasks, a simulator (e.g. Gazebo [19], SimSpark [20], MuJoCo [21]) is often available. Within a simulator, robots can engage in unrestricted exploration, gaining access to a vast array of state-action data. These data can compensate for the limitations of the offline dataset in hand. Although the combination of real-world data and simulated data seems promising, there are two major challenges within this approach. Firstly, although the simulator provides a risk-free environment for exploration, low-quality or aimless exploration data might not be sufficient and effective in addressing the OOD issues within the offline dataset. Thus, striking a balance between the *exploration* in the simulator and the *exploitation* of offline datasets becomes crucial. Secondly, accurately modeling the complex dynamics of the real world is often intractable or expensive. In contrast, *inaccurate simulators* are relatively easier to obtain and more efficient (e.g., MuJoCo for RL). However, if we indiscriminately treat the inaccurate simulated data equally with the real offline data, the *dynamics gap* [22] between the real and simulated environments might adversely impact the effectiveness of offline policy learning.

To address these issues, we propose a novel method called **O**ffline **R**einforcement learning with **I**naccurate **S**imulator (ORIS), as shown in Fig. 1, which aims to 1) collect more effective data from the inaccurate simulator to achieve exploration-exploitation trade-off, and 2) better use the mixed datasets to alleviate the side-effect of inaccurate simulation. To this end, we first pre-train a Generative Adversarial

This work was supported in part by the Major Research Plan of the National Natural Science Foundation of China (92048301), Anhui Provincial Major Research and Development Plan (202004H07020008), and Anhui Province Development and Reform Commission 2021 New Energy and Intelligent Connected Vehicle Innovation Project.

All authors are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China. E-mail: ({houyiwen,sunhaoyuan,jinmingm}@mail.ustc.edu.cn, wufeng02@ustc.edu.cn). *Feng Wu is the corresponding author.

Network (GAN) [23] model with the offline dataset to fit the state distribution of the offline dataset. Given this, we interact with an inaccurate simulator by a hybrid rollout policy from the initial state distribution provided by the GAN generator, which effectively balances the exploration-exploitation dilemma. Then, we employ the GAN discriminator to adeptly integrate both the offline datasets and the simulated data, reducing the side-effect of inaccurate simulator and enhancing the precision of Q value estimations. We conducted experiments on the D4RL dataset [24], a common offline RL benchmark, as well as real-world robotic manipulations. The experimental results show that our method achieved better performance than the state-of-the-art methods especially given limited amount of data.

II. Background

We formally model the robotic RL problem as a Markov Decision Process (MDP): $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \rho_0, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space; $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the environment dynamics; $\rho_0 \in \Delta(S)$ is the distribution of the initial states; $r(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function; $\gamma \in (0, 1]$ is the discount factor. The goal of RL is to learn a policy $\pi(a|s) : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the cumulative discounted returns: $\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$ from the experiences without directly accessing the model.

To date, off-policy actor-critic algorithms are one of the most commonly used frameworks to solve the RL problem without consideration of how the experiences were generated, which learn a Q-function $Q_\theta(s, a)$ by minimizing the Bellman error and a policy π_ϕ by maximizing the Q-function, where θ and ϕ are the parameters of Q-function and policy, and the loss functions are as follow:

$$\mathcal{L}_Q(\theta) = \mathop{\mathbb{E}}_{(s,a,s')\sim\mathcal{B}} \left[(Q_\theta(s, a) - \mathcal{T}^{\pi_\phi} Q_\theta(s, a))^2 \right] \quad (1)$$

$$\mathcal{L}_\pi(\phi) = \mathop{\mathbb{E}}_{(s)\sim\mathcal{B}, a\sim\pi_\phi(\cdot|s)} \left[-Q_\theta(s, a) \right] \quad (2)$$

where $\mathcal{B}$ is the replay buffer, $\mathcal{T}^\pi$ is the Bellman operator and $\mathcal{T}^\pi Q_\theta(s, a) = r(s, a) + \gamma \, \mathbb{E}_{a'\sim\pi(\cdot|s')} \left[Q_{\bar{\theta}}(s', a') \right]$.

In offline RL, offline dataset $\mathcal{D}_{\text{off}} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{|\mathcal{D}_{\text{off}}|}$ consists of transitions collected from the real environment $\mathcal{M}$ by some unknown behavior policy β. Here, we additionally consider an *inaccurate simulator* modeled as an MDP $\widehat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \widehat{\mathcal{P}}, \rho_0, r, \gamma)$, where the dynamics of the inaccurate simulator $\widehat{\mathcal{P}}$ is different from the dynamics $\mathcal{P}$ in real environment. We can interact with such an inaccurate simulator by policy π to collect the data $\mathcal{D}_{\text{sim}} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{|\mathcal{D}_{\text{sim}}|}$. Our goal is to leverage the augmented dataset $\mathcal{D} = \mathcal{D}_{\text{off}} \cup \mathcal{D}_{\text{sim}}$ to learn a policy for completing tasks in the real environment.

Given a *limited offline dataset* and an *inaccurate simulator*, there are two key challenges: 1) exploitation-exploration trade-off when interacting with the simulator to collect useful data for learning and 2) reducing side-effect when leveraging both the real data and inaccurate simulation data.

III. Method

Here, we propose our method, named **O**ffline **R**einforcement Learning with **I**naccurate **S**imulator (ORIS). As shown in Fig. 2, the GAN generator G and discriminator D are trained for addressing the aforementioned challenges. Specifically, the generator aims to generate states that align with the state distribution of the offline dataset, while the discriminator is tasked with discerning whether a given state is either in-distribution or out-of-distribution. After that, we iteratively collect data using the inaccurate simulator and train the policy with the mixed dataset. During data collection, we interact with the simulator from the restart distribution provided by the generator G to avoid aimless exploration. For the trade-off of exploitation-exploration, the rollout policy $\pi_{rollout}$ is a hybrid policy consisting of the random policy and the current policy π_ϕ. During policy training, we utilize an off-policy actor-critic algorithm, i.e., SAC [25], training with both the offline dataset and the collected simulated data. Here, we adopt the weighted critic loss with discriminator D to down-weight potentially harmful simulated data. Next, we will describe our method in more details.

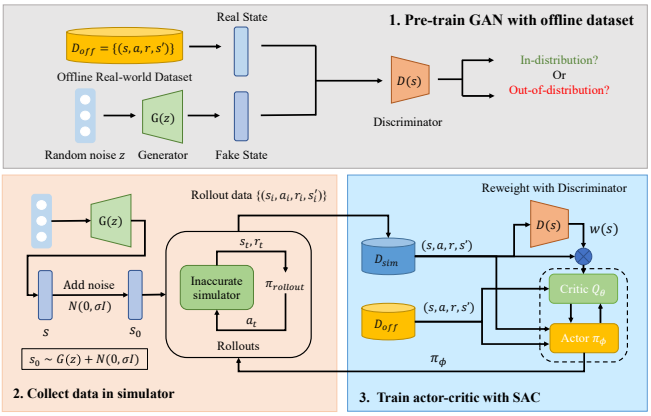


Fig. 2: Overview of our ORIS framework.

A. Pre-train GAN with Offline Dataset

To improve offline RL with inaccurate simulator, the key issues are the exploitation-exploration trade-off when sampling and the inaccurate Q value estimation when minimizing the critic loss due to the dynamics gap. To avoid aimless exploration, we can sample trajectories starting from the state distribution fitting the offline dataset. To alleviate inaccurate Q value estimation, we need to discriminate potentially harmful data. We will elaborate on these issues later.

Fortunately, the structure of the generator and discriminator of GAN provides a promising solution to solve these two problems in a unified manner. Note that we choose GAN for its simplicity and effectiveness though any generative model with generator and discriminator (e.g., WGAN [26]) is compatible with our framework. As shown in Fig. 2, the generator G learns to produce samples that closely resemble the state distribution of the offline dataset, while the discriminator D learns to distinguish between samples generated by G and the real samples from the offline dataset:

$$\min_G \max_D \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \left[\log(D(s)) \right] + \mathop{\mathbb{E}}_{z\sim p(z)} \left[\log(1 - D(G(z))) \right] \quad (3)$$

To solve the min-max optimization problem, we update the generator and the discriminator iteratively. At every iteration k, we update G_ψ^k and D_ω^k as follow:

$$G_\psi^{k+1} \leftarrow \arg\min_\psi \mathop{\mathbb{E}}_{z\sim p(z)} \Big[\log(1 - D_\omega^k(G_\psi^k(z)))\Big] \tag{4}$$

$$D_\omega^{k+1} \leftarrow \arg\max_\omega \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \Big[\log(D_\omega^k(s))\Big] + \mathop{\mathbb{E}}_{z\sim p(z)} \Big[\log(1-D_\omega^k(G_\psi^k(z)))\Big] \tag{5}$$

After the GAN is trained, we use the generator G_ψ to produce initial state distributions for the simulated process. Additionally, the discriminator D_ω is employed to assess the simulated data, in order to reweight it during the off-policy learning process. Next, we will illustrate how we use the generator and discriminator to better leverage the potential of the inaccurate simulator.

B. Collecting Simulated Data with Generator

As aforementioned, using the off-policy algorithm directly with limited offline data can lead to extrapolation errors stemming from the out-of-distribution (OOD) issue. This error cannot be fixed during offline training without additional online interaction. Therefore, our goal is to supplement the offline dataset with data from a simulator. However, it is still challenging to collect useful data for policy training due to the exploitation-exploration dilemma.

Most methods directly reset to initial states ρ_0 when sampling trajectories. However, this will lead to massive aimless exploration around ρ_0, especially for some robotic tasks with sparse reward. Fortunately, offline datasets usually consist of some trajectories completing the target task, which can provide valuable guidance for avoiding such exploration. For exploitation of the offline dataset, we utilize the generator of the pre-trained GAN model, which fits the state distribution of offline datasets. Specifically, we use $G(z)$ to generate the restart distribution $\rho_G \sim G(z)+\mathcal{N}(0,\sigma I)$. When interacting with the simulator starting from $s_0 \sim \rho_G$, it is more likely to sample high-return trajectories, which can be difficult to explore when starting from ρ_0. For exploration within the simulator, we utilize a hybrid rollout policy similar to ε-greedy. Specifically, we choose the random policy with probability p, and the current policy with probability $1-p$. Such a hybrid rollout policy can explore the action space, effectively rectifying the extrapolation errors associated with OOD actions and thereby facilitating the discovery of an improved policy around the offline dataset. All in all, the restart distribution and the hybrid policy balance the exploitation-exploration dilemma in a more effective way.

C. Reweighting Data with Discriminator

Although simulated data can complement limited offline data and alleviate unrealistic overestimation of the Q-value, treating all simulated and real data equally may harm the performance of the policy due to the dynamics gap. The primary dynamics gap between $\mathcal{M}$ and $\widehat{\mathcal{M}}$ originates from the environment dynamics $\mathcal{P}$, that is, $(s,a,r,s') \in \mathcal{D}_{\text{off}}$ and $(s,a,r,\widehat{s'}) \in \mathcal{D}_{\text{sim}}$. Based on the Bellman equation, the Q-value estimates of similar state-action pairs (s,a) are given by: $Q_{\mathcal{M}}(s,a) = r+\gamma\,\mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q(s',a')\right]$ and $Q_{\widehat{\mathcal{M}}}(s,a) = r+\gamma\,\mathbb{E}_{a'\sim\pi(\cdot|\widehat{s'})}\Big[Q(\widehat{s'},a')\Big]$. If we directly minimize the critic loss as delineated in Eq.1, the variance in the target Q-value, i.e., $Q_{\mathcal{M}}(s,a)$ and $Q_{\widehat{\mathcal{M}}}(s,a)$, may result in an ambiguous estimation of $Q(s,a)$. To address this, we propose to assign a lower weight to the transition in $\mathcal{D}_{\text{sim}}$. By doing so, we can reduce the impact on the overall Q-value estimation, and thereby alleviate the potential negative effects caused by the dynamics gap between $\mathcal{M}$ and $\widehat{\mathcal{M}}$.

Specifically, we employ adaptive weighting for simulated data to down-weight any potentially harmful simulated information. The corresponding weighted critic loss is as:

$$\begin{aligned}\mathcal{L}_Q(\theta) = &\mathop{\mathbb{E}}_{(s,a,s')\sim\mathcal{D}_{\text{off}}}\left[(Q_\theta(s,a) - \mathcal{T}^{\pi_\phi}Q_\theta(s,a))^2\right] \\ &+ \mathop{\mathbb{E}}_{(s,a,s')\sim\mathcal{D}_{\text{sim}}} w(s)\left[(Q_\theta(s,a) - \mathcal{T}^{\pi_\phi}Q_\theta(s,a))^2\right]\end{aligned} \tag{6}$$

where $w(s) = \text{clip}(1-2D(s), w_{\min}, w_{\max})$, and $w_{\min}$ is a small positive number. The underlying idea behind the weight $w(s)$ is as follows. For a given transition $(s_{\text{sim}}, a_{\text{sim}}, r_{\text{sim}}, s'_{\text{sim}}) \in \mathcal{D}_{\text{sim}}$, if the state s_{sim} is in-distribution (i.e., the offline dataset already contains transitions of s_{sim}), we assign a small weight to that transition when calculating the critic loss. This prevents inconsistency in Q-value estimation for s_{sim} between the offline data and simulated data. In such cases, $D(s) \to 0.5$ and the weight $w(s) \to w_{\min}$. Conversely, if the state s_{sim} is out-of-distribution and absent from the offline dataset, there is no need to assign a small weight to it, as it will not conflict with offline data. For such states, $D(s) \to 0$ and the weight $w(s) \to w_{\max}$.

Building upon this, we minimize the actor loss by directly minimizing it over the dataset combining both the offline and simulated data, $\mathcal{D}_{\text{off}} \cup \mathcal{D}_{\text{sim}}$. The updated actor loss is as:

$$\mathcal{L}_\pi(\phi) = \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D}_{\text{off}}\cup\mathcal{D}_{\text{sim}},\\ a\sim\pi_\phi(\cdot|s)}} \left[-Q_\theta(s,a) + \lambda \log \pi_\phi(a|s)\right] \tag{7}$$

where λ is the temperature parameter as in SAC. Note that ORIS can be compatible with any off-policy actor-critic algorithm (e.g., TD3 [27]). With the components described above, the main procedures of ORIS are outlined in Algorithm 1.

Algorithm 1 **O**ffline **R**L with **I**naccurate **S**imulator (ORIS)

Input: offline dataset $\mathcal{D}_{\text{off}}$, inaccurate simulator with biased dynamics $\widehat{\mathcal{P}}$, rollout horizon H, rollout count C
Output: policy network π_ϕ

Initialize discriminator $D_\omega(\cdot|s)$ and generator $G_\psi(\cdot|z)$
Initialize policy network π_ϕ and critic network Q_θ
Initialize replay buffer $\mathcal{D}_{\text{sim}} \leftarrow \emptyset$
Pre-train $D_\omega(\cdot|s)$ and $G_\psi(\cdot|z)$ using Eq.4 and Eq.5
for $epoch = 0, 1, \cdots$ **do**
  $\pi_{\text{rollout}} = \begin{cases}\pi_{\text{random}} & \text{with probability } p \\ \pi_\phi & \text{with probability } 1-p\end{cases}$
  for $c = 0, 1, \ldots, C$ **do** ▷ *Collect data in simulator*
    Generate a restart state $s_0 \sim G(z) + \mathcal{N}(0, \sigma I)$
    Rollout H steps in the simulator starting from s_0 by π_{rollout}, and add the rollout to $\mathcal{D}_{\text{sim}}$
  Sample minibatch data $B_{\text{off}} \sim \mathcal{D}_{\text{off}}, B_{\text{sim}} \sim \mathcal{D}_{\text{sim}}$
  Update Q_θ and π_ϕ via minimizing Eq.6 and Eq.7

TABLE I: Average normalized scores of all methods. (g2.0) means the simulator with 2 times gravity.

Task Name	BC	CQL	IQL	TD3+BC	COMBO	SQL	SAC(g2.0)	H2O(g2.0)	ORIS(g2.0)(Ours)
hopper-random	3.7±0.6	7.9±0.4	7.9±0.2	8.5±0.6	17.9±1.4	7.8±0.5	10.0±5.6	20.6±9.8	**31.4±0.4**
hopper-medium-replay	16.6±4.8	88.7±12.9	94.7±8.6	60.9±18.8	89.5±1.8	96.7±3.3	10.0±5.6	46.7±28.0	**100.6±0.3**
hopper-medium	54.1±3.8	53.0±28.5	66.2±5.7	59.3±4.2	97.2±2.2	73.5±3.4	10.0±5.6	21.6±17.7	**99.8±0.8**
hopper-medium-expert	53.9±4.7	105.6±12.9	91.5±14.3	98.0±9.4	111.1±2.9	**111.8±2.2**	10.0±5.6	25.2±24.4	110.1±1.5
walker2d-random	1.3±0.1	5.1±1.3	5.4±1.2	1.6±1.7	7.0±3.6	5.1±0.4	30.2±19.9	12.1±6.3	**30.4±15.0**
walker2d-medium-replay	20.3±9.8	81.8±2.7	73.8±7.1	81.8±5.5	56.0±8.6	77.2±3.8	30.2±19.9	39.2±25.9	**91.5±0.5**
walker2d-medium	70.9±11.0	73.3±17.7	78.3±8.7	83.7±2.1	81.9±2.8	84.2±4.6	30.2±19.9	34.4±15.2	**86.2±5.3**
walker2d-medium-expert	90.1±13.2	107.9±1.6	109.6±1.0	**110.1±0.5**	103.3±5.6	110.0±0.8	30.2±19.9	27.3±17.0	102.8±2.2
halfcheetah-random	2.2±0.0	17.5±1.5	13.1±1.3	11.0±1.1	38.8±3.7	14.4±1.0	**43.3±1.5**	35.2±1.4	39.2±1.7
halfcheetah-medium-replay	37.6±2.1	45.5±0.7	44.2±1.2	44.6±0.5	55.1±1.0	44.8±0.7	43.3±1.5	52.8±5.5	**59.6±2.4**
halfcheetah-medium	43.2±0.6	47.0±0.5	47.4±0.2	48.3±0.3	54.2±1.5	48.3±0.2	43.3±1.5	55.2±4.9	**68.2±2.1**
halfcheetah-medium-expert	44.0±1.6	75.6±25.7	86.7±5.3	90.7±4.3	90.0±5.6	**94.0±0.4**	43.3±1.5	33.0±6.6	74.5±4.9
Average Score	36.5±4.4	59.1±8.9	59.9±4.6	58.2±4.1	66.8±3.4	64.0±1.8	27.8±9.0	33.6±13.6	**74.5±3.1**

IV. Experiments

We pose the following questions and provide affirmative answers in our experiments: Q1) Is ORIS effective for RL with offline data and an inaccurate simulator? Q2) When the offline data is reduced, can ORIS make full use of the simulator to supplement the data? Q3) Is ORIS robust to the inaccuracy of the simulator? Q4) How do the different components affect the performance? Q5) Is ORIS effective in real-world robotic tasks?

A. D4RL Benchmarks and Baselines

We first evaluate our method on the locomotion tasks of the widely used D4RL [24] dataset, i.e., *halfcheetah*, *hopper* and *walker2d*, and use the MuJoCo physics simulator [21]. For each domain, we reconstruct three task simulation environments with intentionally introduced dynamics gaps upon the original locomotion tasks (which serve as the real environments) by modifying the dynamics parameters identical to H2O [22]: 1) **Gravity**: applying 2 times the gravitational acceleration in the simulator; 2) **Friction**: using 0.3 times the friction coefficient; 3) **Action Noise**: adding a random noise sampled from a standard normal distribution $\mathcal{N}(0, 1)$ on every dimension of the action space.

We compare our method with the state-of-the-art online, offline, and hybrid offline-and-online RL methods. For the online RL, we compare with SAC [25], which is trained in the modified simulator, and evaluate the policy in the original environment. For the offline RL, we compare with behavior cloning (BC), CQL [7], IQL [9], TD3+BC [6], COMBO [8], SQL [12], which are trained on the fixed offline dataset. The results of offline baselines are taken directly from their corresponding papers. For the hybrid method, we compare with H2O [22], which uses the same settings as ours. During training, we collect simulated data from the modified (inaccurate) simulator and evaluate the learned policy on the original (accurate) simulator. The policy is trained for 500K steps and evaluated over 10 episodes every 1000 steps. The results are over five random seeds.

B. Results on D4RL Benchmarks

To answer question Q1, we compare ORIS with all the baselines on the D4RL dataset. In Table I, SAC, H2O, and ORIS collect simulated data from the simulator with 2 times the gravitational acceleration, i.e., the (g2.0) suffix. ORIS outperforms the baselines in most of the tasks. Offline RL performs well on *medium-expert* datasets that contain expert trajectories. However, they perform poorly on other types of datasets, which usually contain sub-optimal human demonstrations. The results show that the performance of offline RL is greatly affected by the quality of the dataset. H2O behaves relatively well in *halfcheetah* but poorly in *hopper* and *walker2d* tasks because the agent quickly falls down and is unable to explore valuable data in the simulator when starting from ρ_0. In contrast, ORIS starts the rollout from the states generated by GAN and achieves good performance, which benefits the balance of exploitation of the offline data and exploration of the inaccurate simulator.

In addition, we compare SAC and H2O under different simulators with varying dynamics, and the results are shown in Table II. It can be seen that ORIS consistently outperforms H2O, which demonstrates that our method can take more advantages than H2O from the inaccurate simulator. SAC suffers from the dynamics gap in most cases but sometimes outperforms ORIS on random datasets. This is because our method trusts the offline dataset, which may be misleading especially in the random cases.

C. Results on Small Dataset

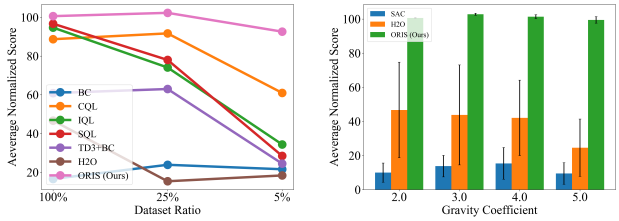
Given a small offline dataset, we aim to assess if our method can effectively leverage the simulator to supplement the data. For the limited dataset, we split the *hopper-medium-replay-v2* dataset into trajectories and then randomly selected subsets comprising 25% and 5% of these trajectories to create two new limited datasets. We compare our method with offline baselines and H2O given the limited datasets.

As shown in Fig. 3(a), the performances of almost all offline RL methods drop dramatically as the amount of data decreases, which shows that offline RL methods are extremely sensitive to data quantity. H2O also performs worse with less data, showing that the performance of H2O also highly relies on the amount of offline data.

By leveraging the simulator, our method outperforms all baselines on all limited datasets. Remarkably, the final scores only dropped 8% when the amount of data decreased from 100% to 5%. This demonstrates that our method can effectively use the simulator to compensate for the extremely little

TABLE II: Average normalized scores of SAC, H2O and our method for simulators with different unreal dynamics.

Unreal Dynamics	**Gravity**			**Friction**			**Action Noise**		
Task Name	**SAC**	**H2O**	**ORIS(Ours)**	**SAC**	**H2O**	**ORIS(Ours)**	**SAC**	**H2O**	**ORIS(Ours)**
hopper-random	10.0±5.6	20.6±9.8	**31.4±0.4**	**56.8±31.8**	15.8±7.3	34.2±7.7	**37.4±18.2**	13.7±3.2	34.4±14.4
hopper-medium-replay	10.0±5.6	46.7±28.0	**100.6±0.3**	56.8±31.8	43.0±28.6	**102.0±1.3**	37.4±18.2	66.3±28.7	**101.6±1.1**
hopper-medium	10.0±5.6	21.6±17.7	**99.8±0.8**	56.8±31.8	24.9±9.7	**101.1±0.8**	37.4±18.2	75.2±16.5	**98.5±3.2**
hopper-medium-expert	10.0±5.6	25.2±24.4	**110.1±1.5**	56.8±31.8	15.7±14.6	**106.2±7.7**	37.4±18.2	31.4±20.9	**109.1±3.9**
walker2d-random	30.2±19.9	12.1±6.3	**30.4±15.0**	**76.1±8.9**	9.3±5.2	22.5±16.8	**19.7±7.8**	11.4±5.5	16.6±4.0
walker2d-medium-replay	30.2±19.9	39.2±25.9	**91.5±0.5**	76.1±8.9	75.6±12.7	**93.5±9.9**	19.7±7.8	36.4±18.6	**95.3±3.6**
walker2d-medium	30.2±19.9	34.4±15.2	**86.2±5.3**	76.1±8.9	30.4±9.3	**89.6±9.9**	19.7±7.8	30.4±6.9	**88.0±5.7**
walker2d-medium-expert	30.2±19.9	27.3±17.0	**102.8±2.2**	76.1±8.9	44.7±14.4	**103.4±1.8**	19.7±7.8	40.4±13.7	**107.7±1.1**
halfcheetah-random	**43.3±1.5**	35.2±1.4	39.2±1.7	41.8±3.7	42.7±15.6	**53.0±4.2**	**41.2±3.8**	9.3±0.4	20.5±0.3
halfcheetah-medium-replay	43.3±1.5	52.8±5.5	**59.6±2.4**	41.8±3.7	53.9±4.9	**65.6±3.0**	41.2±3.8	53.9±4.2	**59.6±0.8**
halfcheetah-medium	43.3±1.5	55.2±4.9	**68.2±2.1**	41.8±3.7	51.1±7.0	**73.2±2.3**	41.2±3.8	60.1±2.7	**66.2±1.3**
halfcheetah-medium-expert	43.3±1.5	33.0±6.6	**74.5±4.9**	41.8±3.7	18.4±6.2	**86.8±3.5**	41.2±3.8	33.7±8.6	**76.0±7.8**
Average Score	27.8±9.0	33.6±13.6	**74.5±3.1**	58.2±14.8	35.5±11.3	**77.6±5.7**	32.8±9.9	38.5±10.8	**72.8±3.9**



(a) Small Datasets (b) Simulator Inaccuracies

Fig. 3: Results on small datasets (25% and 5%) (left) and different simulator inaccuracies (right).

Fig. 4: Ablation experiments for different modules.

data. We attribute this robust performance to the generator's ability to accurately fit the state distribution even with limited offline datasets. Furthermore, the simulated data, collected from restart distribution ρ_G, proves to be a more effective supplement to the limited offline data.

D. Results on Robustness to Simulator Inaccuracy

In the subsection, we investigate the robustness of our method with varying simulators. In complex applications, the gap between the simulator and the real environment can be substantial. Specifically, we compare our method with SAC and H2O on the *hopper-medium-replay-v2* while increasing the simulator's inaccuracy by applying a scaling factor of *Gravity Coefficient (GC)* to the gravitational acceleration in the simulator, i.e., $GC = 3, 4, 5$. Intuitively, the inaccuracy of the simulator grows as GC increases.

As shown in Fig. 3(b), the performance of H2O drops when GC increases, which is because H2O cannot fully utilize the simulator with a large gap and accurately estimate the value function of state-action pair from offline dataset due to the poor coverage of state-action space. Without the information of the real dynamics, SAC is severely affected by the dynamics gap, and cannot repair the impact of the gap. Our method is robust to the gap and the performance only drops slightly when $GC = 5$. These results show that our method can better utilize the inaccurate simulator and is more robust to inaccurate factors.

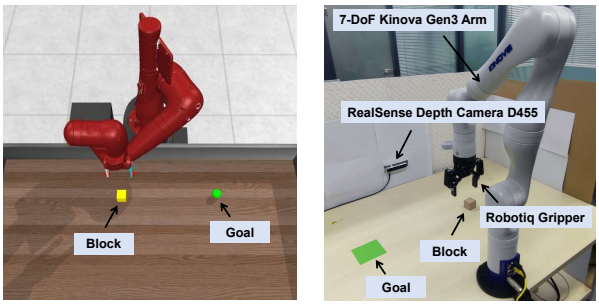
E. Ablation Studies

In the subsection, we verify the effectiveness of the components of our approach on *hopper-medium-replay-v2* datasets. We test 1) **w/o restart**: starting from the initial state $s_0 \sim \rho_0$ to replace the distribution generated by the GAN generator in our method and keep the rest of the modules unchanged; 2) **H2O reweight** $w(s, a, s')$ (Eq. 8 in [22]) for simulated data instead of our discriminator-based re-weighting; 3) **w/o restart & H2O reweight**: replace both start distribution and re-weighting method, which is the same as H2O.

As shown in Fig. 4, the **Original** version of our method achieves the best performance. The performance of the **w/o restart** version drops by a significant margin (46.9%), which shows that initiating from the states produced by the GAN generator can effectively enhance performance. For the re-weighting module, our approach outperforms **H2O reweight** (with a drop of about 9.4%). These results highlight the effectiveness of our re-weighting module.

F. Real-World Robotic Manipulation

We now test the performance of ORIS to a real-world robot, which is typically more complex and involve many factors contributing to the simulation gap (e.g., modeling error, calibration error, control inaccuracy). As shown in Fig. 5, we use a 7-DoF Kinova Gen3 robotic arm equipped with a Robotiq gripper to do the *Pick-and-place* task. The state space is represented by a 7-dimensional vector, including end-effector position (3D), gripper opening state (1D), and position of



(a) MuJoCo Metaworld Simulator (b) Real-world Kinova Arm

Fig. 5: Pick-and-place in simulation (left) and real (right).

the block (3D). The block's pose is estimated using Apritag [28] from RGB images captured by the RealSense depth camera D455. The action space is a 4-dimensional vector, responsible for dictating the velocity of the end-effector (3D) and the status of the gripper (1D, i.e., open or close). The reward function is defined as $r = -0.1 + 0.1 \times \mathbb{I}(d_1 < 0.1) + 20.0 \times \mathbb{I}(d_2 < 0.05)$, where $\mathbb{I}(\cdot)$ is the indicator function, d_1 is the distance between the gripper and the object and d_2 is the distance between the object and the goal. To collect an offline dataset, we employed a rule-based policy created manually, which generated trajectories for placing the object at various positions. For the simulator, we use an off-the-shelf simulation environment: Metaworld [29], based on Mujoco [21], without changing its robot model and parameters. During evaluation, the object is initially placed at 25 different positions in each episode.

The results are presented in Table III. Among offline RL methods trained solely on the offline dataset, CQL and TD3+BC both encounter severe OOD errors, rendering them unable to pick up the block. While BC, IQL, and SQL can reach and grasp the object, they struggled to accurately place it in the correct position. This failure can be attributed to the low-quality nature of the dataset, which includes numerous task-agnostic trajectories. Among the methods that used the inaccurate simulator, SAC was able to successfully complete the task within the simulator. However, when transferring the learned policy to the real world, it faced challenges in both picking up and placing the object due to the dynamics gap. H2O performed poorly as it did not explore sufficiently within the simulator, hindering its ability to learn effective policies. In contrast, ORIS can complete this challenging task due to the proper utilization of both the limited offline data and the inaccurate simulator, which shows the effectiveness of our method in the real-world applications.

V. Related Work

Offline RL aims to address the problem of learning effective policies entirely from previously collected data using some behavior policies, without further online interaction [2], [3]. The main challenge is the extrapolation error [3], the overestimation of the value function of OOD actions, which is caused by the distribution shift between the current policy and the behavior policy [4], [7]. Several methods such as

TABLE III: Average return and success rate (SR) of all methods for the Pick-and-place task. The sub-tasks of "Reach", "Pick" and "Place" success when d_1<0.1, the height of the object z_{obj}>0.05 and d_2<0.05 respectively.

	BC	CQL	IQL	TD3+BC	SQL	SAC	H2O	ORIS
Average return	-6.50	-13.58	-3.84	-14.99	-3.26	4.20	-14.56	**18.18**
SR (Reach)	98.7%	29.3%	100.0%	6.7%	98.7%	100.0%	22.7%	**100.0%**
SR (Pick)	61.3%	0.0%	92.0%	0.0%	97.3%	30.3%	0.0%	**98.7%**
SR (Place)	0.0%	0.0%	0.0%	0.0%	4.0%	27.6%	0.0%	**98.7%**

BCQ [3], BEAR [4], BRAC [5], TD3+BC [6], and LAPO [30] constrain the learned policy to the behavior policy used to collect the dataset. Other methods such as CQL [7] and COMBO [8] constrain the learned policy by making conservative estimates of value functions of OOD actions. Another type of methods such as AWR [10], One-step [11], IQL [9] and SQL [12] does behavior cloning weighted by advantage of the data. Offline RL methods are often conservative and pessimistic, and their performance heavily depends on the quality, size, and coverage of the state-action space of the given offline dataset [16]–[18].

Another line of work does data augmentations to the offline dataset. S4RL [31] adds noise to the states in the offline dataset. Some model-based methods augment the offline dataset by learning a reverse model [32] or bidirectional models [33]. [34] propose mixup augmentation in the Koopman subspace for offline RL. Similar to ours, H2O [22] uses an inaccurate simulator to supplement the dataset, which learns a pair of discriminators to reweight the simulated data. However, H2O does not fully utilize the simulator. Our method uses the same settings with H2O but fully utilizes the simulator to augment the offline dataset by starting from state-distribution [35], [36] and re-weight simulated data according to the state-distribution.

Our work is also related to sim-to-real transfer [37]. Domain randomization randomizes the simulation to cover the real distribution of real-world data, including visual randomization [38]–[40] and dynamics randomization [41]. Some other works based on system identification [42], attempt to build a precise simulator for sim-to-real transfer. Unlike these works, ORIS does not require access to the parameters of the simulator for adjustment and can directly utilize the off-the-shelf simulator for real-world tasks.

VI. Conclusions

In this paper, we proposed a method ORIS to enhance the performance of offline RL by utilizing an inaccurate simulator. Firstly, we train a GAN to fit the state distribution of the offline dataset. Then we use the generator to produce starting states for rollouts in the simulator with a hybrid behavior policy, aiming to balance the exploration-exploitation dilemma of collecting useful data. Furthermore, we employ the discriminator to re-weight the simulated data for more precise Q value estimation. Our experiments on the D4RL benchmarks and the real-world task demonstrate the effectiveness of our approach. In the future, we plan to extend our method to tackle more challenging robotics tasks.

REFERENCES

[1] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3389–3396.

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

[3] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International Conference on Machine Learning (ICML)*, 2019, pp. 2052–2062.

[4] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

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

[6] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," in *Advances in neural information processing systems (NeurIPS)*, 2021, pp. 20 132–20 145.

[7] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 1179–1191.

[8] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn, "Combo: Conservative offline model-based policy optimization," in *Advances in neural information processing systems (NeurIPS)*, 2021, pp. 28 954–28 967.

[9] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *International Conference on Learning Representations (ICLR)*, 2021.

[10] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," *arXiv preprint arXiv:1910.00177*, 2019.

[11] D. Brandfonbrener, W. Whitney, R. Ranganath, and J. Bruna, "Offline rl without off-policy evaluation," in *Advances in neural information processing systems (NeurIPS)*, 2021, pp. 4933–4946.

[12] H. Xu, L. Jiang, J. Li, Z. Yang, Z. Wang, V. W. K. Chan, and X. Zhan, "Offline RL with no OOD actions: In-sample learning via implicit value regularization," in *International Conference on Learning Representations (ICLR)*, 2023.

[13] H. Xu, L. Jiang, L. Jianxiong, and X. Zhan, "A policy-guided imitation approach for offline reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022, pp. 4085–4098.

[14] S. Emmons, B. Eysenbach, I. Kostrikov, and S. Levine, "Rvs: What is essential for offline rl via supervised learning?" in *International Conference on Learning Representations (ICLR)*, 2021.

[15] R. Yang, Y. Lu, W. Li, H. Sun, M. Fang, Y. Du, X. Li, L. Han, and C. Zhang, "Rethinking goal-conditioned supervised learning and its connection to offline rl," in *International Conference on Learning Representations (ICLR)*, 2021.

[16] S. Y. Arnob, R. Islam, and D. Precup, "Importance of empirical sample complexity analysis for offline reinforcement learning," *arXiv preprint arXiv:2112.15578*, 2021.

[17] J. Chen and N. Jiang, "Information-theoretic considerations in batch reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2019, pp. 1042–1051.

[18] K. Schweighofer, M.-c. Dinu, A. Radler, M. Hofmarcher, V. P. Patil, A. Bitto-Nemling, H. Eghbal-zadeh, and S. Hochreiter, "A dataset perspective on offline reinforcement learning," in *Conference on Lifelong Learning Agents*, 2022, pp. 470–517.

[19] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154.

[20] Y. Xu and H. Vatankhah, "Simspark: An open source robot simulator developed by the robocup community," in *RoboCup 2013: Robot World Cup XVII 17*, 2014, pp. 632–639.

[21] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.

[22] H. Niu, Y. Qiu, M. Li, G. Zhou, J. HU, X. Zhan, *et al.*, "When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022, pp. 36 599–36 612.

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

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

[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning (ICML)*, 2018, pp. 1861–1870.

[26] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning (ICML)*, 2017, pp. 214–223.

[27] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning (ICML)*, 2018, pp. 1587–1596.

[28] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3400–3407.

[29] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning (CoRL)*, 2020, pp. 1094–1100.

[30] X. Chen, A. Ghadirzadeh, T. Yu, J. Wang, A. Y. Gao, W. Li, L. Bin, C. Finn, and C. Zhang, "Lapo: Latent-variable advantage-weighted policy optimization for offline reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022, pp. 36 902–36 913.

[31] S. Sinha, A. Mandlekar, and A. Garg, "S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics," in *Conference on Robot Learning (CoRL)*, 2022, pp. 907–917.

[32] J. Wang, W. Li, H. Jiang, G. Zhu, S. Li, and C. Zhang, "Offline reinforcement learning with reverse model-based imagination," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 29 420–29 432.

[33] J. Lyu, X. Li, and Z. Lu, "Double check your state before trusting it: Confidence-aware bidirectional offline model-based imagination," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022, pp. 38 218–38 231.

[34] J. Jang, J. Han, and J. Kim, "K-mixup: Data augmentation for offline reinforcement learning using mixup in a koopman invariant subspace," *Expert Systems with Applications*, vol. 225, p. 120136, 2023.

[35] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6292–6299.

[36] A. Sharma, R. Ahmad, and C. Finn, "A state-distribution matching approach to non-episodic reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2022, pp. 19 645–19 657.

[37] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 737–744.

[38] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[39] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 969–977.

[40] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," in *Conference on Robot Learning (CoRL)*, 2018, pp. 734–743.

[41] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3803–3810.

[42] M. Kaspar, J. D. M. Osorio, and J. Bock, "Sim2real transfer for reinforcement learning without dynamics randomization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4383–4388.