A Trajectory Perspective on the Role of Data Sampling Techniques in Offline Reinforcement Learning

Jinyi Liu
College of Intelligence and Computing, Tianjin University
Tianjin, China
jyliu@tju.edu.cn

Yi Ma
College of Intelligence and Computing, Tianjin University
Tianjin, China
mayi@tju.edu.cn

Jianye Hao
College of Intelligence and Computing, Tianjin University
Tianjin, China
jianye.hao@tju.edu.cn

Yujing Hu
NetEase Fuxi AI Lab
Hangzhou, China
huyujing@corp.netease.com

Yan Zheng†
College of Intelligence and Computing, Tianjin University
Tianjin, China
yanzheng@tju.edu.cn

Tangjie Lv
NetEase Fuxi AI Lab
Hangzhou, China
hzlvtangjie@corp.netease.com

Changjie Fan
NetEase Fuxi AI Lab
Hangzhou, China
fanchangjie@corp.netease.com

ABSTRACT

In recent years, offline reinforcement learning (RL) algorithms have gained considerable attention. However, the role of data sampling techniques in offline RL has been somewhat overlooked, despite their potential to enhance online RL performance. Recent research in offline RL indicates that applying sampling techniques directly to state-transitions does not consistently improve performance. Therefore, to better leverage limited offline trajectory data, we investigate the impact of data sampling processes on offline RL algorithms from a trajectory perspective. In this paper, we introduce a memory technique, (Prioritized) Trajectory Replay (TR/PTR), to facilitate trajectory data storage and sampling. Building on TR, we delve into the potential of trajectory backward sampling, a method that has already proven effective in online RL, in the offline RL domain. Furthermore, to improve the sampling efficiency, we examine the influence of prioritized sampling based on various trajectory priority metrics on offline training. Integrating with existing algorithms, our findings demonstrate that data sampling and updates based on vanilla TR can contribute to more stable training. Also, our proposed 13 trajectory priority metrics for PTR exhibit outstanding performance on their respective applicable types of dataset, with the best-case scenario resulting in performance improvements exceeding 25%. These performance gains are achieved at a slight extra cost during the data sampling process, highlighting the significant advantages of trajectory-based data sampling for offline RL.

†Corresponding author: Yan Zheng (yanzheng@tju.edu.cn).
*This work is done when Jinyi Liu was intern in NetEase Fuxi AI Lab.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

KEYWORDS

Offline Reinforcement Learning; sampling techniques

ACM Reference Format:
Jinyi Liu, Yi Ma, Jianye Hao, Yujing Hu, Yan Zheng†, Tangjie Lv, and Changjie Fan. 2024. A Trajectory Perspective on the Role of Data Sampling Techniques in Offline Reinforcement Learning. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Reinforcement learning (RL) has made significant developments in recent years and has been widely applied in various scenarios [11, 29]. From a functional perspective, researchers typically study RL algorithms from three aspects: data collection, data sampling, and training algorithms. Data collection encompasses various methods for obtaining or generating diverse and comprehensive data, such as exploration [9, 23, 37] and data augmentation [15]. Data sampling refers to studying different sampling schemes for existing data, to improve the learning efficiency [16, 30]. Training algorithms are what most researchers focus on, seeking to optimize objectives through various techniques, including Q-value estimation [24], evolutionary policy [20, 21] and more.

To overcome the high cost of interacting with the environment in real scenarios, offline RL has received wide attention as a way to learn good policy based on the fixed dataset [17]. The study of offline RL can also be broadly categorized into the aforementioned three groups, but with slight differences. In offline RL, the existing literature primarily concentrates on devising training algorithms based on conservative value estimation [2, 13, 14], policy constraints [6, 7, 35], and other aspects [1, 3, 26]. In terms of data collection, pure offline training does not require exploration for new data, and research on data augmentation receives certain attention [22, 34]. Nevertheless, research on data sampling techniques under offline RL [5] is still at its early stages and lacks a unified

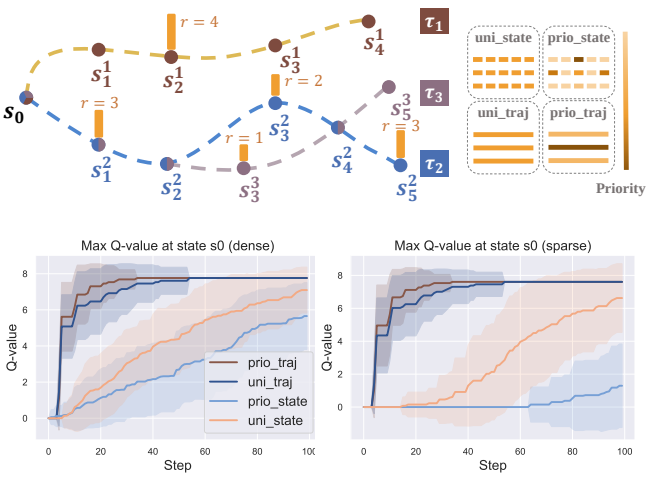

Figure 1: A motivating example. Top: the illustration of state transitions in three trajectories started at state s_0, and four different sampling techniques. Bottom: curves of the estimated maximum Q-value at s_0 learned on these trajectories. The solid line is the averaged value over 50 seeds, and the shaded area represents the standard deviation. The oracle value, capturing the discounted return following trajectory τ_2, is slightly less than 8.

and comprehensive conclusion. It is worth noting that experiences from online RL have revealed that data sampling techniques can indeed enhance performance [16, 30]. Therefore, this study primarily focuses on data sampling techniques to gain a more comprehensive understanding of their role in offline RL.

To elucidate the effect of data sampling techniques on limited data, we present an illustrative example in Figure 1. The results indicate that sampling state-transitions directly, whether being uniform (*uni_state*) or prioritized (*prio_state*), results in inferior learning of the Q-value at the initial state s_0. Even prioritized sampling (*prio_state*) offers little advantage over uniform sampling, which aligns with the findings in [5]. Such sampling could potentially overlook critical states (e.g., s_1) that act as connections between other states and s_0, consequently hindering the effective propagation of subsequent reward signals to s_0. In contrast, when sampling along trajectories (*uni_traj* and *prio_traj*), the propagation of reward signals to s_0 accelerates, converging faster to the optimal value, which is more prominent on sparse reward trajectories. Comparatively, *prio_traj* efficiently samples the trajectory with higher returns, further enhancing training efficiency. Our observation underscore the significance of considering the entire trajectory when determining the order of data sampling, which provides more informative insights into offline RL, in contrast to solely focusing on individual (s, a) pairs.

To fully discover the potential of data sampling techniques from a trajectory perspective for offline RL, we propose a plug-and-play memory, Prioritized Trajectory Replay (PTR), and conduct a detailed experimental analysis on these techniques, ultimately improving the performance of existing offline RL algorithms. We begin by implementing Trajectory Replay (TR), which serves as the fundamental memory for storing and sampling data in trajectories. Specifically, we sample and evaluate the data within a trajectory in a backward order. For instance, given two consecutive states, the latter one is sampled and evaluated first, followed by the former one. This allows each state to benefit from the estimated subsequent states, leading to improved convergence speed and performance especially on sparse rewards tasks.

Building on TR, we further explore the potential of data sampling from two distinct perspectives. Firstly, from the algorithm updating standpoint, inspired by [16], we modify the target Q computation method by balancing the original target Q-value and SARSA Q target value [32]. The SARSA Q-value enables us to restrict the update process to solely utilize trajectory data, thereby avoiding the sampling of out-of-distribution (OOD) actions and mitigating the extrapolation error. Secondly, from the trajectory sampling perspective, we propose Prioritized Trajectory Replay (PTR), wherein we consider various characteristics of trajectories (e.g., quality or uncertainty) and investigate 13 distinct trajectory priority definitions. Extensive evaluations lead us to the discovery that prioritizing the sampling of trajectories with higher upper quartile mean or mean value of rewards, as well as trajectories with lower uncertainty, effectively accelerates the learning of offline RL algorithms.

The primary contribution of this paper lies in a concrete analysis of several trajectory-based data sampling techniques in offline RL, which are integrated into a unified and plug-in memory structure - PTR. We emphasize that our work does not aim to propose a universally state-of-the-art (SOTA) algorithm. Instead, our study focuses on examining the advantage of basic trajectory sampling in TR, and exploring the role of various trajectory priorities in PTR. Our objective is to offer valuable insights for the development of robust data sampling techniques in offline RL in the future.

2 RELATED WORK

This section provides an overview of fundamental data sampling methods and offline RL algorithms, which inspire our considerations of trajectory-based priorities in offline RL.

Data sampling in online RL. Data sampling studies how to efficiently sample data to improve training. In RL, this can be traced back to [25], which improves planning efficiency by selecting the next updated state based on priorities. For model-free RL, PER [30] proposes to replay transitions with high temporal-difference (TD) error more frequently, which significantly improves the training efficiency of DQN [24]. Based on PER, several algorithms have been proposed, which optimize the error-based sampling [18, 28], or propose other priority metrics [5, 27]. These algorithms mostly focus on priority-based state-transition sampling. Other research explores data sampling methods from the perspective of trajectory, and the most typical ones are EBU [16] and TER [10], which propose backward sampling of the trajectory. Backward sampling enables the update process to more promptly and accurately utilize the reward signals of subsequent states, leading to accelerated learning.

Offline RL. In recent years, a large amount of research has focused on offline RL algorithms that learn promising policies from a fixed offline dataset. A key issue that offline RL algorithms need

to address is how to avoid the impact of estimation errors (extrapolation errors) of unseen data in the dataset on policy learning, since collecting new data is impossible [7, 17]. Common solutions can be divided into several categories: the first category imposes a policy constraint, which constrains the distribution of learned policies to be similar to the distribution of behavioral policies in the dataset [6, 13]; the second category applies support constraint, which require the state transition or action values of the learned policy to be restricted to the actions that appear in the dataset [12, 35]; the third category involves conservative Q-value [1, 2, 14, 19] or MDP model [36] estimation, which avoid selecting OOD actions during the policy update process by conservatively estimating unseen data.

In offline training, relatively less attention is given to data sampling techniques. Existing literature [5] evaluates several data sampling techniques from a traditional state-transition perspective based on TD3+BC, but consistent conclusions have not been reached regarding their effectiveness. Considering the return of trajectories, ReD [38] uses return-based data re-balance for to improve the sampling probability of transitions on trajectories with higher returns. Determining the sampling order based on the trajectory return can be regarded as a special case of approach we undertake. Under the perspective of trajectory, we investigate data sampling from two aspects: backward sampling on trajectories, and prioritized trajectory sampling based on trajectory quality or uncertainty.

3 PRELIMINARY

3.1 Offline Reinforcement Learning

Offline RL refers to RL conducted on offline data. In offline RL, we have collected several trajectories, called the dataset $\mathcal{D} = \{\tau_j\}_{j=1}^{N}$. Each trajectory is a sequence of state-action-reward with length l: $\tau = \{(s_i, a_i, r_i)\}_{i=0}^{l-1}$, where s_i represents the state at time i, a_i represents the action taken by the agent at that state, r_i represents the immediate reward, and s_{i+1} represents the next state. The optimization objective of offline reinforcement learning is usually to maximize the expected cumulative rewards through the dataset $\mathcal{D}$, without collecting new data.

3.2 Prioritized Experience Replay

Off-policy RL algorithms store historical experience in a replay memory. In traditional training, training data is uniformly sampled from the replay memory [24]. PER (Prioritized Experience Replay) [30] prioritizes sampling data from the memory based on the TD-error of state-transitions, leading to significant improvement in DQN training efficiency. Specifically, the distribution $P(j)$ used for sampling transition j is defined as follows:

$$P(j) = \frac{p_j^\alpha}{\sum_k p_k^\alpha}, s.t., p_k = |r_k + \gamma_k Q_{\text{target}}(s_{k+1}, \arg\max_a Q(s_{k+1}, a)) - Q(s_k, a_k)| + \epsilon \tag{1}$$

where p_j is the priority of transition j, α determines how much prioritization is used, γ is the discount factor, $Q(\cdot)$ and $Q_{\text{target}}(\cdot)$ represent the Q-value and target Q-value for state-action pair (s, a), and ϵ is a small constant that prevents priorities from being zero. In this way, PER can prioritize high-priority transitions based on the scale of TD-error, thereby accelerating the agent's learning process. In our research, we also propose to employ prioritized sampling during the training of offline RL algorithms. What distinguishes our proposal from PER is our use of trajectory perspective to define priority p, which leads to a better adaptation to offline learning. Additionally, we place emphasis on presenting a comparison about different sampling strategies, thus to ensure a fair analysis, we set the hyper-parameters α to 1.

4 TRAJECTORY REPLAY

In this section, we introduce Trajectory Replay (TR), a replay memory for storing offline data as complete trajectories and sampling batch data from a trajectory perspective. First, we describe the details of TR, including the technique of backward trajectory sampling used to sample from stored trajectories. We then introduce a weighted target computation based on TR, akin to EBU [16].

4.1 Trajectory Replay (TR)

To implement batch data sampling based on trajectory sequence, we maintain a basic replay memory storing the data in the form of trajectory, called Trajectory Replay, which differs from the traditional memory storing data as separate transitions. In EBU [16], it is suggested that sampling data in backward order on a trajectory is beneficial for online learning. Therefore, in this paper, we adopt backward sampling along this insight, as the default sampling order of TR. Such sampling based on TR is expected to offer certain advantages by making more timely and comprehensive use of information from successor state transitions.

Figure 2 illustrates the process of data sampling based on TR, in which $|N|$ trajectories in the offline dataset are stored. The available trajectories set stores all trajectories that have not been sampled currently. At the beginning, we sample $|\mathcal{B}|$ trajectories from it and add them to the sampled trajectories set. At each time step, the last state transition of $|\mathcal{B}|$ trajectories are moved to the data batch $\mathcal{B}$ for algorithm training. When all transitions of a trajectory in the sampled trajectories set have been moved to $\mathcal{B}$, a new trajectory is required to be sampled from the available trajectories set to fill up the sampled trajectories set.

In such vanilla TR-based data sampling, the update objectives of critic or policy model is not changed, and any RL algorithm is applicable. Given the data batch $\mathcal{B}$, a typical optimization objective following deterministic actor-critic algorithm [31] is as follows:

$$\mathcal{J}(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{B}} \left[\frac{1}{2} \left(Q_\theta(s_t, a_t) - Q_{\text{tg}}(s_t, a_t) \right)^2 \right], \quad Q_{\text{tg}}(s_t, a_t) = r_t + \gamma Q_{\bar{\theta}}(s_{t+1}, \pi(s_{t+1})), \tag{2}$$

where θ and $\bar{\theta}$ represent the parameters of critic network and target critic network, respectively, and $\pi(\cdot)$ represents the policy distribution.

Trajectory replay serves as the cornerstone for studying data sampling techniques in this study, providing the fundamental framework for data storage, sampling, and updates from a trajectory perspective. Subsequently, we delve deeper into two main aspects as shown in Figure 2: ❶ the training process based on TR in Section 4.2, and ❷ the process of sampling trajectories according to different trajectory priority metrics in Section 5.

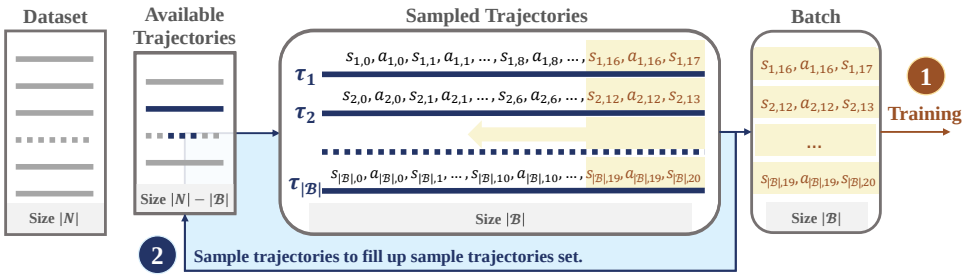

Figure 2: Overview of the process of data sampling based on Trajectory Replay. Offline data is stored and sampled as trajectories, and basically, we conduct a backward order for sampling trajectories to acquire the necessary batch data of each training round.

4.2 Weighted Backward Update based on TR

A key challenge in offline RL is the extrapolation error arising from Q-value estimation. Although TR implements a basic trajectory-based sampling process, it does not take into account alleviating the extrapolation error, which can limit offline RL performance. Our research shows that a SARSA-like target can be introduced based on TR's trajectory backward sampling process to completely avoid accessing OOD actions when calculating TD error during training. The critic target in Eq. 2 is now as follows:

$$Q_{\text{target}}(s_t, a_t) = r_t + \gamma Q_{\text{target}}(s_{t+1}, a_{t+1}), \tag{3}$$

where $Q_{\text{target}}(s_{t+1}, a_{t+1})$ is the target Q-value, calculated in the next time step in the same trajectory. Optimizing Eq. 3 can be considered as an implicit support constraint as OOD actions are avoided.

Based on it, we modify the computation of the target Q by balancing the original target in Eq. 2 and the vanilla SARSA target, to introduce the implicit support constraint in Eq. 3. We obtain the following weighted form of target estimation:

$$\begin{aligned} Q_{\text{target}}(s_t, a_t) = r_t + \gamma[&(1-\beta)Q_{\text{target}}(s_{t+1}, a_{t+1}) \\ &+ \beta Q_{\bar{\theta}}(s_{t+1}, \pi(s_{t+1}))], \end{aligned} \tag{4}$$

where β is the hyper-parameter that controls how much the original target should be used.

5 PRIORITIZED TRAJECTORY REPLAY

Building upon Trajectory Replay, we have developed a weighted target to improve learning efficiency. In this section, we shift our focus from the updating process to enhancing the trajectory sampling process depicted in Figure 2. To enhance the performance through more properly sampling trajectories, we introduce several novel **trajectory priority** metrics to prioritize trajectories during sampling. These metrics primarily consider two key factors: the quality of the trajectory return and the degree of trajectory uncertainty.

Instead of uniformly sampling from the available trajectories, we introduce a probabilistic sampling approach that incorporates trajectory priority metrics for trajectory τ_j, denoted as $\text{pri}(\tau_j)$, to prioritize trajectories. To avoid potential biases towards high priority trajectories, we use the ranking order, denoted as $\text{rank}(\text{pri}(\tau_j))$, as a measure to determine the sampling probability p_{τ_j}, rather than relying solely on the absolute value of $\text{pri}(\tau_j)$. Assigning appropriate priority to each trajectory through its rank ensures that all available trajectories are able to be utilized effectively. To this end, we define the prioritized sampling probability distribution $P(\tau_j)$ for all available trajectories as follows:

$$P(\tau_j) = \frac{p_{\tau_j}}{\sum_k p_{\tau_k}}, \text{ s.t., } p_{\tau_j} = \frac{1}{\text{rank}(\text{pri}(\tau_j))}. \tag{5}$$

5.1 Priority based on Trajectory Quality

Higher quality trajectories have been found to be more beneficial for offline RL [7]. To this end, we propose utilizing trajectory quality as a priority metric during the data sampling process to prioritize high-quality trajectories. Recognizing that using return as a metric may unfairly prioritize longer trajectories, we introduce alternative measures such as the mean of trajectory rewards, including upper quartile mean (UQM) and upper half mean (UHM). In this regard, we define six distinct metrics to assess trajectory quality, as follows:

- **Return**. The undiscounted sum rewards of the trajectory.
- **Avg. reward**. The averaged reward of the trajectory.
- **UQM reward**. The averaged reward of the top 25% state transitions in the trajectory.
- **UHM reward**. The averaged reward of the top 50% state transitions in the trajectory.
- **Min reward**. The minimum reward of the trajectory.
- **Max reward**. The maximum reward of the trajectory.

Trajectory priority $\text{pri}(\tau_j)$ is established based on these metrics to prioritize the sampling of high-quality trajectories. In particular, *Return* and *Avg. reward* reflect the overall performance of the trajectory, while *UQM reward*, *UHM reward*, and *Max reward* reflect the quality of the best transitions in the trajectory. On the other hand, *Min reward* prioritizes trajectories with higher minimum reward values, indicating a more stringent criterion for high quality.

5.2 Priority based on Trajectory Uncertainty

Within the context of offline RL, uncertainty serves as a pivotal metric by quantifying the level of disagreement present among Q estimates. Such disagreement indicates a lack of confidence in the estimated Q-value for a given state-action pair. Thus, a higher degree of uncertainty indicates greater unreliability in the knowledge of the state-action pair, making it unwise and infeasible to use it for policy execution. Taking a trajectory perspective, we define trajectory priority based on uncertainty, such that trajectories exhibiting lower degrees of uncertainty are assigned greater priority during sampling. Specifically, we establish trajectory priority

Table 1: Normalized average returns of TR and baselines on Gym Mujoco, Antmaze, Adroit tasks in D4RL. The averaged performance and standard deviation of multiple runs are reported.

Task Name	TD3+BC (Reproduced)	TD3+BC (TR)	EDAC (Reproduced)	EDAC (TR)	IQL (Reproduced)	IQL (TR)
halfcheeh-medium-v2	48.14±0.2	48.01±0.4	66.73±2.07	64.01±2.42	48.36±0.2	47.90±0.3
hopper-medium-v2	59.60±6.5	60.93±3.5	78.19±18.81	89.92±14.41	58.98±2.4	66.12±2.6
walker2d-medium-v2	83.70±2.5	83.87±1.2	71.60±31.38	87.16±2.28	78.35±5.4	79.78±1.8
halfcheetah-medium-replay-v2	44.51±0.7	43.92±0.6	64.31±1.39	57.79±4.26	43.45±0.7	43.47±0.3
hopper-medium-replay-v2	52.55±19.5	61.80±14.6	97.16±9.45	32.01±5.47	88.36±12.3	97.30±5.8
walker2d-medium-replay-v2	79.86±6.7	56.87±30.4	25.55±24.39	5.39±5.89	78.77±9.2	75.41±14.3
halfcheetah-medium-expert-v2	91.53±5.6	93.35±0.7	71.72±8.70	68.87±15.288	91.80±4.0	95.19±0.4
hopper-medium-expert-v2	94.57±9.8	104.45±11.6	28.44±7.211	28.65±10.82	86.86±24.6	84.37±28.4
walker2d-medium-expert-v2	110.57±0.7	110.16±0.4	114.00±1.02	112.64±1.05	112.11±0.7	110.43±0.3
Total	**665.03**	**663.36**	**617.71**	546.44	687.03	**699.97**
pen-cloned-v1	60.39±26.4	66.84±28.7	0.12±6.12	1.02±4.24	66.00±5.8	73.46±3.0
hammer-cloned-v1	0.52±0.2	0.42±0.1	0.21±0.01	0.22±0.37	3.40±0.2	1.84±1.1
door-cloned-v1	-0.11±0.0	-0.05±0.1	-0.35±0.13	-0.33±0.02	0.07±0.3	0.29±0.4
relocate-cloned-v1	-0.25±0.0	-0.23±0.0	-0.05±0.08	-0.17±0.10	-0.06±0.0	-0.06±0.0
pen-expert-v1	131.03±18.7	133.39±15.5	-1.07±1.52	6.36±8.84	136.51±4.8	138.24±4.8
hammer-expert-v1	128.62±0.4	128.63±0.3	0.25±0.15	0.39±0.54	127.87±0.2	127.61±0.5
door-expert-v1	106.10±0.3	106.26±0.4	3.63±6.33	16.27±28.25	105.53±0.2	105.56±0.4
relocate-expert-v1	105.74±3.8	103.34±5.8	-0.35±0.00	-0.35±0.00	105.46±1.0	103.66±0.7
Total	532.05	**538.60**	2.38	23.41	544.78	**550.60**
antmaze-umaze-v0	60.39±26.4	66.84±28.7	-	-	74.43±11.8	74.60±8.7
antmaze-umaze-diverse-v0	0.00±0.0	0.00±0.0	-	-	56.12±5.9	49.40±9.4
antmaze-medium-play-v0	35.50±34.6	46.00±39.9	-	-	69.40±7.4	62.60±11.3
antmaze-medium-diverse-v0	17.00±26.5	21.00±31.0	-	-	73.20±4.7	68.20±14.9
antmaze-large-play-v0	0.00±0.0	34.40±12.1	-	-	38.64±12.2	41.40±10.5
antmaze-large-diverse-v0	0.00±0.0	11.40±8.7	-	-	41.14±13.6	53.20±9.8
Total	83.90	**205.40**	-	-	**352.93**	349.40

metrics based on the mean, upper quartile mean (UQM), and lower quartile mean (LQM) of all state-action pairs' uncertainty values present within the trajectory, as follows:

- **Lower mean unc**. The reciprocal of the average uncertainty of all state-action pairs.
- **Lower LQM unc**. The reciprocal of the average uncertainty of bottom 25% state-action pairs.
- **Lower UQM unc**. The reciprocal of the average uncertainty of top 25% state-action pairs.

Alternatively, we can assign higher priority to trajectories with higher uncertainty, obtaining the following trajectory priority metrics by taking the reciprocals of the aforementioned values: *Higher mean unc.*, *Higher LQM unc.*, and *Higher UQM unc.*, respectively.

By incorporating trajectory priority, we upgrade TR to **Prioritized Trajectory Replay (PTR)**, achieving probabilistic sampling of trajectories. We anticipate that prioritizing trajectories with higher quality would be more suitable for sparse reward tasks, where the reward signals exhibit significant variations on these trajectories and thus require an emphasis on trajectories with higher value for learning during probabilistic sampling. Also, we expect that prioritizing trajectories with lower uncertainty can be effective in dense reward tasks, which can aid in accurately estimating the level of uncertainty. Conversely, prioritizing trajectories with higher uncertainty is not expected to work well, as errors from data with high uncertainty can accumulate severely for finite data and hinder learning. Section 6 provides a comprehensive and detailed empirical evaluation of these distinct trajectory priority metrics.

6 EXPERIMENT

To reveal the consistency between our analysis and the performance of TR/PTR, and demonstrate the advantage over baselines, we conduct experiments to address the following questions:

***RQ1* (Trajectory replay)**: Can vanilla trajectory sampling of TR bring an improvement to offline RL algorithms by efficiently using information from successor state transitions?

***RQ2* (Weighted target based on TR)**: Can modifying the critic target to SARSA-style target or weighted target bring further improvement to the performance?

***RQ3* (Prioritized trajectory replay)**: Without changing the critic target, can the prioritized trajectory sampling bring consistent performance improvement, and what are the characteristics of different trajectory priority metrics?

6.1 Baseline Algorithms and Implementation Details

Benchmark. In order to comprehensively analyze the advantages and disadvantages of various proposed data sampling mechanisms from a trajectory perspective, we conduct empirical evaluations on the D4RL benchmark [4], focusing primarily on the Mujoco `-v2` datasets (dense reward), Adroit `-v1` datasets (sparse reward), and AntMaze `-v0` datasets (sparse reward).

Baselines. Baseline algorithms include (i) TD3+BC [6], which uses regularization terms of behavior cloning to constrain the learning policy, (ii) IQL [12], which uses expectile regression to focus on

Table 2: Normalized average returns of SARSA-style and weighted target critic based on TR on Gym Mujoco, Antmaze, Adroit tasks in D4RL. The averaged performance and standard deviation of 5 runs are reported.

Task Name	TD3+BC (SARSA)	TD3+BC (weighted)	TD3+BC (Paper)	IQL (Paper)
halfcheeh-m	45.85±0.30	48.13±0.38	48.3	47.4
hopper-m	61.60±7.73	61.60±7.73	59.3	66.3
walker2d-m	84.63±4.89	84.63±4.89	83.7	78.3
halfcheetah-m-r	34.95±1.71	43.91±0.79	44.6	44.2
hopper-m-r	31.15±9.72	49.83±18.08	60.9	94.7
walker2d-m-r	46.45±25.02	**86.28±2.47**	81.8	73.9
halfcheetah-m-e	52.34±4.69	**93.63±1.04**	90.7	86.7
hopper-m-e	105.50±6.16	**105.50±6.16**	98.0	91.5
walker2d-m-e	106.80±3.10	**110.76±0.32**	110.1	109.6
Total	567.96	**684.26**	677.4	**692.4**
antmaze-u	71.17±3.63	**92.53±2.21**	78.6	87.5
antmaze-u-d	44.10±21.78	**58.45±7.85**	71.4	62.2
antmaze-m-p	0.00±0.00	**71.08±9.20**	10.6	71.2
antmaze-m-d	0.00±0.00	**72.33±6.81**	3.0	70.0
antmaze-l-p	0.00±0.00	**28.80±6.36**	0.2	39.6
antmaze-l-d	0.00±0.00	**18.87±29.75**	0.0	47.5
Total	19.21	**342.06**	193.8	**378.0**
pen-cloned	65.67±36.25	**75.75±34.63**	-	37.3
hammer-cloned	0.70±0.76	0.93±1.05	-	2.1
door-cloned	-0.15±0.02	-0.08±0.05	-	1.6
relocate-cloned	-0.16±0.05	-0.04±0.06	-	-0.2
pen-expert	134.61±15.15	**145.84±13.97**	-	-
hammer-expert	127.36±0.76	129.18±0.66	-	-
door-expert	105.69±1.92	105.69±1.92	-	-
relocate-expert	104.22±2.67	104.22±2.67	-	-
Total	537.92	**562.49**	-	-

in-sample actions and avoids querying the values of unseen actions, (iii) EDAC [1], which implements conservative estimation of critic based on ensemble Q networks.

The most critical hyper-parameters in TD3+BC is α used to control the weights of RL and imitation learning. On Mujoco tasks, we set α to 2.5 and the reproduced results are similar to those reported in the original paper. On the Antmaze and Adroit datasets, we conduct a hyperparameter search in the ranges 0.0001, 0.05, 0.25, 2.5, 25, 36, 50, 100, resulting in better results than previously reported. The most important hyper-parameters of EDAC algorithm are the n used to control the number of ensembles, and η, the weight of the ensemble gradient diversity term. To make it easier to verify the advantage, we set n to 3 for `mujoco-medium-replay-v2` datasets and 10 for others. We use the default hyper-parameters for IQL, as recorded in the original paper. All hyper-parameters mentioned above are listed in the Appendix.

Implementation details. To ensure code conciseness, readability, and a fair and identical experimental evaluation across algorithms, we reproduce the baseline algorithms based on the CORL repository [33] and implement various sampling techniques. Unless otherwise specified, all results reported in the following for the baseline algorithms are the results we have reproduced ourselves. For our methods, various data sampling techniques are implemented based on a plug-and-play memory module, TR/PTR, within about only 200 lines of code. Such module can be easily integrated with any offline RL algorithms by simply replacing the initialization and sampling processes of the original replay buffer with those of PTR. The only hyper-parameter β is used to construct the weighted critic target, and we conduct grid searches mainly using values from 0.05, 0.25, 0.5, 0.75, 0.95, 0.98 in the experiments. In basic TR and trajectory-based priority sampling, no hyper-parameters needs to be fine-tuned. The supplementary materials include our code.

6.2 Evaluation of Trajectory Replay

We first evaluate TR, the basic trajectory perspective-based data sampling technique with trajectory backward sampling. We present the reproduced results of the baseline algorithm and the performance improvements achieved by combining TR in Table 1.

For the majority of datasets, the vanilla trajectory sampling and the trajectory backward update processes based on TR do not lead to performance degradation. In fact, there are even performance gains observed, particularly notable in scenarios with sparse rewards such as the TD3+BC(TR) on the "Antmaze" dataset and the EDAC(TR) on the "Adroit" dataset. This supports the efficacy of trajectory backward sampling as an approach for efficiently utilizing reward information from subsequent state transitions, which is consistent with previous research on online scenarios.

However, for some dense reward datasets, the advantages of TR, specifically the enhanced reward propagation, are limited. On `walker2d-medium-replay-v2` and `hopper-medium-replay-v2`, there is even a noticeable decrease in performance. This highlights the urgent need for deeper investigate of data sampling techniques based on TR to improve offline RL algorithms' performance.

6.3 Evaluation of Backward Weighted Update

Following TR, we introduce two target critic forms, namely a vanilla SARSA-style target (Eq. 3) and a weighted target (Eq. 4). The former strictly constrains actions by utilizing all action information in the trajectory during the update process, while the latter balances the original target and vanilla SARSA-style target, thereby relaxing the action range constraint of the former. The corresponding weight values, denoted by β, are provided in the Appendix.

Results in Table 2 show that the vanilla SARSA-style target based on TR, exhibits stable performance on Adroit and simple Antmaze datasets. However, when applied to more complex tasks, the SARSA-style target exhibits instability. This instability primarily stems from the algorithm's overly simplistic and brute-force approach of avoiding out-of-distribution actions, which is inadequate for ensuring stable optimization in challenging tasks.

Compared to basic TR and SARSA-style target, the weighted target strategy significantly improves the performance and overcomes the issue of performance degradation in complex tasks. It achieves the best performance on nearly all datasets, and its results are comparable to, or even better than, those achieved by IQL, which is considered to be much better than TD3+BC.

Nevertheless, a notable limitation of the weighted target is its reliance on the tuning of the weight β. Thus, it is imperative to consider optimizing the target form or exploring alternative data sampling techniques based on TR.

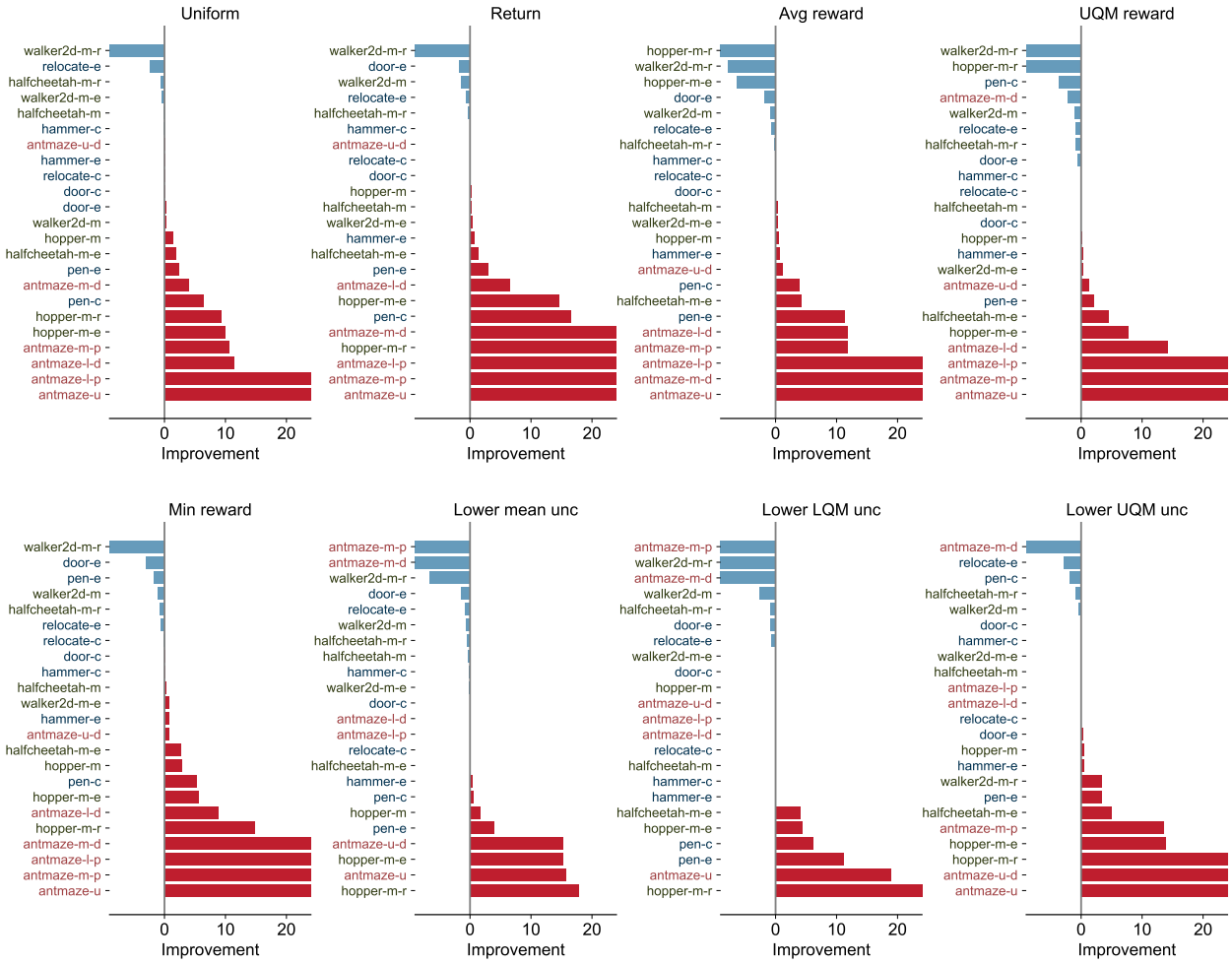

Figure 3: Comparison of the performance of PTR+TD3+BC compared to TD3+BC under different trajectory priority metrics. The results are averaged for 5 runs. The Antmaze and Adroit datasets are highlighted in red and deep blue, respectively. Without causing confusion, we also differentiate the datasets under three benchmarks by different colors (Mujoco, Adroit, Antmaze).

6.4 Evaluation of Prioritized Trajectory Sampling

In this section, we evaluate PTR, which optimizes the trajectory sampling process of TR. We implement two types of trajectory priority metrics based on trajectory quality and trajectory uncertainty. Table 3 presents the summary of performance for datasets on each of the three environments. Figure 3 illustrates the performance advantages of PTR under some excellent trajectory priority metrics on various datasets, compared to the baseline algorithm TD3+BC.

On sparse-reward datasets (Antmaze and Adroit), it is advisable to prioritize higher quality trajectories during sampling. Those sampling approaches mitigate the instability of TD3+BC training and improves performance compared to uniform sampling of TR. Among them, prioritizing trajectories with higher mean rewards (*Avg.* or *UQM reward*) is more beneficial and stable. Furthermore, prioritizing trajectories without low rewards (*Min reward*) on the trajectory has a significant impact on some datasets and is more stable overall than other trajectory quality metrics on TD3+BC.

On dense-reward datasets, prioritizing trajectories with lower uncertainty gains better or equivalent performance. This highlights the ability of uncertainty to capture the characteristics of dense reward trajectories. The metric *Lower UQM unc.*, which seeks trajectories with lower uncertainty, is the best performing metric in these cases. On the other hand, estimating uncertainty of data with sparse reward signals can be challenging, and thus, the performance of uncertainty-based metrics is limited on sparse-reward datasets.

Additionally, for tasks when there are significant differences in trajectories returns, such as `mujoco-medium-expert` datasets, prioritized sampling based on trajectory quality is also a good option, to prioritize the use of higher-quality data. However, these may harm performance on the `mujoco-replay` datasets.

Table 3: Overall performance comparison of various trajectory priority metrics based on TD3+BC and IQL. The reported results are the sum of the average performance over 5 runs.

	env.	baseline	TR	Return	Avg_r	UQM_r	UHM_r	Min_r	Max_r	L_mean	L_LQM	L_UQM	U_mean	U_LQM	U_UQM
TD3BC	Mujoco	665.03	663.36	643.72	630.66	623.35	633.63	677.83	603.02	**691.75**	**692.85**	**710.89**	542.22	541.25	540.16
	Antmaze	83.90	205.40	**254.33**	**254.73**	222.60	219.20	**256.93**	75.73	62.40	50.40	168.87	48.20	169.00	43.80
	Adroit	532.05	538.60	**549.89**	545.62	529.58	**557.20**	532.85	518.19	534.52	**547.93**	531.25	540.11	547.75	30.79
IQL	Mujoco	687.03	699.97	723.49	688.29	712.98	**718.61**	613.51	653.34	**724.93**	**719.04**	719.25	595.81	614.43	589.05
	Antmaze	352.93	349.40	362.60	375.60	**386.60**	**385.80**	338.40	299.80	220.20	202.20	260.20	277.60	336.00	240.80
	Adroit	544.78	**550.60**	**551.36**	**551.45**	548.42	**550.39**	544.79	**550.12**	549.59	546.88	548.23	549.05	549.21	541.34

Table 4: The comparison of various priority metrics

The dataset property	**TR** Uniform	**PTR** Quality	**PTR** Uncertainty
Sparse reward trajectories	✔	✔	✘
Dense reward trajectories	✘	-	✔
Many but not all high quality trajectories	-	✔	-

Table 5: Computational costs.

Sampling Method	Average Training Time(s/epoch)
TD3+BC	33.91
TR	35.24
PTR (quality)	34.88
PTR (uncertainty)	37.94

We also notice, that solely considering the maximum reward along trajectories *Max reward* is overly simplistic, and pursuing trajectories with higher uncertainty for prioritized sampling often performs poorly. This aligns with our consistent understanding of the role of uncertainty in offline training, namely, data with high uncertainty can be detrimental to offline training due to substantial extrapolation errors caused.

In summary, PTR can be seamlessly integrated with various algorithms at a minimal implementation cost, offering a plug-and-play solution that significantly enhances performance by simply replacing the data sampling process. Table 4 provides a concise summary of the most suitable datasets for different sampling techniques. Sparse reward data favors prioritized sampling based on trajectory quality, while dense reward data leans towards priority sampling of trajectories with lower uncertainty. Noting that no single metric dominates across all dataset types, emphasizing the importance of adopting proper sampling techniques to achieve high gains. Trajectory-based sampling, overall, yields substantial performance gains in offline RL training.

6.5 Computational Cost Comparison

We conduct experiments on a single machine with a NVIDIA GeForce 1080Ti 11GB GPU, and record the computational cost of different sampling techniques of PTR based on TD3+BC in Table 5. We record the average training time for each epoch (i.e., 1000 training steps).

The results indicate that trajectory-based sampling incurs a slightly higher computational cost, with an increase of 1 seconds per epoch compared to TD3+BC. This is primarily due to the maintenance of the available trajectories set during the sampling process. Besides, the uncertainty-based prioritized trajectory sampling requires less than 3 seconds more per epoch in contrast to quality-based sampling. This additional cost can be attributed to the continuous updating of the uncertainty value, which determines the sampling probability. These results imply that our PTR can extract more comprehensive information from limited data while maintaining a relatively low extra computational cost.

7 CONCLUSION AND LIMITATION

This paper investigates the effects of data sampling techniques, including trajectory-based backward sampling and priority-based trajectory sampling, on offline RL from the perspective of trajectories. We propose PTR to integrate these trajectory-based data sampling techniques as a plug-and-play replay memory module, which can be easily combined with any offline RL algorithm. Evaluation on D4RL datasets shows that trajectory-based backward sampling performs well, especially on sparse reward tasks. For sampling trajectories, using reward mean, UQM, minimum value, or the reciprocal of uncertainty mean as metrics to guide probability sampling generally produces better results. This study aims to contribute and inspire further research on offline RL algorithms from the perspective of data sampling techniques. We also hope to inspire more attention from various fields to data sampling techniques, such as evolutionary RL [8].

This work still has some limitations that need to be addressed in future research, such as the lack of theoretical discussion on the convergence of the proposed weighted target and the challenge to improve performance on extremely hard datasets like `door-cloned-v1`. Moreover, guiding the process of collecting offline data based on our findings on trajectory quality and uncertainty is of great importance in real industrial scenarios.

ACKNOWLEDGMENTS

This work is supported by the National Key R&D Program of China (Grant No. 2022ZD0116402), the National Natural Science Foundation of China (Grant Nos. 92370132) and the Xiaomi Young Talents Program of Xiaomi Foundation.

REFERENCES

[1] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. 2021. Uncertainty-Based Offline Reinforcement Learning with Diversified Q-Ensemble. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. 7436–7447.

[2] Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhi-Hong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. 2022. Pessimistic Bootstrapping for Uncertainty-Driven Offline Reinforcement Learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[3] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. 15084–15097.

[4] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *CoRR* abs/2004.07219 (2020).

[5] Yuwei Fu, Di Wu, and Benoit Boulet. 2022. Benchmarking Sample Selection Strategies for Batch Reinforcement Learning. https://openreview.net/forum?id=WxBFVNbDUT6

[6] Scott Fujimoto and Shixiang Shane Gu. 2021. A Minimalist Approach to Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. 20132–20145.

[7] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-Policy Deep Reinforcement Learning without Exploration. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 2052–2062.

[8] Jianye Hao, Pengyi Li, Hongyao Tang, Yan Zheng, Xian Fu, and Zhaopeng Meng. 2023. ERL-Re$ˆ2$: Efficient Evolutionary Reinforcement Learning with Shared State Representation and Individual Policy Representation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[9] Jianye Hao, Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Zhaopeng Meng, Peng Liu, and Zhen Wang. 2023. Exploration in Deep Reinforcement Learning: From Single-Agent to Multiagent Domain. *IEEE Transactions on Neural Networks and Learning Systems* (2023), 1–21. https://doi.org/10.1109/TNNLS.2023.3236361

[10] Zhang-Wei Hong, Tao Chen, Yen-Chen Lin, Joni Pajarinen, and Pulkit Agrawal. 2022. Topological Experience Replay. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

[11] B. Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Kumar Yogamani, and Patrick Pérez. 2022. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Trans. Intell. Transp. Syst.* 23, 6 (2022), 4909–4926.

[12] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

[13] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 11761–11771.

[14] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative Q-Learning for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[15] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. 2020. Reinforcement Learning with Augmented Data. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[16] Su Young Lee, Sung-Ik Choi, and Sae-Young Chung. 2019. Sample-Efficient Deep Reinforcement Learning via Episodic Backward Update. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 2110–2119.

[17] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *CoRR* abs/2005.01643 (2020).

[18] Ang A. Li, Zongqing Lu, and Chenglin Miao. 2021. Revisiting Prioritized Experience Replay: A Value Perspective. *CoRR* abs/2102.03261 (2021).

[19] Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. 2021. Dealing with the Unknown: Pessimistic Offline Reinforcement Learning. In *Conference on Robot Learning, 8-11 November 2021, London, UK (Proceedings of Machine Learning Research, Vol. 164)*. PMLR, 1455–1464.

[20] Pengyi Li, Jianye Hao, Hongyao Tang, Xian Fu, Yan Zheng, and Ke Tang. 2024. Bridging Evolutionary Algorithms and Reinforcement Learning: A Comprehensive Survey. *CoRR* abs/2401.11963 (2024).

[21] Pengyi Li, Jianye Hao, Hongyao Tang, Yan Zheng, and Xian Fu. 2023. RACE: Improve Multi-Agent Reinforcement Learning with Representation Asymmetry and Collaborative Evolution. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, 19490–19503.

[22] Shixi Lian, Yi Ma, Jinyi Liu, Yan Zheng, and Zhaopeng Meng. 2023. HIPODE: Enhancing Offline Reinforcement Learning with High-Quality Synthetic Data from a Policy-Decoupled Approach. *CoRR* abs/2306.06329 (2023).

[23] Jinyi Liu, Zhi Wang, Yan Zheng, Jianye Hao, Chenjia Bai, Junjie Ye, Zhen Wang, Haiyin Piao, and yang sun. 2024. OVD-Explorer: Optimism should not be the Sole Pursuit of Exploration in Noisy Environments. In *The 38th Association for the Advancement of Artificial Intelligence (AAAI): 2024*. AAAI Press.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nat.* 518, 7540 (2015), 529–533.

[25] Andrew W. Moore and Christopher G. Atkeson. 1993. Prioritized Sweeping: Reinforcement Learning With Less Data and Less Time. *Mach. Learn.* 13 (1993), 103–130.

[26] Fei Ni, Jianye Hao, Yao Mu, Yifu Yuan, Yan Zheng, Bin Wang, and Zhixuan Liang. 2023. MetaDiffuser: Diffusion Model as Conditional Planner for Offline Meta-RL. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, 26087–26105.

[27] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. 2018. Self-Imitation Learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 3875–3884.

[28] Yangchen Pan, Jincheng Mei, Amir-massoud Farahmand, Martha White, Hengshuai Yao, Mohsen Rohani, and Jun Luo. 2022. Understanding and mitigating the limitations of prioritized experience replay. In *Uncertainty in Artificial Intelligence, Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands (Proceedings of Machine Learning Research, Vol. 180)*. PMLR, 1561–1571.

[29] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. 2021. Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning. *IEEE Access* 9 (2021), 153171–153187.

[30] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[31] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. 2014. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014 (JMLR Workshop and Conference Proceedings, Vol. 32)*. JMLR.org, 387–395.

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

[33] Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. 2022. CORL: Research-oriented Deep Offline Reinforcement Learning Library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*.

[34] Kerong Wang, Hanye Zhao, Xufang Luo, Kan Ren, Weinan Zhang, and Dongsheng Li. 2022. Bootstrapped Transformer for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*.

[35] Chenjun Xiao, Han Wang, Yangchen Pan, Adam White, and Martha White. 2023. The In-Sample Softmax for Offline Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*.

[36] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. 2021. COMBO: Conservative Offline Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. 28954–28967.

[37] Yifu Yuan, Jianye Hao, Fei Ni, Yao Mu, Yan Zheng, Yujing Hu, Jinyi Liu, Yingfeng Chen, and Changjie Fan. 2023. EUCLID: Towards Efficient Unsupervised Reinforcement Learning with Multi-choice Dynamics Model. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[38] Yang Yue, Bingyi Kang, Xiao Ma, Zhongwen Xu, Gao Huang, and Shuicheng Yan. 2022. Boosting Offline Reinforcement Learning via Data Rebalancing. *CoRR* abs/2210.09241 (2022).