# Sample Efficient Offline-to-Online Reinforcement Learning

Siyuan Guo, Lixin Zou, Hechang Chen, Bohao Qu, Haotian Chi, Philip S. Yu, *Life Fellow, IEEE* and Yi Chang, *Senior Member, IEEE*

**Abstract**—Offline reinforcement learning (RL) makes it possible to train the agents entirely from a previously collected dataset. However, constrained by the quality of the offline dataset, offline RL agents typically have limited performance and cannot be directly deployed. Thus, it is desirable to further finetune the pretrained offline RL agents via online interactions with the environment. Existing offline-to-online RL algorithms suffer from the low sample efficiency issue, due to two inherent challenges, i.e., exploration limitation and distribution shift. To this end, we propose a sample-efficient offline-to-online RL algorithm via Optimistic Exploration and Meta Adaptation (OEMA). Specifically, we first propose an optimistic exploration strategy according to the principle of optimism in the face of uncertainty. This allows agents to sufficiently explore the environment in a stable manner. Moreover, we propose a meta learning based adaptation method, which can reduce the distribution shift and accelerate the offline-to-online adaptation process. We empirically demonstrate that OEMA improves the sample efficiency on D4RL benchmark. Besides, we provide in-depth analyses to verify the effectiveness of both optimistic exploration and meta adaptation. Our codes are available at https://github.com/guosyjlu/OEMA.

**Index Terms**—Offline-to-online reinforcement learning, Sample efficiency, Optimistic exploration, Meta learning.

## 1 Introduction

Offline reinforcement learning (RL), where agents learn from a previously collected dataset, has been widely studied due to its safety for online systems and achieved great success in some domains, such as robotic manipulation [1], recommender system [2], [3], and healthcare [4], [5]. However, constrained to the quality of offline dataset, the performance of offline RL agents is typically sub-optimal. Hence, it is generally beneficial and necessary to further finetune the offline pretrained RL agents via online interactions before the deployment. We refer such a process to offline-to-online RL, which is depicted in Fig. 1.

The pretraining-finetuning paradigm has become a standard pipeline in modern computer vision [6], [7] and natural language processing [8], [9]. However, directly borrowing their successful experience is non-trivial for offline-to-online RL since studies [10], [11] have shown that naive online finetuning methods cannot quickly adapt to the online RL settings, and may even result in performance degradation. Such failure can be attributed to the distribution shift, i.e., the discrepancy between the offline dataset and the online replay buffer. Particularly, the poor value estimates for the out-of-distribution (OOD) online samples will be further amplified when performing policy evaluation with bootstrapping, leading to severe extrapolation error. As a result, the policy improvement process fails to benefit from the online interactions. To address the distribution shift issue, existing offline-to-online RL methods can be divided into two branches. The first branch is to design a unified offline pretraining and online finetuning method. For example, advantage weighted actor critic (AWAC) [12] performs implicitly constrained policy updates to avoid OOD state-action pairs during both offline pretraining and online finetuning stages. However, this branch of work has limited applicable scenarios as it cannot be directly applied to agents pretrained with other offline RL methods. In contrast, the other branch is to design general online finetuning methods for arbitrarily given pretrained RL agents. For example,

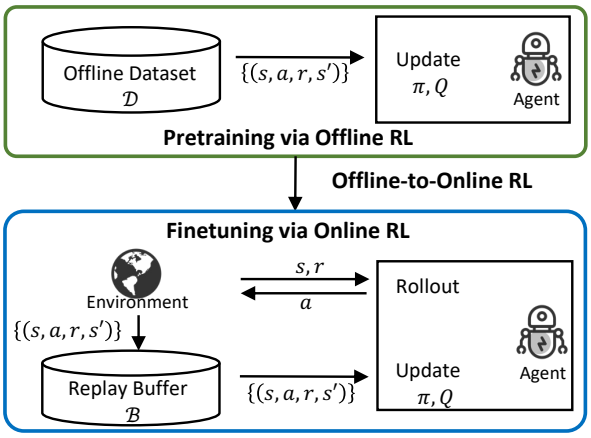


Fig. 1. Pictorial illustration of offline-to-online RL. We first pretrain agents via the offline RL algorithm and then finetune them via the online RL algorithm. Both algorithms are typically built on off-policy RL algorithms.


- *Siyuan Guo, Bohao Qu, Haotian Chi, Hechang Chen and Yi Chang are with the School of Artificial Intelligence, Jilin University, China and Engineering Research Center of Knowledge-Driven Human-Machine Intelligence, MOE, China. Siyuan Guo and Yi Chang are also with the International Center of Future Science, Jilin University, China. E-mail: guosyjlu@gmail.com, qubohao@126.com, chiht19@gmail.com, chenhc@jlu.edu.cn, yichang@jlu.edu.cn.*
- *Lixin Zou is with the School of Cyber Science and Engineering, Wuhan University, China. E-mail: zoulixin15@gmail.com.*
- *Philip S. Yu is with the Department of Computer Science, University of Illinois at Chicago, USA. Email: psyu@uic.edu.*
- *Hechang Chen and Yi Chang are the corresponding authors.*

Zhao et al. [13] propose to apply adaptive behavior cloning regularization (ABCR) loss term during online finetuning, which explicitly constrains the policy to avoid performing OOD actions.

Despite their success in offline-to-online RL, both constraints on policy prevent agents from fully utilizing the online interactions, resulting in low sample efficiency and limited performance. To derive a sample-efficient offline-to-online RL method, there are two main challenges remaining unsolved: (1) *Exploration limitation*: Offline RL typically applies heavy constraints to off-policy RL algorithms to avoid OOD state-action pairs, such as the conservatism mechanism in CQL [14] and policy regularization in TD3+BC [15]. As the behavior policy for exploration is typically derived by the target policy, such constrained pretrained policies tend to perform conservative actions, which makes the behavior policy unable to seek novel state and actions that might yield high rewards and lead to long-term gains [16]. Hence, it calls for a better solution on how to derive a behavior policy that can sufficiently explore the environment to accelerate the finetuning process. (2) *Distribution shift*: Furthermore, the distribution shift issue makes it difficult for offline pretrained agents to quickly adapt to the online finetuning setting, which causes low sample efficiency. Worse still, when we employ more exploratory behavior policy to interact with the environment, this issue may be further amplified. Thus, sample efficient offline-to-online adaptation deserves to investigate.

To address the aforementioned challenges, we propose a sample-efficient offline-to-online RL algorithm via **O**ptimistic **E**xploration and **M**eta **A**daptation (OEMA). Our contributions are summarized as follows:

1) *We propose an optimistic exploration strategy according to the principle of optimism in the face of uncertainty to resolve the exploration limitation.*

Specifically, we propose to solve the optimization problem derived by the principle of optimism in the face of uncertainty [17], [18], [19] in an iterative manner. To keep the on-policyness of the behavior policy, we compose the behavior policy by the target policy, and a perturbation model to decide the exploration direction. The perturbation model is trained to maximize an approximate upper bound of the value function, which models the epistemic uncertainty [20]. In this way, we can derive a more exploratory behavior policy in a stable manner and achieve better sample efficiency.

2) *A meta learning based adaptation method is proposed to accelerate the reduction of the distribution shift between offline and online data.*

In particular, we keep two replay buffers, an overall replay buffer to save both offline and online transitions, and a recent replay buffer to save recent online transitions. We employ the policy improvement based on the overall replay buffer. Meanwhile, we introduce an auxiliary meta objective [21], [22] to ensure that the direction taken to achieve this also leads to policy improvement in the recent replay buffer. As such, correction in the value estimate of recent states can be timely delivered in the policy, which reduces the distribution shift and accelerates the offline-to-online adaptation process.

3) *Experimental results on D4RL benchmark demonstrate the superiority of OEMA.*

We conduct extensive experiments on D4RL [23] benchmark. Experimental results demonstrate that OEMA outperforms state-of-the-art offline-to-online RL algorithms in terms of sample efficiency. Besides, we provide a series of in-depth analyses, including ablation study, hyper-parameter analysis and controlled experiments, to verify the effectiveness of both components in OEMA.

The remainder of this paper is organized as follows. Section 2 elaborates the preliminaries including RL, Actor-critic, offline RL and offline-to-online RL. Section 3 details the proposed method OEMA. Section 4 presents the experiment results to demonstrate the superiority of OEMA. In Section 5, we introduce the related works about offline-to-online RL, optimistic exploration in RL, and meta learning for RL. Section 6 draws conclusions and suggests future work.

## 2 PRELIMINARIES

In this section, we elaborate the preliminaries, including RL, Actor-Critic, offline RL, and offline-to-online RL.

### 2.1 RL

We consider the standard RL setup, where an agent interacts with the environment to maximize the cumulative rewards. Formally, the RL problem is formulated as a Markov decision process (MDP), defined by a tuple $(S; A; R; p; \ )$ [24]. At each timestep $t$, the agent observes a state $s_t \ 2 \ S$, and performs an action $a_t \ 2 \ A$ based on the policy $\ $. Then, the environment rewards the agent with $r_t = R(s_t; a_t)$ and transitions to the next state $s_{t+1}$ according to the transition probability $p(s_{t+1} j s_t; a_t)$. As such, we can obtain the transition $(s_t; a_t; r_t; s_{t+1})$ at timestep $t$. The agent's objective is to maximize the expected discounted return $\mathbb{E}\left[\sum_{t=0}^{1} \ ^t r_t\right]$, where $\ 2 \ [0; 1)$ is the discounted factor to balance the importance of the current rewards and the future rewards. We measure this objective by a value function

$$Q \ (s; a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \ ^t r_t j s_0 = s; a_0 = a \ \right]; \qquad (1)$$

which measures the expected discounted return after taking the action $a$ in state $s$. The classic online RL can be divided into two categories, namely, on-policy RL and off-policy RL. On-policy RL algorithms update the agent with experience from its current rollouts, while off-policy RL algorithms can update the agent according to the replay buffer that stores the experience collected at any point during training.

### 2.2 Actor-Critic

We mainly consider Actor-Critic based off-policy RL algorithms. Among them, TD3 [25] is a representative algorithm that learns a deterministic policy $\ (s)$ parameterized by $\ $, and two Q-functions $Q_{\ 1}(s; a); Q_{\ 2}(s; a)$ parameterized by $\ _1$ and $\ _2$. Specifically, given the replay buffer $B$, the critic is updated based on the clipped double Q-learning algorithm

$$L(\ _i) = \mathbb{E}_{(s;a;r;s^0) \ B}\left[\frac{1}{2}(Q_{\ i}(s; a) \quad y)^2\right]; \qquad (2)$$

where the temporal difference (TD) target $y$ is defined as

$$y = r + \min_{i=1;2} Q_{\ i}(s^0; \quad (s^0)); \qquad (3)$$

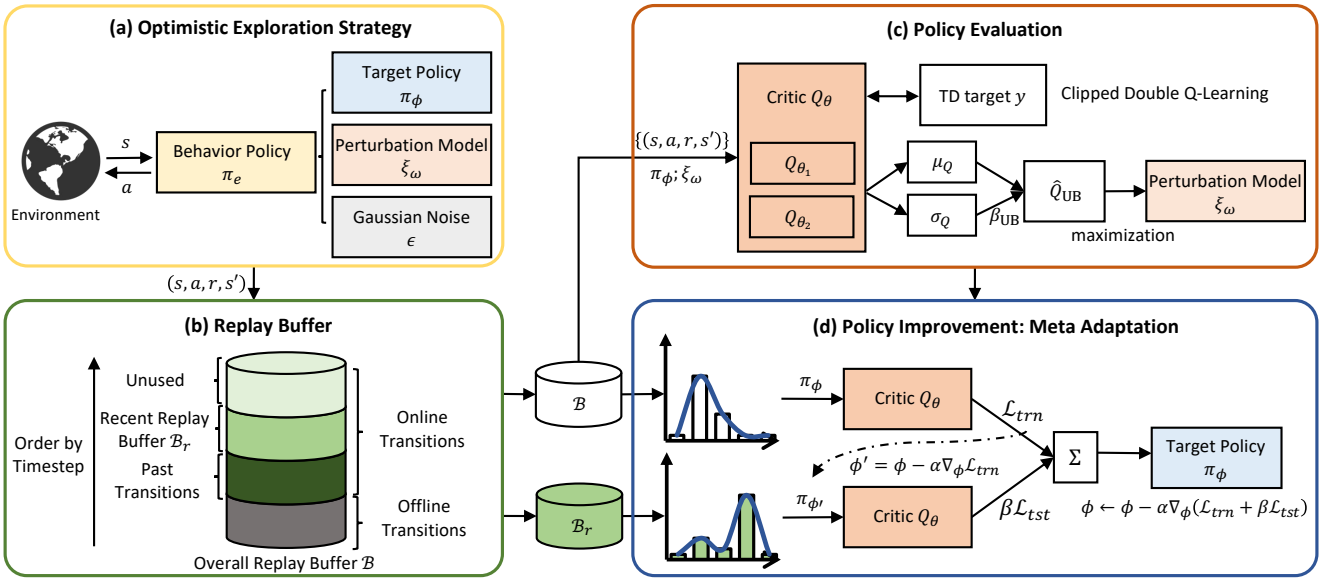


Fig. 2. The overall framework of OEMA, which contains (a) Optimistic Exploration Strategy, (b) Replay Buffer, (c) Policy Evaluation, and (d) Meta Adaptation based Policy Improvement. As shown in (a), we compose the behavior policy by target policy, perturbation model and Gaussian noise to interact with the environment. After that, we store the transitions in the replay buffer (b). Then in (c), we perform clipped double Q-learning based policy evaluation and optimization of the perturbation model. Finally, we perform the meta adaptation based policy improvement as shown in (d).

and the actor is learned to maximize the current Q-function,

$$L(\ ) = \quad \mathrm{E}_{s\ B}[Q_{\ 1}(s;\quad (s))]: \tag{4}$$

As such, TD3 performs constant policy iterations that alternate between policy evaluation and policy improvement.

### 2.3 Offline RL

Modern offline RL algorithms are typically built on the off-policy RL algorithm [14], [15], [26], [27]. Offline RL refers to the scenario where the agent cannot interact with the environment but is given a fixed-size dataset $D = f(s; a; r; s^0)g$, which is beneficial for many cost-sensitive tasks, such as robotic manipulation [23], recommender system [2], [3] and healthcare [4], [5]. However, it is problematic to directly utilize off-policy RL algorithms for offline RL due to the extrapolation error, which refers to the error in the estimated values for OOD state-action pairs. Worse still, since the TD learning for policy evaluation involves a bootstrapping term $Q(s^0;\quad (s^0))$, such extrapolation errors will be further amplified via bootstrapping, resulting in extremely unrealistic policy evaluation, and in turn, unreliable policy improvement.

Existing offline RL algorithms solve this problem by applying various constraints or regularization to avoid performing OOD actions. For example, TD3+BC [15] constrains the policy improvement of TD3 with a behavior cloning term that forces the policy to choose actions close to the offline dataset $D$. The policy improvement process of TD3+BC can be formulated by

$$L(\ ) = \quad \mathrm{E}_{(s;a)2D}\ \overset{\mathrm{h}}{Q}_{\ 1}(s;\quad (s))\quad {}_{\mathrm{BC}}(\quad (s)\quad a)^2\ \overset{\mathrm{i}}{;} \tag{5}$$

where ${}_{\mathrm{BC}}$ is a hyper-parameter for balancing both losses.

### 2.4 Offline-to-Online RL

The performance of offline pretrained RL agents is constrained to the quality of the offline dataset. Thus, it is desirable to perform further online finetuning before deployment. Formally, offline-to-online RL aims at studying the setting where the agent , typically along with the Q function $Q$ , is first pretrained with the offline dataset $D$ via a given offline RL algorithm and then finetuned via online interactions with the environment. The online finetuning process is based on off-policy RL algorithms to bring the training scheme in correspondence with that during offline pretraining. In addition, the online replay buffer $B$ is typically initialized with the offline dataset $D$ to enable effective experience reusage.

## 3 Methodology

In this section, we first introduce the framework of OEMA. Then, we elaborate the two parts of the proposed offline-to-online RL algorithm OEMA, namely, optimistic exploration and meta adaptation. We give a detailed algorithm description in the end.

### 3.1 The Framework of OEMA

Given agents pretrained by offline RL algorithms such as TD3+BC [15], OEMA aims at further finetuning the policy and Q function $Q$ via online interactions with better sample efficiency. With TD3 [25] as the backbone during the online finetuning process, the overall framework of OEMA is presented in Fig. 2. In each environment step, the agent first interacts with the environment via the proposed optimistic exploration strategy as shown in Fig. 2 (a). Then in Fig. 2 (b), the transition is stored in the overall replay buffer with the timestep order. Finally in Fig. 2 (c) and (d), we perform one step of policy iteration, including clipped

double Q-learning based policy evaluation, optimization of the perturbation model based on the principle of optimism in the face of uncertainty, and meta adaptation based policy improvement.

In the following subsections, we introduce the two essential components in detail, i.e., optimistic exploration strategy and meta adaptation method, which tackle the two inherent challenges for sample ef ciency, i.e., exploration limitation and distribution shift, respectively.

### 3.2 Optimistic Exploration with Limited Budget

As mentioned in Section 2.3, of ine RL algorithms typically apply various constraints or regularization terms to advanced off-policy RL algorithms to eliminate effects of the extrapolation error [28]. Given the target policy (s), TD3 takes $_e(s) =$ (s) + as the behavior policy, where N (0; ) is the Gaussian policy noise and is a small value for controlling the exploration noise. Since is trained to perform conservative actions close to the of ine dataset during of ine pretraining, the behavior policy $_e$ fails to seek novel states and actions that might yield high rewards and lead to long-term gains.

To solve this problem, we propose to design an exploratory behavior policy based on the principle of optimism in the face of uncertainty [17], [18], [19]. Formally, the derivation of the behavior policy can be de ned as

$$\begin{aligned} &_e = \arg\max \hat{Q}_{UB}(s; \ (s)); \\ &\text{s:t:} \ \ \frac{1}{2}k \ \ (s) \quad (s)k \quad ; \end{aligned} \tag{6}$$

where $\hat{Q}_{UB}$ is an approximate upper con dence bound of $Q_1$ and $Q_2$ to model the epistemic uncertainty, and controls the deviation between the behavior policy and the target policy. As such, the overall objective of the behavior policy is to increase the chance of choosing informative actions with high epistemic uncertainty, while constraining the deviation between behavior policy and target policy.

To begin with, we focus on obtaining the approximate upper con dence bound $\hat{Q}_{UB}$. Following previous works [20], [29], we utilize a Gaussian distribution to model the epistemic uncertainty. The mean belief is de ned by $_Q(s;a) = \frac{1}{2}(Q_1(s;a) + Q_2(s;a))$, while the standard deviation is derived by

$$\begin{aligned} _Q(s;a) &= \sqrt{\sum_{i=1;2} \frac{1}{2}(Q_i(s;a) \quad _Q(s;a))^2} \\ &= \frac{1}{2} Q_1(s;a) \quad Q_2(s;a) : \end{aligned} \tag{7}$$

As such, we can de ne the approximate upper bound as

$$\hat{Q}_{UB}(s;a) = \ _Q(s;a) + \ _{UB} \ _Q(s;a); \tag{8}$$

where $_{UB}$ 2 $R^+$ controls the optimism level. Note that when $_{UB} =$ 1, we have

$$\begin{aligned} \hat{Q}_{UB}(s;a) \ _{UB=\ 1} &= \ _Q(s;a) \quad _Q(s;a) \\ &= \frac{1}{2}(Q_1(s;a) + Q_2(s;a)) \quad \frac{1}{2}|Q_1(s;a) \quad Q_2(s;a)| \\ &= \min(Q_1(s;a); Q_2(s;a)); \end{aligned} \tag{9}$$

which is precisely part of the TD target used by TD3 [25] and SAC [30] to utilize the approximate lower con dence bound for avoiding overestimation bias. Similarly, when $_{UB}$ = 1, we can obtain

$$\hat{Q}_{UB}(s;a) \ _{UB=1} = \max(Q_1(s;a); Q_2(s;a)); \tag{10}$$

which shows that the approximate upper con dence bound with $_{UB}$ = 1 is equivalent to the maximum of the two values. As such, a higher value of $_{UB}$ indicates more optimism on the epistemic uncertainty.

Then, we focus on solving the optimization problem in Eq. 6. Optimistic actor critic (OAC) [20] derives an approximate closed form solution of the behavior policy $^{OAC}_e$, which is in the form of

$$^{OAC}_e(s) = \ (s) + \frac{p}{2} \frac{r_a \hat{Q}_{UB}(s;a)_{a= \ (s)}}{r_a \hat{Q}_{UB}(s;a)_{a= \ (s)}}: \tag{11}$$

However, a recent study [29] points out that OAC suffers from the instability issue in practice since its derivation involves approximation and has a gap with the exact solution, which results in less ef cient exploration. To eliminate this issue, we propose to solve the optimization in an iterative manner to derive an empirically feasible behavior policy. A naive solution is to convert the constrained optimization problem in Eq. 6 into an unconstrained one by adding a behavior cloning penalty term, which can be formalized as

$$^{naive}_e(s) = \arg\max \hat{Q}_{UB}(s; \ (s)) \quad k \ (s) \quad (s)k; \tag{12}$$

where controls the penalty for the deviation between the behavior policy and target policy. However, since the target policy is constantly updated via policy improvement, such a behavior cloning based penalty term fails to close the gap between the behavior policy and target policy, which further violates the on-policyness of the behavior policy de ned by the constraint in Eq. 6. As a result, the naive solution performs poorly in practice.

In order to keep the on-policyness of the behavior policy while remaining optimistic in the face of uncertainty, we propose a perturbation based optimistic exploration strategy. As depicted in Fig. 2 (a), the behavior policy $_e$ is composed of three parts: target policy , perturbation model $_!$, and Gaussian policy noise N (0; ). Based on the state and the action taken by the target policy, the perturbation model outputs a small adjustment to the target policy as the exploration direction. Thus, the behavior policy can be formulated as

$$_e(s) = \ (s) + \ _!(s; \ (s)) + \ ; \tag{13}$$

where the perturbation model $_!$ is trained for maximization of the approximate upper con dence bound in Eq. 8 to be optimistic in the face of the epistemic uncertainty. The loss function of the perturbation model is

$$L(!) = \ E_{s \ B}\left[\hat{Q}_{UB}(s; \ _e(s))\right]: \tag{14}$$

In summary, we solve the optimization problem in an iterative manner to avoid unnecessary approximation in the closed form solution of OAC. Since we utilize the perturbation model $_!$ to decide the exploration direction beyond the

target policy , our behavior policy $_e$ is naturally close to the target policy , which satis es the constraints in Eq. 6. Hence, we empirically achieve more stable exploration than OAC. In this way, the proposed perturbation based optimistic exploration strategy enables the agents to suf ciently explore the environment while avoiding the instability issue of OAC, which further improves the sample ef ciency of of ine-to-online RL.

### 3.3 Meta Adaptation for Distribution Shift Reduction

Distribution shift, i.e., the discrepancy between the of ine dataset and the online replay buffer, is the most important issue in of ine-to-online RL that leads to low sample ef ciency [10], [11], [13]. Worse still, since we employ a more exploratory behaviour policy based on the optimistic exploration strategy, such distribution shift is further ampli ed. To accelerate the reduction of distribution shift, we propose a meta learning [21], [22] based adaptation method.

Speci cally, during the online netuning process, we keep two replay buffers, an overall replay buffe $B$ to save all the of ine and online transitions, and a recent replay buffer $B_r$ to save recent online transitions. As such, there exists obvious data distribution shift between $B$ and $B_r$ due to the constant policy improvement during online netuning. To improve the sample ef ciency, we expect that of ine pretrained agents can quickly adapt to the online settings, i.e., policy improvement based on the overall replay buffer $B$ can also bring policy improvement in terms of the recent replay buffer $B_r$. This can be achieved by introducing the following meta learning based adaptation method.

#### 3.3.1 Meta-train

We rst employ policy improvement based on the overall replay buffer $B$ with

$$L_{trn}( ) = \quad E_{s B}[Q_{1}(s; \quad (s))] : \tag{15}$$

Based on this, we can perform one gradient update step via stochastic gradient descent (SGD) [31] as

$$^{0} = \quad r \quad L_{trn}( ); \tag{16}$$

where is the learning rate of the policy parameters .

#### 3.3.2 Meta-test

Then, we simulate the testing process in the online settings via recent transitions in $B_r$, which enables the policy to learn to adapt to the online environment. The meta-test process can be formulated by

$$L_{tst}( ^{0}) = \quad E_{s B_r}[Q_{1}(s; \quad _{0}(s))] ; \tag{17}$$

where the loss function is calculated using the updated parameter $^{0}$ from meta-train. Note that the optimization process based on the meta-test loss function above involves the second derivative with respect to .

#### 3.3.3 Meta-optimization

The meta-train and meta-test loss functions above can be simultaneously optimized by the following meta-optimization objective

$$= \arg\min L_{trn}( ) + \quad L_{tst}( \quad r \quad L_{trn}( )); \tag{18}$$

**Algorithm 1** Sample Ef cient Of ine-to-online RL via Optimistic Exploration and Meta Adaptation

Require: Pretrained policy , Q function $Q_{1}$, $Q_{2}$, and target network $\hat{}$; $Q\hat{}_{1}$; $Q\hat{}_{2}$; of ine dataset $D$
1: Initialize the overall replay buffer $B$ with $D$.
2: Initialize the recent replay buffer $B_r$.
3: Initialize the perturbation model $_{!}$.
4: for each iteration do
5: Select action with perturbation and exploration noise $a = \quad (s) + \quad _{!}(s; \quad (s)) + \quad ; \quad N(0; \quad )$ and observe reward $r$ and next state $s^{0}$.
6: Store transition tuple $(s; a; r; s^{0})$ in $B$ and $B_r$.
7: Update $Q_{i}$ $(i = 1; 2)$ with Eq. 2 based on $B$.
8: Update $_{!}$ with Eq. 14 based on $B$.
9: Meta-train:
10: Calculate meta-train loss with Eq. 15 based on $B$.
11: Update parameter $^{0} = \quad r \quad L_{trn}( )$.
12: Meta-test:
13: Calculate meta-test loss with Eq. 17 based on $B_r$.
14: Meta-optimization:
15: Update with Eq. 19.
16: Update the target networks
$\hat{}_{i} = \quad _{i} + (1 \quad )\hat{}_{i}$, $i = 1; 2$,
$\hat{} = \quad + (1 \quad )\hat{}$.
17: end for

where is the meta weight to balance the meta-train and meta-test loss. Hence, the overall optimization process for policy parameter can be formulated as

$$\frac{@(L_{trn}( )) + \quad L_{tst}( \quad r \quad L_{trn}( ))}{@} : \tag{19}$$

Here, we use the same learning rate for both meta-train and meta optimization to avoid extra hyperparameter.

In summary, as shown in Fig. 2 (d), we can divide the meta learning process above into two tasks. Meta-train learns to improve the policy via the overall replay buffer $B$ containing transitions from both of ine dataset and the online replay buffer. And the meta-test further updates the policy such that after the policy improvement in meta-train, the policy can also achieve better performance in the online settings $B_r$. In this way, correction in the value estimate of recent states can be timely delivered in the policy, which eliminates the distribution shift and accelerates the of ine-to-online adaptation, achieving better sample ef ciency.

Since off-policy RL algorithms involve policy evaluation and policy improvement, it is intuitive to expect that the proposed meta adaptation method can be similarly applied to the policy evaluation process for distribution shift reduction. However, the policy evaluation relies on bootstrapping based TD learning. Hence, the meta adaptation method for policy evaluation makes the estimation of the TD target biased, leading to poor performance and even non-convergence in practice. In contrast, the meta adaptation for policy improvement can automatically correct the potential bias through feedbacks from online interactions. Thus, the proposed meta adaptation method is built on the policy improvement process instead of the policy evaluation process.

Fig. 3. D4RL tasks, halfcheetah, hopper, walker2d, and maze2d with u-shape, medium, large maze.

### 3.4 The Overall Algorithm

We summarize the overall algorithm of OEMA in Algorithm 1. Note that there are two replay buffers in OEMA. The first is the overall replay buffer B to save both offline and online transitions. A naive solution is to use the vanilla replay buffer that maintains all the transitions from the offline dataset and constantly stores online transitions. However, as the quality of the offline dataset is typically limited, such a replay buffer may result in potential bias and low sample efficiency. To solve this problem, we follow the previous successful experience of balanced replay buffer [11] and down-sampling based replay buffer [13]. We provide a comprehensive comparison of these three replay buffers in Section 4.5. And we utilize the balanced replay buffer in practice. The second is the recent replay buffer $B_r$ to save recent online transitions. With these two replay buffers, we can utilize the meta adaptation method to accelerate the offline-to-online adaptation process.

To decrease the unnecessary space complexity, we implement both replay buffers as depicted in Fig. 2 (b). Specifically, we store all the transitions in the overall replay buffer with timestep order, namely, offline transitions, past online transitions and recent online transitions in order. As such, we only need to store the recent online transitions once and thus, avoid introducing a physical recent replay buffer.

## 4 Experiments

In this section, we conduct extensive experiments to demonstrate the sample efficiency of the proposed offline-to-online RL algorithm OEMA. Besides, we give in-depth analyses on optimistic exploration and meta adaptation. In the end, we investigate the impact of different replay buffers on the performance of OEMA.

### 4.1 Experiment Settings

#### 4.1.1 Environment and Dataset

We evaluate the sample efficiency of offline-to-online RL algorithms on D4RL [23] benchmark, which provides various continuous-control tasks and datasets. Among them, we mainly focus on three MuJoCo [32] locomotion tasks, i.e., halfcheetah, hopper and walker2d, which are widely used by RL community. Besides, we also include Maze2D tasks, with u-shape, medium and large maze. Fig. 3 illustrates the six experimental environments. In terms of MuJoCo tasks, D4RL provides five offline datasets with different quality for each task: random, medium, medium-replay, medium-expert and expert. Specifically, the random dataset is collected by a randomly initialized policy; the medium dataset is collected by an early-stopped soft actor-critic (SAC) [30] agent with medium-level performance; the medium-replay dataset consists of all the transitions in the replay buffer after training the aforementioned medium-level SAC agent; the medium-expert dataset consists of mixing equal amounts of expert demonstrations and sub-optimal data; the expert dataset is collected by a fully trained SAC agent. Among them, offline RL agents trained with the expert dataset already achieve expert-level performance, and there is no need for further finetuning. Thus, we follow previous works [10], [11], [13] to omit the evaluation on the expert dataset. In terms of Maze2D tasks, we consider two settings of sparse and dense rewards. We use D4RL datasets of v0 version for MuJoCo tasks, and v1 version for Maze2D tasks.

#### 4.1.2 Evaluation Protocols

We take the standard TD3+BC to pretrain the offline RL agent for all the 18 settings with one million gradient steps. Note that we follow the previous work [13] to omit the state normalization during offline pretraining for simplicity. After offline pretraining, we finetune the agents via 300,000 environment steps for MuJoCo tasks, and 100,000 environment steps for Maze2D tasks. We evaluate the agents every 5000 timesteps and each evaluation consists of 10 episodes. We take the best return over all the evaluations, and the average return over last 10 evaluations to evaluate the sample efficiency of different offline-to-online RL algorithms. Besides, we include average D4RL score (a normalized score with 0 to represent random performance and 100 to represent expert-level performance) improvement over the offline performance for clearer comparison. Note that we follow previous works [11], [13] to evaluate the sample efficiency based on performance within a fixed number of environment steps.

#### 4.1.3 Baselines

We compare our OEMA algorithm with the following four baselines.

> TD3+BC-ft . It performs further TD3+BC finetuning steps with additional online interactions.
> TD3-ft . It finetunes the offline agent based on the standard TD3 [25] algorithm.
> Balanced Replay (BR) [11]. This is the state-of-the-art offline-to-online RL algorithm that prioritizes near-on-policy transitions from the replay buffer to balance both offline and online samples.
> Adaptive Behavior Cloning Regularization (ABCR) [13]. This is the state-of-the-art offline-to-online RL algorithm that adaptively weights a behavior cloning regularization based loss term to solve the distribution shift issue.

We implement all the baselines above on top of the standard TD3 algorithm and run them from the same

TABLE 1
Best return on D4RL benchmark. We report both mean and standard deviation over 5 seeds.

| | Of ine | Best return | | | | |
|---|---|---|---|---|---|---|
| | TD3+BC | TD3-ft | TD3+BC-ft | BR | ABCR | OEMA (ours) |
| halfcheetah-random | 891.77 | 9359.09 241.77 | 4624.57 14.10 | 11559.56 256.89 | 11039.65 378.84 | 12205.95 183.35 |
| halfcheetah-medium | 4900.93 | 7139.52 49.76 | 5402.67 25.25 | 8201.63 169.12 | 8115.51 50.63 | 9399.60 225.82 |
| halfcheetah-medium-replay | 4632.27 | 8117.65 129.97 | 6047.79 23.90 | 7841.06 126.67 | 7422.23 85.79 | 8513.79 80.44 |
| halfcheetah-medium-expert | 10007.90 | 10007.90 0.00 | 11973.29 228.20 | 11278.00 282.82 | 10061.98 108.16 | 11018.72 274.46 |
| hopper-random | 336.61 | 3047.95 583.42 | 376.85 1.34 | 3386.45 83.98 | 3325.45 136.47 | 3544.96 49.74 |
| hopper-medium | 2852.76 | 3357.82 37.29 | 3290.15 6.95 | 3388.03 60.42 | 3398.23 75.28 | 3451.11 67.41 |
| hopper-medium-replay | 910.63 | 3311.27 39.80 | 3174.04 81.72 | 3319.20 24.46 | 3313.80 36.34 | 3472.34 138.45 |
| hopper-medium-expert | 3625.19 | 3659.51 22.90 | 3675.24 15.27 | 3708.87 27.80 | 3744.64 12.65 | 3694.62 31.13 |
| walker2d-random | 111.88 | 359.49 107.03 | 452.72 163.55 | 757.11 134.05 | 505.35 131.14 | 906.51 54.86 |
| walker2d-medium | 3702.98 | 3907.36 45.73 | 3814.31 19.95 | 4398.98 213.38 | 3989.75 76.59 | 4433.11 149.66 |
| walker2d-medium-replay | 836.65 | 4771.41 169.87 | 4165.89 135.49 | 4784.13 262.65 | 4639.01 142.84 | 4944.94 193.81 |
| walker2d-medium-expert | 4258.64 | 4695.25 359.00 | 5096.30 46.59 | 5618.93 160.72 | 5315.80 300.85 | 5485.63 179.85 |
| MuJoCo improvement | / | 215.37% | 114.81% | 274.55% | 245.65% | 299.92% |
| maze2d-umaze | 81.88 | 243.37 9.70 | 101.97 2.83 | 249.81 3.48 | 252.18 4.11 | 253.61 3.16 |
| maze2d-umaze-dense | 92.77 | 434.95 29.06 | 411.83 11.44 | 487.70 7.78 | 464.04 14.32 | 485.01 16.03 |
| maze2d-medium | 142.15 | 209.64 14.49 | 110.71 4.47 | 216.64 17.98 | 234.61 5.29 | 236.16 9.79 |
| maze2d-medium-dense | 138.97 | 338.98 14.08 | 204.40 7.19 | 380.69 23.61 | 387.61 12.57 | 400.98 30.31 |
| maze2d-large | 286.77 | 575.17 17.65 | 537.87 9.28 | 575.30 25.44 | 595.96 19.75 | 597.07 14.98 |
| maze2d-large-dense | 274.52 | 407.59 31.65 | 417.99 8.87 | 459.15 20.58 | 481.15 32.25 | 516.23 23.31 |
| Maze2D improvement | / | 250.35% | 150.02% | 282.15% | 283.98% | 297.21% |
| Average improvement | / | 232.86% | 132.42% | 278.35% | 264.82% | 298.57% |

TABLE 2
Average return over last 10 evaluations on D4RL benchmark. We report both mean and standard deviation over 5 seeds.

| | Of ine | Average return over last 10 evalutations | | | | |
|---|---|---|---|---|---|---|
| | TD3+BC | TD3-ft | TD3+BC-ft | BR | ABCR | OEMA (ours) |
| halfcheetah-random | 891.77 | 8819.63 132.92 | 4528.30 22.54 | 11121.75 318.66 | 10713.24 344.26 | 11683.28 278.59 |
| halfcheetah-medium | 4900.93 | 6930.40 56.37 | 5298.50 18.89 | 7984.44 81.08 | 7918.09 130.20 | 9130.34 198.22 |
| halfcheetah-medium-replay | 4632.27 | 7888.45 146.89 | 5965.29 11.85 | 7726.50 133.72 | 7268.70 73.13 | 8250.61 109.13 |
| halfcheetah-medium-expert | 10007.90 | 4919.67 294.04 | 10024.12 89.26 | 10705.92 258.82 | 8825.90 649.40 | 10413.45 329.84 |
| hopper-random | 336.61 | 2462.26 661.07 | 372.76 1.12 | 2812.50 520.88 | 2835.23 487.28 | 3319.63 207.93 |
| hopper-medium | 2852.76 | 3003.47 173.59 | 3253.26 3.90 | 3010.77 251.50 | 3160.27 221.22 | 3252.55 31.27 |
| hopper-medium-replay | 910.63 | 2936.81 206.30 | 2278.74 365.29 | 2944.20 225.03 | 3174.63 160.69 | 3281.98 312.27 |
| hopper-medium-expert | 3625.19 | 3299.10 106.82 | 3599.51 37.07 | 3056.05 745.96 | 3617.94 56.27 | 3537.93 89.20 |
| walker2d-random | 111.88 | 167.69 30.15 | 215.44 81.32 | 181.38 113.91 | 72.65 26.54 | 610.00 116.80 |
| walker2d-medium | 3702.98 | 3583.60 155.86 | 3606.54 55.59 | 4280.09 257.71 | 3580.80 284.97 | 4102.83 201.98 |
| walker2d-medium-replay | 836.65 | 3985.54 245.05 | 3985.09 133.45 | 4457.28 390.71 | 4221.69 110.98 | 4421.43 198.07 |
| walker2d-medium-expert | 4258.64 | 3493.79 544.74 | 4691.95 128.49 | 4532.44 342.67 | 3950.24 440.21 | 4706.36 306.31 |
| MuJoCo improvement | / | 162.39% | 82.85% | 201.93% | 187.64% | 257.03% |
| maze2d-umaze | 81.88 | 197.21 9.44 | 90.81 3.34 | 203.07 15.06 | 226.40 25.44 | 236.90 7.05 |
| maze2d-umaze-dense | 92.77 | 305.51 40.06 | 390.19 14.65 | 410.15 26.08 | 341.56 44.74 | 435.10 25.14 |
| maze2d-medium | 142.15 | 177.08 15.31 | 95.84 6.57 | 189.66 18.86 | 204.73 13.09 | 212.45 11.64 |
| maze2d-medium-dense | 138.97 | 231.56 20.84 | 164.17 7.06 | 327.62 32.77 | 302.10 24.04 | 352.74 20.91 |
| maze2d-large | 286.77 | 412.46 33.73 | 488.95 22.33 | 456.42 46.31 | 524.94 45.22 | 487.26 24.81 |
| maze2d-large-dense | 274.52 | 293.74 43.00 | 395.44 9.28 | 389.50 36.05 | 382.46 32.70 | 436.14 20.30 |
| Maze2D improvement | / | 149.69% | 124.18% | 214.95% | 202.53% | 248.45% |
| Average improvement | / | 156.04% | 103.52% | 208.44% | 195.09% | 252.74% |

codebase. Note that we omit the heavy ensemble mechanism in BR and ABCR for fair comparison. We provide all the implementation details and hyperparameters at https://github.com/guosyjlu/OEMA. Note that we do not tune any hyper-parameter of the backbone RL algorithm, i.e., TD3 for fair comparison.

We conduct all the experiments on one machine with 2 NVIDIA 3090 GPUs. We use the following software versions: Python 3.8, PyTorch 1.10.0 [33], Gym 0.19.0 [34], MuJoCo 2.2.0 [32] and mujoco-py 2.1.2.14.

### 4.2 Main Results and Analyses

To validate the sample efficiency of OEMA, we conduct extensive experiments compared with four baselines. Table 1 and 2 show the best return and average return of fine-tuning results of offline pretrained RL agents over 12 MuJoCo locomotion settings and 6 Maze2D settings. Note that we also include the average D4RL score improvement for clearer comparison. We can observe that OEMA outperforms or matches other algorithms in all the 18 settings. It is noteworthy

Fig. 4. Results of online netuning with different exploration strategies on three D4RL MuJoCo locomotion tasks with random datasets. We plot the mean and standard deviation across 5 seeds. Best viewed in color.

thy that OEMA achieves 20.22% and 44.30% improvement over the best performing baseline BR in terms of the best return and average return D4RL score improvement, respectively. This veri es the effectiveness of OEMA for improving the sample ef ciency. The reasons are two-fold: (i) the optimistic exploration strategy can seek novel states and actions that might yield high rewards and lead to long-term gains, achieving better sample ef ciency; (ii) the meta learning based adaptation method can reduce the distribution shift and accelerates the of ine-to-online adaptation.

In addition, we nd that most of the of ine RL agents bene t from online netuning, especially for those sub-optimal agents pretrained with random datasets. This is reasonable since RL succeeds according to the process of trial and error, and the online netuning enables the agents to timely discover and correct errors, and thus, improve the performance. Besides, TD3-ft fails to bring netuning improvement in some settings, which demonstrates that the exploration limitation and of ine-to-online distribution shift may hurt the performance.

Last but not least, we derive that it is easier for behavior cloning based algorithms, i.e., TD3+BC-ft and ABCR, to achieve desirable performance in the settings with high-quality datasets. However, they fail to bring signi cant improvement in the other settings. In contrast, OEMA can achieve the best or at least competitive performance in all the 18 settings.

### 4.3 In-depth Analysis on Optimistic Exploration

In this subsection, we give an in-depth analysis on the proposed optimistic exploration strategy via ablation study and hyperparameter tuning on optimism level. Note that we mainly focus on MuJoCo locomotion tasks in the following subsections, given their extensive adoption [11], [14], [15].

#### 4.3.1 Ablation Study on Exploration Strategy

We rst analyze the impact of different exploration strategies via an ablation study. In particular, we replace the optimistic exploration strategy of OEMA by (i) w/o-OE, the vanilla exploration method in TD3, i.e., $_e(s) = (s) + ; N(0; )$, and (ii) OAC [20], an exploration strategy from the approximate closed form solution in the form of Eq. 11. We strictly follow the hyperparameters in the original paper. Fig. 4 shows the online netuning curves on three tasks with the random of ine dataset. OAC fails in all three settings, which can be attributed to its unsafe exploration at action boundaries. Note that this observation is identical to [29]. Additionally, OEMA outperforms the vanilla exploration method, w/o-OE, in terms of both sample ef ciency and nal performance over all three settings. Particularly, w/o-OE attains a sudden performance drop in the halfcheetah while OEMA maintains relatively stable performance improvement. This veri es the effectiveness of the proposed optimistic exploration strategy for nding novel states and actions that might yield high rewards and achieve long-term gains.

Fig. 5. Performance comparison with different optimism levels $_{UB}$ on three D4RL MuJoCo locomotion tasks with four different-quality datasets. We report the averaged best D4RL score across 5 seeds with 300K environment steps.

Fig. 6. Performance comparison with different meta weights on three D4RL MuJoCo locomotion tasks with four different-quality datasets. We report the averaged best D4RL score across 5 seeds with 300K environment steps.

#### 4.3.2 Hyper-parameter Analysis on Optimism Level $_{UB}$

Furthermore, we investigate the performance of OEMA with different optimism levels $_{UB}$. We set $_{UB}$ $2$ $f1; 2; :::; 10g$ and plot the best D4RL score on three tasks with different datasets in Fig. 5. Overall, the optimal optimism level depends on both task and quality of the dataset. Speci cally, halfcheetah prefers larger optimism level to encourage stronger optimistic exploration on state-action pairs with high epistemic uncertainty, while hopper and walker2d prefer more conservative optimistic exploration due to the complexity of the environment. In terms of the dataset quality, optimistic exploration strategy with high optimism level fails in the medium-replay datasets. This is reasonable since medium-replay datasets mix both the sub-optimal data and random data, resulting in extremely high epistemic uncertainty, and thus, limited performance. In practice, we suggest choosing the optimism level with the consideration of both task complexity and distribution of the of ine dataset. Conservative optimism level is preferred

TABLE 3
Performance comparison with different adaptation methods on D4RL MuJoCo locomotion tasks with different-quality datasets. We report the average D4RL score over last 10 evaluations with 300K environment steps. All the results are averaged across 5 seeds.

| | Best D4RL score | | | | Avg D4RL score over last 10 evaluations | | | |
|---|---|---|---|---|---|---|---|---|
| | w/o-MA | MC | control | OEMA | w/o-MA | MC | control | OEMA |
| halfcheetah-random | 99.84 1.04 | 93.44 1.67 | 100.96 1.92 | 100.57 1.48 | 96.10 1.26 | 87.76 2.52 | 97.81 1.77 | 96.36 2.24 |
| halfcheetah-medium | 76.07 0.50 | 43.37 1.08 | 78.09 1.03 | 77.97 1.82 | 74.25 0.83 | 22.36 1.17 | 76.13 0.92 | 75.80 1.60 |
| halfcheetah-medium-replay | 68.57 0.73 | 67.08 1.43 | 69.11 0.88 | 70.83 0.65 | 66.55 1.05 | 65.15 1.39 | 67.42 0.91 | 68.71 0.88 |
| halfcheetah-medium-expert | 87.94 2.27 | 82.87 0.00 | 87.76 3.09 | 91.01 2.21 | 81.81 3.13 | 46.89 12.77 | 83.55 2.44 | 86.13 2.66 |
| hopper-random | 109.06 3.26 | 90.37 27.11 | 107.49 1.83 | 109.55 1.53 | 98.03 8.04 | 70.99 36.12 | 96.52 12.76 | 102.62 6.39 |
| hopper-medium | 108.23 2.13 | 100.82 6.66 | 106.64 2.99 | 106.66 2.07 | 100.28 5.40 | 49.12 46.07 | 98.77 7.76 | 100.56 0.96 |
| hopper-medium-replay | 106.93 4.04 | 104.62 4.49 | 109.24 4.67 | 107.31 4.25 | 95.75 11.14 | 57.55 37.29 | 99.18 7.86 | 101.47 9.59 |
| hopper-medium-expert | 114.12 0.80 | 112.01 0.00 | 113.69 0.60 | 114.14 0.96 | 109.21 2.66 | 27.07 30.28 | 109.18 3.81 | 109.33 2.74 |
| walker2d-random | 18.44 2.37 | 17.21 1.60 | 17.39 2.38 | 19.71 1.19 | 8.54 4.88 | 5.88 2.79 | 5.19 2.51 | 13.25 2.54 |
| walker2d-medium | 91.75 2.98 | 80.63 0.00 | 95.23 2.11 | 96.53 3.26 | 85.63 5.66 | 19.04 8.00 | 86.89 4.52 | 89.34 4.40 |
| walker2d-medium-replay | 101.83 4.29 | 99.64 3.80 | 99.04 5.44 | 107.68 4.22 | 88.44 11.36 | 47.44 14.97 | 88.70 11.17 | 96.28 4.31 |
| walker2d-medium-expert | 116.43 2.19 | 99.52 9.08 | 115.59 2.69 | 119.46 3.92 | 87.02 5.77 | 26.18 33.04 | 85.02 13.09 | 102.48 6.67 |
| Average D4RL score | 91.60 | 82.63 | 91.69 | 93.45 | 82.53 | 43.79 | 82.86 | 86.86 |

with complex task and diverse dataset distribution. Note that as the target policy gradually converges, we should encourage the behavior policy to be less exploratory. Thus, it is intuitively better to dynamically decay the optimism level during online netuning, which is a potential future research direction.

### 4.4 In-depth Analysis on Meta Adaptation

In this subsection, we give an in-depth analysis on the proposed meta adaptation method via ablation study, controlled experiment on batch information, and hyperparameter tuning on meta weight.

#### 4.4.1 Ablation Study on Meta Adaptation

We rst conduct an ablation study to investigate the impact of different of ine-to-online adaptation methods. Particularly, we omit the meta adaptation method in OEMA to derive (i) w/o-MA , and we replace the meta adaptation method by (ii) Meta Critic (MC) [35]. Although MC is not speci cally proposed for of ine-to-online adaptation, it can similarly accelerate the policy improvement in off-policy RL algorithms. Table 3 lists the results of the best and average D4RL score over last 10 evaluations across the 12 settings. Note that the column denoted by control will be discussed in the Section 4.4.2. Speci cally, OEMA signi cantly outperforms w/o-MA and MC, which veri es the effectiveness of the proposed meta adaptation method. Moreover, MC fails in most of the settings. This is because MC only enables the feature extraction of the policy network to adapt to the online settings, which may bring potential bias and even result in performance degradation.

#### 4.4.2 Controlling for Batch Information

Since the meta adaptation method in OEMA brings more potential information via two mini-batches from the overall replay buffer and recent replay buffer, we conduct controlled experiments to perform vanilla policy improvement on both mini-batches, denoted by control in Table 3. Note that we can also regard control as a naive of ine-to-online adaptation method. As shown in Table 3, control outperforms w/o-MA in some settings, but fails to bring signi cant improvement in most of the settings. From the perspective of best D4RL score across 12 settings, control only improves w/o-MA by 0.09, while the proposed meta adaptation method improves that by 1.95. Similarly, from the perspective of average D4RL score across 12 settings, control only improves w/o-MA by 0.23, while the proposed meta adaptation method improves that by 4.23. This reveals the limitation of naively using both mini-batches for of ine-to-online adaptation. Moreover, the results demonstrate that the proposed meta adaptation method bene ts not only from the potential information contained in two mini-batches, but more from the meta-learned of ine-to-online adaptation direction.

#### 4.4.3 Hyper-parameter Analysis on Meta Weight

We investigate the performance of OEMA with different meta weights . We set 2 f0:01; 0:05; 0:1; 0:5; 1:0g and plot the best D4RL score on three tasks with different datasets in Fig. 6. The performance of OEMA is not signi cantly impacted by the meta weight in most of the settings. This indicates that the meta weight is not a sensitive hyperparameter. However, when we set = 0, it is just the w/o-MA above, which is 4.23 lower than OEMA with = 1:0 in average. This veri es the effectiveness of the meta adaptation method. It is also desirable to point out that the of ine-to-online gap gradually decreases with the constant netuning process. Thus, it is intuitively better to dynamically tune the meta weight during online netuning. We leave this for future work, which is a potential future research direction.

### 4.5 Comparison Among Different Replay Buffers

In this subsection, we investigate the impact of different replay buffers on the performance of OEMA. In particular, we take the following replay buffers for comparison: (i) naive initializes the overall replay buffer with all transitions in the of ine dataset. (ii) downsampling [13] initializes the overall replay buffer with 5% randomly sampled transitions from the of ine dataset. (iii) BR [11] performs the same initialization process as naive, but it prioritizes the near-on-policy transitions during the online netuning process. Table 4 shows the best return and average return netuning

TABLE 4
Performance comparison of OEMA with different replay buffer. We show the best return, and average return over last 10 evaluations on D4RL MuJoCo locomotion tasks with 300K environment steps. All the results are averaged over 5 seeds. Best results are highlighted.

| | Best return | | | Avg return over last 10 evaluations | | |
|---|---|---|---|---|---|---|
| | naive | downsampling | BR | naive | downsampling | BR |
| halfcheetah-random | 10247.48 | 11904.89 | 12205.95 | 9757.65 | 11398.49 | 11683.28 |
| halfcheetah-medium | 7746.81 | 9016.80 | 9399.60 | 7543.44 | 7886.69 | 9130.34 |
| halfcheetah-medium-replay | 7638.85 | 7981.95 | 8513.79 | 7404.60 | 7711.37 | 8250.61 |
| halfcheetah-medium-expert | 10148.91 | 10148.91 | 11018.72 | 4382.13 | 7479.86 | 10413.45 |
| hopper-random | 3261.87 | 3383.01 | 3544.96 | 2459.79 | 2888.44 | 3319.63 |
| hopper-medium | 3336.14 | 3534.12 | 3451.11 | 2945.64 | 3373.52 | 3252.55 |
| hopper-medium-replay | 3275.43 | 3460.66 | 3472.34 | 2566.93 | 3117.71 | 3281.98 |
| hopper-medium-expert | 3679.30 | 3710.34 | 3694.62 | 3408.15 | 3567.00 | 3537.93 |
| walker2d-random | 675.43 | 660.18 | 906.51 | 145.05 | 222.62 | 610.00 |
| walker2d-medium | 3937.81 | 3988.44 | 4433.11 | 3447.63 | 3389.54 | 4102.83 |
| walker2d-medium-replay | 4269.31 | 4422.45 | 4944.94 | 4013.62 | 4150.12 | 4421.43 |
| walker2d-medium-expert | 4355.05 | 5227.20 | 5485.63 | 2733.17 | 3823.55 | 4706.36 |

results of of ine pretrained RL agents over three MuJoCo locomotion tasks with four different-quality datasets. Overall, naive fails in most of the settings, which veri es the necessity of balancing the ratio of online transitions to of ine transitions to achieve better sample ef ciency. Additionally, we nd that the downsampling mechanism can eliminate this issue to some extend. However, OEMA with downsampling replay buffer still suffers from the quality of the of ine dataset. Among them, OEMA with BR achieves the best performance in most of the settings. This is reasonable since BR can dynamically adjust the distribution of the sampled transitions according to the target policy, thus achieving better sample ef ciency. As a result, we utilize BR as the overall replay buffer in practice.

## 5 Related Work

In this section, we introduce three aspects of the related work, including of ine-to-online RL, optimistic exploration in RL, and meta learning for RL.

### 5.1 Of ine-to-Online RL

Of ine RL aims to learn a policy from a previously collected xed-size dataset without any further interaction with the environment [28]. Built on top of the off-policy RL algorithms [25], [30], [36], recent works propose various constraints to avoid the distribution shift, such as conservatism mechanism [14], [37], policy regularization [15], [38], [39], etc. However, of ine RL typically has limited performance due to sub-optimal datasets. Hence, it is necessary to further netune the of ine RL agents before the deployment.

Generally, existing of ine-to-online RL algorithms can be divided into two branches. The rst branch is to design a uni ed of ine pretraining and online netuning method [12], [37], [40]. However, this branch of work has limited applicable scenarios. The other branch is to design general online netuning methods for arbitrarily given pretrained RL agents. The state-of-the-art model-free of ine-to-online RL algorithms are the balanced replay [11] and the adaptive behavior cloning regularization [13]. Both of them are designed to solve the distribution shift issue. In contrast, OEMA aims to solve both exploration limitation and distribution shift issue. There is also a model-based of ine-to-online RL algorithm named MOORe [10]. However, since model-based RL algorithms are typically more sample-ef cient than model-free RL, we do not take it for comparison. Besides, online decision transformer [41] studies the of ine-to-online settings, but it is built on the upside-down RL [42], and thus, we do not take it for comparison.

### 5.2 Optimistic Exploration in RL

Optimistic exploration is built on the principle of optimism in the face of uncertainty, which means exploring the environment with the highest guess for best actions to balance the exploration and exploitation under constant trials and errors [17], [18], [19], [43], [44]. OAC [20] proposes to estimate the epistemic uncertainty for exploration and gives an approximate closed form solution. However, a recent study [29] points out that OAC tends to choose actions at the action boundaries, resulting in low sample ef ciency, and proposes another approximate closed form solution built on top of the trust region optimization. Different from them, we propose a perturbation based optimistic exploration strategy in an iterative manner, which naturally satis es the constraints of the optimization and attains better stability.

### 5.3 Meta Learning for RL

Meta learning, a.k.a, learning to learn, aims to improve the learning algorithm based on the previous experience of multiple learning process [45]. In terms of RL, the most direct application of meta learning is meta RL [21], [22], [46], which leverages a set of training tasks to learn a policy that can quickly adapt to new test tasks. We borrow the idea of meta RL to develop the meta adaptation method. However, there are essential differences between us because our problem setting only involves one task. Additionally, there are some recent works utilizing meta learning to boost the performance of existing RL algorithms. For example, meta SAC [47] applies meta learning to automatically tune the entropy hyperparameter in SAC. Moreover, meta critic [35] trains an extra critic via meta learning to improve actor-critic based off-policy RL algorithms. Different from both, we utilize meta learning to reduce the distribution shift and accelerate the of ine-to-online adaptation.

## 6 Conclusion and Future Work

In this paper, we identify the exploration limitation and the distribution shift as the major obstacle towards sample-efficient offline-to-online RL. To this end, we propose an efficient offline-to-online RL algorithm via optimistic exploration and meta adaptation. The optimistic exploration strategy enables the agents to explore novel state-action pairs that might yield high rewards while attaining the stability. Moreover, the meta learning based adaptation method reduces the distribution shift and accelerates the offline-to-online adaptation. We empirically demonstrate that the proposed OEMA algorithm attains state-of-the-art offline-to-online performance on the popular D4RL benchmark. Besides, extensive experiments are conducted to verify the effectiveness of the proposed optimistic exploration strategy and meta learning based adaptation method.

For future work, we expect to investigate the impact of the initialization with different offline pretrained agents on the offline-to-online performance, which may provide us a more instructive direction on the evaluation of the offline RL algorithms. Besides, we also expect to explore the application of offline-to-online RL in real-world production systems, such as recommender system and search system.

## Acknowledgments

This work is partially supported by the National Natural Science Foundation of China under grants Nos. 61976102 and U19A2065; the International Cooperation Project of Jilin Province under grant No. 20220402009GH.

## References

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

[2] T. Xiao and D. Wang, "A general offline reinforcement learning framework for interactive recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4512–4520.

[3] C. Gao, W. Lei, J. Chen, S. Wang, X. He, S. Li, B. Li, Y. Zhang, and P. Jiang, "Cirs: Bursting filter bubbles by counterfactual interactive recommender system," *arXiv preprint arXiv:2204.01266*, 2022.

[4] S. Tang and J. Wiens, "Model selection for offline reinforcement learning: Practical considerations for healthcare settings," in *Machine Learning for Healthcare Conference*. PMLR, 2021, pp. 2–35.

[5] M. Fatemi, M. Wu, J. Petch, W. Nelson, S. J. Connolly, A. Benz, A. Carnicelli, and M. Ghassemi, "Semi-markov offline reinforcement learning for healthcare," in *Conference on Health, Inference, and Learning*. PMLR, 2022, pp. 119–137.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2020.

[7] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 000–16 009.

[8] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[9] Y. Sun, S. Wang, S. Feng, S. Ding, C. Pang, J. Shang, J. Liu, X. Chen, Y. Zhao, Y. Lu *et al.*, "Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation," *arXiv preprint arXiv:2107.02137*, 2021.

[10] Y. Mao, C. Wang, B. Wang, and C. Zhang, "Moore: Model-based offline-to-online reinforcement learning," *arXiv preprint arXiv:2201.10070*, 2022.

[11] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin, "Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble," in *Conference on Robot Learning*. PMLR, 2022, pp. 1702–1712.

[12] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[13] Y. Zhao, R. Boney, A. Ilin, J. Kannala, and J. Pajarinen, "Adaptive behavior cloning regularization for stable offline-to-online reinforcement learning," *arXiv preprint arXiv:.2210.13846*, 2022.

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

[15] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 20 132–20 145, 2021.

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

[17] T. L. Lai, H. Robbins *et al.*, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[18] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Machine learning*, vol. 49, no. 2, pp. 209–232, 2002.

[19] R. I. Brafman and M. Tennenholtz, "R-max-a general polynomial time algorithm for near-optimal reinforcement learning," *Journal of Machine Learning Research*, vol. 3, no. Oct, pp. 213–231, 2002.

[20] K. Ciosek, Q. Vuong, R. Loftin, and K. Hofmann, "Better exploration with optimistic actor critic," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[21] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[22] D. Li, Y. Yang, Y.-Z. Song, and T. Hospedales, "Learning to generalize: Meta-learning for domain generalization," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[23] 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.

[24] S. Thrun and M. L. Littman, "Reinforcement learning: an introduction," *AI Magazine*, vol. 21, no. 1, pp. 103–103, 2000.

[25] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

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

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

[28] R. F. Prudencio, M. R. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *arXiv preprint arXiv:2203.01387*, 2022.

[29] N. Kappes and P. Herrmann, "Trust region optimization of optimistic actor critic," 2022.

[30] 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*. PMLR, 2018, pp. 1861–1870.

[31] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[32] 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*. IEEE, 2012, pp. 5026–5033.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[34] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[35] W. Zhou, Y. Li, Y. Yang, H. Wang, and T. Hospedales, "Online meta-critic learning for off-policy actor-critic methods," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 662–17 673, 2020.