Adaptive Policy Learning for Offline-to-Online Reinforcement Learning

Han Zheng[1*], Xufang Luo[2], Pengfei Wei[3],
Xuan Song[4], Dongsheng Li[2], Jing Jiang[1]

[1]University of Technology Sydney, [2]Microsoft Research Asia, [3]National University of Singapore,
[4]Southern University of Science and Technology

Abstract

Conventional reinforcement learning (RL) needs an environment to collect fresh data, which is impractical when online interactions are costly. Offline RL provides an alternative solution by directly learning from the previously collected dataset. However, it will yield unsatisfactory performance if the quality of the offline datasets is poor. In this paper, we consider an offline-to-online setting where the agent is first learned from the offline dataset and then trained online, and propose a framework called Adaptive Policy Learning for effectively taking advantage of offline and online data. Specifically, we explicitly consider the difference between the online and offline data and apply an adaptive update scheme accordingly, that is, a pessimistic update strategy for the offline dataset and an optimistic/greedy update scheme for the online dataset. Such a simple and effective method provides a way to mix the offline and online RL and achieve the best of both worlds. We further provide two detailed algorithms for implementing the framework through embedding value or policy-based RL algorithms into it. Finally, we conduct extensive experiments on popular continuous control tasks, and results show that our algorithm can learn the expert policy with high sample efficiency even when the quality of offline dataset is poor, e.g., random dataset.

Introduction

Conventional online reinforcement learning (RL) methods (Haarnoja et al. 2018; Fujimoto, van Hoof, and Meger 2018) usually learn from experiences generated by online interactions with the environment. They are impractical in some real-world applications, e.g., dialog (Jaques et al. 2019) and education (Mandel et al. 2014), where interactions are costly. Furthermore, many offline data have already been generated by one or more policies, but online RL agents usually cannot utilize them directly (Fujimoto, Meger, and Precup 2019). In contrast, supervised learning achieves remarkable successes across a range of domains (Kotsiantis et al. 2007; Brown et al. 2020) by directly leveraging existing large-scale datasets, such as ImageNet (Deng et al. 2009). To apply such data-driven learning paradigm with RL objectives, many offline RL methods have been proposed and aroused much attention recently (Levine et al. 2020). They try to learn policies from static datasets which are pre-collected by arbitrary policies.

Existing offline RL studies mainly focus on remedying the issue of distribution mismatch or out-of-distribution (OOD) actions by employing a pessimistic update scheme (Kumar et al. 2019, 2020) or in combination with imitation learning (Fujimoto, Meger, and Precup 2019). However, when the dataset is fixed and sub-optimal, it is nearly impossible for offline RL to learn the optimal policy (Kidambi et al. 2020). Even worse, when a large portion of actions or states is not covered within the training set distribution, offline RL methods usually fail to learn good policies (Kumar et al. 2020; Fu et al. 2020; Levine et al. 2020). Moreover, policy evaluation under the offline RL setting is also challenging. Although some methods, such as off-policy evaluation algorithms (Dann et al. 2014), are proposed for this problem, they are still not ideal for the practical purpose.

Some recent works address the above issues by employing an offline-to-online setting. Such methods (Lee et al. 2021; Nair et al. 2020) focus on pre-training a policy using the offline dataset and fine-tuning the policy through further online interactions. Even though their methods alleviate the above problems, they have not considered the different advantages of offline and online data and utilised both well. Specifically, offline data can prevent agents from prematurely converging to sub-optimal policies thanks to the potential data diversity, while online data can stabilize training and accelerate convergence (Nair et al. 2020; Sutton and Barto 2011). They can contribute to better policy learning in different ways. But prior methods only stress one of them. For instance, Nair et al. (2020) employs a pessimistic strategy to update the policy, which may be problematic for improving policy performance during the online phase. Unlike Nair et al. (2020), Lee et al. (2021) only selects near-on-policy data from the offline dataset during online phase to alleviate the distribution shift in transition from offline to online. Such a strategy ignores a large part of the offline dataset, whose potential data diversity is important for learning better policies.

To tackle the above problems, in this paper, we propose that *the advantage of online and offline data should be emphasized in an adaptive way.* First, considering their different characteristics, separate updating strategies should be

[*]This work was done when Han Zheng was interning with Microsoft Research Asia. Email: z.han0903@gmail.com

employed for online and offline data, respectively. To do so, we present a novel framework called *Adaptive Policy Learning* (APL) that takes advantages of them effectively. The core idea is simple: when learning from online data, the agent is updated in an optimistic way, and when learning from the offline dataset, the agent is optimized by a pessimistic strategy. The intuition behind this is that near-on-policy data can be collected via online interactions, so an optimistic strategy is used here for better policy improvement, and potentially useful offline datasets can be collected by arbitrary policies, so a pessimistic strategy is used to exploit all these data. In addition, to distinguish between offline and online data in a simple way, we design a two-level replay buffer for APL framework. We further provide a value-based and policy-based implementations for APL framework through embedding state-of-the-art (SOTA) online and offline RL methods into the framework. Experimental results demonstrate that our algorithm can learn expert policies in most tasks with a high sample efficiency regardless of the quality of offline datasets. More specifically, our algorithm performs better by using only 20% online interactions compared with the previous offline-to-online method (Nair et al. 2020).

Our contributions can be summarized as below:

- We propose a simple framework called Adaptive Policy Learning (APL) that considers different advantages of offline and online data for policy learning, and can effectively make use of them.
- We further provide a value-based and a policy-based algorithms to implement APL framework, showing APL is compatible with various RL methods.
- We test our algorithm on the popular continuous control tasks MuJoCo (Todorov, Erez, and Tassa 2012) and compare it with some strong baselines. The results clearly show that the APL framework is effective under the offline-to-online setting.

Related Work

Online RL In general, online RL algorithms can be divided into two categories, i.e., on-policy and off-policy algorithms. On-policy methods (Schulman et al. 2015, 2017) update the policy using data collected by its current behavior policy. As ignoring the logged data collected by its history behavior policies, they usually have a lower sample efficiency than the off-policy RL. On the other hand, off-policy methods (Fujimoto, van Hoof, and Meger 2018; Chen et al. 2021a) enable the policy to learn from experience collected by history behavior policies. However, they cannot learn from history trajectories collected by other agents' behavior policies (Fujimoto, Meger, and Precup 2019; Kumar et al. 2020). Consequently, the need for huge online interaction makes online RL impractical for some real-world applications, such as dialog agents (Jaques et al. 2019) or the education system (Mandel et al. 2014).

Offline RL Offline RL algorithms assume the online environment is unavailable and learn policies only from the pre-collected dataset. As the value estimation error cannot be corrected using online interactions, these methods utilize a pessimistic updating strategy to relieve the distribution mismatch problem (Fujimoto, Meger, and Precup 2019; Kumar et al. 2019). Model-free offline RL methods generally employ value or policy penalties to constrain the updated policy close to the data collecting policy (Wu, Tucker, and Nachum 2019; Kumar et al. 2020; Fujimoto, Meger, and Precup 2019; He and Hou 2020; Ghosh et al. 2022). Model-based methods use predictive models to estimate uncertainties of states and then update the policy in a pessimistic way based on them (Kidambi et al. 2020; Yu et al. 2020; Chen et al. 2021b). Those offline RL methods cannot guarantee a good performance, especially when the data quality is poor (Kumar et al. 2020). Besides, policy evaluation when the online environment is unavailable is also challenging. Even though off-policy evaluation (OPE) methods (Dann et al. 2014) present alternative solutions, they are still far from perfect.

The above issues in online and offline RL motivate us to investigate the offline-to-online setting.

Offline-to-online RL Some works focus on the mixed setting where the agent is first learned from the offline dataset and then trained online. Nair et al. (2020) propose an advantage-weighted actor-critic (AWAC) method that restricts the policy to select actions close to those in the offline data by an implicit constraint. Kostrikov, Nair, and Levine (2022) present IQL, implementing a weighted behavioral cloning step for better online policy improvement, which can also be used in offline-to-online setting. When online interactions are available, such conservative designs may adversely affect the performance. OFF2ON (Lee et al. 2021) employs a balanced replay scheme to address the distribution shift issue. It uses offline data by only selecting near-on-policy samples. Unlike these works, our method addresses all online and offline data, and explicitly considers the difference between them by adaptively applying optimistic and pessimistic updating schemes for online and offline data, respectively. Besides, our framework is more flexible because it can be easily applied to policy or value-based methods. Moreover, our algorithm is much more robust to different hyper-parameters (see the experiment section). Particularly, without careful tuning, our algorithm can achieve better or comparable performance than prior methods, while OFF2ON uses a specialized network architecture, and the hyper-parameters are fine-tuned. Matsushima et al. (2021) focus on optimizing deployment efficiency, i.e., the number of distinct data-collection policies used during learning, by employing a behavior-regularized policy updating strategy. However, they ignore existing offline dataset, and dose not focus on improving sample efficiency, while both are addressed in our paper. Some works (Zhu et al. 2019; Vecerik et al. 2017; Rajeswaran et al. 2018; Kim et al. 2013) can also learn from online interactions and offline data. However, they need expert demonstrations instead of any dataset, limiting their applicability.

Preliminaries

In RL, interactions between the agent and environment are usually modelled using Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, p_{\mathrm{M}}, r, \gamma)$, with state space $\mathcal{S}$ (state $\mathbf{s} \in \mathcal{S}$), action space $\mathcal{A}$ (action $\mathbf{a} \in \mathcal{A}$). At each discrete time step, the agent takes an action $\mathbf{a}$ based on the current state $\mathbf{s}$, and the state changes into $\mathbf{s}'$ according to the transition dynamics $p_{\mathrm{M}}(\mathbf{a}' \mid \mathbf{s}, \mathbf{a})$, and the agent receives a reward $r \in \mathbb{R}$. The agent's objective is to maximize the return, which is defined as $R_t = \sum_{n=0}^{\infty} \gamma^n r_{t+n}$, where t is the time step, and $\gamma \in [0, 1)$ is the discounted factor. The mapping from $\mathbf{s}$ to $\mathbf{a}$ is denoted by the stochastic policy $\pi : \mathbf{a} \sim \pi(\cdot|\mathbf{s})$. Policy can be stochastic or deterministic, and we use the stochastic from in this paper for generality. Each policy π have a corresponding action-value function $Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi[R_t \mid \mathbf{s}, \mathbf{a}]$, which is the expected return following the policy after taking action $\mathbf{a}$ in state $\mathbf{s}$. The action-value function of policy π can be updated by the Bellman operator $\mathcal{T}^\pi$:

$$\mathcal{T}^\pi Q(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\mathbf{s}'}[r + \gamma Q(\mathbf{s}', \pi(\mathbf{s}'))] \tag{1}$$

Q-learning (Sutton and Barto 2011) directly learns the optimal action-value function $Q^*(\mathbf{s}, \mathbf{a}) = \max_\pi Q^\pi(\mathbf{s}, \mathbf{a})$, and such Q-function can be modelled using neural networks (Mnih et al. 2015).

In principle, off-policy methods, such as Q-learning, can utilize experiences collected by any policy, and thus they usually maintain a replay buffer $\mathcal{B}$ to store and repeatedly learn from experiences collected by behavior policies (Agarwal, Schuurmans, and Norouzi 2020). Such capability also enables off-policy methods in the offline setting by storing offline data into the buffer $\mathcal{B}$ and not updating the buffer during learning since no further interactions are available here (Levine et al. 2020). However, this simple adjusting generally results in poor performance due to bootstrapping errors from OOD actions, especially when the dataset is not diverse (Kumar et al. 2020; Agarwal, Schuurmans, and Norouzi 2020; Fujimoto, Meger, and Precup 2019), and this is also the problem tackled in most offline RL works.

This paper focuses on the offline-to-online setting, where the agent learns from both the offline and online datasets. We use off-policy methods in the online part because they can utilize data more effectively and generally have higher sample efficiency than on-policy ones. Thus, if no additional marks, online RL methods refer to off-policy algorithms in the rest of this paper.

Methodology

In this section, we first give an illustrative example to explain our motivation. Then, we introduce the proposed APL framework, trying to couple online and offline RL in an adaptive way. Finally, we present detailed algorithms for implementing APL framework.

An illustrative example We test the SOTA offline and online RL methods under the offline-to-online setting, i.e., Conservative Q-learning (CQL) (Kumar et al. 2020) and Randomized Ensembled Double Q-learning (REDQ) (Chen et al. 2021a), respectively. Specifically, we first pre-train the agent with the offline dataset for $100K$ steps. Then, the agent is fine-tuned online by alternately conducting the interaction and updating process. The agent interacts with the environment for $1K$ steps, and is updated for $10K$ steps. The total online interactions steps is around $100K$.

The offline-to-online setting described above should be a favourable one for policy learning because both diverse of-

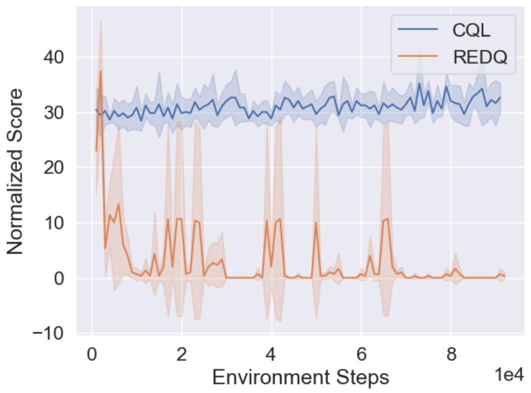

Figure 1: Learning curves for the online agent initialized with the D4RL (Fu et al. 2020) dataset hopper-medium-replay-v0. Scores are averaged over five random seeds and the shaded areas represent the standard deviation. The normalized score of 100 is the average returns of a domain-specific expert while normalized score of 0 corresponds to the average returns of an agent taking uniformly random actions across the action space.

fline data and online interactions are available. Therefore, we expect the following results: the initial offline training with the fixed dataset will provide relatively good but not expert performance for the agent, and then the agent can be further improved by the online process since online interaction data can be obtained. At last, we may acquire a well-performed policy, with the normalized score close to or better than 100 (i.e., the performance of an expert).

Nonetheless, the experiment results in Figure 1 do not show what we expected. Specifically, starting points of the two curves show that the offline algorithm CQL and online algorithm REDQ can obtain normalized scores greater than 0 (i.e., the performance of a random policy). This indicates that by learning from the offline dataset, the agent can obtain a relatively good starting point as expected. And these starting scores (below 30) are also far from the expert score, which leave a large room for further promotion in the online phase. However, both algorithms have trouble in the online process. REDQ suffers from severe instability, and in the end, the policy almost degenerates into a random one. Although the CQL agent can keep stable during learning, its improvement is very limited, which indicate it is ineffective for leveraging online interaction.

These results indicate that a pure online RL algorithm may be problematic for handling offline and online data in a single training process. Furthermore, the pure offline RL algorithm cannot effectively use valuable online interaction data due to its conservative updating strategy.

Adaptive Policy Learning Framework

We present the *Adaptive Policy Learning* (APL) framework in this subsection, trying to tackle the above problem by adaptively leveraging online and offline data. The underlying idea is simple and can be described as follows. When online interactions are available, we try to obtain near-on-policy data from them, and choose an optimistic updating strategy since these data reflect the true situation of the current policy. By contrast, when data are sampled from the offline dataset, we use a more pessimistic updating strategy. In this way, we can make full use of both online and offline data and explicitly consider their differences by separately applying suitable updating schemes. It is worth stressing that being optimistic here means we rely on an online RL update strategy which is optimistic compared with offline RL methods, while being pessimistic means an offline RL update strategy.

Then, we give a general formalization for the above idea. Since this idea can be applied to both value-based and policy-based methods, we use C to represent a policy or state-action value function, and define a unified updating rule as follows:

$$C^{k+1} \leftarrow \mathcal{F}\left(\mathbb{A}(C^k) + \mathcal{W}(\mathbf{s},\mathbf{a})\mathbb{B}(C^k)\right), \tag{2}$$

where k is the times of iterative updates. For the right-hand side, the first term $\mathbb{A}(C)$ stands for the optimistic updating strategy. It is the common learning target used in online RL, such as the Bellman error for value-based methods. The second term $\mathbb{B}(Q)$ stands for the pessimistic updating strategy. It is a penalty term to make the learned agent take actions close to the dataset or take conservative actions. Besides, to express the idea of adaptive learning, we apply a weight function $\mathcal{W}(\mathbf{s},\mathbf{a})$ to the penalty term $\mathbb{B}(Q)$. Specifically, when we use near-on-policy data, $\mathcal{W}(\mathbf{s},\mathbf{a})$ will be a small value, and the updating relies more on $\mathbb{A}(C)$, leading to a relatively optimistic updating strategy. On the contrary, when we use the offline data, $\mathcal{W}(\mathbf{s},\mathbf{a})$ will be a large value, and the updating strategy is relatively pessimistic. In addition, we use $\mathcal{F}$ to denote the general updating operator, such as argmin or argmax. Using this general operator, the updating rule for both value and policy-based methods can be unified in Eq. 2. Note that we only introduce the general framework here, and detailed algorithms are presented in next sections.

Online-Offline Replay Buffer We now introduce a simple but effective online-offline replay buffer (OORB) to distinguish between near-on-policy and offline data. OORB consists of two buffers. One is the online buffer that collects the near-on-policy data generated via online interactions. To ensure the data in the online buffer is near-on-policy, we set this buffer to be small, and newly collected online data are stored into it by following the first-in-first-out rule. The other is the offline buffer containing previously collected offline dataset which can come from arbitrary policies, and all data generated via interactions in the online phase.

Data are sampled from OORB following a Bernoulli distribution. With a probability p, data are sampled from the online buffer, and with probability $1-p$, they are sampled from the offline buffer. The effect of p on the final performance is tested via ablation studies in the experiment section.

Value-Based Implementation

We first present an action-value based implementation for APL framework by incorporating CQL (Kumar et al. 2020) as the value-based offline RL method and an ensemble online RL algorithm REDQ (Chen et al. 2021a). We name the implementation Greedy-Conservative Q-ensemble Learning (GCQL).

Algorithm 1: Greedy Conservative Q-ensemble Learning

```
// Online interaction steps in each iteration $T_{\text{on}}$, updating steps in each iteration $T_{\text{off}}$, initial offline training steps $T_{\text{initial}}$
// OORB threshold $p$, OORB starting sampling size $T_{\text{s}}$
**Input:** Total online interaction steps $S_T$
Initialize online buffer $B_{\text{on}}$ to empty, offline buffer $B_{\text{off}} \leftarrow$ offline dataset
Initialize policy $\pi$, Q-functions $Q_i, i \in N$
Set the updating step counter $t$
Set the total online interaction step counter $S_{\text{on}} \leftarrow 0$
Train the agent for $T_{\text{initial}}$ steps using the offline dataset
**repeat**
  $t \leftarrow 0$
  Interact with the environment for $T_{\text{on}}$ steps
  Store collected experiences to both online buffer $B_{\text{on}}$ and offline buffer $B_{\text{off}}$
  $S_{\text{on}} \leftarrow S_{\text{on}} + T_{\text{on}}$
  **for** $t < T_{\text{off}}$ **do**
    Sample a random value $p_s \sim \mathbb{U}(0,1)$
    **if** $p_s < p$ and $S_{\text{on}} > T_{\text{s}}$ **then**
      Sample a batch of $(\mathbf{s},\mathbf{a})$ from online buffer $B_{\text{on}}$
      Set the $\mathcal{W}(\mathbf{s},\mathbf{a})$ to 0
    **else**
      Sample a batch of $(\mathbf{s},\mathbf{a})$ from offline buffer $B_{\text{off}}$
      Set the $\mathcal{W}(\mathbf{s},\mathbf{a})$ to 1
    **end if**
    Update the Q-functions $Q_i, i \in N$ by Eq. 6
    Update the policy $\pi$ by Eq. 7
    $t \leftarrow t + 1$
  **end for**
**until** $S_{\text{on}} > S_T$
```

Specifically, C in Eq. 2 is the state-action value Q here, and we use the updating function in REDQ as the optimistic strategy. Since REDQ is an ensemble method which uses a set of Q-functions, we use i as the index of the Q-functions, and the size of the set is N. Hence,

$$\mathbb{A}(Q_i^k) = \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\text{OORB},\mathbf{a}'\sim\pi^k(\cdot|s')}\left[\left(Q_i^k(\mathbf{s},\mathbf{a}) - \mathcal{B}^\pi\hat{Q}^k(\mathbf{s}',\mathbf{a}')\right)^2\right], \tag{3}$$

where k is the times of iterative updates same as Eq. 2, and "$\sim$ OORB" represents sampling data from OORB. Operator $\mathcal{B}^\pi\hat{Q}^k(\mathbf{s}',\mathbf{a}')$ is

$$r + \gamma \min_{i\in\mathcal{M}} \hat{Q}_i^k(\mathbf{s}',\mathbf{a}'), \quad \mathbf{a}' \sim \pi^k(\cdot \mid \mathbf{s}'). \tag{4}$$

Here, π^k is the current policy, and $\hat{Q}$ denotes a target Q function for stabilizing the learning process (Mnih et al. 2015). Following REDQ's design, we randomly select two Q-functions from a set of them for ensemble, with $\mathcal{M}$ representing the set of selected indexes.

Then, we adopt the conservative regularizer in CQL as the second term in Eq. 2. For every Q-function in the set:

$$\mathbb{B}(Q_i^k) = \alpha \mathbb{E}_{\mathbf{s}\sim \text{OORB}} \left[\log \sum_{\mathbf{a}'} \exp(Q_i^k(\mathbf{s}, \mathbf{a}')) - \mathbb{E}_{\mathbf{a}\sim \text{OORB}}[Q_i^k(\mathbf{s}, \mathbf{a})] \right], \tag{5}$$

where action $\mathbf{a}'$ is sampled from the current policy, i.e., $\mathbf{a}' \sim \pi^k(\cdot|s)$. Hence, the overall updating function of Q is:

$$Q_i^{k+1} = \arg\min_{Q_i^k} \left\{ \mathbb{A}(Q_i^k) + \mathcal{W}(\mathbf{s}, \mathbf{a}) \mathbb{B}(Q_i^k) \right\} \tag{6}$$

Finally, the update function of policy is shown as follows:

$$\pi^{k+1} = \arg\max_{\pi^k} \mathbb{E}_{\mathbf{a}\sim \pi^k(\cdot|\mathbf{s})} \left[\mathbb{E}_{i\in N} \left[Q_i^k(\mathbf{s}, \mathbf{a}) \right] - \alpha \log \pi^k(\mathbf{a} \mid \mathbf{s}) \right]. \tag{7}$$

Policy-Based Implementation

Besides value-based methods, APL framework can also be easily implemented with policy-based methods. Here we take TD3BC (Fujimoto and Gu 2021) as an example, and name our algorithm Greedy-Conservative TD3BC (GCTD3BC). Specifically, TD3BC uses the policy update step in TD3, and adds a behavior cloning (BC) term to regularize the policy. Therefore, GCTD3BC takes the policy update function in TD3 as the optimistic term, and takes the BC regularizer as the pessimistic term. So, the policy update rule in GCTD3BC is

$$\pi^{k+1} = \underset{\pi^k}{\text{argmax}} \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim \text{OORB}} \left[\mathbb{A}(\pi^k) + \mathcal{W}(\mathbf{s}, \mathbf{a}) \mathbb{B}(\pi^k) \right], \tag{8}$$

where $\mathbb{A}(\pi^k) = \lambda Q^k(\mathbf{s}, \pi(\mathbf{s}))$ is the value function in TD3, and $\mathbb{B}(\pi^k) = -(\pi^k(\mathbf{s}) - \mathbf{a})^2$ is the BC regularizer. Hyper-parameter λ is used to adjust two terms.

We now introduce how $\mathcal{W}(\mathbf{s}, \mathbf{a})$ is set in both implementations. We use a very simple method: when data is sampled from the online buffer, $\mathcal{W}(\mathbf{s}, \mathbf{a})$ is set to 0, otherwise 1. This is why we use the term *Greedy-Conservative* to describe our algorithms. Specifically, we greedily exploit the near-on-policy data with the online RL scheme without any conservative regularizer. And we conservatively exploit the offline data by employing the offline RL methods. Formally, this strategy can be explained as below:

$$\mathcal{W}(\mathbf{s}, \mathbf{a}) \leftarrow \begin{cases} 0 & \text{if } (\mathbf{s}, \mathbf{a}) \sim \text{online buffer} \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

To be more clear, we summarize GCQL in Algorithm 1, and explain the main steps as follows. Firstly, we learn from the existing offline data for T_{initial} steps to leverage them. To make good use of the offline data, we usually set T_{initial} to a large value, e.g., $100K$. Secondly, we begin the following interleaving learning process. We conduct the online interaction for T_{on} steps and store newly collected experiences to OORB, and then we update the agent for T_{off} steps. For higher sample efficiency, we set the number of online interaction steps T_{on} to a small value, e.g., $1K$, and the number of updating steps T_{off} to a value larger than T_{on}, e.g., $10K$. If the batch of data for updating the policy and Q-functions comes from the online buffer, $\mathcal{W}(\mathbf{s}, \mathbf{a})$ is set to 0, otherwise 1.

Algorithm GCTD3BC is similar to Algorithm 1 except that GCTD3BC uses the action-value objective in TD3 (Fujimoto, van Hoof, and Meger 2018) and the policy updating rule is defined in Eq. 8.

Experiments

In this section, we design experiments to verify the effectiveness of our framework from three perspectives: (1) the performance compared with competitive baselines; (2) ablation studies to test the effect of key components used in our methods; (3) the influence of different hyper-parameters.

Tasks All experiments were done on the continuous control task set MuJoCo (Todorov, Erez, and Tassa 2012), and the offline dataset comes from the popular offline RL benchmark D4RL (Fu et al. 2020). Here, we test three tasks, i.e., walker2d, hopper and halfcheetah, and each task takes four different kinds of offline dataset, which are random-v0, medium-v0, medium-replay-v0 and medium-expert-v0.

Settings We set T_{on} in Algorithm 1 to $1K$. To better exploit the offline dataset, we set T_{initial} and T_{off} to $100K$ and $10K$, respectively. For OORB, we set $p = 0.5$ for GCQL and $p = 0.1$ for GCTD3BC, and T_{s} to $10K$ for both of them. The size of online and offline buffer is set to $20K$ and $3M$, respectively. We input S_T as $100K$. The above configurations keep the same across all tasks, datasets and methods.

Compared Methods We compare our algorithms, i.e., GCQL and GCTD3BC, with 8 competitive baselines, i.e., CQL, REDQ_ON, REDQ, TD3_ON, TD3BC, AWAC (Nair et al. 2020), OFF2ON (Lee et al. 2021) and IQL (Kostrikov, Nair, and Levine 2022). REDQ_ON denotes that we rerun the officially released codes by the authors (Chen et al. 2021a) without changing hyper-parameters, and do not use any offline data to pre-train for this online method. We use the "_ON" to indicates the agent is trained purely online without offline pre-training, and methods without "_ON" are in the offline-to-online setting. Similarly, TD3_ON is a pure online agent trained with TD3. For a fair comparison, we also include a re-implemented version in GCQL, i.e. REDQ, as a baseline, where the number of Q-functions is set to 5 for computational efficiency (as in GCQL) and pre-training with offline data is leveraged. The results of AWAC are directly taken from their paper. For OFF2ON, since the authors did not release the code for pre-training, we contact the authors and use pre-trained agents provided by them, and then use their codes for the rest of training. For other methods, results are obtained by rerunning their codes under the offline-to-online setting. We use default hyper-parameters for all methods without further tuning.

Overall Performance

We list scores in Table 1, and include all learning curves in Appendix. First, it is clear that our methods GCQL and GCTD3BC perform much better than baselines, well demonstrating the effectiveness of them. Second, we notice that our methods are more robust comparing with baselines. Specifically, offline RL methods (i.e., CQL and TD3BC) can obtain higher scores than pure online methods (i.e., REDQ_ON and TD3_ON) when the dataset is not random,

Environment	GCQL	GCTD3BC	CQL	REDQ_ON	REDQ	TD3_ON	TD3BC	AWAC	OFF2ON	IQL
walker2d-r	31±27	5±3	7±9	**71±11**	5±3	7±2	6±3	12	20±13	7±3
hopper-r	58±30	35±22	10±1	78±37	2±1	10±2	11±0	63	**81±21**	10±2
halfcheetah-r	**90±10**	69±8	46±4	59±2	32±1	39±1	35±3	53	85 ±3	28±7
walker2d-m	**94±6**	90±7	83±1	71±11	2±3	7±2	79±2	80	89±2	51±13
hopper-m	83±11	**99±3**	70±23	78±37	3±1	10±2	80±13	91	59 ±9	42±9
halfcheetah-m	**66±3**	62±2	25±8	59±2	46±1	39±1	43±1	41	58±2	40±0
walker2d-me	93±12	102±2	105±1	71±11	12±3	7±2	**110±3**	78	101±24	58±22
hopper-me	**110±1**	**110±2**	109±5	78±37	40±15	10±2	**110±0**	**112**	82 ±21	72±16
halfcheetah-me	**102±1**	**103±2**	92±2	59±2	9±3	39±1	98±2	41	100±1	38±17
walker2d-mr	**97±16**	90±9	57±5	71±11	13±2	7±2	60±5	-	71±32	30±13
hopper-mr	72±20	**87±11**	37±4	78±37	0±1	10±2	38±1	-	**60±23**	31±10
halfcheetah-mr	**59±2**	53±1	50±0	**59±2**	28±25	39±1	47±0	-	57±1	42±2
Total	**955±139**	905±71	691±63	624±200	192±59	224±20	717±33	-	863±152	449±114

Table 1: Averaged normalized score over last three evaluation iterations and 5 random seeds. ± captures the standard deviation over seeds. The highest performing scores are in bold. The results of AWAC are taken from their paper (Nair et al. 2020). REDQ_ON and TD3_ON are pure online methods without any offline pre-training. We rerun them using implementations from authors to ensure identical evaluation process. Suffix "-r" = random-v0 dataset, "-m" = medium-v0 dataset, "-me" = medium-expert-v0, and "-mr" = medium-replay-v0. All learning curves are showed in Appendix.

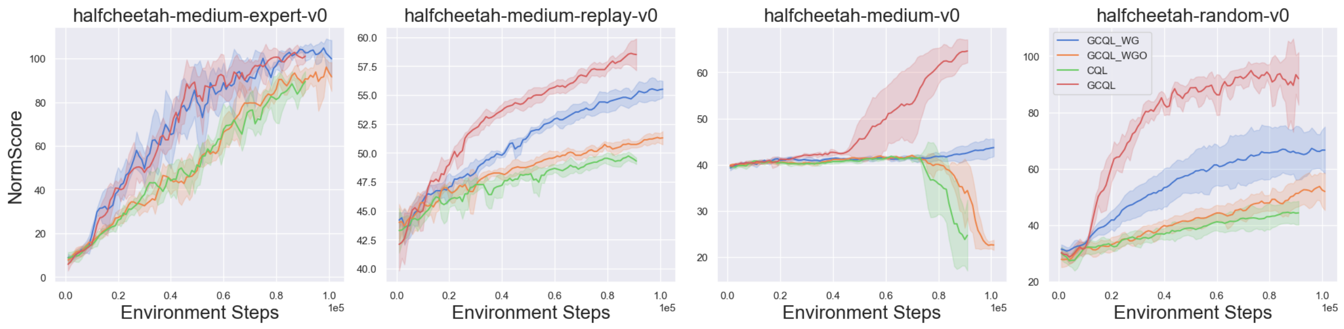

Figure 2: Ablation studies on Halfcheetah. GCQL_WG: GCQL without the greedy updating strategy. GCQL_WGO: GCQL without the greedy updating strategy and online buffer. Results on four tasks are averaged over three random seeds.

but they have poor performance when the dataset is random. In contrast, offline-to-online methods (e.g., GCQL and OFF2ON) can perform well regardless of random dataset or not. Note that OFF2ON is fine-tuned. For instance, the critic's network architecture is different from that in the original CQL paper. Our methods use default hyper-parameters in the original paper without fine-tuning, and can perform better than those fine-tuned methods. Third, it is surprising that single-agent method GCTD3BC have higher scores and faster learning speed than ensemble method GCQL on some tasks, such as hopper-medium-v0. All results shows APL can benefit more from the online and offline data and gain high sample efficiency. Also, the general APL framework can be successfully applied to the various RL methods.

Ablation Study on Key Components in APL

To investigate the effect of key components in our method, we conduct the following ablation studies on GCQL. The main differences between APL framework and prior offline RL methods is the adaptive update schema and online buffer in OORB. To this end, we design two variants of GCQL to investigate such two components' effect. **GCQL_WG**: We remove the adaptive update schema by fixing $\mathcal{W}$ to 1, resulting in the variant of GCQL **w**ithout **g**reedy updating strategy. **GCQL_WGO**: We further remove the online buffer, resulting in the variant of GCQL **w**ithout **g**reedy updating strategy and **o**nline buffer.

Learning curves of all four kinds of offline datasets for the task Halfcheetah are shown in Figure 2. It is easy to deduce that when the dataset includes the expert data, all variants perform well. The underlying reason is obvious as offline RL methods can perform quite well when data quality is good. However, for other datasets with worse quality (e.g., medium-replay, medium and random), online buffer and greedy update schema play an important role in boosting the performance and stabilizing the learning process. Results show that for these worse datasets, both GCQL_WG and GCQL_WGO have a clear performance drop compared with GCQL. Specifically, it is observed that, for medium datasets, both CQL and GCQL_WGO suffer a serious stability issue,

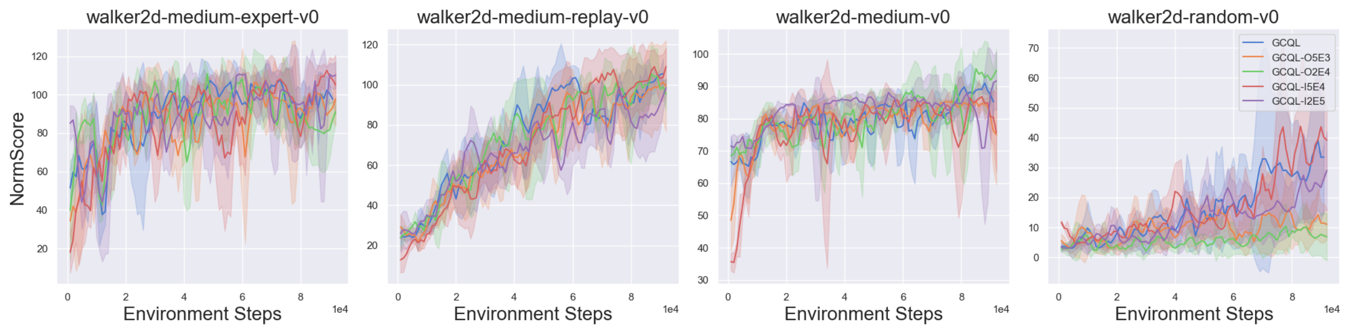

Figure 3: Learning curves for agents with different initial offline training steps T_{initial} and updating steps in each iteration T_{off}. GCQL-i2e5 and GCQL-i5e4 mean that T_{initial} is $2e5$ and $5e4$, respectively. And GCQL-o2e4 and GCQL-o5e3 mean T_{off} is set to corresponding values. Results are averaged over three random seeds.

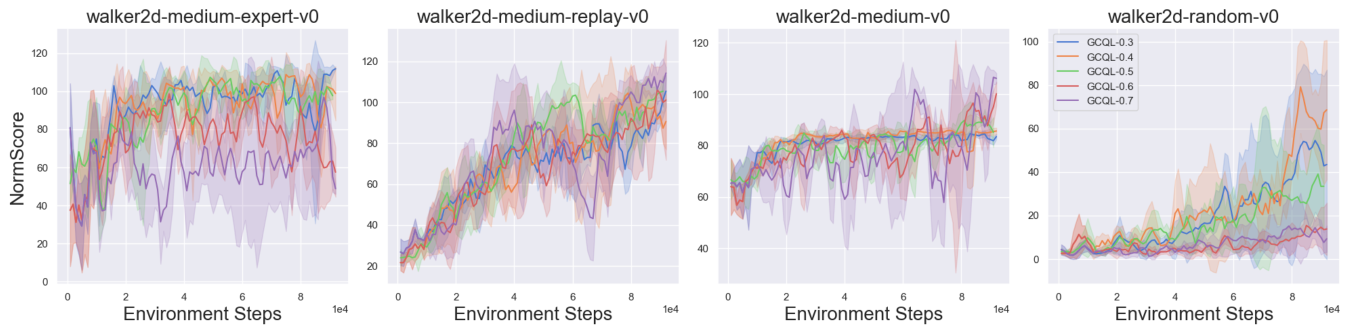

Figure 4: Learning curves for agents with different OORB threshold p. GCQL-X means that p is set to X. Results are averaged over three random seeds.

which verifies that the online buffer indeed stabilizes the learning process. Besides, GCQL_WG performs better than GCQL_WGO, which indicates that the online buffer can not only stabilize the learning process but also improve the performance. Moreover, GCQL still outperforms GCQL_WG by a large margin on the sub-optimal dataset, which verifies the efficiency of our greedy update scheme. In summary, greedy update schema and online buffer are both crucial for improving the sample efficiency in GCQL.

Analysis on Hyper-parameters

As we do not use careful fine-tuning techniques (e.g., grid search) for hyper-parameters in our methods, one may wonder how the hyper-parameters affect the performance. To this end, we conduct experiments to investigate their influence on GCQL. Here, we use the most complicated one among three tasks, i.e., walker2d, in this experiment, and we analyze initial offline training steps T_{initial}, updating steps in each iteration T_{off} and OORB threshold p in the section.

Learning curves for analyzing T_{initial} and T_{off} are shown in Figure 3. We try a larger and smaller value than the default one to test its impacts. Specifically, T_{initial} is tested with $2e5$ and $5e4$, and T_{off} is tested with $2e4$ and $5e3$. We can see that the performance of GCQL is insensitive to the T_{initial} and T_{off}, especially for datasets with not poor quality, e.g., datasets except for the random one. Besides, Figure 4 shows curves for analyzing p, which have bigger impact on the performance than T_{initial} and T_{off}. p is tested with 0.3, 0.4, 0.5, 0.6 and 0.7, and results show that methods with higher p perform better on the medium and medium-replay datasets, while those with lower p perform better on the other datasets. Particularly, only $p = 0.5$ can achieve a clear performance improvement in both the medium and random datasets. Overall, the default $p = 0.5$ is the most appropriate setting, suggesting that taking the online and offline data equally important may be the best option for GCQL in most cases.

Conclusion and Future Work

In this paper, we propose an Adaptive Policy Learning (APL) framework for offline-to-online reinforcement learning. In APL, the advantages of online and offline data are considered in an adaptive way, so that they are well-utilized for policy learning. Furthermore, a value-based and a policy-based algorithm are provided for implementing APL framework. Finally, we conduct comprehensive experiments and results demonstrate that our methods can obtain best sample efficiency in the offline-to-online setting comparing with several competitive baselines. In the future, we will continue to further improve the robustness of APL. For example, we will try to minimize the impact of offline dataset's quality on the performance. At last, we hope this work could bridge the gap between offline and online RL.

References

Agarwal, R.; Schuurmans, D.; and Norouzi, M. 2020. An optimistic perspective on offline reinforcement learning. In *ICML*.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Chen, X.; Wang, C.; Zhou, Z.; and Ross, K. W. 2021a. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. *ICLR*, abs/2101.05982.

Chen, X.-H.; Yu, Y.; Li, Q.; Luo, F.-M.; Qin, Z.; Shang, W.; and Ye, J. 2021b. Offline Model-based Adaptable Policy Learning. *Advances in Neural Information Processing Systems*, 34: 8432–8443.

Dann, C.; Neumann, G.; Peters, J.; et al. 2014. Policy evaluation with temporal differences: A survey and comparison. *Journal of Machine Learning Research*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

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

Fujimoto, S.; and Gu, S. S. 2021. A Minimalist Approach to Offline Reinforcement Learning. arXiv:2106.06860.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-Policy Deep Reinforcement Learning without Exploration. In *ICML*.

Fujimoto, S.; van Hoof, H.; and Meger, D. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*.

Ghosh, D.; Ajay, A.; Agrawal, P.; and Levine, S. 2022. Offline rl policies should be trained to be adaptive. In *International Conference on Machine Learning*, 7513–7530. PMLR.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*.

He, Q.; and Hou, X. 2020. POPO: Pessimistic Offline Policy Optimization. *ArXiv*, abs/2012.13682.

Jaques, N.; Ghandeharioun, A.; Shen, J. H.; Ferguson, C.; Lapedriza, A.; Jones, N.; Gu, S.; and Picard, R. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*.

Kidambi, R.; Rajeswaran, A.; Netrapalli, P.; and Joachims, T. 2020. MOReL : Model-Based Offline Reinforcement Learning. *NeurIPS*, abs/2005.05951.

Kim, B.; massoud Farahmand, A.; Pineau, J.; and Precup, D. 2013. Learning from Limited Demonstrations. In *NeurIPS*.

Kostrikov, I.; Nair, A.; and Levine, S. 2022. Offline reinforcement learning with implicit q-learning. *International Conference on Learning Representations*.

Kotsiantis, S. B.; Zaharakis, I.; Pintelas, P.; et al. 2007. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1): 3–24.

Kumar, A.; Fu, J.; Soh, M.; Tucker, G.; and Levine, S. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, 11761–11771.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative Q-Learning for Offline Reinforcement Learning. *NeurIPS*.

Lee, S.; Seo, Y.; Lee, K.; Abbeel, P.; and Shin, J. 2021. Offline-to-Online Reinforcement Learning via Balanced Replay and Pessimistic Q-Ensemble. *arXiv preprint arXiv:2107.00591*.

Levine, S.; Kumar, A.; Tucker, G.; and Fu, J. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *ArXiv*, abs/2005.01643.

Mandel, T.; Liu, Y.-E.; Levine, S.; Brunskill, E.; and Popovic, Z. 2014. Offline policy evaluation across representations with applications to educational games. In *AAMAS*, volume 1077.

Matsushima, T.; Furuta, H.; Matsuo, Y.; Nachum, O.; and Gu, S. 2021. Deployment-efficient reinforcement learning via model-based offline optimization. *ICLR*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Nair, A.; Dalal, M.; Gupta, A.; and Levine, S. 2020. Accelerating Online Reinforcement Learning with Offline Datasets. *CoRR*, abs/2006.09359.

Rajeswaran, A.; Kumar, V.; Gupta, A.; Vezzani, G.; Schulman, J.; Todorov, E.; and Levine, S. 2018. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *Robotics: Science and Systems*.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International conference on machine learning*, 1889–1897. PMLR.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R. S.; and Barto, A. G. 2011. Reinforcement learning: An introduction.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Vecerik, M.; Hester, T.; Scholz, J.; Wang, F.; Pietquin, O.; Piot, B.; Heess, N.; Rothörl, T.; Lampe, T.; and Riedmiller, M. 2017. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*.

Wu, Y.; Tucker, G.; and Nachum, O. 2019. Behavior Regularized Offline Reinforcement Learning. *ArXiv*, abs/1911.11361.

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J.; Levine, S.; Finn, C.; and Ma, T. 2020. MOPO: Model-based Offline Policy Optimization. *NeurIPS*, abs/2005.13239.

Zhu, H.; Gupta, A.; Rajeswaran, A.; Levine, S.; and Kumar, V. 2019. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, 3651–3657. IEEE.

Learning Curves

For better comparison, we compare ensemble GCQL with ensemble methods, such as OFF2ON or REDQ_ON and compare GCTD3BC with single-agent methods, such as TD3BC and TD3_ON.

Learning curves for GCQL

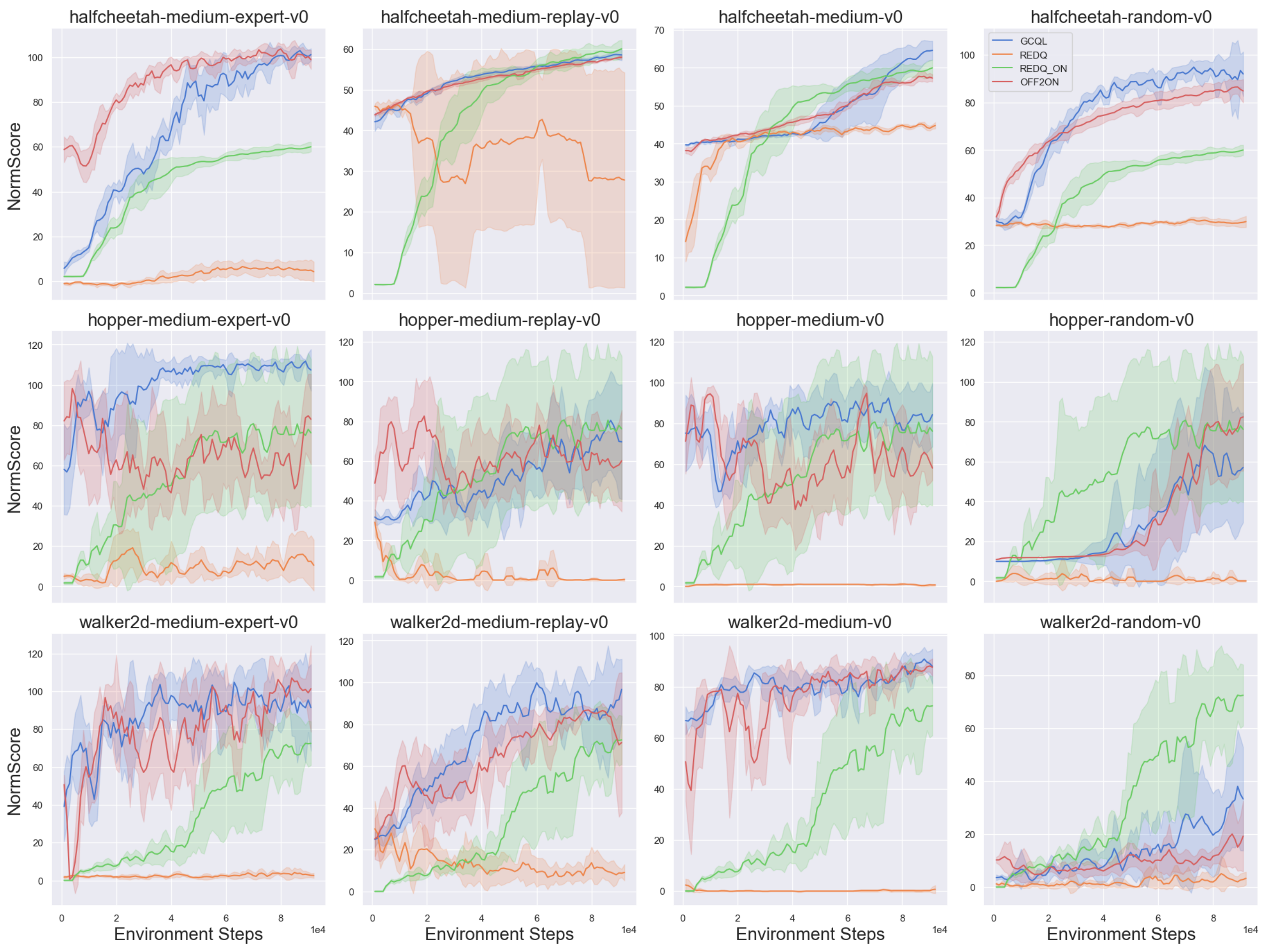

Figure 5: Learning curves for GCQL. The shaded areas represent the standard deviation across different seeds. REDQ_ON means the agent is trained using REDQ purely online without the offline pre-training, while other baselines use the offline-to-online setting.

Learning curves for GCTD3BC

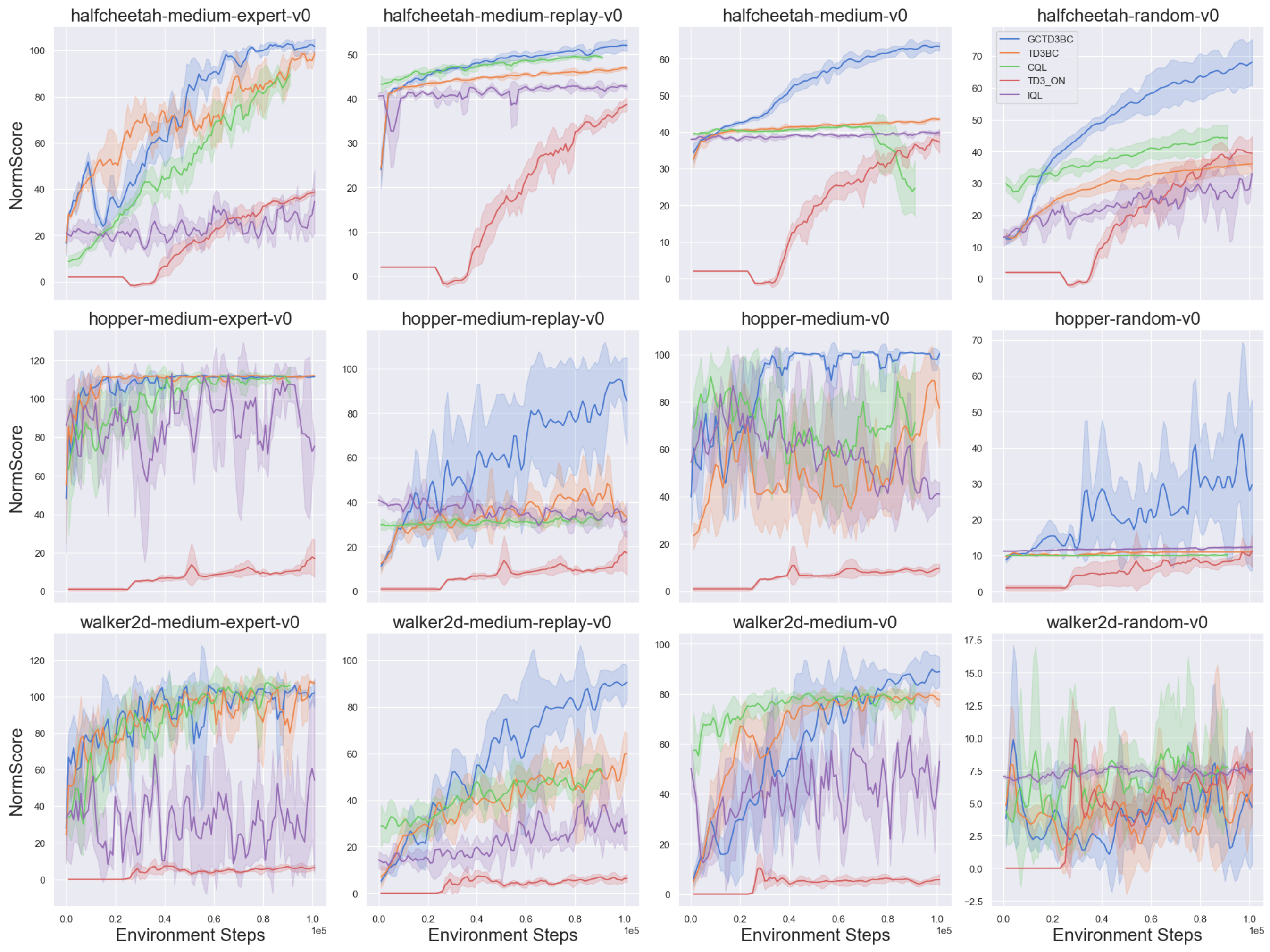

Figure 6: Learning curves for GCTD3BC. The shaded areas represent the standard deviation across different seeds. TD3_ ON means the agent is trained using TD3 purely online without the offline pre-training, while other baselines use the offline-to-online setting.