ADAPTIVE Q-LEARNING FOR INTERACTION-LIMITED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Conventional reinforcement learning (RL) needs an environment to collect fresh data, which is impractical when an online interaction is costly. Offline RL provides an alternative solution by directly learning from the logged dataset. However, it usually yields unsatisfactory performance due to a pessimistic update scheme or/and the low quality of logged datasets. Moreover, how to evaluate the policy under the offline setting is also a challenging problem. In this paper, we propose a unified framework called Adaptive Q-learning for effectively taking advantage of offline and online learning. Specifically, we explicitly consider the difference between the online and offline data and apply an adaptive update scheme accordingly, i.e., a pessimistic update strategy for the offline dataset and a greedy or no pessimistic update scheme for the online dataset. When combining both, we can apply very limited online exploration steps to achieve expert performance even when the offline dataset is poor, e.g., random dataset. Such a framework provides a unified way to mix the offline and online RL and gain the best of both worlds. To understand our framework better, we then provide an initialization following our framework. Extensive experiments are done to verify the effectiveness of our proposed method.

1 INTRODUCTION

Conventional online reinforcement learning (RL) methods (Haarnoja et al., 2018; Fujimoto et al., 2018) usually learn from experiences generated by interactions with the online 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. Recently, offline RL (Levine et al., 2020) has aroused much attention. It targets the above challenge by making the agent learn from an offline dataset collected by other policies in a purely data-driven manner. The difference between online RL and offline RL is shown in Figure 1.

Existing offline RL studies try to target the distribution mismatch or out-of-distribution actions issue by employing a pessimistic update scheme (Kumar et al., 2019; 2020) or in combination with imitation learning (Fujimoto et al., 2019). However, when the dataset is fixed, offline RL cannot learn the optimal policy (Kidambi et al., 2020), and even worse, when the dataset's quality is poor, offline RL usually gains a relatively poor performance (Kumar et al., 2020; Fu et al., 2020; Levine et al., 2020). On the other hand, it is challenging to evaluate the learned policy when learning totally from the offline dataset. Even though some research topics, e.g., off-policy evaluation (Dann et al., 2014), study how to evaluate the learned policy without the interaction with the environment, it is still not ideal for the practical purpose.

Some recent works try to address the above issues by employing an offline-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 issues to some extent, their main bottleneck is that they do not consider the different characteristics of offline and online data. For instance, pre-existing offline data can prevent agents from converging prematurely due to the potential diverse offline dataset, while online data can improve stability and accelerate convergence. Generally, these different data are mixed and used by a pessimistic strategy to update the policy in their methods, which may be problematic since using a pessimistic strategy for online data may harm the policy performance (Nair et al., 2020). Moreover,

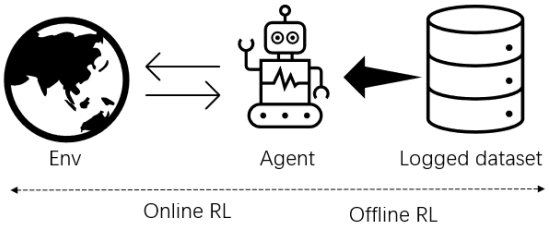


Figure 1: Online RL collect the data by interacting with the environment and they don't utilize the existing logged dataset while offline RL only exploit logged dataset without any future performance improvement. By contrast, we focus on obtaining the best of both worlds.

with sufficiently large and diverse offline data, a high-performing policy can be learned just using a pure online RL algorithm, while the online data (near-on-policy) also play a key role in improving the online RL's stability especially when dataset is not diverse (Fujimoto et al., 2019; Agarwal et al., 2020). Hence, we should take full advantage of both offline and online dataset.

To tackle the above problems, in this paper, we emphasize that *online and offline RL should be coupled organically*. First, a separate updating strategy should be employed for online and offline data, respectively, considering their different characteristics. To do so, we present a framework called adaptive Q-learning that integrates the advantage of offline learning and online learning effectively. When learning from the offline dataset, we conduct a pessimistic update strategy. In contrast, we use a greedy or non-pessimistic update strategy when learning from the online dataset. Second, we design a novel replay buffer to distinguish the offline from online datasets in a simple way. By utilizing such a novel framework and buffer design, the agent can achieve an expert policy using limited online interaction steps regardless of the quality of the offline dataset. In the experiments, our proposed framework can achieve better performance by using only one fifth number of interactions compared with the previous method (Nair et al., 2020).

Our contributions can be summarized as below:

- We propose a unified framework called adaptive Q-learning with limited online interaction steps that can effectively benefit from both the offline dataset and limited online data.
- Based on the general framework, we initialize a practical algorithm, called Greedy-conservative Q-ensemble learning (GCQL) that builds on top of state-of-the-art offline RL method CQL (Kumar et al., 2020) and online RL method REDQ (Chen et al., 2021)
- We empirically verify the effectiveness of our method by comprehensive experiments on the popular offline benchmark D4RL (Fu et al., 2020).

2 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 behaviour policies, they usually have a lower sample efficiency than the off-policy RL. On the other hand, off-policy methods (Fujimoto et al., 2018; Chen et al., 2021) enable the policy to learn from experience collected by history behavior polices, however, they cannot learn well from history trajectories collected by other agents' behavior policies (Fujimoto et al., 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 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 here, these methods tend to utilize a pessimistic updating strategy to relieve the distribution mismatch problem (Fujimoto et al., 2019; Kumar et al., 2019). To implement such a strategy,

model-free offline RL methods generally employ value or policy penalties to constrain the updated policy close to the data collecting policy (Wu et al., 2019; Kumar et al., 2020; Fujimoto et al., 2019; He & Hou, 2020). And 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). 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.

Above issues of online and offline RL motivate us to investigate the offline-online mixed setting.

Offline-online mixed RL Matsushima et al. (2021) propose a deployment-efficiency setting that also focuses on a practical setting: how many times the data-collection policy has been changed. They employ a KL-based trust-region optimization (Schulman et al., 2015) to update the policy. However, it ignores the offline dataset and focuses on the deployment times of the behavior policy instead of the interaction steps. Lee et al. (2021) aims at fine-tuning a pre-trained agent by online experiences and focus on the distribution shift issue. They employ a balanced experience replay buffer and select the near-on-policy samples from offline data. Even the samples are near-on-policy, they still apply a pessimistic strategy. Nair et al. (2020) propose an advantage-weighted actor-critic (AWAC) method that restricts the policy to stay close to the actions observed in the data by an implicit constraint. When online interaction is available, such constraints may have adverse effects on the performance. In contrast, our method explicitly considers the difference between offline and online learning and adaptive apply conservative or non-conservative update schemes. Some works (Zhu et al., 2019; Vecerik et al., 2017; Rajeswaran et al., 2018; Kim et al., 2013) also can learn from online interaction and offline dataset, however, they need the expert demonstrations instead of any dataset which limit their applicability.

3 PRELIMINARIES

In reinforcement learning, the interaction with environment usually assumed to be a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, p_M, r, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, and transition dynamics $p_M\left(s' \mid s, a\right)$. At each discrete time step, the agent receive a reward $r\left(s, a, s'\right) \in \mathbb{R}$ after conducting an action a in state s and arriving at next state s'. The agent's objective is to obtain the maximum return which is defined by $R_t = \sum_{i=t+1}^{\infty} \gamma^i r\left(s_i, a_i, s_{i+1}\right)$. The $\gamma \in [0, 1)$ is a discounted factor that down-weight the future rewards.

The policy can be deterministic, in which case it is usually denoted by μ: $a = \mu(s)$ or it may be stochastic, in which case it is usually denoted by π: $a \sim \pi(\cdot|s)$. Each policy π have a corresponding action-value function $Q^\pi(s, a) = \mathbb{E}_\pi\left[R_t \mid s, a\right]$, the expected return when following the policy after taking action a in state s. The policy π's action-value function can be updated by the Bellman operator $\mathcal{T}^\pi$:

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

When applying a greedy optimal strategy, i.e., $Q^*(s, a) = \max_\pi Q^\pi(s, a)$, it is known as the Q-learning that optimize the policy for maximizing the Q value (Sutton & Barto, 2011). Such a policy and Q function can be approximated with neural networks (Mnih et al., 2015). The above setting is typically off-policy or online RL as they need interaction with the environment to collect data. In principle, off-policy RL algorithms maintain a replay buffer $\mathcal{B}$ storing the exploration experiences collected by behavior policy.

Offline RL algorithms are based on the off-policy RL techniques but without the online interaction assumption (Levine et al., 2020). In another word, the replay buffer $\mathcal{B}$ size is fixed, and no further interaction with the environment under the offline RL setting. The data from the replay buffer can be collected by one or more behavior policies. Offline RL provides a way to free RL agents from data collection. The offline-online mixed setting focuses on pre-train a policy using the logged dataset and then fine-tuning this pre-trained policy through future online interaction (Nair et al., 2020; Lee et al., 2021).

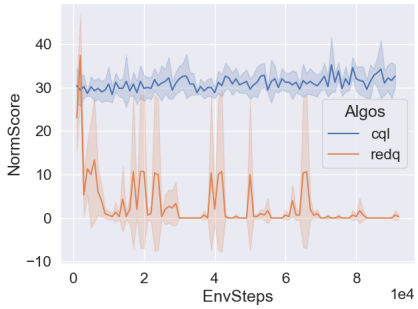


Figure 2: Learning curves on the D4RL (Fu et al., 2020) task Hopper-medium-replay-v0. The reported results are the averaged performance across five random seeds and the shaded areas represent the standard deviation across different seeds. The normalized score of 100 is the average returns of a domain-specific expert while normalized score of 0 corresponds to the average returns of an agent taking actions uniformly at random across the action space.

4 METHODOLOGY

In this section, we first give an illustrative example to explain our motivation. After that, we introduce the proposed general framework, namely Adaptive Q-learning, trying to couple online and offline RL in an organic way. Then, we present details about the Online-Offline Replay Buffer, targeting retaining and distinguishing online and offline data simultaneously. Finally, we implement the framework by incorporating SOTA online and offline RL algorithms and introduce the proposed Greedy-Conservative Q-ensemble Learning in detail.

4.1 AN ILLUSTRATIVE EXAMPLE

We test the SOAT offline RL method CQL (Kumar et al., 2020) and off-policy RL method REDQ (Chen et al., 2021) under the offline-online fine-tuning setting using one of the D4RL dataset hopper-medium-replay-v0 that consists of diverse experiences. Specifically, we firstly pre-train the agent with the offline dataset for 100000 steps, and then online explore 1000 steps. After that, we do offline update 10000 steps. We repeat such an interleaving process until the summed online exploration steps reach 90K.

The experiment results are showed in Figure 2. We can see both CQL and REDQ have troubles during training under this setting, even though both online and offline data are available here, which should be beneficial for policy learning. Particularly, during the initial offline training, they are both capable of learning, i.e., the starting point's performance is better than a random policy whose score is 0. Such result verified Agarwal et al. (2020)'s claim: when the dataset is diverse and large, online RL is capable of learning effectively from offline dataset. However, when an online interaction is available, the online REDQ suffers severe instability issue, and performance drops significantly. In contrast, the CQL agent maintains a stable learning curve but with a very limited performance improvement due to its conservative strategy. We expect offline RL or online RL can be capable of learning from the fresh on-policy data well, i.e., gain a score close to or better than the expert policy's score: 100. However, they only achieved a very limited performance improvement or failed to learn during the online interaction process. Such observation indicates that current online or offline RL methods may be ineffective for benefiting from hybrid learning.

4.2 ADAPTIVE Q-LEARNING FRAMEWORK

Now, we introduce our proposed framework. The underlying idea is simple. When the agent learns from near-on-policy samples, we choose a more greedy or no-pessimistic strategy. By contrast, when sampling from the offline dataset, we tend to a more pessimistic strategy (Jin et al., 2021). Through such an adaptive strategy, we can make full use of both online and offline data. Here, we take the value penalty as an example (Wu et al., 2019) to explain our framework. The learning objective of adaptive Q learning consists of two parts. One is the regular online Q objective presented by

$\mathbb{A}(Q)$,e.g., the bellman backup error. The other is the value penalty term presented by $\mathbb{B}(Q)$, e.g., the Q value regularizer. Then, an adaptive Q-learning objective can be defined by the following equation.

$$Q^{k+1} \leftarrow \arg\min_{Q^k} \mathcal{W}(s,a)\mathbb{B}(Q^k) + \mathbb{A}(Q^k) \tag{2}$$

The function $\mathcal{W}(s,a)$ weight the Q regularizer based on the sampled data. Specifically, when the sampled data is near-on-policy, we rely on the online Q objective more, which means the $\mathcal{W}(s,a)$ will choose a small value, e.g., close to 0. On the contrary, we will choose a bigger value for $\mathcal{W}(s,a)$ when the samples are from the offline buffer.

It is worth noting that we can easily adapt our framework to policy objective. The equation 2 can be adjusted to $\pi^{k+1} \leftarrow \arg\max_{\pi^k} \mathcal{W}(s,a)\mathbb{B}\left(\pi^k\right) + \mathbb{A}\left(\pi^k\right)$, where $\mathbb{A}(\pi^k)$ is the normal policy objective and $\mathbb{B}(\pi^k)$ is the policy penalty term. We apply such variant to TD3+BC (Fujimoto & Gu, 2021) which is a policy penalty method. The results indicates that our framework boost its performance by a large margin with limited online interaction. The details are presented in Appendix D.

Now, the key problem becomes how to define near-on-policy samples. To solve this issue, we introduce an online-offline replay buffer(OORB).

4.3 OORB: Online-Offline Replay Buffer

OORB aims to distinguish the limited online data from offline data. A similar tool can be found in Zheng et al. (2020b;a); Zhang et al. (2019) where they use a two-level replay buffer for boosting the learning of the off-policy agents. However, there is a major difference between ours and theirs: their methods are episode-based, and they select the episode with high episode return, while ours focuses on deciding whether the experiences are near-on-policy or not. In this sense, our OORB is much simpler and effective in our setting. Specifically, the online-offline replay buffer consists of two replay buffers. One is the online replay buffer that collects the online exploration experience. The other is the offline replay buffer consisting of the newly generated online exploration data and the logged dataset collected by any policies. To ensure the data in the online buffer is near-on-policy, we set its size to be much smaller than the size of the offline buffer. We store the fresh exploration experiences following a first-in-first-out rule. When new experiences arrive, the earliest saved experiences will be overridden.

To sample from OORB, we follow a Bernoulli distribution, sampling data from the online buffer with a probability p, and the offline buffer with $1-p$. To benefit from both online and offline dataset in a balanced way , we set p as 0.5 in our experiments. To verify other p's effect, we also conduct extra experiments to investigate the performance with different p in section 5.4. The experiment results indicate that $p = 0.5$ is the most suitable option in most cases, confirming our assumption: offline data and online data are both crucial for RL learning.

4.4 GCQL: Greedy-Conservative Q-ensemble Learning

For understanding our framework better, we provide a practical algorithm. According to equation 2, we need to define two parts in Q's update objective. The first is the pessimistic regularization term used in an offline learning setting, and the second is a normal online Q objective. For the first term, we can employ a penalized value function, e.g., a divergence function between distributions over actions (Wu et al., 2019), or the conservative objective used in Kumar et al. (2020). The second term is the normal Q-learning objective, such as the Bellman backup error. We present an initialization on top of the state-of-the-art offline RL agent CQL (Kumar et al., 2020), and the online RL agent REDQ (Chen et al., 2021), called Greedy conservative Q-ensemble learning (GCQL). Specifically, we replace the Q target function of CQL with the random-ensemble Q functions of REDQ, i.e., the randomized min Q values, and keep the rest of the objective of CQL($\mathcal{H}$). Formally, for each Q_i, the updated objective can be defined by following equation:

$$\begin{aligned} Q_i^{k+1} \leftarrow \arg\min_{Q_i^k} \mathcal{W}(\mathbf{s},\mathbf{a})\alpha\mathbb{E}_{\mathbf{s}\sim\mathcal{D}_{\text{OORB}}}\left[\log\sum_{\hat{\mathbf{a}}}\exp(Q(\mathbf{s},\hat{\mathbf{a}})) - \mathbb{E}_{\mathbf{a}\sim\mathcal{D}_{\text{OORB}}}[Q(\mathbf{s},\mathbf{a})]\right] \\ + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}_{\text{OORB}},\mathbf{a}'\sim\pi_\phi(\cdot|s')}\left[\left(Q_i^k(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^\pi\hat{Q}^k(\mathbf{s}',\mathbf{a}')\right)^2\right] \end{aligned} \tag{3}$$

Algorithm 1: Greedy-conservative Q-ensemble learning

Initialization:
Initialize policy π_ϕ, ensemble Q functions $Q_{\theta_i}, i \in N$,offline training steps t
Online exploration steps T_{on}, offline update steps T_{off}
Initialize $T_{initial}$, sample possibility p, start sampling steps:T_s
Initialize online buffer B_{on} to empty, offline buffer $B_{\text{off}} \leftarrow$ offline dataset
Initialize online buffer size $S_{\text{on}} \leftarrow 0$
Initial offline learning:
Train the agent for $T_{initial}$ steps using the logged dataset
while *True* **do**
  $t \leftarrow 0$
  Explore T_{on} steps online
  Store the T_{on} steps experiences to both online buffer B_{on} and offline buffer B_{off}
  $S_{on} \leftarrow S_{on} + T_{on}$
  for $t < T_{off}$ **do**
    Sample a random value $p_s \sim \mathbb{U}(0, 1)$
    if $p_s < p$ *and* $S_{on} > T_s$ **then**
      Sample a batch (s, a) from online buffer B_{on}
      Set the $\mathcal{W}(s, a)$ to 0
    end
    else
      Sample a batch (s, a) from offline buffer B_{off}
      Set the $\mathcal{W}(s, a)$ to 1
    end
    Update the Q functions by equation 3
    Update the policy by equation 5
    t += 1
  end
end

where the action $\dot{a}$ used in the first left Q function is sampled from current policy, i.e., $\dot{a} \sim \pi_\phi(\cdot|s), s \sim \mathcal{D}_{\text{OORB}}$, $\mathcal{D}_{\text{OORB}}$ is the OORB replay buffer, and $\hat{\mathcal{B}}^\pi \hat{Q}^k(\mathbf{s}, \mathbf{a})$ is defined by

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

We randomly select two Q functions from the ensemble Q functions and the $\mathcal{M}$ represents the selected Q functions' index just following the REDQ's setting. The $\hat{Q}$ stands for a target Q function for stabilizing the learning process (Mnih et al., 2015). The policy update is defined by

$$\pi_{\phi_{k+1}} \leftarrow \arg\max_{\pi_{\phi_k}} \mathbb{E}\left[\mathbb{E}_{i \in N}\left[Q_i(\mathbf{s}, \mathbf{a})\right] - \alpha \log \pi_{\phi_k}(\mathbf{a} \mid \mathbf{s})\right], \quad \mathbf{a} \sim \pi_{\phi_k}(\cdot \mid \mathbf{s}) \tag{5}$$

When sampled data is from the online replay buffer, we set $\mathcal{W}(s, a)$ to 0, otherwise 1, that is why we called it greedy-conservative. In another word, we greedy exploit the near-on-policy data by the normal online RL scheme without any conservative regularizer. On the contrary, we conservatively exploit the offline data by employing the offline RL regularizer. Formally, this strategy can be explained by following definition 6

$$\mathcal{W}(s, a) \leftarrow \begin{cases} 0 & \text{if } (s, a) \text{ are sampled from online replay buffer} \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

The practical algorithm is shown in Algorithm 1. Here, we also explain the main steps in the Algorithm as follows. Firstly, We first learn from the existing logged data for $T_{initial}$ steps to leverage offline dataset. To make good use of the offline data, we usually set a big value for $T_{initial}$, e.g., 100000 steps. Secondly, We begin the following interleaving learning steps. We conduct the online exploration process for T_{on} steps and store the new experiences to OORB. We set the online

Environment	GCQL(Ours)	CQL	REDQ	AWAC
walker2d-random	**53±27**	16±9	19±3	12
hopper-random	**84±40**	11±1	12±17	63
halfcheetah-random	**100±2**	46±4	34±1	53
walker2d-medium	**94±6**	83±1	5±3	80
hopper-medium	**105±1**	100±1	3±1	91
halfcheetah-medium	**66±3**	42±0	46±1	41
walker2d-expert	**117±2**	113±1	6±1	103
hopper-expert	**114±1**	113±1	12±6	112
halfcheetah-expert	**110±0**	108±1	1±0	106
walker2d-medium-expert	**117±4**	113±1	12±3	78
hopper-medium-expert	**115±1**	114±0	40±15	112
halfcheetah-medium-expert	**107±1**	95±2	9±3	41
walker2d-medium-replay	**114±6**	64±5	53±11	-
hopper-medium-replay	**96±9**	39±4	38±9	-
halfcheetah-medium-replay	**59±2**	50±0	50±1	-

Table 1: Performance of policies trained using 90K online interaction steps for GCQL, CQL and REDQ, and 500K online interaction for AWAC whose results are taken from Nair et al. (2020).± captures the standard deviation over seeds. The reported results are the average test performance across five random seeds in our experiments. The learning curves are showed in Appendix A.

exploration steps T_{on} to a small value, e.g., 1000. In contrast, the offline update step T_{off} is set to be larger than T_{on}, e.g., 10000. We sample a batch from our OORB and update our policy and Q functions. If the sampled batch comes from the online buffer, then we set the weight value $\mathcal{W}(s, a)$ to 0, otherwise 1. Repeat the above interleaving learning steps.

5 Experiments

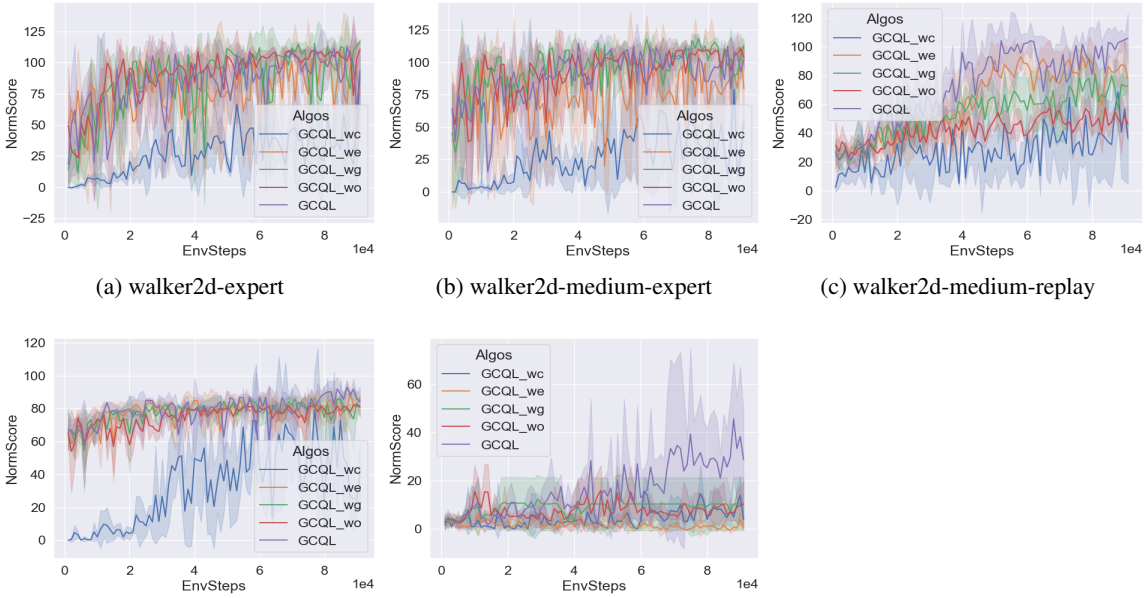
In this section, we design experiments to verify the effectiveness of our method from three perspectives: 1. the performance superiority compared with other baselines; 2. ablation studies to explain the effectiveness of each component used in our method. 3. the impact of different hyper-parameters.

5.1 Settings

All experiments were done on the continuous control benchmark: Mujoco (Todorov et al., 2012). The logged history dataset is the popular offline dataset D4RL (Fu et al., 2020). To make interaction limited, we set the online interaction step for each iteration to a small value, i.e., 1000. To better exploit the offline dataset, we set the offline training step to a large value, i.e., 100000 for T_{intial}, and 10000 for T_{off}. 1 million for entire offline-updates steps, and 90K for entire online exploration steps. As we take the online and offline datasets equally important, we set the sampling possibility from the online buffer as 0.5. We further set the size of the online buffer as 20000 and the size of the offline buffer as 3e6. We follow the default setting of all the baselines, except for the number of the ensemble Q, which is 5 in our setting. The above configurations are fixed for all tasks. As our main purpose is to present a new framework instead of gaining state-of-the-art performance, we do not fine-tune these hyper-parameters.

5.2 Overall performance

Table 1 shows the comparison results of Ours, CQL, AWAC, and REDQ. The AWAC's results are directly taken from their paper (Nair et al., 2020). We test on the continuous control tasks from D4RL, including random-v0, medium-v0, expert-v0, medium-expert-v0, and medium-replay-v0. As the online exploration is available, we take the maximization over testing scores during the whole training process, and report the average of these max scores across different seeds in Table 1. For a



(a) walker2d-expert (b) walker2d-medium-expert (c) walker2d-medium-replay

(d) walker2d-medium (e) walker2d-random

Figure 3: Ablation study on Walker2d task.

comprehensive understanding, we include the whole learning curves in Appendix A. The reported results are the averaged test performance across five random seeds against the online interaction steps. These learning curves demonstrate that our method also achieves a better final average score at the last iteration in most cases. As shown in Table 1, our method gains a better performance than the baselines in all tasks. Particularly, for the medium-replay and random dataset, our GCQL outperforms the baselines by a large margin. By contrast, CQL also achieves an expert performance on the high-quality dataset, e.g., expert and medium-expert tasks. For these high-quality datasets, our method achieves a comparable or slightly better performance over CQL. In terms of the online RL method: REDQ, it fails on almost all tasks. Even though we employ an online limited exploration process, conventional online RL cannot benefit from such limited experiences. On the contrary, such limited online experiences could bring catastrophic consequences, e.g., on the halfcheetah-medium-replay task, REDQ suffers a significant instability issue. On the other hand, when dataset quality is high, e.g., expert or medium-expert dataset, our method can leverage the benefit of the offline RL that gains an expert performance with very limited interaction steps, such as around 2e4 steps. By contrast, when dataset quality is poor, e.g., the random dataset or medium-replay. Our method can also take the advantage of the limited online experiences as much as possible. For instance, only our method can learn effectively with less than 1e5 online steps on random or medium-replay dataset. In sum, our method can benefit the most from the limited online experience while still maintain the offline learning's ability.

5.3 Ablation Studies

To estimate each component's contribution in our method, we conduct an ablation study on the walker2d. To do this, we design four variants of our method. **GCQL-we**: GCQL without the ensemble where we only use two Q functions same as the setting of CQL. **GCQL-wo**: GCQL without the online replay buffer where we only sample from the offline buffer. **GCQL-wg**: GCQL without the greedy where we fix the weighed $\mathcal{W}$ to 1. **GCQL-wc**: GCQL without the conservative term where we fix the $\mathcal{W}$ to 0.

These results are shown in figure 3, and we evaluate it on five datasets on walker2d across three random seeds. Firstly, from figure 3, it is easy to deduce that when the quality of data is high, i.e., including the expert dataset, almost all variants perform well except the GCQL-wc. The underlying reason is obvious as we firstly pre-train the policy by offline data. If no such conservative restriction,

then the pre-trained policy would suffer a serious distribution mismatch issue (Kumar et al. (2020)) and gain a poor pre-trained policy at last. By contrast, with the conservative scheme, the policy would obtain a relatively good pre-trained policy. That is why the starting point of GCQL-wc is much lower than other variants.

Secondly, when the dataset is diverse, i.e., the dataset is collected by different behavior policies and include data from different distributions, such as the medium-replay dataset, the greedy scheme (including the offline-online replay buffer and the greedy update strategy) plays an important role in such case while the ensemble feature seems have limited influence. For instance, the performance declined significantly for GCQL-wo and GCQL-wg on Walker2d-medium-replay.

Thirdly, when the dataset is not diverse, e.g., the medium dataset (data is collected by one medium policy), almost all tasks grow slowly except GCQL-wc. The underlying reason may be a lack of diversity, as Agarwal et al. (2020) declaimed diverse dataset plays a key role in RL. However, according to the learning curves, our method still has a clear performance improvement at the latter learning stage. For the random dataset, all variants fail due to the poor quality of the dataset. Instead, our method achieves a clear performance improvement under the random dataset, which shows the effectiveness of each component in this case.

5.4 Analysis on hyper-parameters

As we do not fine-tune the hyper-parameters in our experiments, one may wonder how the hyper-parameters affect the performance. To this end, we conduct experiments to discuss their influence. The detailed learning curves are shown in Appendix B. As one main characteristic of our method is limited interaction, we fix the online exploration step as 1000 for each iteration in the Mujoco benchmark. We then investigate the performance with the initial update steps $T_{initial}$, the offline update steps T_{off}, and the sample possibility from online buffer p. Specifically, the $T_{initial}$ was tested with 2e5 and 5e4, T_{off} was tested with 2e4 and 5e3 and p was tested with 0.3,0.4,0.5,0.6 and 0.7.

According to the evaluation results, we may conclude that our framework is insensitive to the $T_{initial}$ and T_{off}, especially when the dataset is not poor, e.g., not the random dataset. By contrast, p has a bigger impact on the performance. Particularly, higher p performs better than the lower ones on the medium and medium-replay tasks, while the lower ones perform better on the other tasks. That indicates that when dataset quality is very good or very bad, more offline update performs better. On the contrary, more online update is preferred for the medium-quality dataset. On the other hand, only the variant of p=0.5 can achieve a clear performance improvement in both the medium and random datasets. Overall, the default setting p=0.5 is the most suitable setting in terms of all tasks, which indicates that taking the online and offline data equally important may be the best option in most cases.

6 Conclusions

This paper first discusses the disadvantages of online RL and offline RL, and then the shortcomings of current offline-online mixed methods. After that, we propose a unified framework that can take advantage of online and offline learning effectively. Furthermore, a practical initialization that greedily exploits the online experiences and conservatively exploits the offline experiences is presented. We conduct comprehensive experiments to verify the effectiveness of our method. In terms of the shortcomings, although our framework can take advantage of the offline and online dataset under the interaction-limited setting, the quality of the offline dataset still has a big impact on the performance. For instance, the policy would converge to expert performance very soon when learning from the expert dataset. By contrast, our method is relatively slower when learning from the random dataset. Importantly, we also show that offline and online (near-on-policy) datasets are both crucial for RL learning. Thus, through a unified way, we can gain the best from both sides. At last, we hope this work could contribute to bridging the gap between the practice and DRL research.

REFERENCES

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *ICML*, 2020.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. *ICLR*, abs/2101.05982, 2021.

Christoph Dann, Gerhard Neumann, Jan Peters, et al. Policy evaluation with temporal differences: A survey and comparison. *Journal of Machine Learning Research*, 2014.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning, 2021.

Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

Scott Fujimoto, D. Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Qiang He and X. Hou. Popo: Pessimistic offline policy optimization. *ArXiv*, abs/2012.13682, 2020.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? *International Conference on Machine Learning*, 2021.

R. Kidambi, A. Rajeswaran, Praneeth Netrapalli, and T. Joachims. Morel : Model-based offline reinforcement learning. *NeurIPS*, abs/2005.05951, 2020.

Beomjoon Kim, Amir massoud Farahmand, Joelle Pineau, and Doina Precup. Learning from limited demonstrations. In *NeurIPS*, 2013.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pp. 11761–11771, 2019.

Aviral Kumar, Aurick Zhou, G. Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *NeurIPS*, 2020.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. *arXiv preprint arXiv:2107.00591*, 2021.

Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.

Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline policy evaluation across representations with applications to educational games. In *AAMAS*, volume 1077, 2014.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *ICLR*, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *CoRR*, abs/2006.09359, 2020.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *Robotics: Science and Systems*, 2018.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Y. Wu, G. Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *ArXiv*, abs/1911.11361, 2019.

Tianhe Yu, G. Thomas, Lantao Yu, S. Ermon, J. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *NeurIPS*, abs/2005.13239, 2020.

Zhizheng Zhang, Jiale Chen, Zhibo Chen, and Weiping Li. Asynchronous episodic deep deterministic policy gradient: Towards continuous control in computationally complex environments. *arXiv preprint arXiv:1903.00827*, 2019.

Han Zheng, Jingping Jiang, Pengfei Wei, Guodong Long, and C. Zhang. Competitive and cooperative heterogeneous deep reinforcement learning. In *AAMAS*, 2020a.

Han Zheng, Pengfei Wei, Jing Jiang, Guodong Long, Q. Lu, and C. Zhang. Cooperative heterogeneous deep reinforcement learning. *NeurIPS*, 2020b.

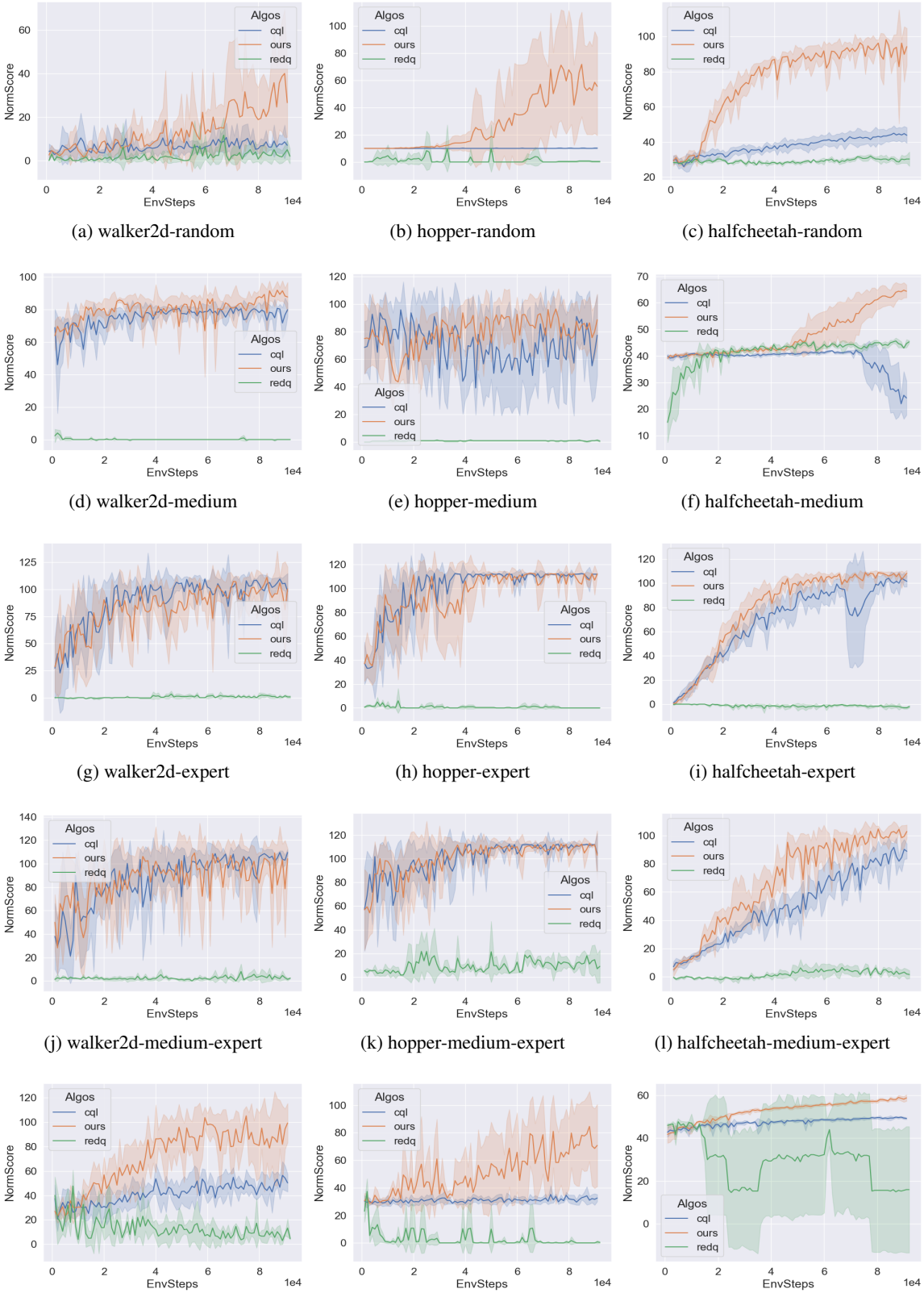
Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3651–3657. IEEE, 2019.

A LEARNING CURVES FOR ALL TASKS

Figure 4 indicate the whole learning curves for GCQL, REDQ and CQL.

B EXTRA EXPERIMENTS ON HYPER-PARAMETERS

Figure 5 and 6 shows the different hyper-parameters' setting on the performance.



(a) walker2d-random (b) hopper-random (c) halfcheetah-random

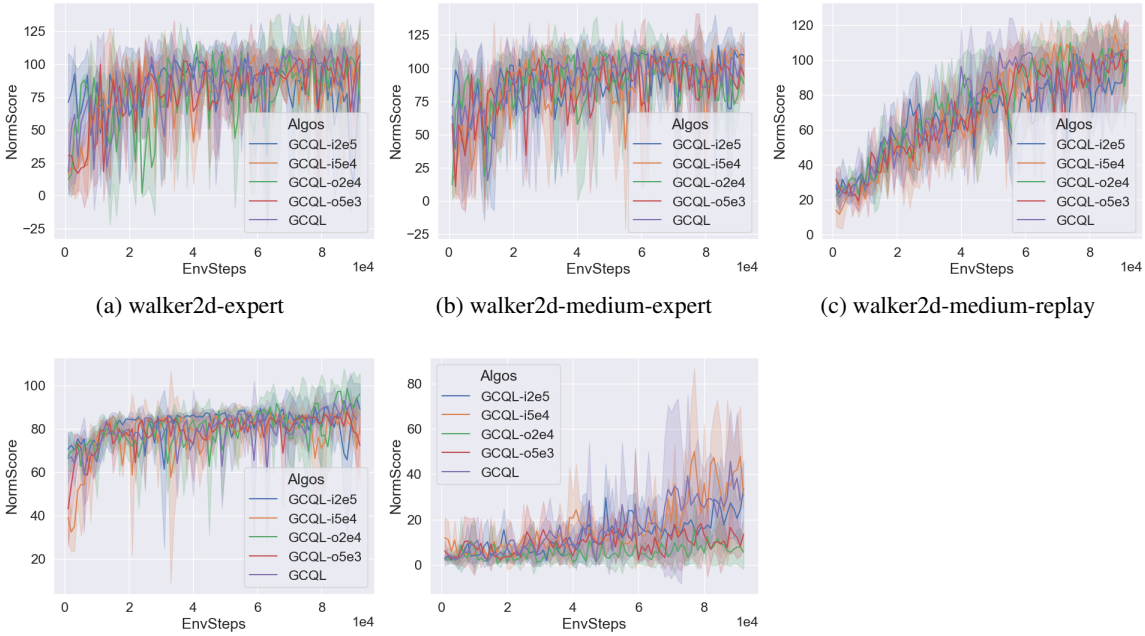
(d) walker2d-medium (e) hopper-medium (f) halfcheetah-medium

(g) walker2d-expert (h) hopper-expert (i) halfcheetah-expert

(j) walker2d-medium-expert (k) hopper-medium-expert (l) halfcheetah-medium-expert

(m) walker2d-medium-replay (n) hopper-medium-replay (o) halfcheetah-medium-replay

Figure 4: Training curves on D4RL continuous control benchmark across five random seeds. The shaded areas represent the standard deviation across different seeds.



(a) walker2d-expert (b) walker2d-medium-expert (c) walker2d-medium-replay

(d) walker2d-medium (e) walker2d-random

Figure 5: Different offline update steps on walker2d task across three random seeds.GCQL-i2e5:initial steps is 2e5, GCQL-i5e4: initial stesp is 5e4, GCQL-o2e4: offline update steps is 2e4, GCQL-o5e3: offline update steps is 5e3.

C EXTRA ABLATION STUDY ON HALFCHEETAH

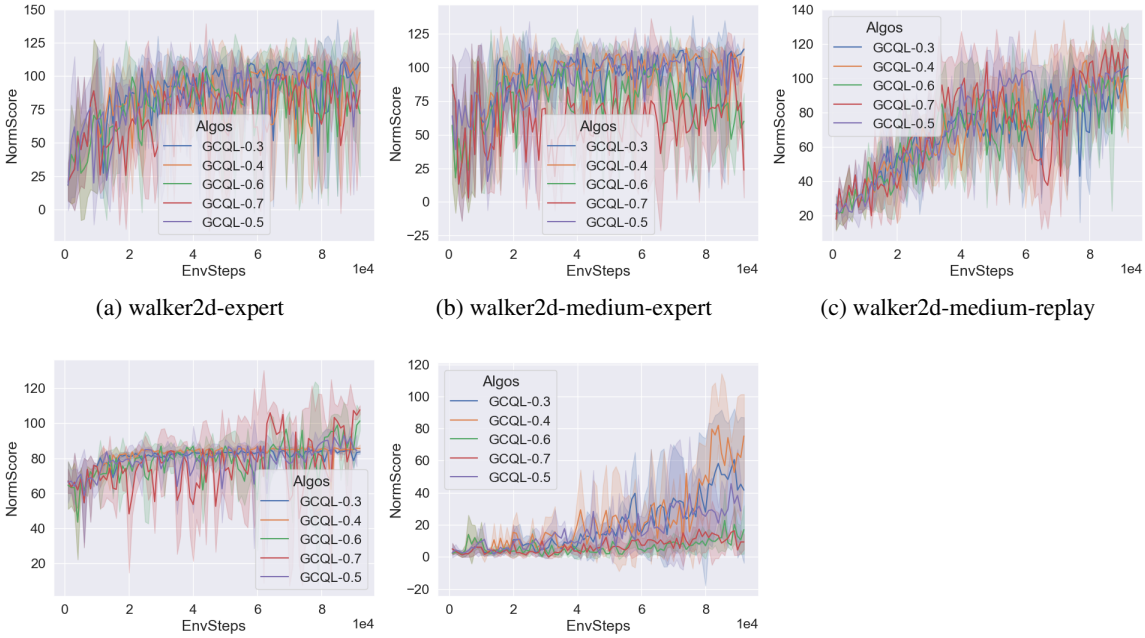
Figure 7 present extra ablation study on halfcheetah. GCQL-wg is the GCQL without the greedy update scheme but still with the online-offline two-level buffer, while GCQL-wgo is the GCQL without the greedy scheme and online-offline replay buffer. From Figure 7 c,d,e, it is clear that the greedy scheme plays an important role in boosting the performance when the dataset is not optimal or near-optimal. On the other hand, from Figure 7 d, we can see the online-offline replay buffer is crucial for stabilizing the learning process where the GCQL-wgo and CQL suffer serious stability issues which may be caused by extrapolation error (Fujimoto et al., 2019).

D EXTRA EXPERIMENTS ON TD3+BC

Extra experiments on TD3+BC (Fujimoto & Gu, 2021) (without states normalization) for the sub-optimal dataset, i.e., random, medium, and medium-replay, as the offline RL method can perform very well on expert or near-on-expert datasets. The learning curves indicate that our greedy-conservative framework can still improve policy-penalty-based methods by a large margin in most tasks. Moreover, it is very simple to apply our framework to TD3+BC where less than 20 lines codes are needed. We modify the policy objective from $\pi = \text{argmax}_\pi \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\lambda Q(s,\pi(s)) - (\pi(s)-a)^2\right]$ to $\pi = \text{argmax}_\pi \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\lambda Q(s,\pi(s)) - \mathcal{W}(s,a)(\pi(s)-a)^2\right]$, where $\mathcal{W}(s,a)$ follows our setting introduced in section 4.4.

E EXTRA EXPERIMENTS ON OFF2ON

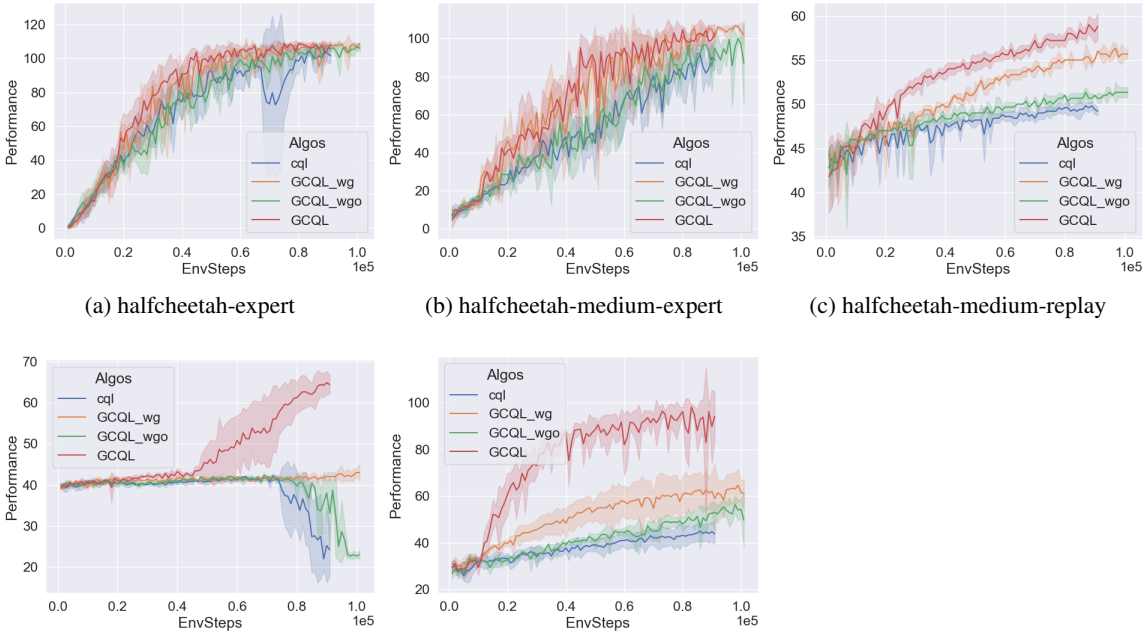
Here, we compared our method with off2on (Lee et al., 2021) which also employs an online update scheme to fine-tune a pre-trained agent. As the author did not provide the pre-training code for



(a) walker2d-expert (b) walker2d-medium-expert (c) walker2d-medium-replay

(d) walker2d-medium (e) walker2d-random

Figure 6: Different possibility setting on walker2d task across three random seeds.GCQL-X means the sample possibility from online buffer is X.

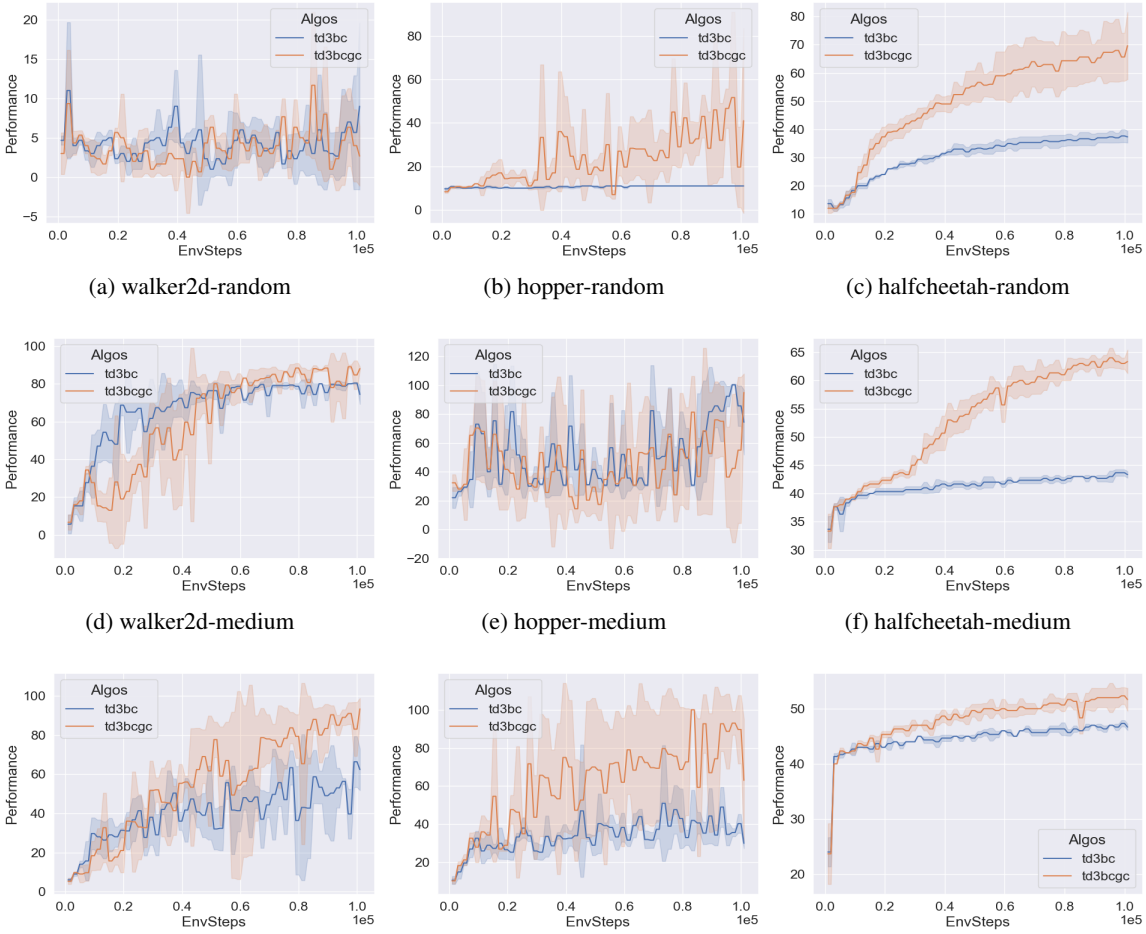


(a) halfcheetah-expert (b) halfcheetah-medium-expert (c) halfcheetah-medium-replay

(d) halfcheetah-medium (e) halfcheetah-random

Figure 7: Extra ablation study on Halfcheetah.

CQL, we use the pre-trained CQL agents provided by off2on's author for online fine-tuning. All the hyper-parameters follow the default setting except the online interaction steps. Figure 9 indicated the whole learning curves. From this figure, we can see GCQL has a comparable or better performance



(a) walker2d-random (b) hopper-random (c) halfcheetah-random

(d) walker2d-medium (e) hopper-medium (f) halfcheetah-medium

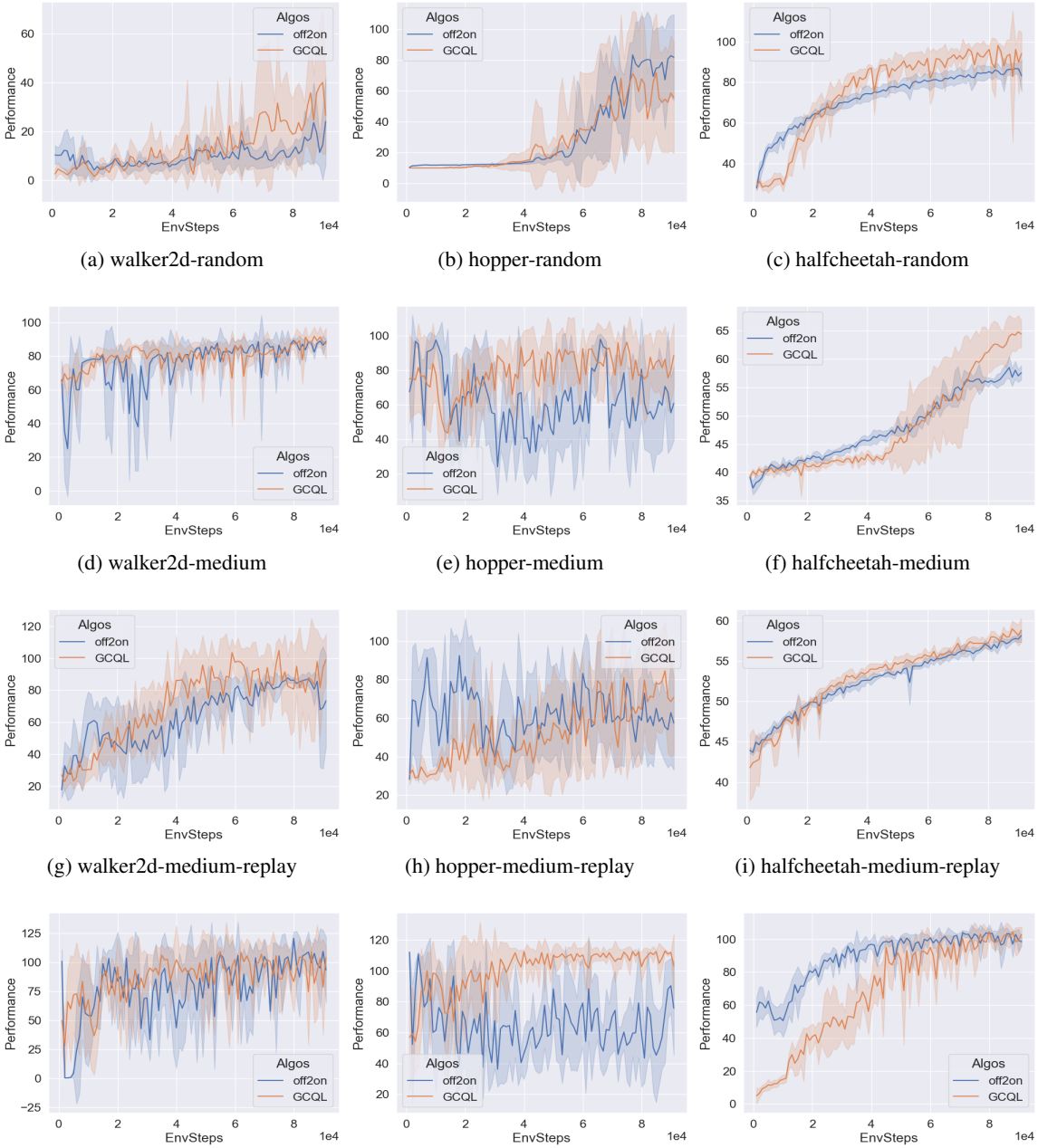
(g) walker2d-medium-replay (h) hopper-medium-replay (i) halfcheetah-medium-replay

Figure 8: Training curves for TD3+BC on D4RL continuous control benchmark across three random seeds on tasks: random-v0, medium-v0 and medium-replay-v0. td3bc means the algorithm introduced by (Fujimoto & Gu, 2021) while td3bcgc is the variant with our greedy-conservative framework and online-buffer replay buffer.

than off2on. It is worth noting that off2on fine-tuning their method, for instance, the critic's neural network architecture is different from the original CQL paper. In contrast, we did not fine-tune these parameters, and all settings are the same for all tasks.

F EXTRA EXPERIMENTS ON OFFICIAL ONLINE REDQ

In this section, we conduct extra experiments on the online REDQ (Chen et al., 2021). To guarantee the reproduction performance, we use the official code from the author, and all hyper-parameters are following its default setting, e.g., the number of Q is 10, and the utd-ratio is 20. In contrast, the number of Q in GCQL is 5, and utd-ratio is 10. REDQ-online updates its policy following a conventional online RL setting. From Figure 10, it is clear that GCQL achieves a better or comparable performance than REDQ-online except for the walker2d-random dataset. Specifically, when the dataset is good, e.g., including the expert dataset, GCQL can outperform the REDQ-online by a large margin. On the other hand, when the offline dataset is of poor quality, i.e., random-dataset,



(a) walker2d-random (b) hopper-random (c) halfcheetah-random

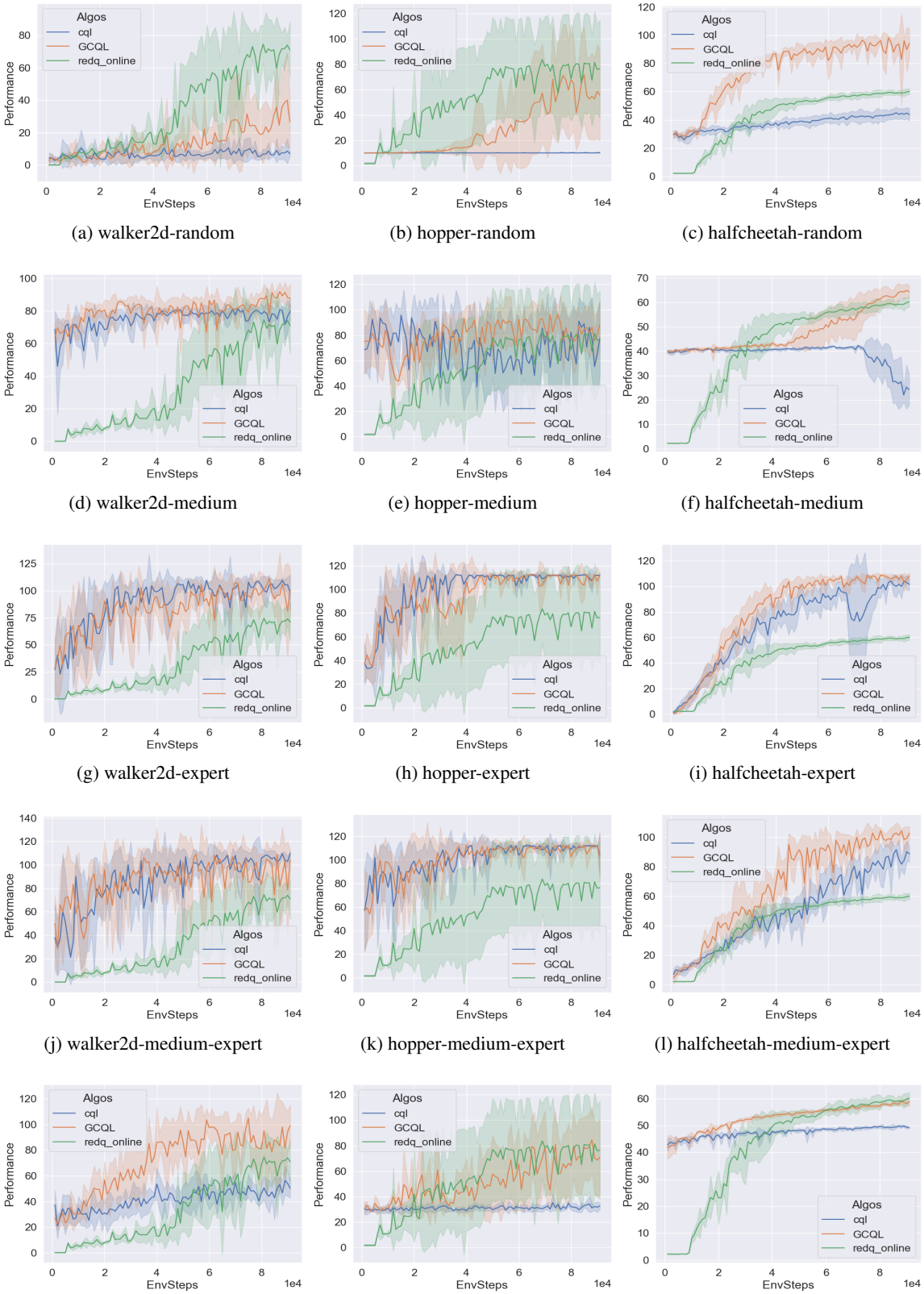
(d) walker2d-medium (e) hopper-medium (f) halfcheetah-medium

(g) walker2d-medium-replay (h) hopper-medium-replay (i) halfcheetah-medium-replay

(j) walker2d-medium-expert (k) hopper-medium-expert (l) halfcheetah-medium-expert

Figure 9: Training curves for off2on (Lee et al., 2021) on D4RL continuous control benchmark across four random seeds on tasks: random-v0, medium-v0,medium-expert-v0 and medium-replay-v0.

GCQL can still learn a comparable or better policy than REDQ-online in two of three tasks, i.e., the hopper-random and halfcheetah-random.



(a) walker2d-random (b) hopper-random (c) halfcheetah-random

(d) walker2d-medium (e) hopper-medium (f) halfcheetah-medium

(g) walker2d-expert (h) hopper-expert (i) halfcheetah-expert

(j) walker2d-medium-expert (k) hopper-medium-expert (l) halfcheetah-medium-expert

(m) walker2d-medium-replay (n) hopper-medium-replay (o) halfcheetah-medium-replay

Figure 10: REDQ-online learn from scratch without the offline training while GCQL and CQL learn from both offline dataset and online interaction.