PRETRAINING A SHARED Q-NETWORK FOR DATA EFFICIENT OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent breakthroughs in supervised learning domains such as computer vision and natural language processing follow the consistent paradigm: pretrain a neural network with a large dataset and fine-tune it onto downstream tasks with a relatively small dataset. Offline reinforcement learning (RL) can be an alternative approach for learning the best policy with the static dataset in sequential decision-making problems, akin to supervised learning. Following the paradigm, previous works have focused on constructing a large dataset or pretraining networks with the static dataset and fine-tuning them with online interactions. However, it is still vague that offline RL can exhibit data efficiency, e.g. robustness to static dataset size. In this paper, we propose a simple yet effective plug-and-play method that pretrains a Q-network under an offline RL scheme, improving task performance and data efficiency. Our method consists of two core functionalities: Transforming the Q-network structure to a shared network architecture and pretraining weights of the shared network by a supervised regression task that predicts the forward dynamics of a task. We provide an analysis of how our method enables improved performance even in a small dataset in terms of the projected Bellman equation. We also empirically demonstrate that the proposed method improves the performance of existing popular offline RL methods on the D4RL and Robomimic benchmarks with an average improvement of 135.94% on the D4RL benchmark. Moreover, we demonstrate the proposed method boosts data efficiency in offline RL with varying data collection strategies.

1 INTRODUCTION

In the deep learning fields including computer vision (Chen et al., 2020; Saharia et al., 2022; Ramesh et al., 2022; Meng et al., 2023), and natural language processing (Radford et al., 2018; Devlin et al., 2019; Brown et al., 2020; Radford et al., 2021), a dominant paradigm has been widely used to boost the performance of deep learning algorithms: pretraining a scalable large model with a large dataset and fine-tuning them to a specific task. In contrast, typical reinforcement learning (RL) considers an online learning nature that involves iterative processes between experience collections and policy improvements through online interactions with the environment (Sutton et al., 1998). Unfortunately, online interaction is impractical in several cases since data collection requires expensive costs and retains potential risks of the agent, e.g. hardware corruption. Offline RL provides a solution by avoiding online interactions with the environment (Levine et al., 2020).

Offline RL aims to learn a policy from pre-collected data from an unknown behavior policy without further interactions with the environment. Recent research has focused on a large dataset and a scalable neural network model under an offline RL scheme, following the paradigm of the supervised learning domain (Chebotar et al., 2023; Padalkar et al., 2023; Team et al., 2024). On the other hand, pretraining with offline RL and fine-tuning with online RL has been investigated to improve sample efficiency of online interactions (Nakamoto et al., 2024b; Xie et al., 2021; Rafailov et al., 2023b; Ball et al., 2023b). Besides, pieces of work have focused on the dataset itself, e.g. an imbalanced dataset, unlabeled data, and even data corruption under offline RL scheme (Hong et al., 2023; Yu et al., 2022; Yang et al., 2023).

However, we denote that there is not enough work to improve data efficiency in offline RL; for instance, robustness on dataset size or guaranteed performance even in a small dataset. Investigation into data-efficient offline RL is necessary since collecting experience charges expensive costs and unfavorable exploration in the real world, hampering the possibility of offline RL in the real world. Furthermore, numerous algorithms often depend on the large dataset to verify their superior performance while the amount of required data for brilliant performance totally depends on a benchmarking dataset. In this work, we propose a simple yet effective plug-and-play method that pretrains a Q-network under an offline RL scheme, improving task performance and data efficiency. To this end, we devise a shared Q-network structure that outputs the predicted next state and Q-value as illustrated in Figure 1. The proposed approach consists of two phases: pretraining the Q-network with forward dynamics and training existing offline RL algorithm.

We empirically demonstrate that the proposed method improves the performance of existing popular offline RL methods on the D4RL (Fu et al., 2020), and Robomimic (Mandlekar et al., 2021), benchmarks with an average improvement of 135.94% on the D4RL benchmark. We demonstrate that our data-efficient method maintains the performance with fragments of the dataset across the data quality of the optimality on the D4RL dataset. Moreover, We investigate our method across the data collection strategies on the ExoRL datasets (Yarats et al., 2022), assuming a small dataset would have more narrow state-action coverage than a large dataset. As a result, we demonstrate that our method improves the performance even in a narrow data distribution coverage. Additionally, we provide an analysis of how our method enables improved performance even in a small dataset, concerning the projected Bellman equation.

2 RELATED WORKS

Offline RL. Offline RL aims to learn a policy with static data without further interactions with the environment. Previous approaches have mainly addressed the distribution shift problem, which is caused from the idea that queries of the Q-function over out-of-distribution actions may yield overly optimistic values during offline training (Fujimoto et al., 2019; Kumar et al., 2019; Levine et al., 2020; Kumar et al., 2020; Fujimoto & Gu, 2021; Kostrikov et al., 2021a). Recently, scalability to a large dataset and neural network model has been studied (Chebotar et al., 2023; Padalkar et al., 2023; Team et al., 2024). On the other hand, pretraining with offline RL and fine-tuning with online RL improves sample efficiency in online interaction (Nakamoto et al., 2024b; Xie et al., 2021; Rafailov et al., 2023b; Ball et al., 2023b). In contrast, dissimilar experiments over the way to consuming the static dataset have been conducted, e.g. an imbalanced dataset, unlabeled data, and even data corruption under an offline RL scheme (Hong et al., 2023; Yu et al., 2022; Yang et al., 2023). However, we focus on the data efficiency in offline RL (i.e. robustness on small datasets). In this work, we propose a simple yet effective plug-and-play method for pretraining Q-network to overcome the data efficiency problem.

Sample efficient RL. A common issue in most RL algorithms is sample efficiency: excessive interactions with the environment are required to learn an optimal policy. For this reason, sample efficiency has been an active research topic in RL. Model-based RL, (Sutton, 1991; Deisenroth & Rasmussen, 2011; Hafner et al., 2019b;a; Hansen et al., 2022), is a common approach to resolve sample inefficiency by learning a (latent) dynamics model and using it to generate additional transition samples. Otherwise, effective pretraining, (Schwarzer et al., 2021; Yarats et al., 2021b), and data augmentation, (Laskin et al., 2020; Kostrikov et al., 2021b), play a critical role in improving sample efficiency in RL. In recent, offline-to-online, (Lee et al., 2022; Ball et al., 2023a; Rafailov et al., 2023a; Feng et al., 2023; Nakamoto et al., 2024a), and foundation model, (Ahn et al., 2022; Seo et al., 2022; Brohan et al., 2023b;a; Bhateja et al., 2023), have tackled this problem where the poor data efficiency of online RL regime is alleviated by leveraging large offline data. In contrast, we define the data efficiency problem in offline RL as the ability of an offline RL algorithm whether an agent can learn the desired policy even with a small dataset.

3 MARKOV DECISION PROCESS

We consider the Markov decision process, where the agent sequentially takes actions to maximize cumulative discounted rewards. In a Markov decision process with the state-space $\mathcal{S} := \{1, 2, \ldots, |\mathcal{S}|\}$ and action-space $\mathcal{A} := \{1, 2, \ldots, |\mathcal{A}|\}$, the decision maker selects an action $a \in \mathcal{A}$ at the current state $s \in \mathcal{S}$, then the state transits to the next state $s' \in \mathcal{S}$ with probability $P(s'|s, a)$, and the transition incurs a reward $r(s, a, s') \in \mathbb{R}$, where $P(s'|s, a)$ is the state transition probability from the current state $s \in \mathcal{S}$ to the next state $s' \in \mathcal{S}$ under action $a \in \mathcal{A}$, and $r(s, a, s')$ is the reward function. For convenience, we consider a deterministic reward function and simply write $r(s_k, a_k, s_{k+1}) =: r_k, k \in \{0, 1, \ldots\}$.

A deterministic policy, $\pi : \mathcal{S} \to \mathcal{A}$, maps a state $s \in \mathcal{S}$ to an action $\pi(s) \in \mathcal{A}$. The objective of the Markov decision problem is to find a deterministic (or stochastic) optimal policy, π^*, such that the cumulative discounted rewards over infinite time horizons is maximized, i.e.,

$$\pi^* := \arg\max_{\pi \in \Theta} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_k \middle| \pi \right],$$

where $\gamma \in [0, 1)$ is the discount factor, Θ is the set of all deterministic policies, $(s_0, a_0, s_1, a_1, \ldots)$ is a state-action trajectory generated by the Markov chain under policy π, and $\mathbb{E}[\cdot|\pi]$ is an expectation conditioned on the policy π. Moreover, Q-function under policy π is defined as

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_k \middle| s_0 = s, a_0 = a, \pi \right], \quad (s, a) \in \mathcal{S} \times \mathcal{A}.$$

4 PRETRAINING Q-NETWORK WITH FORWARD DYNAMICS HELPS IMPROVING DATA EFFICIENCY

Algorithm 1 Pretraining Q-network scheme for Offline RL

```
Input: Dataset $\mathcal{D}$ of transition $(s, a, s')$, Learning rate $\alpha$
Initialize parameters $\varphi, \psi$
for each gradient step do
    Sample a mini-batch $\mathcal{B} \sim \mathcal{D}$
    Compute the forward dynamics estimation error
        $\mathcal{L}_{pre}(\varphi, \psi) = \sum_{(s,a,s') \in \mathcal{B}} (s' - (g_\psi \circ h_\varphi)(s, a))^2$
    Update weights of the shared network and forward network
        $\varphi \leftarrow \varphi - \alpha \nabla_\varphi \mathcal{L}_{pre}(\varphi, \psi), \quad \psi \leftarrow \psi - \alpha \nabla_\psi \mathcal{L}_{pre}(\varphi, \psi)$
end for
Output: Pretrained weights $\varphi$ of the shared network
```

In this paper, we propose a simple yet effective pretraining method adapting features of forward dynamics into the initialization of Q-network to improve data efficiency in offline RL. To this end, we design Q-network that partially shares a network with the forward dynamics estimation model. In particular, the forward model is constructed as follows:

$$\hat{s}' = (g_\psi \circ h_\varphi)(s, a), \quad (s, a) \in \mathcal{S} \times \mathcal{A}, \tag{1}$$

where $\hat{s}'$ is the estimated next state, g_ψ is a parameterized linear function, and h_φ is shared with the Q-network, which is defined as

$$Q_{\varphi,\theta}(s, a) = (f_\theta \circ h_\varphi)(s, a), \quad (s, a) \in \mathcal{S} \times \mathcal{A}, \tag{2}$$

where f_θ is also a parameterized linear function that represents the linear output layer and h_φ represents the fully connected neural network layers shared with the forward model in equation 1. The overall structures of the neural networks are illustrated in Figure 1.

In the proposed method, the forward model $g_\psi \circ h_\varphi$ is pretrained by minimizing the mean squared prediction error loss function

$$\mathcal{L}_{pre}(\varphi, \psi) = \sum_{(s,a,s') \in \mathcal{D}} (s' - (g_\psi \circ h_\varphi)(s, a))^2 \tag{3}$$

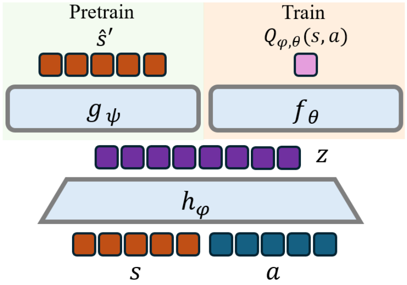


Figure 1: **Overview of our pretraining method.** Our method splits the original Q-network into two core architectures: a shared network that extracts the representation z from the concatenated vector of state s and action a and separated heads for training the forward model network and Q-network, respectively.

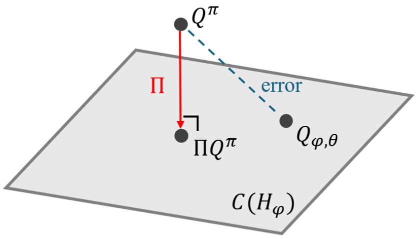


Figure 2: **Reduced approximation error with the expanded column space of H_φ.** In linear approximation, there exists Q^π outside of the column space of H_φ. To deal with this problem, the projected Bellman equation projects Q^π to ΠQ^π which exists in the column space of H_φ.

over the pre-collected dataset $\mathcal{D}$ which includes a given set of the transition (s, a, s'). Afterwards, the pretrained parameter φ can be used as an initial or fixed parameter for standard RL algorithms based on the Q-network structure in equation 4 without any modification. The overall pretraining process is summarized in Algorithm 1 for offline RL. We also note that similar principles can be applied for online RL as well, and the corresponding algorithm is given in Appendix A.

Later in this paper, we empirically demonstrate that combining the proposed pretraining method with existing offline RL methods can effectively improve their performances. Moreover, we demonstrate that our method indeed improves data efficiency through some experiment settings in offline RL.

4.1 ANALYSIS: BASED ON THE PROJECTED BELLMAN EQUATION

In this section, underlying insights behind the proposed method are discussed. For simplicity and convenience of presentation, we assume that the state and action spaces are discrete and finite, and the transition is deterministic. Our analysis is based on the observation that Q-function with neural networks can be generally represented by equation 2. Defining the feature vector $z = h_\varphi(s, a) \in \mathbb{R}^m$, it can be rewritten as

$$Q_{\varphi,\theta}(s,a) = \sum_{i=1}^{m} \theta_i h_{\varphi,i}(s,a) = \langle \theta, h_\varphi(s,a) \rangle, \quad (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{4}$$

When φ is fixed, then the above structure can be viewed as a linear function approximation with the feature function $h_{\varphi,i}$. In the proposed method, $h_{\varphi,i}$ is indeed pretrained by minimizing the loss in equation 3 and then fixed while learning Q-function in equation 4. Therefore, the interpretation based on the linear function approximation is expected to be a reasonable model to explain the phenomenon in the proposed method.

It is well-known that with linear function approximation, the corresponding standard Bellman equation

$$Q_{\varphi,\theta}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P^\pi(s'|s,a) \sum_{a' \in A} Q_{\varphi,\theta}(s',a')$$

may not admit a solution in general. However, typical TD-learning algorithms are known to converge to the unique fixed point of the projected Bellman equation. In particular, considering the vector form of the Bellman equation, $Q_{\varphi,\theta} = R + \gamma P^\pi Q_{\varphi,\theta}$, the projected Bellman equation (Melo & Ribeiro (2007)) is known to admit a solution

$$Q_{\varphi,\theta} = \Pi(R + \gamma P^\pi Q_{\varphi,\theta})$$

Table 1: **The Rank of the latent space of Q-network on the D4RL benchmark.** We compare the rank of the latent space between a vanilla TD3+BC and TD3+BC adapted with our method over 512 samples. As a result, adapting our method significantly increases the rank of the latent space, leading to reduced approximation error.

	Halfcheatah		Hopper		Walker2d	
	TD3+BC	TD3+BC (+ours)	TD3+BC	TD3+BC (+ours)	TD3+BC	TD3+BC (+ours)
Random	59	**236**	69	**192**	72	**82**
Medium	55	**249**	85	**227**	55	**254**
Medium Replay	49	**252**	77	**249**	77	**255**
Medium Expert	58	**236**	86	**232**	52	**253**
Expert	44	**198**	104	**198**	68	**225**

where Π is the projection onto the column space, $C(H_\varphi)$, of the feature matrix H_φ defined as

$$H_\varphi := \begin{bmatrix} \vdots \\ h_\varphi(s,a)^T \\ \vdots \end{bmatrix}.$$

The corresponding solution is known to have the error bound

$$||Q_{\varphi,\theta} - Q^\pi||_\infty \leq \frac{1}{1-\gamma}||\Pi Q^\pi - Q^\pi||_\infty, \tag{5}$$

where Q^π is the true Q-function corresponding to the target policy π. As can be seen from the above bound, the error depends on the feature matrix H_φ. We can observe that the smaller the distance between $C(H_\varphi)$ and Q^π, the smaller the error between $Q_{\varphi,\theta}$ and Q^π. Therefore, a proper choice of the feature function is key to the successful estimation of Q^π.

With the neural network function approximation, typical value-based RL algorithms update both φ and θ simultaneously via TD-learning algorithms. Since the feature functions, $h_{\varphi,i}$, are in general nonlinear and non-convex in φ, it may sometimes converge to a local optimal solution. This in turn implies that appropriate initialization or pretraining of the feature functions, $h_{\varphi,i}$, can play an important role for estimating Q-function with smaller approximation errors on the right-hand side of equation 5 by avoiding suboptimal local solutions.

We conjecture that the pretraining approach with the forward model introduced in the previous section can effectively shape the feature functions so that the column space $C(H_\varphi)$ can cover higher dimensional vector space in $\mathbb{R}^{|S\times A|}$. As shown in Figure 2, this eventually results in a reduction of the solution error on the right-hand side of equation 5. To support this, we empirically compare the rank of the Q-network in the latent space between a vanilla TD3+BC and the pretrained TD3+BC with our method over 512 data samples. As a result, adapting our method shows significantly higher rank than the rank of the vanilla method. We suggest that the proposed method tends to expand $C(H_\varphi)$ and increases the probability of reducing the approximation error on the right-hand side of equation 5, leading to more precise Q-function estimation.

5 EXPERIMENTS

In this section, we evaluate our method over existing offline RL methods with the popular offline RL benchmarks, D4RL, and more complex domain, Robomimic. Furthermore, we examine the proposed method over the partial fragments of D4RL and ExoRL datasets for data-efficient offline RL. We introduce a detailed experimental setup and baselines in the following paragraphs and provide empirical results subsequently.

Experimental setup. We have considered heterogeneous tasks and diverse datasets for precise comparisons. For the locomotion task, the proposed method is compared with existing methods in the popular D4RL benchmark (Fu et al., 2020). Three different embodied agents and five distinct datasets are considered in order to validate the effectiveness of the proposed method: *HalfCheetah, Hopper,*

Table 2: **Averaged normalized scores on the D4RL benchmark over 5 seeds.** In each column corresponding to different RL methods, values on the left-hand side are scores of the baseline methods directly taken from the literature. The values on the right-hand side of each column represent scores of the proposed methods combined with the baselines. The increased scores compared to the baselines are highlighted in blue font, and they are reported with the mean and standard deviations over five random seeds.

		AWAC	CQL	IQL	TD3+BC
Random	HalfCheetah	2.2→51.10±0.89	21.7±0.9→31.94±2.63	→18.28±1.02	10.2±1.3→14.83±0.54
	Hopper	9.6→59.47±33.79	10.7±0.1→30.20±2.66	→10.67±0.41	11.0±0.1→31.56±0.16
	Walker2d	5.1→13.11±3.91	2.7±1.2→19.56±4.49	→8.88±0.71	1.4±1.6→11.23±5.05
Medium	HalfCheetah	37.4→54.63±1.45	37.2±0.3→39.93±18.84	47.4→48.85±0.16	42.8±0.3→49.17±0.26
	Hopper	72.0→101.73±0.20	44.2±10.8→90.58±2.23	66.4→78.62±2.21	99.5±1.0→71.52±2.16
	Walker2d	30.1→89.51±0.88	57.5±8.3→84.66±0.67	78.3→83.63±1.14	79.7±1.8→87.09±0.60
Medium Replay	HalfCheetah	→55.75±1.30	41.9±1.1→47.60±0.37	44.2→45.48±0.17	43.3±0.5→45.84±0.26
	Hopper	→106.67±0.59	28.6±0.9→98.63±2.12	94.7→99.43±1.71	31.4±3.0→100.16±1.60
	Walker2d	→100.31±2.11	15.8±2.6→87.66±1.30	73.9→87.95±1.68	25.2±5.1→92.01±1.58
Medium Expert	HalfCheetah	36.8→90.05±1.89	27.1±3.9→82.75±6.51	86.7→95.25±0.14	97.9±4.4→96.89±0.92
	Hopper	80.9→113.23±0.22	111.4±1.2→111.06±0.81	91.5→105.77±11.31	112.2±0.2→113.02±0.19
	Walker2d	42.7→111.88±0.28	68.1±13.1→91.63±42.48	109.6→112.09±0.93	101.1±9.3→111.58±0.35
Expert	HalfCheetah	78.5→93.48±0.11	82.4±7.4→97.09±1.03	→97.40±0.13	105.7±1.9→98.86±0.55
	Hopper	85.2→112.86±0.10	111.2±2.1→112.10±0.35	→113.34±0.46	112.2±0.2→113.35±0.28
	Walker2d	57.0→111.22±0.35	103.8±7.6→110.64±0.28	→112.80±1.08	105.7±2.7→111.00±0.15
Total		→1265.01±48.07	764.3±61.5→1136.03±86.78	→1118.46±23.25	979.3±33.4→1148.12±14.65

Walker2d for agents and *random, medium-replay, medium, medium-expert, expert* for datasets. For the tabletop manipulation tasks, we have evaluated the proposed method in the Robomimic benchmark, (Mandlekar et al., 2021), where off-the-shelf offline RL methods are already implemented. Two different tabletop tasks and mixed-quality datasets are considered to verify the scalability of the proposed method: *Lift, Can* for tasks and *Machine-Generated (MG)* for datasets. For data-efficient offline RL, we have evaluated the proposed method across the optimal quality of the datasets of D4RL *Gym locomotion tasks*, and the dataset collection strategies for *walker walk (i.e. SMM, RND, ICM)* and *point mass maze (i.e. Proto, Diayn)* in ExoRL (Yarats et al., 2022). See Appendix C for a more detailed setup for tasks and datasets.

Baselines. We have designed extensive experiments on the D4RL benchmark to verify the effectiveness of the proposed method built on top of the popular offline RL methods, including AWAC (Nair et al., 2020), CQL (Kumar et al., 2020), TD3+BC (Fujimoto & Gu, 2021), and IQL (Kostrikov et al., 2021a). To verify the benefits of the proposed method, we compared the normalized scores between the vanilla method and the one combined with the proposed pretraining method. Similar to the D4RL benchmark, the success rate are compared on the Robomimic benchmark, where IQL, TD3+BC, BCQ (Fujimoto et al., 2019), and IRIS (Mandlekar et al., 2020), were used in combination with the proposed methods. On the ExoRL benchmark, we used TD3 (Fujimoto et al., 2018), for *walker walk* task, and CQL for *point mass maze* tasks. See Appendix E for more implementation details.

5.1 D4RL

The normalized scores between the vanilla and the one combined with our method are compared in Table 2 for each environment and dataset, where the scores of the baselines were taken directly from the literature. One can observe that the proposed method combined with the baselines improves the corresponding the original methods, achieving an average improvement of $\mathbf{135.94}\%$, across diverse environments and datasets. Specifically, one can observe that all methods including AWAC ($+306.45\%$), CQL ($+132.77\%$), IQL ($+9.21\%$), and TD3+BC ($+95.34\%$) exhibit significantly increased performance on average compared to the results reported in the original papers. We have taken all normalized scores of TD3+BC, AWAC, and CQL from the reported scores in (Fujimoto & Gu, 2021). In addition, we have borrowed the score of IQL from (Kostrikov et al., 2021a).

Furthermore, the proposed method demonstrates outstanding performance, achieving an average improvement of $\mathbf{222.39}\%$, on mixed-quality data regimes, which are mostly desirable in offline RL: AWAC ($+530.67$), CQL ($+191.85$), IQL ($+9.20$), and TD3+BC ($+157.83\%$) on average for *random, medium and medium-replay* datasets. The results suggest that the proposed method effectively

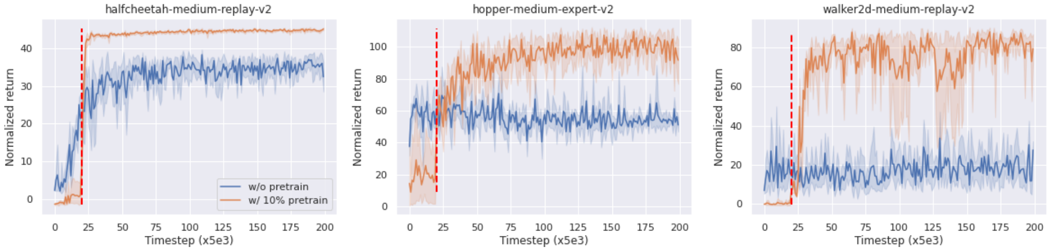


Figure 3: **Learning curves of TD3+BC.** The blue and orange curves are, respectively, the normalized scores of TD3+BC and TD3+BC pretrained with the proposed method. The vertical red reference lines split the pretraining and main training phases. After the pretraining phase, TD3+BC combined with the proposed method quickly outperforms the vanilla TD3+BC by a large margin.

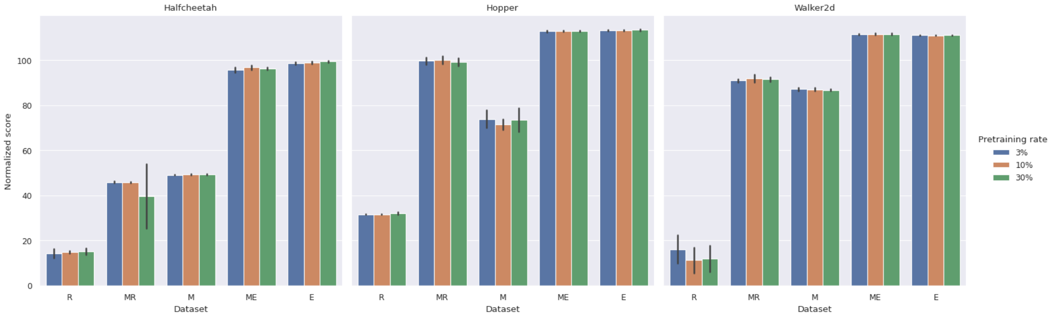


Figure 4: **Averaged normalized scores across pretraining rates.** R, M, MR, ME, and E represent random, medium, medium replay, medium expert, and expert datasets on the D4RL benchmark, respectively.

adopts beneficial features to avoid suboptimal local solutions, enabling further improvements without imposing any modification of the original methods.

The learning curves of TD3+BC are illustrated in Figure 3 to verify the effectiveness of the proposed method. After the pretraining period (indicated by the red vertical lines), one can notice that the learning curves rapidly increase and achieve higher returns compared to the original methods. These results suggest that our method accelerates training and enhances performance with only a few lines of modifications on top of the baselines. Full graphs of TD3+BC are provided on Figure 11 in Appendix F.

We also applied our method with different pretraining ratios (i.e., 3%, 30%) on TD3+BC over 5 seeds. The results are presented on Figure 4 and Table 9 in Appendix H. Notably, regardless of the pretraining ratio, the proposed method demonstrates improved performance over different pretraining rates. Overall, the pretraining ratio of 3% yields a slightly higher total sum of averaged scores while the results of the 10% ratio yield the lowest standard deviation.

5.2 ROBOMIMIC

Additional experiments are conducted on large-scale robotic manipulation tasks to verify the effectiveness of the proposed method for complex tasks. The proposed method is evaluated with tasks containing suboptimal transitions, where the proposed method improves the baselines on the D4RL benchmark. The averaged success rate of four offline RL baselines is reported in Figure 5 with and without applying the proposed method. As can be seen, all the methods with the proposed pretraining method are improved over the baselines in seven out of eight cases. Therefore, we conclude that the proposed method also effectively performs in solving more complex tasks. We also have conducted experiments on Adroit, 24-DOF environment, in Appendix D. The results also demonstrate that the proposed method is effective in solving complex tasks.

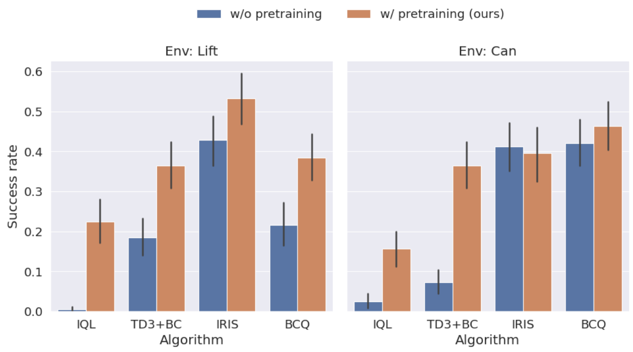


Figure 5: **Averaged success rate on the Robomimic benchmark.** We evaluate both vanilla methods without pretraining (blue) and methods with pretraining (orange). 7 out of 8 cases depict notably improved performance in both environments.

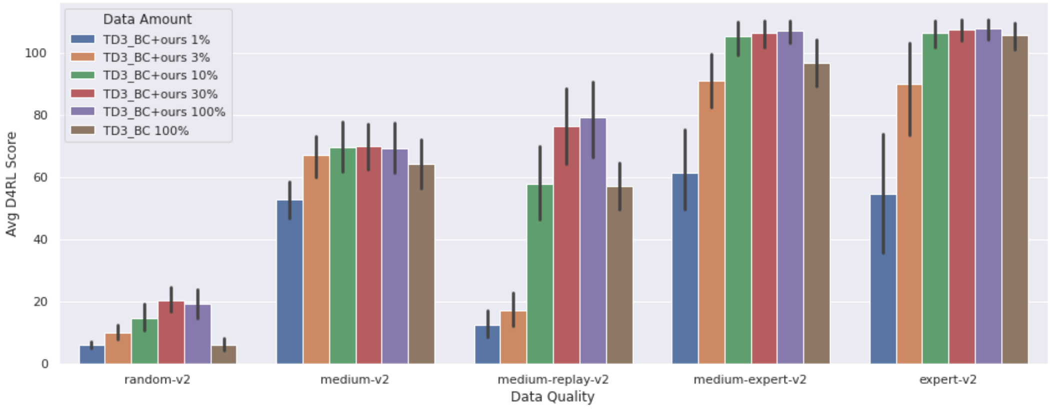


Figure 6: **Averaged normalized scores in reduced datasets across data quality.** This figure shows that overall performance of our method across reduced dataset sizes *(i.e., 1%, 3%, 10%, 30%, 100%)* for three environments *(i.e., halfcheetah, hopper, walker2d)* in D4RL. From the overall results, we conclude that our method guarantees better performance even in 10% of the datasets regardless of the data quality of the dataset, and even 1% for the *random* datasets and 3% for the *medium* datasets.

5.3 DATA EFFICIENCY ACROSS THE OPTIMAL QUALITY OF THE DATASETS

To validate that the proposed method is indeed data efficient regardless of the dataset quality, we have examined the proposed method with TD3+BC in reduced datasets *(i.e., 1%, 3%, 10%, 30%, 100% of each dataset)* across the data quality *(i.e., random, medium, medium replay, medium expert, expert)* on D4RL over 5 seeds. To construct the reduced datasets, we uniformly have sampled the transition segments (*i.e.,* (s, a, r, s')) from the each dataset. On the *random* datasets (a leftmost section in Figure 12), training with the proposed method with only 1% of the dataset outperforms the vanilla TD3+BC trained with full datasets at *halfcheetah* and *warker2d* environments. On the *medium* datasets (right to the *random* in Figure 12), the proposed method shows similar or improved results compared to the vanilla TD3+BC with full datasets by only using 3% size of the datasets. On other datasets (i.e. *medium-replay*, *medium-expert*, and *expert*), training the proposed method with 10% datasets totally outperforms the vanilla TD3+BC with full datasets. From the overall results in Figure 6, we conclude that our method guarantees better performance even in 10% of the datasets regardless of the data quality of the dataset.

5.4 DATA EFFICIENCY ACROSS THE DATASET COLLECTION STRATEGIES

We have assumed that a typical small dataset would have more narrow state-action coverage than a large dataset. Therefore, we have considered a goal-reaching offline agent in a maze environment

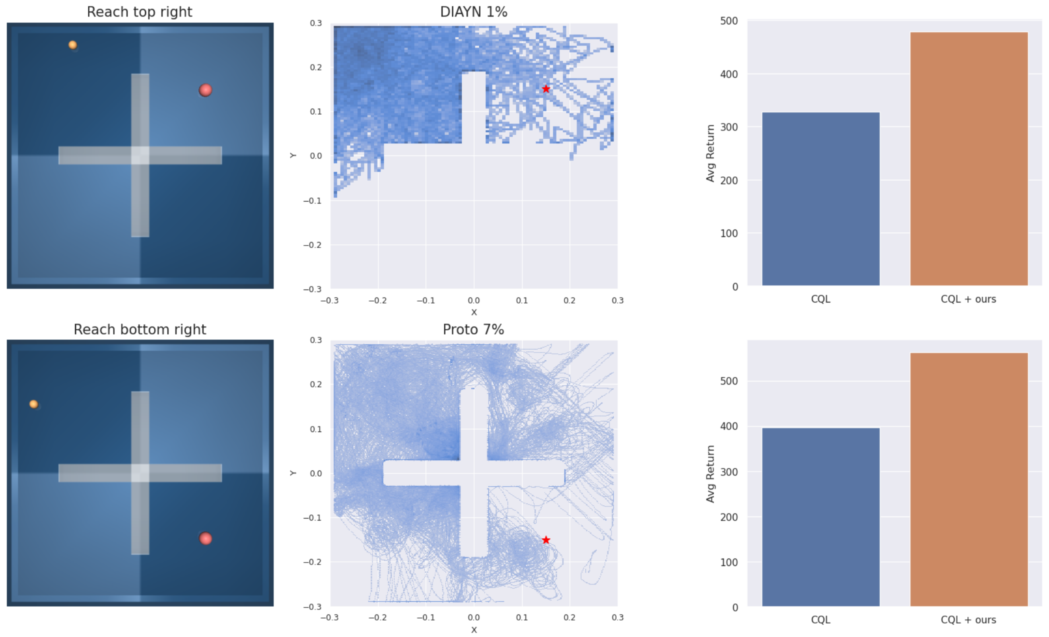


Figure 7: **Effectiveness of the proposed method over different data collection strategies.** (Left) Visualized goal-reaching point mass agents and trajectories with different goals, portions, and exploration methods. (Right) Averaged return of CQL trained with two datasets with and without the proposed pretraining method.

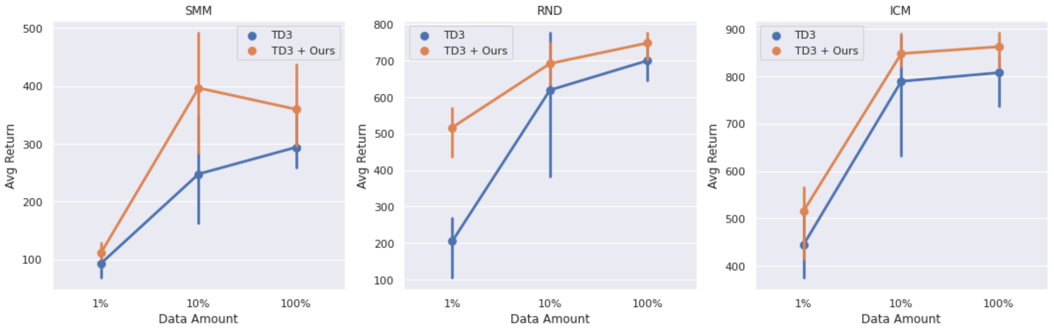


Figure 8: **Average returns in reduced datasets across the dataset collection strategies.** We evaluate our method over different dataset collection strategies *(i.e., SMM, RND, ICM)*. TD3 with our method outperforms the vanilla TD3 overall and even training with 10% of datasets outperform the vanilla TD3 with full datasets. From the results, we demonstrate that our method is data efficient regardless of the dataset collection strategies.

with different exploration strategies. Figure 7 visualizes the trajectories of each reduced dataset collected by DIAYN (Eysenbach et al., 2018), and Proto (Yarats et al., 2021a), strategies *(i.e., 1% of DIAYN, 7% of Proto)*. In comparison with Figure 2 in Yarats et al. (2022), our reduced dataset settings cover narrow state distribution. The top right figure of DIAYN shows that there are a few trajectories around the *top right goal* and the bottom left right figure of Proto also shows that there are a few trajectories around the *bottom right goal* in Figure 7. To demonstrate our method is effective even in narrow state distribution, we evaluated the proposed method on reduced *point mass maze* datasets described in Figure 7 over short (*reach top right*) and long (*reach bottom right*) goals with CQL. Figure 7 demonstrates that our method significantly improves the performance even with narrow state distribution.

Based on the assumption we have made, we also have evaluated our method across dataset collection strategies since each dataset has different distribution. In ExoRL (Yarats et al., 2022), we chose TD3 as a comparison algorithm and SMM (Lee et al., 2019), RND (Burda et al., 2018), and ICM (Pathak et al., 2017), as *walker walk* task datasets. In (Yarats et al., 2022), ICM shows best performance, followed by RND, SMM and TD3 shows best performance in ICM. We compare TD3 to TD3 with our method in reduced datasets (*i.e., 1%, 10%, 100%)* over 3 seeds. To construct reduced datasets, we select the data from the front. Figure 8 shows the results. For all datasets, training our method with only 10% of datasets outperforms TD3 with full datasets. Specially in RND, even training with 1% of datasets shows significantly high averaged return. From the results, we conclude that our method indeed data efficient regardless of the dataset collection strategies.

6 CONCLUSION

In this paper, we propose a data efficiency problem of whether offline RL can maintain performance even in small datasets under an offline RL scheme. To the authors' best knowledge, we first define the data efficiency problem in offline RL and propose an effective method for settling the problem. We suggest the pretraining Q-network method using a forward dynamics prediction task. To pretrain the Q-network, we design a novel shared network architecture that outputs predictions of the next state and Q-values. This structure make our method easy to apply to any existing offline RL algorithms.

To demonstrate that our method improves the performance even in reduced datasets, we conduct experiments with various setting in offline RL. From the results, we demonstrate that our method significantly improve the performance of existing offline RL algorithms over D4RL and Robomimic benchmarks. Furthermore, we demonstrate that our method is indeed data efficient across the data qualities in D4RL and data collection strategies in ExoRL. We leave future work to validate that our method can outperform in real-world applications, e.g. robotic manipulation.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances. (arXiv:2204.01691), August 2022. doi: 10.48550/arXiv.2204.01691. URL `http://arxiv.org/abs/2204.01691`. arXiv:2204.01691 [cs].

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. (arXiv:2302.02948), May 2023a. doi: 10.48550/arXiv.2302.02948. URL `http://arxiv.org/abs/2302.02948`. arXiv:2302.02948 [cs].

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023b.

Chethan Bhateja, Derek Guo, Dibya Ghosh, Anikait Singh, Manan Tomar, Quan Vuong, Yevgen Chebotar, Sergey Levine, and Aviral Kumar. Robotic offline rl from internet videos via value-function pre-training. (arXiv:2309.13041), September 2023. URL `http://arxiv.org/abs/2309.13041`. arXiv:2309.13041 [cs].

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong

Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. (arXiv:2307.15818), July 2023a. doi: 10.48550/arXiv.2307.15818. URL `http://arxiv.org/abs/2307.15818`. arXiv:2307.15818 [cs].

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. (arXiv:2212.06817), August 2023b. doi: 10.48550/arXiv.2212.06817. URL `http://arxiv.org/abs/2212.06817`.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pp. 3909–3928. PMLR, 2023.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pp. 1691–1703. PMLR, 2020. URL `http://proceedings.mlr.press/v119/chen20s.html`.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. (arXiv:1810.04805), May 2019. doi: 10.48550/arXiv.1810.04805. URL `http://arxiv.org/abs/1810.04805`. arXiv:1810.04805 [cs].

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Yunhai Feng, Nicklas Hansen, Ziyan Xiong, Chandramouli Rajagopalan, and Xiaolong Wang. Finetuning offline world models in the real world. October 2023. doi: 10.48550/arXiv.2310.16029. URL `http://arxiv.org/abs/2310.16029`.

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. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20132–20145. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/hash/a8166da05c5a094f7dc03724b41886e5-Abstract.html`.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2052–2062. PMLR, May 2019. URL `https://proceedings.mlr.press/v97/fujimoto19a.html`.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. (arXiv:1912.01603), 2019a. URL `https://arxiv.org/abs/1912.01603`.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b. URL `https://proceedings.mlr.press/v97/hafner19a.html`.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. (arXiv:2203.04955), July 2022. doi: 10.48550/arXiv.2203.04955. URL `http://arxiv.org/abs/2203.04955`. arXiv:2203.04955 [cs].

Zhang-Wei Hong, Aviral Kumar, Sathwik Karnik, Abhishek Bhandwaldar, Akash Srivastava, Joni Pajarinen, Romain Laroche, Abhishek Gupta, and Pulkit Agrawal. Beyond uniform sampling: Offline reinforcement learning with imbalanced datasets. *Advances in Neural Information Processing Systems*, 36:4985–5009, 2023.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. (arXiv:2110.06169), October 2021a. doi: 10.48550/arXiv.2110.06169. URL `http://arxiv.org/abs/2110.06169`. arXiv:2110.06169 [cs].

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. (arXiv:2004.13649), March 2021b. URL `http://arxiv.org/abs/2004.13649`. arXiv:2004.13649 [cs, eess, stat].

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*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/c2073ffa77b5357a498057413bb09d3a-Abstract.html`.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. (arXiv:2006.04779), August 2020. doi: 10.48550/arXiv.2006.04779. URL `http://arxiv.org/abs/2006.04779`. arXiv:2006.04779 [cs, stat].

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Proceedings of the 5th Conference on Robot Learning*, pp. 1702–1712. PMLR, January 2022. URL `https://proceedings.mlr.press/v164/lee22d.html`.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. (arXiv:2005.01643), November 2020. doi: 10.48550/arXiv.2005.01643. URL `http://arxiv.org/abs/2005.01643`. arXiv:2005.01643 [cs, stat].

Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4414–4420. IEEE, 2020.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

Francisco S Melo and M Isabel Ribeiro. Q-learning with linear function approximation. 2007.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. pp. 14297–14306, 2023. URL `https://openaccess.thecvf.com/content/CVPR2023/html/Meng_On_Distillation_of_Guided_Diffusion_Models_CVPR_2023_paper.html`.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36, 2024a. URL `https://proceedings.neurips.cc/paper_files/paper/2023/hash/c44a04289beaf0a7d968a94066a1d696-Abstract-Conference.html`.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36, 2024b.

Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. URL `https://www.mikecaptain.com/resources/pdf/GPT-1.pdf`.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, and Jack Clark. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021. URL `http://proceedings.mlr.press/v139/radford21a`.

Rafael Rafailov, Kyle Beltran Hatch, Victor Kolev, John D. Martin, Mariano Phielipp, and Chelsea Finn. Moto: Offline pre-training to online fine-tuning for model-based robot learning. pp. 3654–3671. PMLR, December 2023a. URL `https://proceedings.mlr.press/v229/rafailov23a.html`.

Rafael Rafailov, Kyle Beltran Hatch, Victor Kolev, John D Martin, Mariano Phielipp, and Chelsea Finn. Moto: Offline pre-training to online fine-tuning for model-based robot learning. In *Conference on Robot Learning*, pp. 3654–3671. PMLR, 2023b.

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. *arXiv preprint arXiv:1709.10087*, 2017.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. (arXiv:2205.11487), May 2022. doi: 10.48550/arXiv.2205.11487. URL `http://arxiv.org/abs/2205.11487`. arXiv:2205.11487 [cs].

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.

Younggyo Seo, Kimin Lee, Stephen L. James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pp. 19561–19579. PMLR, 2022. URL `https://proceedings.mlr.press/v162/seo22a.html`.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning. vol. 135, 1998.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Tengyang Xie, Nan Jiang, Huan Wang, Caiming Xiong, and Yu Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *Advances in neural information processing systems*, 34:27395–27407, 2021.

Rui Yang, Han Zhong, Jiawei Xu, Amy Zhang, Chongjie Zhang, Lei Han, and Tong Zhang. Towards robust offline reinforcement learning under diverse data corruption. *arXiv preprint arXiv:2310.12955*, 2023.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 11920–11931. PMLR, 2021a.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pp. 10674–10681, 2021b.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Chelsea Finn, and Sergey Levine. How to leverage unlabeled data in offline reinforcement learning. In *International Conference on Machine Learning*, pp. 25611–25635. PMLR, 2022.

A PRETRAINING Q-NETWORK FOR ONLINE RL (OFF-POLICY)

Algorithm 2 Pretraining phase for Online RL (Off-policy)

```
Input: Learning rate $\alpha$
Initialize parameters $\varphi, \psi$ and a buffer $\mathcal{D}$
for each gradient step do
    Uniformly sample a random action and collect a transition
    $a \sim U(a_{min}, a_{max})$
    $s' \sim p(s'|s, a)$
    Update the buffer with a collected transition
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a, r, s')\}$

    Sample a mini-batch $\mathcal{B} \sim \mathcal{D}$
    Compute the forward dynamics prediction error
        $\mathcal{L}_{pre}(\varphi, \psi) = \sum_{(s,a,s') \in \mathcal{B}} (s' - (g_\psi \circ h_\varphi)(s, a))^2$
    Update weights of the shared network and forward network
        $\varphi \leftarrow \varphi - \alpha \nabla_\phi \mathcal{L}_{pre}(\varphi, \psi), \quad \psi \leftarrow \psi - \alpha \nabla_\psi \mathcal{L}_{pre}(\varphi, \psi)$
end for
Output: Pretrained weights $\varphi$ of the shared network, collected buffer $\mathcal{D}$
```

Algorithm 3 Pretraining phase for Online RL (Off-policy) with pre-collected dataset

```
Input: Dataset $\mathcal{D}_{pre}$ of transition $(s, a, s')$, Learning rate $\alpha$
Initialize parameters $\varphi, \psi$
for each gradient step do
    Sample a mini-batch $\mathcal{B} \sim \mathcal{D}_{pre}$
    Define the loss function
        $\mathcal{L}_{pre}(\varphi, \psi) = \sum_{(s,a,s') \in \mathcal{B}} (s' - (g_\psi \circ h_\varphi)(s, a))^2$
    Take the gradient descent step
        $\varphi \leftarrow \varphi - \alpha \nabla_\phi \mathcal{L}_{pre}(\varphi, \psi), \quad \psi \leftarrow \psi - \alpha \nabla_\psi \mathcal{L}_{pre}(\varphi, \psi)$
end for
Output: Pretrained weights $\varphi$ of the shared network
```

We extended our pretraining method to popular online off-policy RL methods by incorporating the pretraining phase ahead of the main training phase. During the pretraining phase of the online agent, a trajectory dataset was obtained by either initializing the replay buffer with actively collected interaction data by uniformly sampling a random action or offline static dataset.

For experiments on online RL using an off-policy setting, we adopted soft actor-critic (SAC) Haarnoja et al. (2018) and twin delayed deep deterministic policy gradient algorithm (TD3) Fujimoto et al. (2018). We compare these algorithms with and without our pretraining method on OpenAI Gym MuJoCo tasks. For a fair comparison, all algorithms were trained for 1 million time steps on each task over 5 seeds.

Table 3 presents the results of the experiments following Algorithm 2 which collects the pretraining dataset by uniformly sampling random actions. Incorporating our pretraining phase shows better performance in more than half of the results. Additionally, we trained both SAC and TD3 with the pre-collected dataset from the D4RL for the pretraining phase along the Algorithm 3. Note that we only used the pre-collected dataset during the pretraining phase. Table 4 shows the best scores among the 5 datasets (i.e., random, medium, medium replay, medium expert, expert). Interestingly, pretraining with the suboptimal-level dataset (medium-replay) shows better performance compared to the expert-level dataset.

Table 3: Results of Off-policy RL application on OpenAI gym MuJoCo tasks

	SAC	TD3
HalfCheetah-v2	10065.77±621.80→11005.51±374.14	10644.63±190.42→11697.71±236.01
Hopper-v2	3357.07±30.64→1419.55±137.55	3365.08±94.69→3454.83±129.34
Walker2d-v2	4279.67±509.51→2697.92±674.29	4193.11±435.31→4481.19±190.93
Ant-v2	4191.17±986.11→4399.56 766.24	5172.78±659.02→4407.40±759.64
Humanoid-v2	5545.70±85.00→479.09 83.86	5247.14±187.64→5816.16±199.25
Pusher-v2	-190.77±88.51→-133.96 29.00	-22.94±0.52→-22.85±1.25

Table 4: Results of Off-policy RL pretrain with the D4RL OpenAI gym MuJoCo datasets

	SAC	TD3
HalfCheetah-v2	10402.79±1675.67	11820.06±269.76
Hopper-v2	3405.95±70.87	3465.25±149.87
Walker2d-v2	4785.15±247.37	4559.38±1007.69

From the above experiments, we conjecture that pretrained online RL (off-policy) has limitations when they only exploit random action data for pretraining. A marginal state distribution induced by uniformly sampling random actions is close to the initial state distribution, limiting the diversity in the dataset and eventually leading to an increase in forward dynamics uncertainty. Consequently, there are fewer opportunities to learn the good features of forward dynamics with random action datasets than suboptimal-level datasets. This explains why Table 3 shows worse results than Table 4.

We also applied another approach introduced in section B to online RL settings. The results, shown in Table 5, indicate that more than half exhibit enhanced performance compared to reported scores in Table 3.

Table 5: Results of Off-policy RL with Additional Loss

	SAC	TD3
HalfCheetah-v2	8498.68±3195.13	9588.53±866.30
Hopper-v2	3539.39±133.47	3523.67±202.52
Walker2d-v2	4847.86±135.52	3819.68±552.84
Ant-v2	3710.73±917.35	5401.0±844.56
Humanoid-v2	5576.98±106.31	5489.73±38.28
Pusher-v2	-158.66±55.02	-25.47±34.00

B ANOTHER DESIGN CHOICE USING OUR SHARED Q-NETWORK STRUCTURE

In this section, we introduce another approach that also utilizes features of forward dynamics using the shared networks as in the previous pretraining method. In this approach, we use the following modified loss that adds the forward model loss to the loss for the Q-function estimation:

$$\mathcal{L}_Q = \mathcal{L}_{TD} + \mathcal{L}_{dynamics} \tag{6}$$

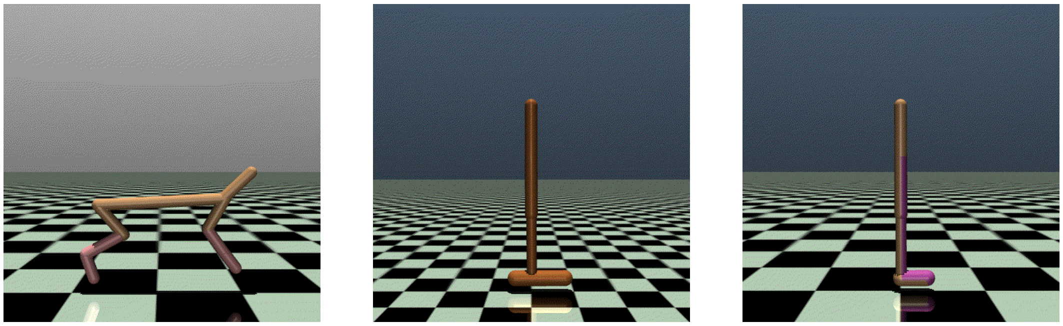
In this way, the shared network is trained throughout the entire training period without the pretraining phase. We adopt TD3+BC for evaluation and the results are presented in table 6. On TD3+BC, this approach also outperforms almost all of the vanilla scores. Simply adding the supervised loss term of state prediction without any multiplier or technique demonstrates improved performance. Consequently, we suggest that the proposed shared Q-network can be expanded in other directions and we expect that it holds significant potential for further research.

C TASKS AND DATASETS

In this section, we provide detailed experimental setups for the tasks and datasets. Illustrated environments can be found in Figure 9

Table 6: **Averaged normalized scores of TD3+BC with additional loss on D4RL benchmark.** We depict increased scores compared to their original scores in blue color and report mean and standard deviations over 5 random seeds.

	Random	Medium	Medium Replay	Medium Expert	Expert
HalfCheetah-v2	11.45±0.51	48.23±0.33	44.93±0.29	93.55±1.00	96.59±0.25
Hopper-v2	31.54±0.42	70.86±2.17	90.39±7.34	113.44±0.35	113.28±0.20
Walker2d-v2	13.46±6.58	82.65±1.65	86.11±1.54	111.88±0.63	110.98±0.22



(a) HalfCheetah. (b) Hopper. (c) Walker2d.

(d) Lift. (e) Can.

Figure 9: **Illustrations of each environment.** Top and bottom rows are 2D locomotion and 3D manipulation environments, respectively.

C.1 D4RL

D4RL consists of 8 separate tasks. In this work, we utilized one of them for the main experiments; OpenAI Gym MuJoCo continuous control tasks. It consists of 4 different environments (i.e., HalfCheetah, Walker2d, Hopper, and Ant) and 5 heterogeneous datasets in terms of data quality for each environment. Each dataset is collected along the below strategies:

- Random (1M samples): Collected from a randomly initialized policy.
- Expert (1M samples): Collected from a policy trained to completion with SAC.
- Medium (1M samples): Collected from a policy trained to approximately 1/3 the performance of the expert.
- Medium-Expert (almost 2M samples): A 50-50 split of medium and expert data.
- Medium-Replay (almost 3M samples): Collected from the replay buffer of a policy trained up to the performance of the medium agent.

All environments have the same episode limit of 1000 and the goal of each locomotion agent is to run as fast as possible without falling to the ground. More detailed information can be found at https://github.com/Farama-Foundation/D4RL.

C.2 ROBOMIMIC

Robomimic provides a large-scale and diverse collection of task demonstrations spanning multiple human or robotic demonstrations of varying quality. We considered machine-generated (MG) datasets generated by training an SAC agent for each task and then using intermediate policies to generate mixed-quality datasets. We selected this dataset for evaluation since the proposed method demonstrated superior performance with suboptimal datasets on the D4RL benchmark. All environments have the same episode limit of 400. The goal of the Lift environment is lifting the cube above a certain height and the goal of the Can environment is placing the can into the corresponding container. More detailed information can be found at https://github.com/ARISE-Initiative/robomimic.

C.3 EXORL

They provide exploratory datasets for 6 DeepMind Control Stuite domains (*i.e., Cartpole, Cheetah, Jaco Arm, Point Mass Maze, Quadruped, Walker*) and totally 19 tasks. For each domain, they collected datasets by running 9 unsupervised RL algorithms (*i.e., APS, APT, DIAYN, Disagreement, ICM, ProtoRL, Random, RND, SMM*) from URLB for total of 10M steps. More detailed information can be found at https://github.com/denisyarats/exorl?tab=readme-ov-file.

D EXPERIMENTS ON ADROIT IN D4RL

We conducted additional experiments on adroit in D4RL Fu et al. (2020) benchmark to validate that the proposed method can be adopted to different complex domains. An illustration of the Adroit environment can be found in Figure 10. The Adroit domain involves controlling a 24-DoF robotic hand with 4 different control tasks (i.e., Pen, Door, Hammer, and Relocate) and 3 heterogeneous datasets as following:

- Human: Collected with the 25 human demonstrations provided in the DAPG Rajeswaran et al. (2017) repository.
- Cloned: a 50-50 split between demonstration data and 2500 trajectories sampled from a behavioral cloned policy on the demonstrations. The demonstration trajectories are copied to match the number of behavioral cloned trajectories.
- Expert: Collected with 5000 trajectories sampled from an expert that solves the task, provided in the DAPG repository.

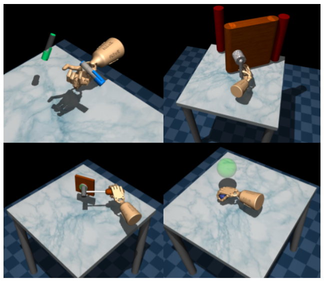


Figure 10: **The tasks of Adroit. (top left)** Pen - aligning a pen with a target orientation, **(top right)** Door - opening a door, **(bottom left)** Hammer - hammering a nail into a board, **(bottom right)** Relocate - moving a ball to a target position.

For experiments, we compared AWAC, IQL, and TD3+BC with/without our pretraining method over 5 seeds. Table 7 yields averaged normalized scores for each task. Overall, learning with our pretraining phase demonstrates enhanced performance. From these results, we conclude that the proposed method can be effective in complex domains not only tabletop but dexterous manipulation as well.

Table 7: **Averaged normalized scores on Adroit.** Left-hand side scores are scores of vanilla methods. Right-hand side scores are scores of baselines combined with our pretraining method. We depict increased scores compared to their original scores in blue color and report mean and standard deviations over 5 random seeds.

		AWAC	IQL	TD3+BC
Human	Pen	146.19±5.29→157.60±5.28	101.87±14.34→104.66±17.30	20.32±5.97→20.78±10.93
	Hammer	7.98±9.41→36.95±35.13	14.33±5.22→17.78±9.27	2.40±0.16→2.38±0.17
	Door	60.82±12.38→29.96±22.43	6.74±1.31→5.81±3.20	-0.09±0.00→-0.04±0.04
	Relocate	1.51±1.05→3.91±2.21	1.20±1.05→1.52±1.11	-0.29±0.01→-0.18±0.13
Cloned	Pen	145.37±4.19→144.48±3.42	98.38±16.13→97.76±16.90	39.69±18.95→48.18±11.27
	Hammer	10.37±7.88→12.61±8.66	8.94±2.07→11.38±4.46	0.59±0.17→1.17±0.61
	Door	2.95±2.97→9.59±7.73	5.61±3.02→5.00±1.44	-0.23±0.11→-0.03±0.03
	Relocate	0.04±0.09→0.18±0.21	0.91±0.45→1.06±0.40	-0.02±0.09→-0.13±0.09
Expert	Pen	163.99±1.19→163.73±1.88	148.38±2.46→147.79±3.06	131.73±19.15→141.10±10.28
	Hammer	130.08±1.30→130.04±0.48	129.46±0.42→129.50±0.36	33.36±34.61→59.76±52.35
	Door	106.67±0.28→106.95±0.16	106.45±0.29→106.71±0.28	0.99±0.83→0.87±1.48
	Relocate	109.70±1.32→111.27±0.35	110.13±1.52→109.82±1.45	0.57±0.33→0.22±0.13
Total		885.67±47.35→907.26±87.94	732.40±48.27→738.79±59.23	229.03±80.40→274.08±87.49

E IMPLEMENTATION DETAILS

In this section, we provide detailed implementation setups for extensive experiments. Since we suggest a plug-and-play pretraining method for popular offline RL methods, we reuse open-source code for comparative results: TD3+BC[1], IQL[2], AWAC[3], and CQL[4] for D4RL. We use off-the-shelf offline methods in the official repository[5] for the Robomimic environment. We only use open-source baselines which use PyTorch for fair comparisons. On the D4RL, we train each agent with 1M gradient steps for each environment over 5 seeds. Also, we evaluate each agent with 5 rollouts every 5k gradient steps for TD3+BC, AWAC, and CQL and 10k gradient steps for IQL. We report the best scores for all tables and figures. On the Robomimic, we train each agent with 200k gradient steps for each environment over 5 seeds. Also, we evaluate each agent with 50 rollouts over 5 seeds. For all experiments, we used RTX-A5000 GPU for training and evaluation.

---

[1] `https://github.com/sfujim/TD3_BC`

[2] `https://github.com/Manchery/iql-pytorch`

[3] `https://github.com/hari-sikchi/AWAC`

[4] `https://github.com/young-geng/CQL`

[5] `https://github.com/ARISE-Initiative/robomimic`

F LEARNING CURVES

In this section, we provide the full results of learning curves in the section 5.1 for further information.

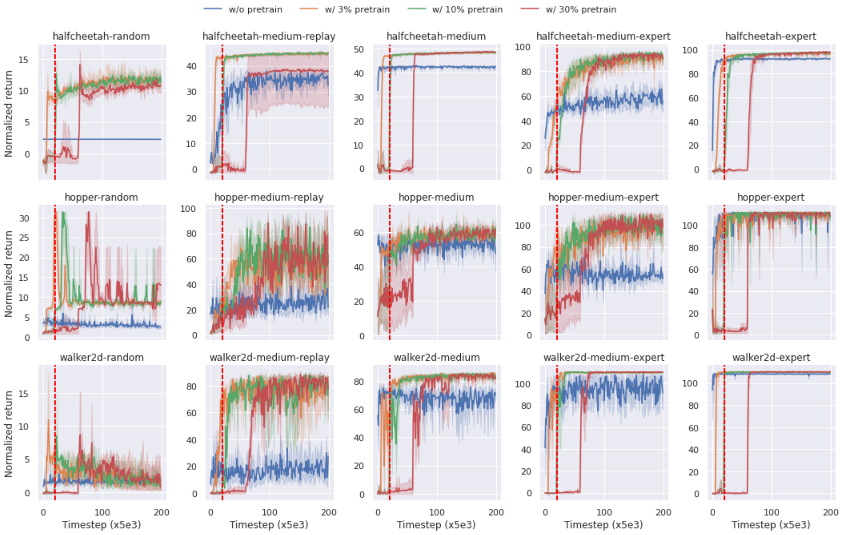


Figure 11: **Learning curves of TD3+BC on the D4RL benchmark.**

G EXPERIMENTS WITH LINEAR APPROXIMATED Q-NETWORK

In this section, We pretrained TD3+BC and froze it except for the last linear layer during the remaining learning time. The blue-colored scores indicate improved scores from the reported scores from the original TD3+BC. Although only the last linear layer of the pretrained TD3+BC was trained and the shared network was frozen, it shows better performance than the vanilla CQL. Moreover, it shows better performance than the others over the suboptimal level of the datasets (i.e., random, medium, medium replay).

Table 8: **Results of pretrained TD3+BC which approximated with linear Q function.**

		AWAC	CQL	IQL	TD3+BC	freezed TD3+BC
Random	HalfCheetah	2.2	21.7±0.9		10.2±1.3	6.03±2.65
	Hopper	9.6	10.7±0.1		11.0±0.1	11.59±10.56
	Walker2d	5.1	2.7±1.2		1.4±1.6	7.18±0.58
Medium	HalfCheetah	37.4	37.2±0.3	47.4	42.8±0.3	42.64±1.19
	Hopper	72.0	44.2±10.8	66.4	99.5±1.0	67.16±3.56
	Walker2d	30.1	57.5±8.3	78.3	79.7±1.8	72.03±0.78
Medium Replay	HalfCheetah		41.9±1.1	44.2	43.3±0.5	40.21±0.79
	Hopper		28.6±0.9	94.7	31.4±3.0	64.41±19.54
	Walker2d		15.8±2.6	73.9	25.2±5.1	41.02±12.05
Medium Expert	HalfCheetah	36.8	27.1±3.9	86.7	97.9±4.4	47.35±8.73
	Hopper	80.9	111.4±1.2	91.5	112.2±0.2	95.07±15.27
	Walker2d	42.7	68.1±13.1	109.6	101.1±9.3	74.75±0.59
Expert	HalfCheetah	78.5	82.4±7.4		105.7±1.9	61.93±10.71
	Hopper	85.2	111.2±2.1		112.2±0.2	113.13±0.39
	Walker2d	57.0	103.8±7.6		105.7±2.7	57.14±44.96
Total			764.3±61.5		979.3±33.4	801.64±132.34

H PRETRAINING RATE

In this section, we provide more details in section 5.1 of the pretraining rate. We conducted each experiment with the same settings in subsection 5.1 over 5 seeds and reported the results that exhibit averaged normalized scores.

Table 9: **Results of TD3+BC with various pretraining rate.**

		w/o pretrain	w/ 3% pretrain	w/ 10% pretrain	w/ 30% pretrain
Random	HalfCheetah	10.2±1.3	14.26±1.86	14.83±0.54	15.22±1.33
	Hopper	11.0±0.1	31.55±0.16	31.56±0.16	32.00±0.50
	Walker2d	1.4±1.6	16.13±5.60	11.23±5.05	11.87±5.28
Medium	HalfCheetah	42.8±0.3	49.14±0.10	49.17±0.26	49.35±0.24
	Hopper	99.5±1.0	73.90±3.54	71.52±2.16	73.53±4.75
	Walker2d	79.7±1.8	87.14±0.63	87.09±0.60	86.79±0.38
Medium Replay	HalfCheetah	43.3±0.5	45.91±0.32	45.84±0.26	39.67±12.74
	Hopper	31.4±3.0	99.69±1.51	100.16±1.60	99.34±1.58
	Walker2d	25.2±5.1	90.91±0.70	92.01±1.58	91.51±0.90
Medium Expert	HalfCheetah	97.9±4.4	95.85±1.05	96.89±0.92	96.35±0.49
	Hopper	112.2±0.2	112.84±0.31	113.02±0.19	112.95±0.22
	Walker2d	101.1±9.3	111.45±0.30	111.58±0.35	111.52±0.37
Expert	HalfCheetah	105.7±1.9	98.68±0.52	98.86±0.55	99.40±0.48
	Hopper	112.2±0.2	113.23±0.31	113.35±0.28	113.43±0.44
	Walker2d	105.7±2.7	111.11±0.17	111.00±0.15	111.06±0.20
Total		979.3±33.4	1151.84±17.08	1148.12±14.65	1144.01±29.90

I EXPERIMENTS WITH VARIOUS SIZES OF DATASETS

In this section, we provide more details in section 5.3 of the dataset size. We conducted each experiment with the same settings in subsection 5.1 over 5 seeds and reported the results that exhibit averaged normalized scores.

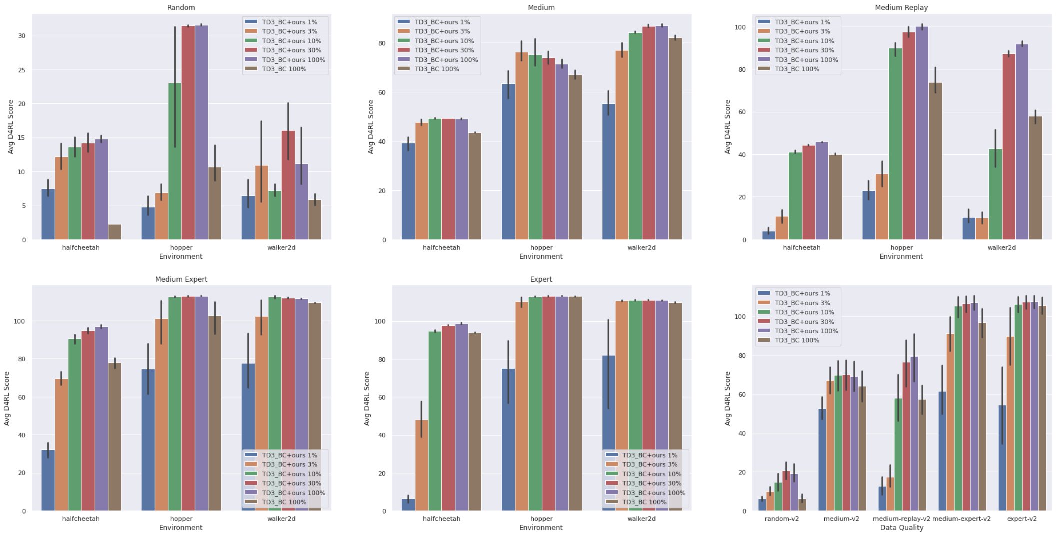


Figure 12: **Averaged normalized scores across dataset optimal quality and sizes.** This figure compares the performance of our method with TD3+BC in reduced datasets *(i.e., 1%, 3%, 10%, 30%, 100% of each dataset)* to vanilla TD3+BC across the data quality *(i.e., random, medium, medium replay, medium expert, expert)* on D4RL. From the overall results (Bottom Right), we conclude that our method guarantees better performance even in 10% of the datasets regardless of the data quality of the dataset.

Table 10: **Results of pretrained AWAC over various size.**

		w/o pretrain	w/ pretrain 10%	w/ pretrain 30%	w/ pretrain
Random	HalfCheetah	2.2	9.71±3.08	36.37±1.47	51.10±0.89
	Hopper	9.6	97.05±3.24	93.35±6.32	59.47±33.79
	Walker2d	5.1	8.57±0.47	8.36±1.30	13.11±3.91
Medium	HalfCheetah	37.4	55.47±1.52	56.64±2.68	54.63±1.45
	Hopper	72.0	101.28±0.78	101.32±0.20	101.73±0.20
	Walker2d	30.1	95.14±1.46	91.38±1.37	89.51±0.88
Medium Replay	HalfCheetah		51.00±0.69	52.12±0.76	55.75±1.30
	Hopper		103.67±1.81	107.69±1.71	106.67±0.59
	Walker2d		104.10±1.57	105.42±1.97	100.31±2.11
Medium Expert	HalfCheetah	36.8	83.18±1.69	86.55±0.94	90.05±1.89
	Hopper	80.9	113.01±0.71	113.34±0.09	113.23±0.22
	Walker2d	42.7	117.26±1.77	114.68±2.18	111.88±0.28
Expert	HalfCheetah	78.5	91.54±1.04	93.46±0.54	93.48±0.11
	Hopper	85.2	113.02±0.17	113.18±0.20	112.86±0.10
	Walker2d	57.0	117.92±2.07	112.55±0.56	111.22±0.35
Total			1261.90±22.05	1286.43±22.28	1265.01±48.07

Table 11: **Results of pretrained IQL over varying dataset sizes.**

		w/o pretrain	w/ pretrain 10%	w/ pretrain 30%	w/ pretrain
Random	HalfCheetah		6.92±0.63	12.65±2.53	18.28±1.02
	Hopper		8.17±0.54	9.93±1.19	10.67±0.41
	Walker2d		8.26±0.64	9.08±0.96	8.88±0.71
Medium	HalfCheetah	47.4	46.51±0.18	47.87±0.21	48.85±0.16
	Hopper	66.4	75.72±3.23	80.76±3.51	78.62±2.21
	Walker2d	78.3	82.62±1.03	83.89±1.69	83.63±1.14
Medium Replay	HalfCheetah	44.2	33.49±1.26	41.16±0.50	45.48±0.17
	Hopper	94.7	80.59±8.25	91.08±3.67	99.43±1.71
	Walker2d	73.9	39.08±10.42	75.33±4.17	87.95±1.68
Medium Expert	HalfCheetah	86.7	87.44±2.52	93.66±0.46	95.25±0.14
	Hopper	91.5	93.89±10.67	91.05±18.78	105.77±11.31
	Walker2d	109.6	111.23±0.83	111.65±0.93	112.09±0.93
Expert	HalfCheetah		77.85±3.82	95.88±0.44	97.40±0.13
	Hopper		109.16±3.25	112.85±1.30	113.34±0.46
	Walker2d		113.76±2.55	112.53±1.35	112.80±1.08
Total			974.68±49.84	1069.36±41.69	1118.46±23.25