Pretraining a Shared Q-Network for Data-Efficient Offline Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Offline reinforcement learning (RL) aims to learn a policy from a static dataset without further interactions with the environment. Collecting sufficiently large datasets for offline RL is exhausting since this data collection requires colossus interactions with environments and becomes tricky when the interaction with the environment is restricted. Hence, how an agent learns the best policy with a minimal static dataset is a crucial issue in offline RL, similar to the sample efficiency problem in online RL. In this paper, we propose a simple yet effective plug-and-play pretraining method to initialize a feature of a Q-network to enhance data efficiency in offline RL. Specifically, we introduce a shared Q-network structure that outputs predictions of the next state and Q-value. We pretrain the shared Q-network through a supervised regression task that predicts a next state and trains the shared Q-network using diverse offline RL methods. Through extensive experiments, we empirically demonstrate that our method enhances the performance of existing popular offline RL methods on the D4RL, Robomimic and V-D4RL benchmarks. Furthermore, we show that our method significantly boosts data-efficient offline RL across various data qualities and data distributions trough D4RL and ExoRL benchmarks. Notably, our method adapted with only 10% of the dataset outperforms standard algorithms even with full datasets.

1 Introduction

Sample efficiency is a crucial issue in reinforcement learning (RL) since typical RL considers an online learning nature that involves iterative processes between experience collections and policy improvements through online interactions with the environment [51]. Unfortunately, requiring excessive online interactions is impractical in several cases since data collection requires expensive costs and retains potential risks of the agent, e.g. hardware corruption. Offline RL is one approach to alleviate this sample efficiency problem, which provides a solution by avoiding online interactions with the environment [35]. In recent years, pretraining with offline RL and fine-tuning with online RL have been investigated to improve sample efficiency of the online interactions [41, 55, 44, 3].

Similar to addressing the sample efficiency problem in online RL, learning offline RL with minimal datasets is necessary since collecting enormous experience charges expensive costs and unfavorable explorations, hampering the possibility of offline RL in the real world. In this paper, we name this problem as data efficiency where an agent tries to learn the best policy with minimal data in the offline RL scheme. Despite the necessity of data efficiency, this problem has not been treated enough in previous works. Although some researchers have evaluated their work empirically on reduced datasets in part of the experiments [1, 29, 31], they have overlooked this data efficiency problem. In the case of online RL, model-based RL and representation learning have proposed the resolution of sample efficiency problem [50, 21, 46, 47]. As in online RL, one can expect that offline model-based

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

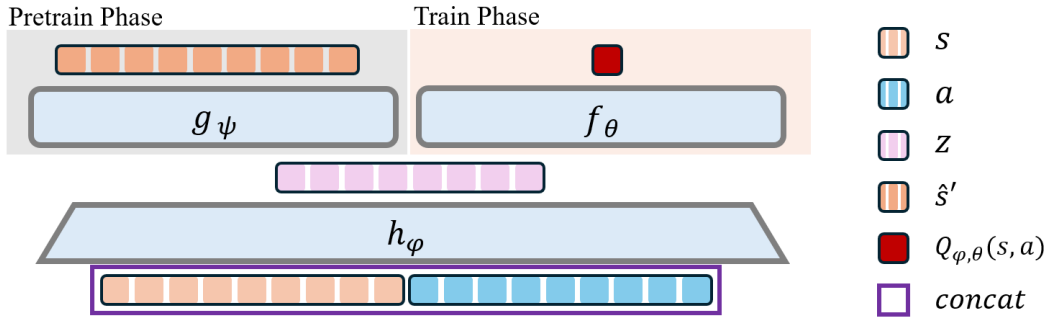

Figure 1: **Overview of our pretraining method.** Our method splits the original Q-network into two core architectures: a shared network that extracts the representation z from the concatenated vector of state s and action a and separated heads for training the transition model network and Q-network,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

RL or representation method might resolve this data efficiency problem [64, 49, 56]. However, Figure 8 demonstrates that both approaches are unable to overcome this problem.

In this work, we propose a simple yet effective plug-and-play method that pretrains a shared Q-network toward data-efficient offline RL. Specifically, the shared Q-network structure is composed of two parts as illustrated in Figure 1. First, a shared deep neural network layer (h_φ) takes the state and action pair as inputs. Second, separate shallow output parts (g_ψ and f_θ) consist of two linear layers that individually output a Q-value for a Q-function and a next state prediction for a transition model. The learning phase of the shared Q-network consists of a pretraining and an RL training phase. In the pretraining phase, the shared network attached with a shallow transition layer (h_φ and g_ψ) is trained through a supervised regression task that predicts the transition model. After the pretraining phase where the shared network is initialized with the pretraining, the shared network is connected with a shallow Q layer (h_φ and f_θ) and trained with an existing offline RL value learning.

We empirically demonstrate that our method improves the performance of existing popular offline RL methods on the D4RL [14], Robomimic [38] and V-D4RL [36] benchmarks. We also show that our method maintains data-efficient performance with fragments of the dataset across the *data quality* on the D4RL dataset. Moreover, we investigate our method across the *data collection strategies* on the ExoRL datasets [58], assuming a small dataset would have a shifted data distribution compared to a large dataset. As a result, we demonstrate that our method improves the performance regardless of the qualities of the datasets and the data distributions. Figure 7 and Figure 10 show that our method with 10% of datasets outperforms vanilla algorithms even with full datasets. Furthermore, Figure 8 demonstrates that our method indeed outperforms the offline model-based RL and representation approaches in reduced datasets.

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 by the idea that queries of the Q-function over out-of-distribution actions may yield overly optimistic values during offline training [17, 30, 35, 31, 15, 27]. Recently, scalability to a large dataset and neural network model has been studied [8, 42, 52]. In other fields, pretraining with offline RL and fine-tuning with online RL is examined to improve sample efficiency in the online interaction step [41, 55, 44, 3]. In contrast, distinct 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 [26, 63, 57]. While prior research [1, 29, 31] often has evaluated their work on reduced datasets as a partial result, the field overlooks the data efficiency problem itself as a main contribution. In contrast, we aim to improve the data efficiency in offline RL (i.e., learning the best policy with minimal data). In this work, we propose a simple yet effective plug-and-play method for pretraining a shared Q-network toward the data-efficient offline RL.

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 [28, 62, 11]. Model-based RL [50, 10, 21, 20, 24] 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 [47, 62] and data augmentation [32, 28] play a critical role in improving sample efficiency in RL. Recently, offline-to-online [34, 3, 44, 13, 41] and foundation model [2, 48, 6, 5, 4] have tackled this problem where the poor sample efficiency of online RL regime is alleviated by leveraging large offline data.

Data efficient Offline RL. In this work, we define data efficiency in offline RL as the ability of an algorithm to learn an optimal policy from a minimal static dataset of pre-collected samples. This contrasts with sample efficiency, which we use to describe the ability to learn effectively with few environment interactions in online RL. Some previous work [46, 47] mention data-efficient approaches. However, they have focused on online RL scheme, therefore, the actual meaning is sample efficiency in our contents. These methods employ self-predictive tasks in latent space to enhance representation learning, relying on techniques like data augmentation [61] and target encoders [25] to improve performance. In contrast, our approach uses supervised learning for pretraining within a shared network architecture, requiring no additional techniques. From the various experimental setting, we demonstrate that our method improves performance in offline RL, effectively addressing our defined data efficiency problem with minimal static datasets.

3 Pretraining Q-network with Transition Model Helps Improving data efficiency

Algorithm 1 Pretraining Q-network scheme for Offline RL

Input: Dataset $\mathcal{D}$ of transition (s, a, s'), learning rate α
Initialize parameters φ, ψ
for each gradient step **do**
Sample a mini-batch $\mathcal{B} \sim \mathcal{D}$
Compute the transition model 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 transition model 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 φ of the shared network

In this paper, we propose a simple yet effective pretraining method adapting features of the transition model 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 estimation of the transition model. In particular, the transition 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 transition model in (1). The overall structures of the neural networks are illustrated in Figure 1.

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

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

over the pre-collected dataset $\mathcal{D}$ which includes a given set of the transition (s, a, s'). Afterward, the pretrained parameter φ can be used as an initial or fixed parameter for standard RL algorithms based on the Q-network structure in (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 our 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.

3.1 Analysis: Based on the Projected Bellman Equation

In this section, we analyze how our method can resolve the data efficiency problem from the perspective of the projected Bellman equation. For simplicity and convenience of presentation, we assume that the state and action spaces are discrete and finite, and the transition is deterministic. However, the principles in this paper can be extended to more general continuous state and continuous action cases. Our analysis is based on the observation that Q-function with neural networks can be generally represented by (2). Following this context, our analysis starts by viewing the conventional MLP into two separate parts: a feature function, h_φ, and a linear function approximator, θ. Defining the feature vector $z = h_\varphi(s, a) \in \mathbb{R}^m$, it can be rewritten as

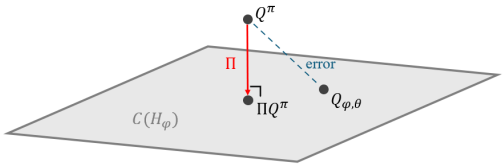

Figure 2: **Reduced approximation error with the expanded column space of H_φ.** In linear approximation, there exists Q^π outside of the column space of H_φ. To deal with this problem, the projected Bellman equation projects Q^π to ΠQ^π which exists in the column space of H_φ.

$$Q_{\varphi,\theta}(s, a) = \sum_{i=1}^{m} \theta_i h_{\varphi,i}(s, a) = \langle \theta, h_\varphi(s, a) \rangle \tag{4}$$

where $(s, a) \in \mathcal{S} \times \mathcal{A}$. When φ is fixed, then the above structure can be viewed as a linear function approximation with the feature function $h_{\varphi,i}$. Our method pretrian $h_{\varphi,i}$ by minimizing the loss in (3). In this work, the output before the last layer of MLP correspond to the latent space (feature) z and the last linear layer is correspond to θ (i.e., $Q_{\theta,\varphi}(s, a) = h_\varphi(s, a)^T\theta$). Therefore, the interpretation based on the linear function approximation is expected to be a reasonable model to explain the phenomenon in our 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 [39] is known to admit a solution

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

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. In other words, expanding the dimension of the feature matrix's column space reduces the Bellman error.

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 (5) by avoiding suboptimal local solutions.

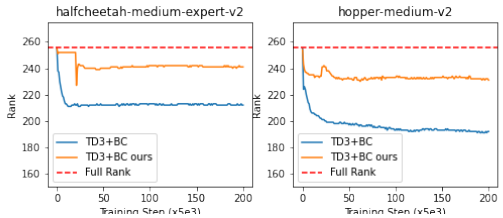

Figure 3: **The Rank of the latent space of Q-network during the training time.** We compare the rank of the latent space between a vanilla TD3+BC and TD3+BC adapted with our method over 512 samples. Our method maintains higher rank of the latent space, leading to reduced approximation error.

Figure 3 exhibits that adapting our method shows a significantly higher rank than the rank of the vanilla method. The full version of the Figure is on Appendix H. From the results, we claim that our method indeed expands the column space $C(H_\varphi)$ and covers higher dimensional vector space in $\mathbb{R}^{|S\times A|}$, leading to more precise Q-function estimation. In other words, we might learn a more precise Q-function with the same amount of samples, and it means that we can get a desirably estimated Q-function with less data. In the following section, we demonstrate our claim with empirical experiments.

We conjecture that the pretraining approach with the transition 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 (5). To support this, we empirically compare the rank of the Q-network in the latent space between vanilla and the pretrained TD3+BC with our method over 512 data samples.

4 Experiments

In this section, we evaluate our method over existing offline RL methods with the popular offline RL benchmarks, D4RL, the more complex domain, Robomimic, and the image-based environment, V-D4RL. Furthermore, we examine our 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 and Baselines. We have considered heterogeneous tasks and diverse datasets for precise comparisons. For the locomotion task, our method is compared with existing methods in the popular D4RL benchmark [14]. Three different embodied agents and five distinct datasets are considered in order to validate the effectiveness of our method: *HalfCheetah, Hopper, Walker2d* for agents and *random, medium-replay, medium, medium-expert, expert* for datasets. We plug our method into the popular offline RL methods, AWAC [40], CQL [31], TD3+BC [15], and IQL [27]. To verify the benefits of our method, we compared the normalized scores between the vanilla method and the one combined with our pretraining method. For the tabletop manipulation tasks, we evaluate our method on the Robomimic benchmark, [38], 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 our method: *Lift, Can* for tasks and *Machine-Generated (MG)* for datasets. We compare the success rate the tasks, where IQL, TD3+BC, BCQ [17], and IRIS [37]. For the image-based environment, we evaluate our method on *Cheetah Run and Walker Walk* tasks in V-D4RL benchmark [36]. We build our method on DrQ+BC. For data-efficient offline RL, we have evaluated our method across the reward qualities of the datasets of D4RL *Gym locomotion tasks* on MOPO [64], MOBILE [49] and ACL [56] to compare our method with offline model-based RL and representation approaches, and the dataset collection strategies for *walker walk (i.e. SMM, RND, ICM)* and *point*

Table 1: **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 our 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	2.6±0.4→51.1±0.9	21.7±0.9→31.9±2.6	10.3±0.9 → 18.3±1.0	2.3±0.0→14.8±0.5
	Hopper	28.6±8.9→59.5±33.8	10.7±0.1→30.2±2.7	9.4±0.4 → 10.7±0.4	10.7±3.2→31.6±0.2
	Walker2d	7.8±0.2→13.1±3.9	2.7±1.2→19.6±4.5	7.9±0.5 → 8.9±0.7	5.9±1.0→11.2±5.1
Medium	HalfCheetah	48.4±0.1→54.6±1.5	37.2±0.3→39.9±18.8	46.6±0.2→48.9±0.2	43.5±0.1→49.2±0.3
	Hopper	88.4±8.8→101.7±0.2	44.2±10.8→90.6±2.2	76.9±5.8→78.6±2.2	67.0±1.9→71.5±2.2
	Walker2d	53.0±33.2→89.5±0.9	57.5±8.3→84.7±0.7	83.8±1.5→83.6±1.1	82.1±1.0→87.1±0.6
Medium Replay	HalfCheetah	46.1±0.3→55.8±1.3	41.9±1.1→47.6±0.4	43.4±0.3→45.5±0.2	40.0±0.5→45.8±0.3
	Hopper	101.3±0.6→106.7±0.6	28.6±0.9→98.6±2.1	96.2±1.9→99.4±1.7	73.9±7.3→100.2±1.6
	Walker2d	88.1±0.6→100.3±2.1	15.8±2.6→87.7±1.3	77.9±2.1→88.0±1.7	58.0±3.6→92.0±1.6
Medium Expert	HalfCheetah	76.4±2.8→90.1±1.9	27.1±3.9→82.8±6.5	94.8±0.2→95.3±0.1	76.8±2.8→96.9±0.9
	Hopper	113.0±0.7→113.2±0.2	111.4±1.2→111.1±0.8	101.8±7.5→105.8±11.3	102.2±9.6→113.0±0.2
	Walker2d	103.3±15.3→111.9±0.3	68.1±13.1→91.6±42.5	111.6±0.7→112.1±0.9	109.5±0.2→111.6±0.4
Expert	HalfCheetah	94.4±0.8→93.5±0.1	82.4±7.4→97.1±1.0	96.4±0.2 → 97.4±0.1	94.0±0.2→98.9±0.6
	Hopper	112.8±0.4→112.9±0.1	111.2±2.1→112.1±0.4	113.1±0.6 → 113.3±0.5	113.0±0.1→113.4±0.3
	Walker2d	110.4±0.0→111.2±0.4	103.8±7.6→110.6±0.3	110.7±0.3 → 112.8±1.1	109.9±0.3→111.0±0.2

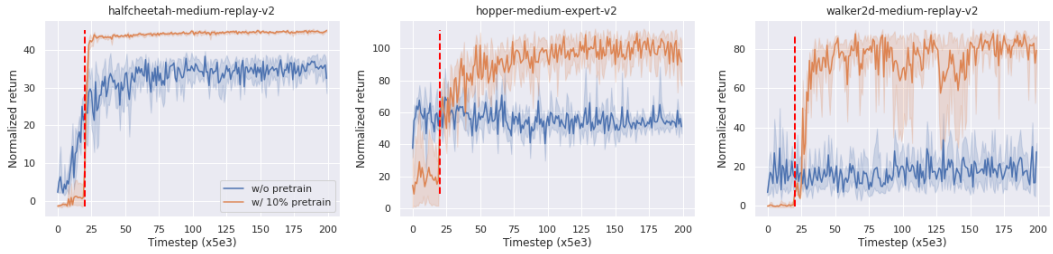

Figure 4: **Learning curves of TD3+BC.** The blue and orange curves are, respectively, the normalized scores of TD3+BC and TD3+BC pretrained with our method. The vertical red reference lines split the pretraining and main training phases. After the pretraining phase, TD3+BC combined with our method quickly outperforms the vanilla TD3+BC by a large margin.

mass maze (i.e. Proto, Diayn) in ExoRL [58] on TD3 [16] and CQL. See Appendix C for a more detailed setup for tasks and datasets and Appendix E for more implementation details.

4.1 Performance Improvement in Offline RL Benchmarks

To demonstrate the effectiveness of our method over existing offline RL methods, we evaluate our method on D4RL and Robomimic datasets. In Table 1, the normalized scores between the vanilla and the one combined with our method are compared for each environment and dataset in D4RL. Our method combined with the baselines (*i.e., AWAC, CQL, IQL, TD3+BC*) improves the corresponding original methods in most cases, across diverse environments and datasets. The blue scores in Table 1 mean increased performance compared to the original baseline algorithms. Specifically, one can observe that CQL exhibits a $+49\%$ increased performance on average compared to the original baselines.

Figure 4 shows the learning curves of TD3+BC and the results verify the effectiveness of our 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 13 in Appendix G.

Additional experiments are conducted on large-scale robotic manipulation tasks in Robomimic benchmark, to verify the effectiveness of our method for complex tasks. our method is evaluated with tasks containing suboptimal transitions, where our 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 our method. As can be seen, all the methods with our pretraining method are

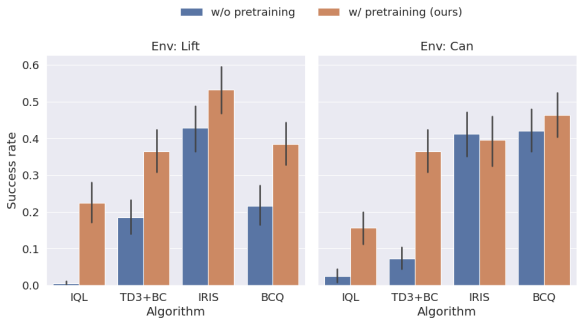

Figure 5: **Averaged success rate on the Robomimic benchmark.** We evaluate both vanilla methods without pretraining (blue) and methods with pretraining (orange). 7 out of 8 cases depict notably improved performance in both environments.

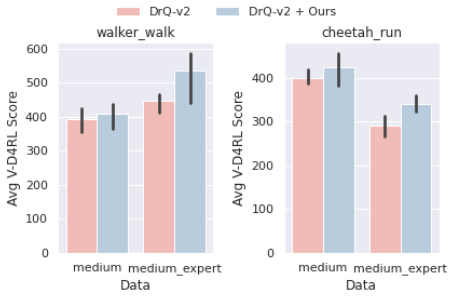

Figure 6: **DrQ-v2 and our method over 3 seeds in the V-D4RL benchmark.** We evaluate our method on the image-based environment, V-D4RL. Figure shows that our method enhances DrQ-v2.

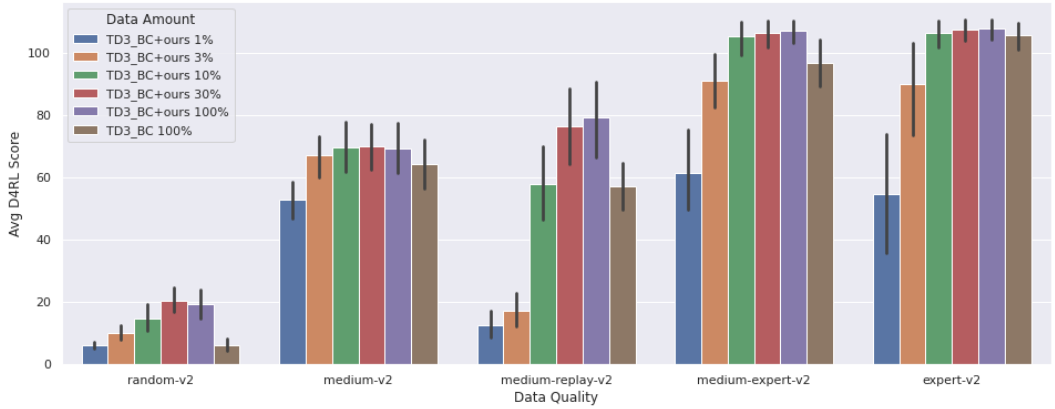

Figure 7: **Averaged normalized scores in reduced datasets across data quality.** This figure shows the overall performance of our method across reduced dataset *(i.e., 1%, 3%, 10%, 30%, 100%)* for three environments *(i.e., halfcheetah, hopper, walker2d)* in D4RL. From the overall results, we conclude 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.

improved over the baselines in seven out of eight cases. Therefore, we conclude that our method also effectively performs in solving more complex tasks. We also have conducted experiments on Adroit, 24-DOF environment, in Appendix D. We have applied our method to AWAC, IQL and TD3+BC, and conducted total 12 experiment (*i.e., four environments* $\times$ *three datasets*) for each algorithm over five seeds. In most cases, our method improves the performance. These results also demonstrate that our method is effective in solving complex tasks.

Our method can be extended to high-dimensional state space formulation. Similar to other vision-based DRL, our method can be built upon popular visual offline RL methods by replacing the original state input with representation input that is extracted from the visual encoder. We experiment on how our method scales to higher dimensions in the V-D4RL benchmark [36]. According to Figure 6, our method successfully improves the performance of the baseline, DrQ-v2 [59].

4.2 data efficiency across the Qualities of the Datasets

To validate the data efficiency of our method, regardless of the dataset quality, we have examined our 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 have uniformly sampled the transition segments (*i.e.,* (s, a, r, s')) from each dataset. Using these reduced, small datasets, we conduct both pre-training and RL training.

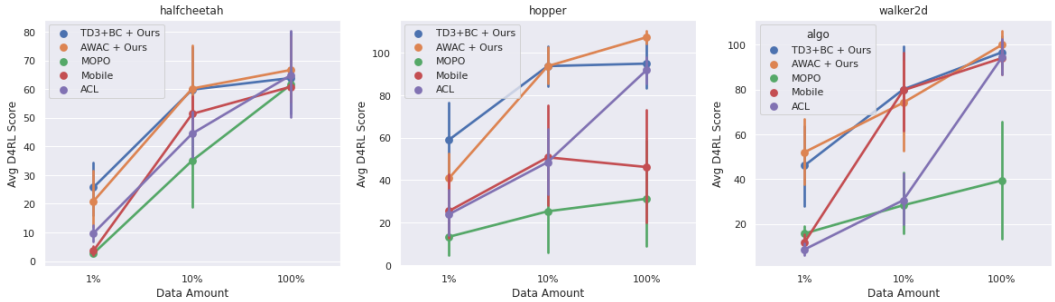

Figure 8: **Comparison of our method with the other approaches on D4RL.** We compare existing model-free offline RLs with our method to offline model-based RLs (*i.e., MOPO, MOBILE*) and a representation RL (*i.e., ACL*) on D4RL over 3 seeds. The graph shows integrated results over *medium, medium-replay, medium-expert* datasets. The results show that our method maintains the performance in reduced datasets, especially 1%, unlike the other approaches.

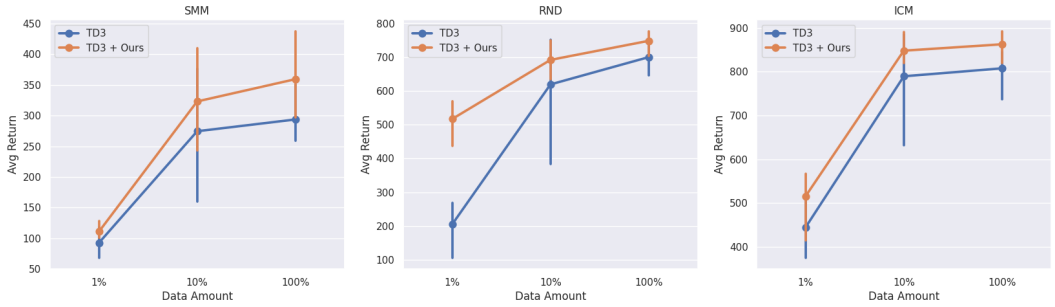

Figure 9: **Average returns in reduced datasets across the dataset collection strategies.** We evaluate our method over different dataset collection strategies *(i.e., SMM, RND, ICM)*. TD3 with our method outperforms the vanilla TD3 overall and even training with 10% of datasets outperforms the vanilla TD3 with full datasets. From the results, we demonstrate that our method is data-efficient regardless of the data distributions.

On the *random* datasets (a leftmost section in Figure 15), training with our 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 15), our method shows similar or improved results compared to the vanilla TD3+BC with full datasets by only using 3% of the datasets. On the other datasets (i.e. *medium-replay*, *medium-expert*, and *expert*), our method with 10% datasets totally outperforms the vanilla TD3+BC with full datasets. From the overall results in Figure 7, we conclude that our method guarantees better performance for offline RL even in 10% amount of the original datasets, regardless of the data quality of the dataset. These results demonstrate that our method is indeed data-efficient and requires minimal amount of static datasets in offline RL scheme.

We also compare our method with offline model-based RL and representation approaches. We apply our method to TD3+BC and AWAC. We adopt MOPO [64] and MOBILE [49] as representatives of offline model-based RL, ACL [56] as a representation representative. We conduct the experiments on D4RL, *medium, medium-replay, medium-expert* datasets over three seeds. Figure 8 shows integrated results over the datasets and Figure 16 shows details. The results show that our method maintains the performance in reduced datasets compared with the other approaches that spend extra training budget (e.g., training and forwarding the transition). As a result, we claim that our method is the most proper choice for data-efficient offline RL.

4.3 data efficiency across the Data Distributions

We assume that a small dataset would have a shifted distribution compared to a large dataset, for instance, some small datasets have narrow support of visited states. Based on the assumption we have made, we evaluate our method across different dataset collection strategies since each dataset has a

different data distribution. In ExoRL [58], we chose TD3 as a comparison algorithm and SMM [33], RND [7], and ICM [43], as *walker walk* task datasets. In [58], ICM shows the best performance, followed by RND, SMM, and TD3 shows the best performance in ICM. We compare TD3 to TD3 with our method in reduced datasets (*i.e., 1%, 10%, 100%)* over three seeds. To construct reduced datasets, we select the data from the front. With same reduced datasets, we conduct both pre-training and RL-training. Figure 9 shows the results. For all datasets, applying our method with only 10% of datasets outperforms vanilla TD3 with full datasets. Especially in RND, even training with 1% of datasets shows a significantly high average return.

Furthermore, we consider a*point mass maze* environment in ExoRL to investigate whether our method is effective even in narrow support of the visited states datasets. Figure 10 visualizes the trajectories of each reduced dataset collected by DIAYN [12], and Proto [60] strategies *(i.e., 1% of DIAYN, 7% of Proto)*. In comparison with Figure 2 in [58], our reduced dataset settings cover more narrow support of visited states. 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 10. To demonstrate our method is effective even with a dataset with this shifted state distribution, we evaluated our method on reduced *point mass maze* datasets described in Figure 10 over short (*reach top right*) and long (*reach bottom right*) goals with CQL. Figure 10 demonstrates that our method shows significant performance even with narrow data distribution. From the results, we conclude that our method is indeed more data-efficient than the other methods regardless of different choices of the data distribution.

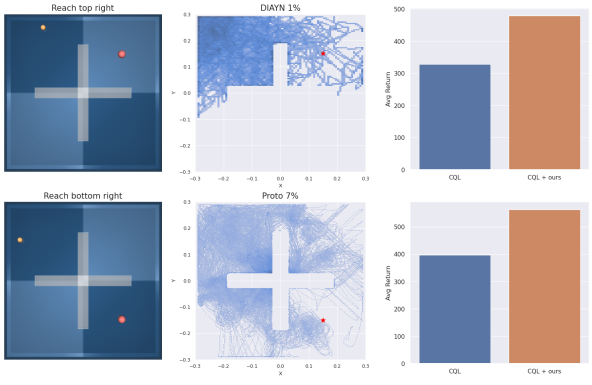

Figure 10: **Effectiveness of our method over narrow support of the visited states datasets.** (Left) Visualized goal-reaching point mass agents and trajectories with different goals, portions, and exploration methods. (Right) Averaged return of CQL trained with two datasets with and without our pretraining method.

5 Conclusion

In this paper, we propose a simple yet effective data-efficient offline RL method that pretrains a shared Q-network with the transition dynamics prediction task, maintaining reasonable performance even with a small training dataset. To pretrain the Q-network, we design a shared network architecture that outputs predictions of the next state and Q-value. This structure makes our method easy to apply to any existing offline RL algorithms and efficiently boosts data efficiency.

To demonstrate the effectiveness of our strategy, we conduct experiments with various settings in offline RL. From the results, we demonstrate that our method significantly improves the performance of existing offline RL algorithms over D4RL, Robomimic and V-D4RL benchmarks. We also demonstrate that our method is indeed data-efficient across the different data qualities from D4RL and the different data distributions from ExoRL.

Limitations & Future Works. This paper might have considered with a standard problem setting for offline RL. Since conventional popular offline RL methods often have chosen model-free architecture that learns the Q value function to learn the best policy, the proposed design organizes to exploit such structures. However, as suggested in [47, 54], offline pretraining can also benefit diverse domains (e.g. unsupervised learning, goal-conditioned RL). Simpleness and plug-and-play capability enable the architecture to be advantageous for extensive applications. We leave future work to expand our method toward various offline RL problems, e.g., offline to online RL, goal-conditioned RL, and real-world applications.

References

[1] R. Agarwal, D. Schuurmans, and M. Norouzi. An optimistic perspective on offline reinforcement learning. In *International conference on machine learning*, pages 104–114. PMLR, 2020.

[2] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as i can, not as i say: Grounding language in robotic affordances. (arXiv:2204.01691), Aug. 2022. arXiv:2204.01691 [cs].

[3] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[4] C. Bhateja, D. Guo, D. Ghosh, A. Singh, M. Tomar, Q. Vuong, Y. Chebotar, S. Levine, and A. Kumar. Robotic offline rl from internet videos via value-function pre-training. (arXiv:2309.13041), Sept. 2023. arXiv:2309.13041 [cs].

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. (arXiv:2307.15818), July 2023. arXiv:2307.15818 [cs].

[6] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. (arXiv:2212.06817), Aug. 2023.

[7] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[8] Y. Chebotar, Q. Vuong, K. Hausman, F. Xia, Y. Lu, A. Irpan, A. Kumar, T. Yu, A. Herzog, K. Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pages 3909–3928. PMLR, 2023.

[9] J. Cheng, R. Qiao, G. Xiong, Q. Miao, Y. Ma, B. Li, Y. Li, and Y. Lv. Scaling offline model-based rl via jointly-optimized world-action model pretraining. *arXiv preprint arXiv:2410.00564*, 2024.

[10] M. Deisenroth and C. 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)*, pages 465–472, 2011.

[11] P. D'Oro, M. Schwarzer, E. Nikishin, P.-L. Bacon, M. G. Bellemare, and A. Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

[12] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[13] Y. Feng, N. Hansen, Z. Xiong, C. Rajagopalan, and X. Wang. Finetuning offline world models in the real world. Oct. 2023.

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

[15] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, page 20132–20145. Curran Associates, Inc., 2021.

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

[17] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, page 2052–2062. PMLR, May 2019.

[18] Z. D. Guo, M. G. Azar, B. Piot, B. A. Pires, and R. Munos. Neural predictive belief representations. *arXiv preprint arXiv:1811.06407*, 2018.

[19] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[20] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. (arXiv:1912.01603), 2019.

[21] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, page 2555–2565. PMLR, 2019.

[22] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[23] N. Hansen, H. Su, and X. Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

[24] N. Hansen, X. Wang, and H. Su. Temporal difference learning for model predictive control. (arXiv:2203.04955), July 2022. arXiv:2203.04955 [cs].

[25] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[26] Z.-W. Hong, A. Kumar, S. Karnik, A. Bhandwaldar, A. Srivastava, J. Pajarinen, R. Laroche, A. Gupta, and P. Agrawal. Beyond uniform sampling: Offline reinforcement learning with imbalanced datasets. *Advances in Neural Information Processing Systems*, 36:4985–5009, 2023.

[27] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. (arXiv:2110.06169), Oct. 2021. arXiv:2110.06169 [cs].

[28] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. (arXiv:2004.13649), Mar. 2021. arXiv:2004.13649 [cs, eess, stat].

[29] A. Kumar, R. Agarwal, D. Ghosh, and S. Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020.

[30] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[31] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. (arXiv:2006.04779), Aug. 2020. arXiv:2006.04779 [cs, stat].

[32] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.

[33] L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[34] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Proceedings of the 5th Conference on Robot Learning*, page 1702–1712. PMLR, Jan. 2022.

[35] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. (arXiv:2005.01643), Nov. 2020. arXiv:2005.01643 [cs, stat].

[36] C. Lu, P. J. Ball, T. G. Rudner, J. Parker-Holder, M. A. Osborne, and Y. W. Teh. Challenges and opportunities in offline reinforcement learning from visual observations. *arXiv preprint arXiv:2206.04779*, 2022.

[37] A. Mandlekar, F. Ramos, B. Boots, S. Savarese, L. Fei-Fei, A. Garg, and D. 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)*, pages 4414–4420. IEEE, 2020.

[38] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[39] F. S. Melo and M. I. Ribeiro. Q-learning with linear function approximation. 2007.

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

[41] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36, 2024.

[42] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[43] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[44] R. Rafailov, K. B. Hatch, V. Kolev, J. D. Martin, M. Phielipp, and C. Finn. Moto: Offline pre-training to online fine-tuning for model-based robot learning. In *Conference on Robot Learning*, pages 3654–3671. PMLR, 2023.

[45] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[46] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

[47] M. Schwarzer, N. Rajkumar, M. Noukhovitch, A. Anand, L. Charlin, R. D. Hjelm, P. Bachman, and A. C. Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.

[48] Y. Seo, K. Lee, S. L. James, and P. Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, page 19561–19579. PMLR, 2022.

[49] Y. Sun, J. Zhang, C. Jia, H. Lin, J. Ye, and Y. Yu. Model-bellman inconsistency for model-based offline reinforcement learning. In *International Conference on Machine Learning*, pages 33177–33194. PMLR, 2023.

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

[51] R. S. Sutton, A. G. Barto, et al. Introduction to reinforcement learning. vol. 135, 1998.

[52] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[53] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[54] P. Wu, A. Majumdar, K. Stone, Y. Lin, I. Mordatch, P. Abbeel, and A. Rajeswaran. Masked trajectory models for prediction, representation, and control. In *International Conference on Machine Learning*, pages 37607–37623. PMLR, 2023.

[55] T. Xie, N. Jiang, H. Wang, C. Xiong, and Y. Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *Advances in neural information processing systems*, 34:27395–27407, 2021.

[56] M. Yang and O. Nachum. Representation matters: Offline pretraining for sequential decision making. In *International Conference on Machine Learning*, pages 11784–11794. PMLR, 2021.

[57] R. Yang, H. Zhong, J. Xu, A. Zhang, C. Zhang, L. Han, and T. Zhang. Towards robust offline reinforcement learning under diverse data corruption. *arXiv preprint arXiv:2310.12955*, 2023.

[58] D. Yarats, D. Brandfonbrener, H. Liu, M. Laskin, P. Abbeel, A. Lazaric, and L. Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

[59] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[60] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 11920–11931. PMLR, 2021.

[61] D. Yarats, I. Kostrikov, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International conference on learning representations*, 2021.

[62] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pages 10674–10681, 2021.

[63] T. Yu, A. Kumar, Y. Chebotar, K. Hausman, C. Finn, and S. Levine. How to leverage unlabeled data in offline reinforcement learning. In *International Conference on Machine Learning*, pages 25611–25635. PMLR, 2022.

[64] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

A Pretraining Q-network for Online RL (Off-Policy)

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

Input: Learning rate α
Initialize parameters φ, ψ 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 φ 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 α
Initialize parameters φ, ψ
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 φ 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) [19] and twin delayed deep deterministic policy gradient algorithm (TD3) [16]. 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 2 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 3 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 2: 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 3: 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 2 shows worse results than Table 3.

We also applied another approach introduced in section B to online RL settings. The results, shown in Table 4, indicate that more than half exhibit enhanced performance compared to reported scores in Table 2.

Table 4: 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 previouse 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 5. 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 our shared Q-network can be expanded in other directions and we expect that it holds significant potential for further research.

Table 5: **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

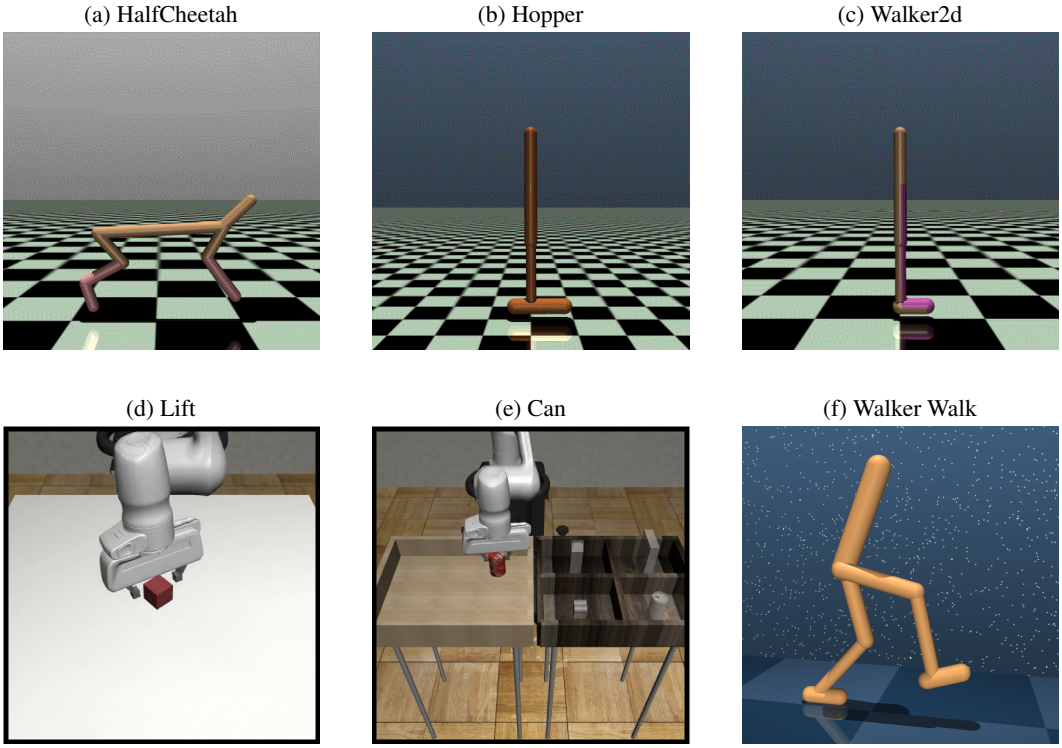

Figure 11: **Illustrations of environments.**

C Tasks and Datasets

In this section, we provide detailed experimental setups for the tasks and datasets. Illustrated environments can be found in Figure 11

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 our 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 [14] benchmark to validate that our method can be adopted to different complex domains. An illustration of the Adroit environment can be found in Figure 12. 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 [45] 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.

For experiments, we compared AWAC, IQL, and TD3+BC with/without our pretraining method over 5 seeds. Table 6 yields averaged normalized scores for each task. Overall, learning with our pretraining phase demonstrates enhanced performance. From these results, we conclude that our method can be effective in complex domains not only tabletop but dexterous manipulation as well.

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

[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`

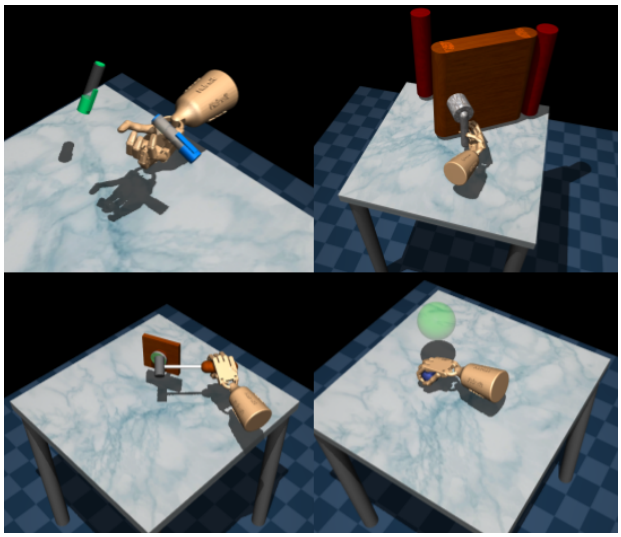

Figure 12: **The tasks of Adroit. (top left)** Pen - aligning a pen with a target orientation, **(top right)** Door - opening a door, **(bottom left)** Hammer - hammering a nail into a board, **(bottom right)** Relocate - moving a ball to a target position.

Table 6: **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

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.

F Discussions

In this section, we address the potential concerns regarding our method's novelty since it closely connects with prior approaches in relevant fields. We provide our detailed discussions in separate subsections of each topic.

Representation Learning. Over recent years, the field has observed a significant amount of literature working on predictive representation in RL. Concerning the similarity with prior works, we claim that the idea of pretraining shared Q-network for improving data efficiency is remarkable. Our method pretrains the neural networks with the next state prediction objective to improve an underlying RL agent's performance and data efficiency similar to [46, 18]. However, [46] has proposed an online

training method in a self-supervised learning manner whereas our method considers supervised learning for pretraining. Since the self-predictive task in [46] is conducted in latent space, representation learning is essentially involved with the task.

Therefore, adopting advanced training techniques including data augmentation [61] and the use of a target encoder [25] significantly affect the RL agent's performance. Additionally, [46] suggests a self-supervised representation learning with the latent transition prediction task in the online RL regime. In comparison, our method alleviates an introduction of extra techniques other than the shared network architecture, proving superior performance in offline RL benchmarks of diverse environments, e.g. locomotion and manipulation tasks.

[18] has presented an unsupervised learning method that encodes the *belief state* capturing sufficient information of the hidden true state from a past interaction history. In other words, the main interest of [18] is how the neural network architecture trained with unsupervised learning extracts adequate information concerning the true state in POMDP, not how the underlying RL method given rich representation performs decision-making problem well. Specifically, the network architecture in [18] is based on GRU, RNN based sequential network, and predicts a next observation o_{t+1} using action a_t and a belief state b_t that contains the partial information of the previous trajectory. Conversely, our method is implemented on MLP with the shared network architecture and predicts the next state s_{t+1} using current state s_t and action a_t without a past history.

Model-based RL. One might argue that our method lacks novelty with the idea of training a neural network with the transition dynamics prediction task. Obviously, the idea of approximating the transition dynamics [50] for downstream RL training is not what we first suggest. However, we contend that our method has a few refuting viewpoints with previous similar works. TDMPC [24] and TDMPC2 [23] are model-based single and multi-task RL approaches, which recursively feed the output of the same network (i.e. the encoder and task embedding network) for the transition model and value learning. The outputs of the shared backbone networks correspond to the latent representation and task embedding vector, respectively, and most latent model-based RL approaches including TDMPC reuse the outputs for the transition model and value learning. On the other hand, our method presents a shared network architecture resembling the dueling architecture [53] to pretrain the shared backbone network with a separated stream (a header) of the transition model and Q-network. Additionally, this paper presents a two-phase training scheme: the transition model combined with the shared network is trained with the transition dynamics prediction task in the first phase and the Q-network, consisting of an MLP header and the shared network initialized with the parameter of the shared network in the first phase, is trained with the downstream RL value learning task in the second phase.

JOWA [9] is an offline world model for multi-task RL with a shared Transformer backbone network for sequential a next-token prediction task. By modeling the decision-making problem to the sequential token prediction task, the backbone network, tokenizer, and header are trained in a supervised manner with the offline dataset. While the main purpose of JOWA is scaling an offline world model across multiple tasks with generalized performance over unseen tasks, this paper intends to improve the data efficiency of conventional offline RL approaches in single-task RL. Furthermore, our method alleviates additional training after offline RL training with a novel two-phase training strategy while JOWA allows few-shot fine-tuning for sample efficient transfer with a multi-game environment. Even with a similar purpose of data efficiency, our method entails a minimal algorithmic change with a consistent training budget compared to previous approaches.

Dreamer [22] has brought a notable advancement in model-based RL. Dreamer suggests a world model for decision-making with a considerate design of the latent transition model and reconstructive objective. Since jointly learning an accurate world model and actor in a multi-task environment is challenging, the expensive cost of collecting samples often becomes problematic. In contrast, our method does not necessitate extra modifications of conventional offline RL and proves its sufficient performance gains in comprehensive experiments. Considering previous improvements in representation learning usually involve state-of-the-art design choices (e.g. data augmentation), this paper would contribute to reasonable architectural achievements for researchers by presenting a minimal training structure with verified performance profit.

G Learning Curves

In this section, we provide the full results of learning curves in the section 4.1 for further information.

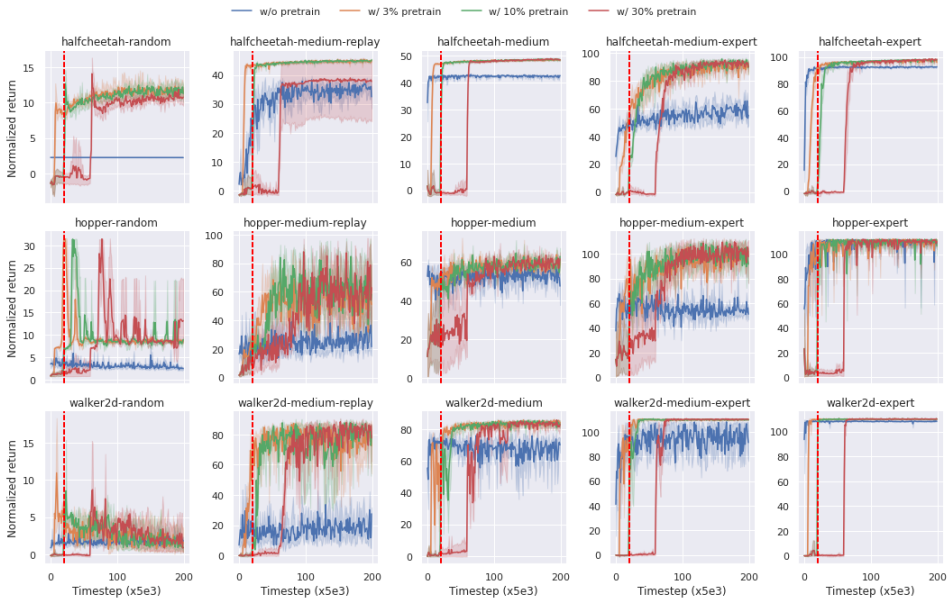

Figure 13: **Learning curves of TD3+BC on the D4RL benchmark.**

H Rank of Latent Space during the Learning Time

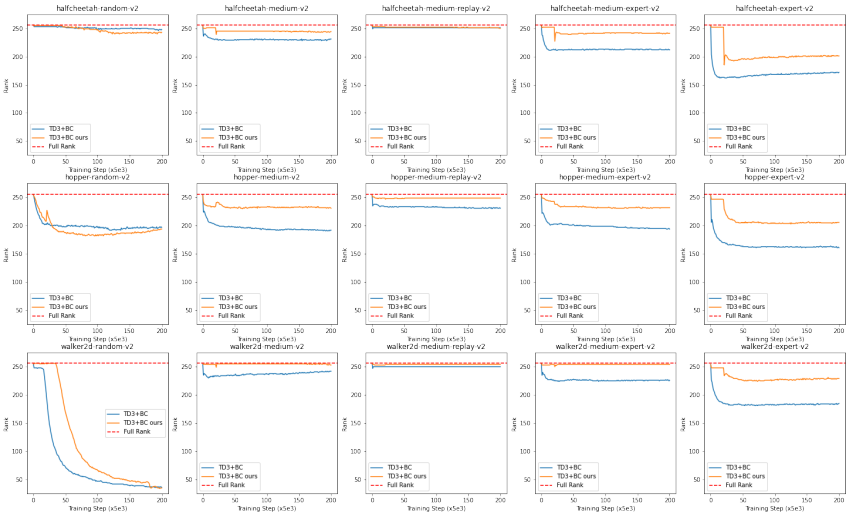

Figure 14: **The Rank of the latent space of Q-network during the training. (Full version)**

I 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 7: **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

J Experiments with Various Amount of Data

In this section, we provide more details in section 4.2 of the dataset size. We conducted each experiment with the same settings in subsection 4.1 over 5 seeds and reported the results that exhibit averaged normalized scores.

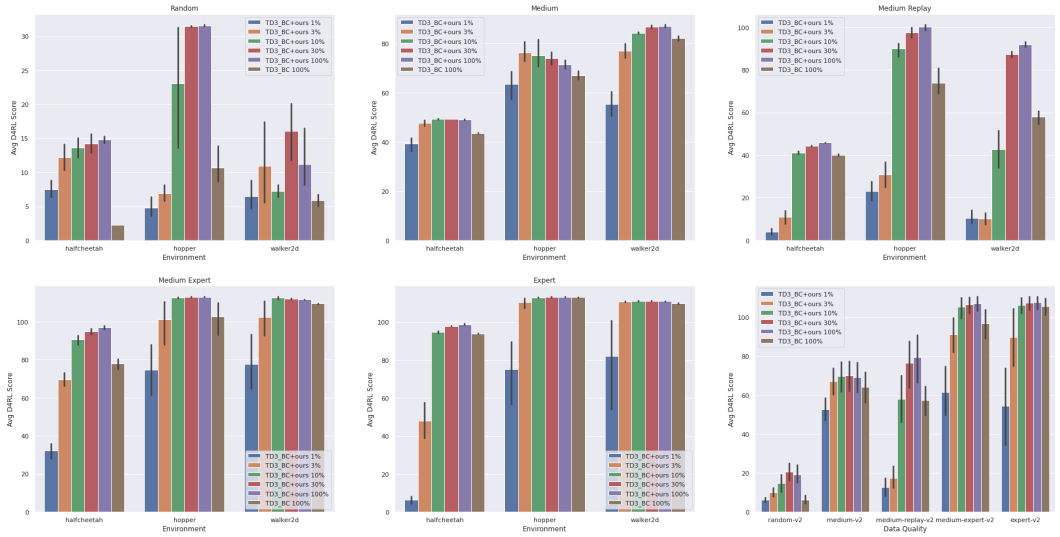

Figure 15: **Averaged normalized scores across dataset optimal quality and sizes.** This figure compares the performance of our method with TD3+BC in reduced datasets *(i.e., 1%, 3%, 10%, 30%, 100% of each dataset)* to vanilla TD3+BC across the data quality *(i.e., random, medium, medium replay, medium 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.

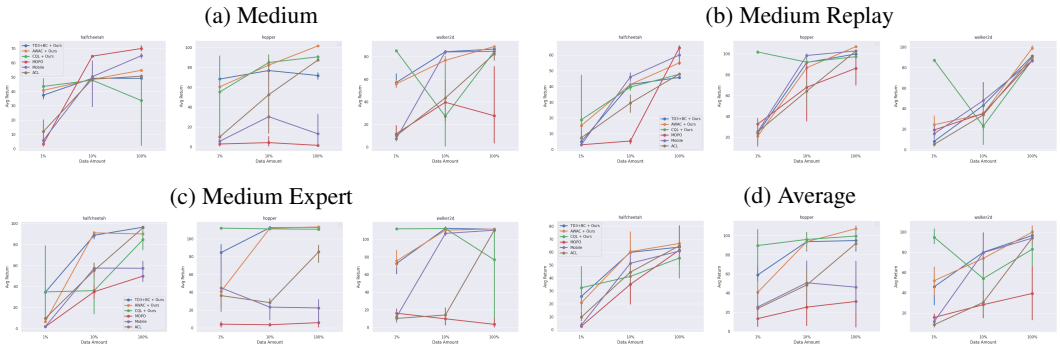

Figure 16: **Comparison with offline model-based RL and representation approaches.** We compare TD3+BC, AWAC, CQL with ours to offline model-based RLs (*i.e., MOPO, Mobile*) and a representation RL (*i.e., ACL*) on D4RL over 3 seeds. The gragh shows results over *medium, medium-replay, medium-expert* datasets. The results show that our method maintains the performance in reduced datasets, especially 1%, unlike the other approaches.

Table 8: **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 9: **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

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the abstract and introduction, we summarize our work and explain our goal of this research.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.

- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We describe the limitations of our work and future work to overcome those issues in section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Although we do not provide any proof in this paper, we use some theoretical results from prior work in Section 3. Before utilizing the theorem of prior work, we explain how the work is related to our work.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.

- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We explain our method in detail, and provide the example code in supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have used open source benchmarks and datasets, and provide the example code in supplementary material. Therefore, data and code are open accessible.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In experiment, section 4, we explained our experiment settings

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In all experiments results, we described mean and standard deviation. See section 4

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In Appendix E, we denoted the resource of computation.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We read the NeurIPS Code of Ethics, and there are no violation in our work.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer:

Justification: Our work will mostly affect engineering field rather than societal since we experimented on specific locomotion tasks.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our method shows robust results.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We wrote reference faithfully.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We described our proposed new assets in section 3. Moreover, we provided the code of implementation in supplemental materials.

    Guidelines:

    - The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our research does not involve crowdsourcing or research with human subjects. We only conducted extensive experiments on MuJoCo locomotion and Robomimic manipulation simulation tasks.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our research does not involve crowdsourcing or research with human subjects. We only conducted extensive experiments on MuJoCo locomotion and Robomimic manipulation simulation tasks.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: Our work is not related with LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.