Discriminator-Weighted Offline Imitation Learning from Suboptimal Demonstrations

Haoran Xu [1] **Xianyuan Zhan** [2] **Honglei Yin** [1] **Huiling Qin** [1]

Abstract

We study the problem of offline Imitation Learning (IL) where an agent aims to learn an optimal expert behavior policy without additional online environment interactions. Instead, the agent is provided with a supplementary offline dataset from suboptimal behaviors. Prior works that address this problem either require that expert data occupies the majority proportion of the offline dataset, or need to learn a reward function and perform offline reinforcement learning (RL) afterwards. In this paper, we aim to address the problem without additional steps of reward learning and offline RL training for the case when demonstrations contain a large proportion of suboptimal data. Built upon behavioral cloning (BC), we introduce an additional discriminator to distinguish expert and non-expert data. We propose a cooperation framework to boost the learning of both tasks, Based on this framework, we design a new IL algorithm, where the outputs of discriminator serve as the weights of the BC loss. Experimental results show that our proposed algorithm achieves higher returns and faster training speed compared to baseline algorithms. Code is available at https://github.com/ryanxhr/DWBC.

1. Introduction

The recent success of reinforcement learning (RL) in many domains showcases the great potential of applying this family of learning methods to real-world applications. A key prerequisite for RL is to design a reward function that specifies what kind of agent behavior is preferred. However, in many real-world applications, designing a reward function is prohibitively difficult (Ng et al., 1999; Irpan, 2018). By contrast, imitation learning (IL) provides a much easier way to leverage the reward function implicitly from the collected demonstrations and has achieved great success in many sequential decision making problems (Pomerleau, 1989; Ng & Russell, 2000; Ho & Ermon, 2016).

However, popular IL methods such as behavioral cloning (BC) (Pomerleau, 1989) and generative adversarial imitation learning (GAIL) (Ho & Ermon, 2016) assume the expert demonstration is optimal. Unfortunately, it is often difficult to obtain sufficient optimal demonstrations for many real-world tasks, because human experts often make mistakes due to various reasons, such as the difficulty of the task, partial observability of the environment, or the presence of distraction. Given such noisy expert demonstrations, which contain records of both optimal and non-optimal behaviors, BC and GAIL all fail to imitate the optimal policy (Wu et al., 2019a; Ma, 2020). Current methods that deal with suboptimal demonstrations either require additional labels, which can be done explicitly by annotating each demonstration with confidence scores by human experts (Wu et al., 2019a), or implicitly by ranking noisy demonstrations according to their relative performance through interacting with the environment (Brown et al., 2019; 2020; Zhang et al., 2021). However, human annotation and environment interaction are laborious and expensive in real-world settings, such as in medicine, healthcare, and industrial processes.

In this work, we investigate a pure offline learning setting where the agent has access to neither the expert nor the environment for additional information. The agent, instead, has only access to a small pre-collected dataset sampled from the expert and a large batch offline dataset sampled from one or multiple behavior policies that could be highly sub-optimal. This strictly offline imitation learning problem arises in many real-world problems, where environment interactions and expert annotations are costly. Prior works that address the problem are based on variants of BC or inverse RL. Sasaki & Yamashina (2021) reuses another policy learned by BC as the weight of original BC objective. However, this requires that expert data occupy the majority proportion of the offline dataset, otherwise the policy will be misguided to imitate the suboptimal data. Zolna et al.

[1]JD Technology, Beijing, China [2]Institute for AI Industry Research (AIR), Tsinghua University, Beijing, China. Correspondence to: Haoran Xu <ryanxhr@gmail.com>, Xianyuan Zhan <zhanxianyuan@air.tsinghua.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

(2020a) first learns a reward function that prioritizes expert data over others and then performs offline RL based on this reward function. This algorithm is extremely expensive to run, requiring solving offline RL in an inner loop, which itself is a challenging problem and prone to training instability (Kumar et al., 2019) and hyperparameter sensitivity (Wu et al., 2019b).

In this paper, we propose an offline imitation learning algorithm to learn from demonstrations that (perhaps) contain a large proportion of suboptimal data without additional steps of reward learning and offline RL training. Built upon the task of BC, we introduce an additional task to learn a discriminator to distinguish expert and non-expert data. We propose a cooperation framework to learn the policy and discriminator cooperatively and boost the performance of both tasks. Based on this framework, we adopt a worst-case error minimization strategy to the policy such that the discriminator can be more robustly learned. This results in a new offline policy learning objective, and surprisingly, we find its equivalence to a generalized BC objective, where the outputs of the discriminator serve as the weights of the BC loss function. We thus term our resulting algorithm Discriminator-Weighted Behavioral Cloning (DWBC). Experimental results show that DWBC achieves higher returns and faster training speed compared to baseline algorithms under different scenarios.

To summarize, the contributions of this paper are as follows.

- We propose a cooperation framework to learn the policy and discriminator cooperatively and boost the performance of both tasks (Section 3.2);
- Based on the proposed framework, we design an effective and light-weighted offline IL algorithm with a worst-case error minimization strategy (Section 3.3);
- We present promising comparison results with comprehensive analysis for our algorithm, which surpasses the state-of-the-art methods (Section 5.3);
- As a by-product, we show that the discriminator in our algorithm can be used to perform offline policy selection, which is of independent interest (Section 5.4).

2. Preliminary

2.1. Problem Setting

We consider the standard fully observed Markov Decision Process (MDP) setting (Sutton et al., 1998), $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, r, \gamma, d_0\}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the MDP's transition probability, r is the reward function, $\gamma \in [0, 1)$ is the discount factor for future reward and d_0 is the initial distribution. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maps from state to distribution over actions. We denote $d^\pi \in \Delta(\mathcal{S} \times \mathcal{A})$ as the discounted state-action distribution of π under transition kernel P, that is, $d^\pi = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t d_t^\pi$, where $d_t^\pi \in \Delta(\mathcal{S} \times \mathcal{A})$ is the distribution of $\left(s^{(t)}, a^{(t)}\right)$ under π at step t. Following the standard IL setting, the ground truth reward function r is unknown. Instead, we have the demonstrations collected by the expert denoted as $\pi_e : \mathcal{S} \to \Delta(\mathcal{A})$ (potentially stochastic and not necessarily optimal). Concretely, we have an expert dataset in the form of i.i.d tuples $\mathcal{D}_e = \{s_i, a_i, s_i'\}_{i=1}^{n_e}$ where (s, a) is sampled from distribution d^{π_e} and s' is sampled from $P(s, a)$.

In our problem setting, we also have an offline static dataset consisting of i.i.d tuples $\mathcal{D}_o = \{s_i, a_i, s_i'\}_{i=1}^{n_o}$ s.t. $(s, a) \sim \rho(s, a), s' \sim P(s, a)$, where $\rho \in \Delta(\mathcal{S} \times \mathcal{A})$ is an offline state-action distribution resulting from some other behavior policies. Note that these behavior policies could be much worse than the expert π_e. Our goal is to only leverage the offline batch data $\mathcal{D}_b = \mathcal{D}_e \cup \mathcal{D}_o$ to learn an optimal policy π with regard to optimizing the ground truth reward r, without any interaction with the environment or the expert.

2.2. A Generalized Behavioral Cloning Objective

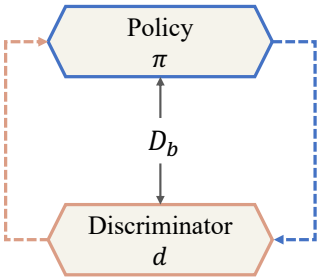
In order to discard low-quality demonstrations and only clone the best behavior available, we consider a generalized behavioral cloning objective to imitate demonstrations unequally, that is,

$$\min_{\pi} \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_b} \left[-\log \pi(a|s) \cdot f(s, a)\right], \tag{1}$$

where $f : \mathcal{S} \times \mathcal{A} \to [0, 1]$ denotes an arbitrary weight function. Existing offline IL methods can simply be recovered by picking one of the valid weight configurations:

- If $f(s, a) = 1$ for $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$, the objective (1) corresponds to the vanilla BC objective.
- If $f(s, a) = \pi'(a|s)$, where π' is an old policy which was previously optimized with $\mathcal{D}_b$, the objective (1) corresponds to the objective of Behavioral Cloning from Noisy Demonstrations (Sasaki & Yamashina, 2021). Since $\sum_a \pi'(a|s) = 1$ for $\forall s \in \mathcal{S}$ is satisfied, $\pi'(a|s)$ can be interpreted as the weights for weighted action sampling.
- If $f(s, a) = \mathbb{1}\left[A^\pi(s, a)\right]$, where $\mathbb{1}$ is the indicator function which creates a boolean mask that eliminates samples which are thought to be worse than the current policy, the objective (1) corresponds to the objective of Offline Reinforced Imitation Learning (Zolna et al., 2020a).

The objective (1) can also be deemed as the objective of Soft Q Imitation Learning (Reddy et al., 2020) with $f(s, a) = 1$ for $(s, a) \in \mathcal{D}_e$ and $f(s, a) = 0$ for $(s, a) \in \mathcal{D}_o$ in online IL literature; or the objective of off-policy actor-critic (Off-PAC) algorithm (Degris et al., 2012) with $f(s, a) = Q^\pi(s, a) \cdot \pi(a|s)/\pi_b(a|s)$ in online RL literature.



New BC Task

$$L_\pi = \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_b}[-\log \pi(a|s) \cdot f(d(s,a,\log\pi))]$$

New Discriminating Task

$$L_d = \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log d(s,a,\log\pi)] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_o}[-\log(1-d(s,a,\log\pi))] - \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log(1-d(s,a,\log\pi))]$$

Figure 1. Illustration of our proposed cooperation framework to alternately learn π and d. In this framework, π uses the outputs of d as the weights to perform a new BC Task; d includes π as additional input to form a new Discriminating Task. This framework is different from GAN-style frameworks in that: 1) π and d are learned cooperatively rather than adversarially; 2) the training of π and d are decoupled into individual objectives rather than sharing one coupled objective.

3. Methodology

We now continue to describe our approach for offline imitation learning from demonstrations that (perhaps) contain large-proportional suboptimal data, without additional steps of reward learning and offline RL training. Built upon the task of BC, we introduce an additional task to learn a discriminator to distinguish expert and non-expert data. We propose a cooperation framework to boost the performance of both tasks. Based on this framework, we adopt a worst-case error minimization strategy to the policy such that the discriminator can be more robustly learned. This results in a new generalized BC objective, we then provide the interpretation of weights in our generalized BC objective, this gives the intuition about why our method can work.

3.1. Learn the Policy and Discriminator Separately

It is obvious that we can avoid the negative impact of suboptimal demonstrations presented in $\mathcal{D}_o$ by only imitating $\mathcal{D}_e$, which can be written as

$$\min_\pi \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log \pi(a|s)]\,. \tag{2}$$

We call the task of learning a policy using objective (2) as **BC Task**. The drawback of BC task is that it does not fully utilize the information from $\mathcal{D}_o$, the resulting policy may not be able to generalize and will suffer from compounding errors due to the potential limited size and state coverage of $\mathcal{D}_e$ (Ross et al., 2011). If we can select those high-reward transitions from $\mathcal{D}_o$ and combine them with $\mathcal{D}_e$, we are expected to get a better policy.

Now let's consider another different task, which aims to learn a discriminator by contrasting expert and non-expert transitions, given by

$$\min_d \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log d(s,a)] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_o}[-\log(1-d(s,a))]\,. \tag{3}$$

Objective (3) is similar to how the discriminator is trained in GAIL (Ho & Ermon, 2016) and GAN (Goodfellow et al., 2014), except that the second term is sampled from a fixed dataset instead of new samples drawn from the learned policy by interacting with the environment.

However, optimizing objective (3) will make the learned discriminator assign 1 to all transitions from $\mathcal{D}_e$ and 0 to all transitions from $\mathcal{D}_o$. This limiting behavior is unsatisfactory because $\mathcal{D}_o$ can contain some successful (high-reward) transitions. This bears similarity to the positive-unlabeled classification problem (Elkan & Noto, 2008), where both positive and negative samples exist in the unlabeled data.

To solve this problem, previous works adopt the approach from positive-unlabeled (PU) learning (du Plessis et al., 2015; Xu & Denil, 2019; Zolna et al., 2020b). The main idea is to re-weight different losses for positive and unlabeled data, in order to obtain an estimate of model loss on negative samples that is not directly available. Applying PU learning to objective (3) yields the following objective:

$$\min_d \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log d(s,a)] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_o}[-\log(1-d(s,a))] - \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log(1-d(s,a))]\,, \tag{4}$$

where η is a hyperparameter, corresponds to the proportion of positive samples to unlabeled samples. We call the task of learning a discriminator using objective (4) as **Discriminating Task**. Intuitively, the second term in (4) could make $d(s,a)$ of state-action pairs from $\mathcal{D}_e$ become 0 if similar state-action pairs are included in $\mathcal{D}_o$, and the third term in (4) balances the impact of the second term, i.e., avoids $d(s,a)$ of state-action pairs from $\mathcal{D}_e$ becoming 0.

However, using information from state and action may be insufficient for the learning of Discriminating task. For example, suppose $\mathcal{D}_e$ comprises near-end transitions of expert trajectories, whereas $\mathcal{D}_o$ comprises near-front transitions of expert trajectories and transitions from non-expert trajectories. In this case, it is hard for the discriminator to

distinguish between expert transitions and non-expert transitions in $\mathcal{D}_o$, as states of transitions in $\mathcal{D}_o$ bear a large similarity (these states are near the initial state), but have a large difference from states in $\mathcal{D}_e$.

To summarize, BC Task aims to imitate the expert behavior from $\mathcal{D}_e$, but ignores the valuable information from $\mathcal{D}_o$; Discriminating Task aims to contrast expert and non-expert transitions from $\mathcal{D}_e$ and $\mathcal{D}_o$, but only uses state-action information as input. Both tasks lack enough information to improve their own performance, which however, can be obtained from the other task, as we will elaborate next.

3.2. Learn the Policy and Discriminator Cooperatively

We propose a cooperation framework to learn the policy and discriminator cooperatively. In this framework, we aim to boost the performance of BC Task and Discriminating Task by incorporating the policy into the training of the discriminator and effectively using the discriminator to help the training of the policy. As illustrated in Figure 1, the policy π uses the discriminator d to perform a new BC Task (i.e., generalized behavioral cloning as introduced in Section 2.2), where the weight is a function of d. The discriminator d also gets information from the policy π as additional input, yielding a new Discriminating Task.

Suppose that d is well-learned to be able to contrast expert and non-expert transitions in $\mathcal{D}_b = \mathcal{D}_e \cup \mathcal{D}_o$, the policy will become better if one can choose an appropriate weight function f to make π only imitate the expert data in $\mathcal{D}_b$. By this way, we are able to use the entire dataset $\mathcal{D}_b$ but get rid of the negative impact of those low-quality data.

Supposed π is learned to be optimal, i.e., assigns large probabilities to expert actions in expert states, the discriminator will receive additional learning signal. It will be easier for the discriminator to contrast expert and non-expert transitions in $\mathcal{D}_o$, as $\pi(a|s)$ will be large if (s, a) are from expert behaviors and small if (s, a) are from non-expert behaviors. Without this information from π, the discriminator is much harder to learn by only using information from (s, a).

A keen reader may find the similarity of our propased framework and GAN-style frameworks (Goodfellow et al., 2014; Ho & Ermon, 2016), where the policy and the discriminator are also jointly learned. However, the learning strategy of our framework has several differences compared with GAN-style frameworks. In GAN, the policy aims to generate expert data and the discriminator aims to distinguish between expert data and policy generated data. If the policy perfectly matches the expert, the discriminator will be unable to distinguish well, and vice versa. This means that GAN adopts an adversarial framework, where task A and task B are contradictory to each other, an improved performance of one task will lead to a deteriorated performance of another task. In contrast to adversarial, our framework is cooperative, task A and task B cooperate with each other to help both tasks, an improved performance of one task will also lead to an improved performance of another task.

Moreover, GAN-style frameworks need to solve a min-max optimization problem (i.e., $\min_d \max_\pi \mathcal{L}(d, \pi)$) and is known to suffer from issues such as traning instability and mode collapse (Arjovsky et al., 2017). Whereas our framewwork allows the decoupled training of π and d. They can both learn with their own objectives in a fully supervised manner (see Figure 1), which is very easy to train and computationally cheap.

3.3. Discriminator-Weighted Behavioral Cloning

It is obvious that, in our proposed framework, there exists multiple valid choices of weight function f that can make the policy imitate those high-reward transitions in $\mathcal{D}_b$. For example, f could be $\mathbb{1}[d > 0.5]$ or $\exp(d/\beta)$, where $\beta > 0$ is a hyperparameter and $\mathbb{1}$ is the indicator function. However, does there exsit one principled solution of f?

Notice that now π appears in the input of d, this means that imitation information from $\log \pi$ will affect $\mathcal{L}_d$, and further impact the learning of d. Hence both d and $\mathcal{L}_d$ become functionals of π (function of a function), i.e., $d(s, a, \log \pi(a|s))$ and $\mathcal{L}_d(d, \log \pi)$. Inspired by the idea of adversarial training, we make the policy π challenge the discriminator d by doing the opposite to minimizing $\mathcal{L}_d$, in other words, we let π maximize $\mathcal{L}_d$ under current d. This can be seen as minimizing the worst-case error (Carlini et al., 2019; Fawzi et al., 2016; Goodfellow et al., 2015), which makes the robustness of the discriminator significantly improved.

Perhaps surprisingly, we found that let π maximize $\mathcal{L}_d$ will give the policy an additional corrective loss, which also leads to a valid choice of weight function f.

Theorem 3.1. *Assume $\mathcal{L}_d(d, \log \pi)$ is twice continously differentiable with respect to d, and d is continuously differentiable with respect to $\log \pi$. With a given discriminator d, then a relaxed neccessary condition for $\mathcal{L}_d(d, \log \pi)$ attains its maxima with respect to π is to require a corrective loss term $\mathcal{L}_w$ is minimized by π, where $\mathcal{L}_w$ is given as follows:*

$$\mathcal{L}_w = \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}\left[\log \pi(a|s) \cdot \left(\frac{\eta}{d} + \frac{\eta}{1-d}\right)\right] - \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_o}\left[\log \pi(a|s) \cdot \frac{1}{1-d}\right]$$

Proof. See the Proposition B.1 and Corollary B.2 in the Appendix for detailed derivation and proof. □

Adding the loss term $\mathcal{L}_w$ to BC task, we get the following

new learning objective of π as:

$$\min_{\pi} \alpha \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log \pi(a|s)] - \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}\left[-\log \pi(a|s)\cdot\frac{\eta}{d\,(1-d)}\right] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_o}\left[-\log \pi(a|s)\cdot\frac{1}{1-d}\right], \quad (5)$$

where α is the weight factor ($\alpha \geq 1$). This new objective essentially transforms the original BC task into a cost-sensitive learning problem (Ling & Sheng, 2008) by imposing the following weight on imitating each state-action transition as

$$\text{BC weights} = \begin{cases} \alpha - \eta/d(1-d), & (s,a) \in \mathcal{D}_e \\ 1/\,(1-d)\,, & (s,a) \in \mathcal{D}_o \end{cases}, \quad (6)$$

where d is clipped to the range of $[0.1, 0.9]$ to satisfy the continuity assumption (see Appendix B.2 for details).

Above behavioral cloning weights induce different behaviors on the imitation of transitions from $\mathcal{D}_e$ and $\mathcal{D}_o$. Suppose d is learning in a virtuous cycle, i.e., assigns large values (close to 1) to expert transitions and small values to non-expert transitions (close to 0). The weight of those expert transitions in $\mathcal{D}_o$ will become large while the weight of those non-expert transitions will become small. For transitions in $\mathcal{D}_e$, their weights can be adjusted by tuning the parameter α. Note that even if the discriminator is learned to be totally wrong (i.e., assign small values to expert transitions and large values to non-expert transitions), which may occur at the very beginning of training, the behavior cloning weights $\alpha - \eta/d(1-d)$ ($\alpha \geq 1, \eta < 1$) will not be drastically changed under value clipping. This means that the policy can still learn from the expert dataset $\mathcal{D}_e$. Even though the weight for $\mathcal{D}_e$ is temporarily incorrect, it will be corrected as the discriminator becomes better and better.

Eq. (5) implies that our approach is also a variant of generalized BC objective, but uses a different form of weights. Unlike Offline Reinforced Imitation Learning (Zolna et al., 2020a), which uses the discriminator as the reward and learns a value function as the weight, our approach uses the discriminator outputs directly as the weight. This can greatly reduce the training time and avoid the overestimation issue in estimating the value function offline (Kumar et al., 2019). We term our algorithm Discriminator-Weighted Behavioral Cloning (DWBC). The pseudocode and implementation details of our algorithm are included in Appendix A.

4. Related Work

4.1. Offline Imitation Learning

Offline IL, which has not received considerable attention, is a promising area because it makes IL more practical to satisfy critical safety desiderata. Offline IL methods can be folded into two paradigms: Behavioral Cloning (BC) and Offline Inverse Reinforcement Learning (Offline IRL).

BC (Pomerleau, 1989) is the simplest IL method that can be used in the offline setting, it considers the policy as a conditional distribution $\pi(\cdot|s)$ over actions, recent work (Florence et al., 2022) enhances BC by using energy-based models (LeCun et al., 2006). BC has shown to have no inferior performance compared to popular IL algorithms such as GAIL (Ho & Ermon, 2016) when clean expert demonstrations are available (Ma, 2020). Unlike BC, offline IRL considers matching the state-action distribution induced by the expert policy, this can be achieved implicitly by adversarial training or explicitly by learning a reward function. Offline IRL algorithms based on adversarial training (Kostrikov et al., 2020; Sun et al., 2021; Swamy et al., 2021; Jarboui & Perchet, 2021) use Intergral Probability Metrics (IPMs) (Sriperumbudur et al., 2009) as a distance measure to solve the dual problem. They introduce a discriminator and aim to find the saddle point of a min-max optimization problem, like GAN (Goodfellow et al., 2014). Jarrett et al. (2020) avoids the need of min-max problem by fixing the policy to be energy-based models, in such case the KL divergence from the demonstrator's state-action distribution to that of the policy can be computed in closed form. However, recent work finds several fundamental mathematical misconceptions in their proposed approach and we refer the reader to Swamy et al. (2021) for more details.

The common problem of these works is that they imitate equally to all demonstrations, this will hinder the performance if the demonstrations contain suboptimal data. To solve this, Sasaki & Yamashina (2021) reuses another policy learned by BC as the weight of original BC objective, however, this requires that expert data occupies the majority proportion of the offline dataset, otherwise the policy will be misguided to imitate the suboptimal data. Zolna et al. (2020a) first constructs a reward function that discriminates expert and exploratory trajectories, then use it to solve an offline RL problem. Instead of the adversarial learning scheme, the reward function can also be learned by cascading to two supervised learning steps (Klein et al., 2013). However, offline IRL based on reward learning is expensive to run, requiring solving offline RL in an inner loop, which itself is a challenging problem and prone to training instability (Kumar et al., 2019) and hyperparameter sensitivity (Wu et al., 2019b). Our algorithm can be seen as a combination of these two algorithms in that we train a discriminator to distinguish expert and non-expert data and use the output of the discriminator as the weight of the generalized BC objective, so as to imitate demonstrations selectively. One recent work (Chang et al., 2021) performs offline IL by adopting techniques from pessimistic model-based offline policy learning (Yu et al., 2020; 2021), our

work does not need to train a dynamics model nor perform the expensive min-max model-based policy optimization. Another recent work (Kim et al., 2022) performs offline IL with a KL constraint to regularize the learned policy to stay close to the behavior policy, it could be overly conservative when $\mathcal{D}_o$ is highly suboptimal.

4.2. Offline Reinforcement Learning

One research area highly related to offline IL is offline RL (Lange et al., 2012; Levine et al., 2020), which considers performing effective RL by utilizing arbitrary given, static offline datasets, without any further environment interactions. Note that in offline RL, the training dataset is allowed to have non-optimal trajectories and the reward for each state-action-next state transition triple is known.

Our algorithm draws connection to a branch of methods in offline RL literature that performs "filtered" behavioral cloning explicitly or implicitly. More specifically, these methods estimate an advantage function, which represents the change in expected return when taking action a instead of following the current policy, and perform weighted regression based on the advantage function, defined as $\mathcal{L}_\pi = \mathbb{E}_{(s,a)\sim\mathcal{D}_b}\left[-\log\pi(a|s)\cdot f\left(A^\pi(s,a)\right)\right]$. The advantage A^π can be estimated by Monte-Carlo methods (Schulman et al., 2017; Peng et al., 2019) or Q-value based methods (Schulman et al., 2015; Nair et al., 2020). The filter function f can be a binary filter (Wang et al., 2020) or an exponential filter (Peng et al., 2019; Nair et al., 2020).

While Chen et al. (2021) and Janner et al. (2021) perform filtered behavioral cloning more implicitly. They cast offline RL as a sequence modeling problem and use Transformer architecture (Vaswani et al., 2017) to perform credit assignment directly via self-attention mechanism. Owing to the memorization power of Transformer in capturing long-term dependencies across timesteps, these methods discard low-quality transitions and conduct behavior cloning only on high-reward transitions.

5. Experiments

We present empirical evaluations of DWBC in a variety of settings. We start with describing our experimental setup, datasets and baselines. Then we evaluate DWBC against other baselines on a range of robotic locomotion tasks with different types of datasets. Finally, we analyze the property of the discriminator, i.e., using the discriminator to do offline policy selection (Fu et al., 2021) owing to the including of $\log\pi$ as input.

5.1. Settings

We construct experiments on both widely-used D4RL MuJoCo datasets (Fu et al., 2020) and more complex Adroit human datasets (Rajeswaran et al., 2017). To verify the effectiveness of our methods, we use three setting to generate $\mathcal{D}_e$ and $\mathcal{D}_o$. Note that we use ground truth reward only to perform the data split step and discard the reward informantion afterward.

- In Setting 1, we use mixed datasets in Mujoco environments. We sort from high to low of all trajectories based on the total reward summed over the entire trajectory. We define a trajectory as well-performing if it is among the top 20% of all trajectories. We then sample every X^{th} trajectory from the well-performing trajectories to constitute $\mathcal{D}_e$ and use the remaining trajectories in the dataset to constitute $\mathcal{D}_o$. Note that with X becomes larger, $\mathcal{D}_o$ will contain more proportion of well-performing data.
- In Setting 2, we use expert and random datasets in Mujoco environments. We first sample 10 trajectories from expert datasets and 1000 trajectories from random datasets. We then random sample X trajectories from those 10 expert trajectories and combine them with those 1000 random trajectories to constitute $\mathcal{D}_o$, we use the remaining $10-X$ trajectories to constitute $\mathcal{D}_e$.
- In Setting 3, we use human datasets in Adroit environments. We use the same procedure to constitute $\mathcal{D}_e$ and $\mathcal{D}_o$ as in Setting 1.

We list all datasets used in this paper and the number of trajectories and transitions in $\mathcal{D}_e$ and $\mathcal{D}_o$ in Appendix C, different X is labeled after the dataset name.

5.2. Baseline and ablated algorithms

We compare DWBC with the following baseline algorithms:

BC-exp & BC-all: Behavioral cloning on expert data or on all data. BC-exp is trained only on $\mathcal{D}_e$. $\mathcal{D}_e$ owns higher quality data but with less quantity, and thus causes serious compounding error problems to the resulting policy. BC-all can generalize better than BC-pos due to access to a much larger dataset, but its performance may be negatively impacted by the low-quality data in $\mathcal{D}_o$.
BCND: BCND is trained on all data, it reuses another policy learned by BC as the weight of BC, its performance will be worse if the suboptimal data occupies the major part of the offline dataset.
ORIL: ORIL learns a reward function and uses it to solve an offline RL problem. It suffers from large computational costs and the difficulty of performing offline RL under distributional shift.
DWBC-old-d: We include one ablation of DWBC that trains d without $\log\pi$ as input, with all others remaining the same. In other words, DWBC-old-d performs new BC Task but old Discriminating Task. This ablation is to understand whether adding $\log\pi$ can make d learn better.

Table 1. Results for Mujoco and Adroit datasets. Scores are undiscounted average returns of the policy at the last iteration of training, averaged over 5 random seeds. We bold the highest values.

	Task name	BC-exp	BC-all	BCND	ORIL	DWBC-old-d	DWBC
Hopper	mixed-2	1547	811	437	1345	2450	**2531**
	mixed-5	1263	811	437	998	2271	**2451**
	mixed-10	1458	811	437	1489	1798	**2231**
	exp-rand-3	1200	314	52	49	1531	**2231**
	exp-rand-6	1070	314	52	51	1604	**1610**
HalfCheetah	mixed-2	4451	4210	4456	/	4980	**5011**
	mixed-5	4553	4210	4456	44	5011	**5018**
	mixed-10	4358	4210	4456	989	**5022**	5017
	exp-rand-3	6072	5753	6007	6013	6021	**6107**
	exp-rand-6	5875	5753	6007	**6110**	5803	5955
Walker2d	mixed-2	2031	784	760	2208	2355	**2436**
	mixed-5	2014	784	760	2481	**3112**	3111
	mixed-10	1611	784	760	2384	3219	**3258**
	exp-rand-3	3078	211	6	955	4547	**4666**
	exp-rand-6	2871	211	6	5	3673	**4250**
Ant	mixed-2	**2682**	2255	917	/	1050	2000
	mixed-5	2381	2255	917	/	1982	**3111**
	mixed-10	2285	2255	917	/	2310	**3417**
	exp-rand-3	1071	151	1045	710	875	**1230**
	exp-rand-6	870	151	1045	639	626	**1127**
Pen	human-2	1888	806	1684	2262	**2571**	2486
	human-3	1780	806	1684	2487	2362	**2617**
	human-5	1531	806	1684	2111	2271	**2487**
Door	human-2	45	31	-4	-12	51	**53**
	human-3	**40**	31	-4	-50	-3	10
	human-5	**38**	31	-4	-3	0	0
Hammer	human-2	-187	-230	-163	-222	**-87**	-88
	human-3	-191	-230	-163	-237	-97	**-96**
	human-5	-213	-230	-163	-159	-60	**24**
Relocate	human-2	4	2	**7**	-4	3	3
	human-3	3	2	**7**	-8	0	1
	human-5	3	2	7	**9**	0	1

5.3. Comparative Evaluations

We show the comparative results in Table 1 and include the learning curves in Appendix C. It can be shown from Table 1 that DWBC outperforms baseline algorithms on most tasks (25 out of 32 tasks), especially in Mujoco datasets (18 out of 20 tasks), showing that DWBC is well suited to make effective use of the expert dataset $\mathcal{D}_e$ and the mixed quality dataset $\mathcal{D}_o$.

As expected, the performance of BC-exp declines as X becomes larger. This is because that a larger X means the number of well-performing transitions is smaller. In some datasets (e.g., `Halfcheetah_exp-rand-6` and `Ant_mixed-10`), there is no clear winner between BC-exp and BC-all, which suggests that the quality of $\mathcal{D}_o$ for the considered tasks varies. BCND performs poorly compared to other methods due to the majority of low-quality data in the mixed datasets. It usually scores below BC-all. ORIL struggles to learn in some tasks (especially in the `Ant` datasets), which suggests their learned reward function does not accurately contrast expert and non-expert data. We also find that the performance of ORIL tends to decrease in some tasks (e.g., `Halfcheetah_mixed-5` and `Ant_exp-rand-6`), this "overfitting" phenomenon also occurs in experiments of offline RL papers (Wu et al., 2019b; Kumar et al., 2019). This is perhaps due to limited data size and model generalization bottleneck (Neyshabur, 2017).

We also find that DWBC-old-d performs worse than DWBC. DWBC improve DWBC-old-d by a large margin especially when $\mathcal{D}_o$ contains more expert data (`mixed-10` datasets and `exp-rand-6` datasets), under which circumstance it is harder for the discriminator to distinguish between expert and non-expert data, without the help of $\log \pi(a|s)$.

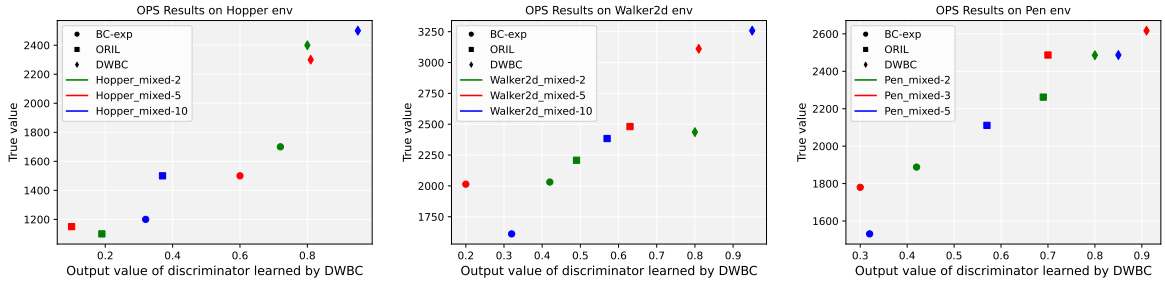


Figure 2. Additional experiment on offline policy selection by the discriminator learned by DWBC.

5.4. Additional Experiments

Offline policy selection by the discriminator. Offline policy selection (OPS) (Paine et al., 2020; Yang et al., 2020; Dereventsov et al., 2021) considers the problem of choosing the best policy from a set of policies given only offline data. This problem is critical in the offline settings (i.e., offline RL and offline IL) because the online execution is often very costly and safety-aware, deploying a problematic policy may damage the real-world systems (Tang & Wiens, 2021). Note that existing offline RL/IL methods break the offline assumption by evaluating different policies corresponding to their rewards in online environment interactions. However, this online evaluation is often infeasible and hence undermines the initial assumption of offline RL/IL.

We find that as a by-product, involving $\log \pi$ in the discriminator d in DWBC brings an appealing characteristic, i.e., *value generalization among policies*. More specifically, d values of known policies can be generalized to unknown policies, we can use expert state-action pairs from $\mathcal{D}_e$ and different policy π as input. The discriminator will assign large values (close to 1) when the evaluated policy is close to the expert policy learned by DWBC, which also means that the evaluated policy is close to the optimal.

To validate our proposed idea, we conduct experiments in `Hopper`, `Walker2d` and `Pen` environment. In `Hopper` and `Walker2d`, we use `mixed-2`, `mixed-5` and `mixed-10` datasets, in `Pen`, we use `mixed-2`, `3` and `mixed-5` datasets. We compare three algorithms (BC-exp, ORIL and DWBC) trained in these datasets, total of 9 policies in each environment. We first train DWBC, then we use the learned discriminator d along with $\mathcal{D}_e$ to compute the value $d(s, a, \log \pi_i(a|s))$ of each policy π_i. We plot average $d(s, a, \log \pi_i(a|s))$ versus the policy's true return in Figure 2. As shown, d values well reflect the rank between almost every two policies. This means that we can first train a DWBC policy and then use the trained discriminator d to do OPS, i.e., select the best policy among given candidate policies, without executing them in the environment to get the actual returns.

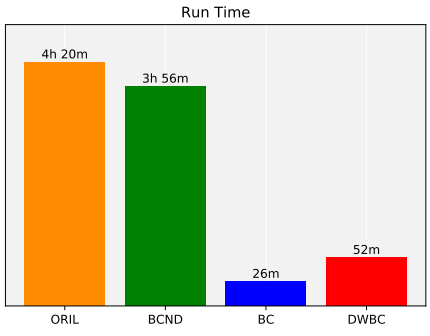


Figure 3. Run time comparison of each offline IL algorithm.

Comparision of run time. We also evaluate the run time of training DWBC and other baseline algorithms for 250,000 training steps (does not include evaluation time cost). All run time experiments were executed on NVIDIA V100 GPUs. For a fair comparison, we use the same policy network size in BC, BCND, ORIL and DWBC. The discriminator network size is also kept the same in ORIL and DWBC. The results are reported in Figure 3. Unsurprisingly, we find the run time of our approach is only slightly more than BC, while other baselines (ORIL, BCND) are over 7 times more costly than BC. The reason that ORIL is costly to run is due to the additional effort to solve an offline RL problem. The high computation cost of BCND is due to its inner iterations of training K policy ensembles ($K = 5$ in our experiment), which is also mentioned in their paper (Sasaki & Yamashina, 2021). This demonstrates the effectiveness of DWBC by only adding a limited cost to the original BC algorithm while providing substantially improved performance.

6. Conclusion and Future Work

In this paper, we propose an effective and light-weighted offline imitation learning algorithm that can learn from suboptimal demonstrations without environment interactions or expert annotations. Experimental results show that our algorithm achieves higher returns and faster training speed

compared to baseline algorithms, under different scenarios. One future work is to derive new algorithms for online IL based on our proposed cooperation framework, as recent studies (Wang et al., 2021; Eysenbach et al., 2021) also reveal the importance of weighting imperfect expert demonstrations in the online IL setting. Another future work is to consider modifying the main task from action matching to state-action distribution matching, which is known to be more robust to distributional shift (Kostrikov et al., 2020).

Acknowledgements

A preliminary version of this work was accepted on Deep RL workshop at NeurIPS 2021. We thank anonymous reviewers for feedback on previous versions of this paper. This work is also supported by gifts from Haomo.AI.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *Proc. of ICML*, 2017.

Brown, D. S., Goo, W., Nagarajan, P., and Niekum, S. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *Proc. of ICML*, 2019.

Brown, D. S., Goo, W., and Niekum, S. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, 2020.

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., and Kurakin, A. On evaluating adversarial robustness. *ArXiv preprint*, 2019.

Chang, J. D., Uehara, M., Sreenivas, D., Kidambi, R., and Sun, W. Mitigating covariate shift in imitation learning via offline data without great coverage. *ArXiv preprint*, 2021.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *ArXiv preprint*, 2021.

Degris, T., White, M., and Sutton, R. S. Off-policy actor-critic. In *Proc. of ICML*, 2012.

Dereventsov, A., Daws Jr, J. D., and Webster, C. Offline policy comparison under limited historical agent-environment interactions. *ArXiv preprint*, 2021.

du Plessis, M. C., Niu, G., and Sugiyama, M. Convex formulation for learning from positive and unlabeled data. In *Proc. of ICML*, 2015.

Elkan, C. and Noto, K. Learning classifiers from only positive and unlabeled data. In *Proc. of KDD*, 2008.

Eysenbach, B., Levine, S., and Salakhutdinov, R. Replacing rewards with examples: Example-based policy search via recursive classification. *ArXiv preprint*, 2021.

Fawzi, A., Moosavi-Dezfooli, S., and Frossard, P. Robustness of classifiers: from adversarial to random noise. In *Proc. of NeuIPS*, 2016.

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. Implicit behavioral cloning. In *Conference on Robot Learning*, pp. 158–168. PMLR, 2022.

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

Fu, J., Norouzi, M., Nachum, O., Tucker, G., Wang, Z., Novikov, A., Yang, M., Zhang, M. R., Chen, Y., Kumar, A., Paduraru, C., Levine, S., and Paine, T. Benchmarks for deep off-policy evaluation. In *Proc. of ICLR*, 2021.

Gelfand, I. M., Silverman, R. A., et al. *Calculus of variations*. Courier Corporation, 2000.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial nets. In *Proc. of NeuIPS*, 2014.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *Proc. of ICLR*, 2015.

Hewitt, E. Rings of real-valued continuous functions. i. *Transactions of the American Mathematical Society*, 64 (1):45–99, 1948.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Proc. of NeuIPS*, 2016.

Irpan, A. Deep reinforcement learning doesn't work yet, 2018.

Janner, M., Li, Q., and Levine, S. Reinforcement learning as one big sequence modeling problem. *ArXiv preprint*, 2021.

Jarboui, F. and Perchet, V. Offline inverse reinforcement learning. *ArXiv preprint*, 2021.

Jarrett, D., Bica, I., and van der Schaar, M. Strictly batch imitation learning by energy-based distribution matching. In *Proc. of NeuIPS*, 2020.

Kim, G.-H., Seo, S., Lee, J., Jeon, W., Hwang, H., Yang, H., and Kim, K.-E. Demodice: Offline imitation learning with supplementary imperfect demonstrations. In *Proc. of ICLR*, 2022.

Klein, E., Piot, B., Geist, M., and Pietquin, O. A cascaded supervised learning approach to inverse reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*, 2013.

Kostrikov, I., Nachum, O., and Tompson, J. Imitation learning via off-policy distribution matching. In *Proc. of ICLR*, 2020.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Proc. of NeuIPS*, 2019.

Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement learning*. 2012.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 2006.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv preprint*, 2020.

Ling, C. X. and Sheng, V. S. Cost-sensitive learning and the class imbalance problem. *Encyclopedia of machine learning*, 2008.

Lowd, D. and Meek, C. Adversarial learning. In *Proc. of KDD*, 2005.

Ma, Y. J. *From Adversarial Imitation Learning to Robust Batch Imitation Learning*. PhD thesis, 2020.

Nair, A., Dalal, M., Gupta, A., and Levine, S. Accelerating online reinforcement learning with offline datasets. *ArXiv preprint*, 2020.

Neyshabur, B. Implicit regularization in deep learning. *ArXiv preprint*, 2017.

Ng, A. Y. and Russell, S. J. Algorithms for inverse reinforcement learning. In *Proc. of ICML*, 2000.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc. of ICML*, 1999.

Paine, T. L., Paduraru, C., Michi, A., Gulcehre, C., Zolna, K., Novikov, A., Wang, Z., and de Freitas, N. Hyperparameter selection for offline reinforcement learning. *ArXiv preprint*, 2020.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *ArXiv preprint*, 2019.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. In *Proc. of NeuIPS*, 1989.

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

Reddy, S., Dragan, A. D., and Levine, S. SQIL: imitation learning via reinforcement learning with sparse rewards. In *Proc. of ICLR*, 2020.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011.

Sasaki, F. and Yamashina, R. Behavioral cloning from noisy demonstrations. In *Proc. of ICLR*, 2021.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. Trust region policy optimization. In *Proc. of ICML*, 2015.

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

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., and Lanckriet, G. R. On integral probability metrics, ϕ-divergences and binary classification. *ArXiv preprint*, 2009.

Sun, M., Mahajan, A., Hofmann, K., and Whiteson, S. Softdice for imitation learning: Rethinking off-policy distribution matching. *ArXiv preprint*, 2021.

Sutton, R. S., Barto, A. G., et al. *Introduction to reinforcement learning*. MIT press Cambridge, 1998.

Swamy, G., Choudhury, S., Wu, Z. S., and Bagnell, J. A. Of moments and matching: Trade-offs and treatments in imitation learning. *ArXiv preprint*, 2021.

Tang, S. and Wiens, J. Model selection for offline reinforcement learning: Practical considerations for healthcare settings. *ArXiv preprint*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Proc. of NeuIPS*, 2017.

Wang, Y., Xu, C., Du, B., and Lee, H. Learning to weight imperfect demonstrations. In *Proc. of ICML*, 2021.

Wang, Z., Novikov, A., Zolna, K., Merel, J., Springenberg, J. T., Reed, S. E., Shahriari, B., Siegel, N. Y., Gülçehre, Ç., Heess, N., and de Freitas, N. Critic regularized regression. In *Proc. of NeuIPS*, 2020.

Wu, Y., Charoenphakdee, N., Bao, H., Tangkaratt, V., and Sugiyama, M. Imitation learning from imperfect demonstration. In *Proc. of ICML*, 2019a.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *ArXiv preprint*, 2019b.

Xu, D. and Denil, M. Positive-unlabeled reward learning. *ArXiv preprint*, 2019.

Yang, M., Dai, B., Nachum, O., Tucker, G., and Schuurmans, D. Offline policy selection under uncertainty. *ArXiv preprint*, 2020.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. MOPO: model-based offline policy optimization. In *Proc. of NeuIPS*, 2020.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. Combo: Conservative offline model-based policy optimization. *ArXiv preprint*, 2021.

Zhang, S., Cao, Z., Sadigh, D., and Sui, Y. Confidence-aware imitation learning from demonstrations with varying optimality. In *Proc. of NeuIPS*, 2021.

Zolna, K., Novikov, A., Konyushkova, K., Gulcehre, C., Wang, Z., Aytar, Y., Denil, M., de Freitas, N., and Reed, S. Offline learning from demonstrations and unlabeled experience. *ArXiv preprint*, 2020a.

Zolna, K., Saharia, C., Boussioux, L., Hui, D. Y.-T., Chevalier-Boisvert, M., Bahdanau, D., and Bengio, Y. Combating false negatives in adversarial imitation learning. *ArXiv preprint*, 2020b.

A. Training procedure details

A.1. Algorithm details

In this section, we present the pseudocode of DWBC in Algorithm 1.

Algorithm 1 Discriminator-Weighted Behavior Cloning (DWBC)

Require: Dataset D_e and D_o, hyperparameter η, α
1: Initialize the imitation policy π and the discriminator d
2: **while** training **do**
3: Sample $(s_e, a_e) \sim D_e$ and $(s_o, a_o) \sim D_o$ to form a training batch $\mathcal{B}$
4: Compute $\log \pi(a|s)$ values for samples in $\mathcal{B}$ using the learned policy π
5: Compute discriminator output values $d(s, a, \log \pi(a|s))$ using sampled (s, a) and computed $\log \pi(a|s)$
6: Update d by minimizing the learning objective $\mathcal{L}_d$ in Eq.(8)
7: Update π by minimizing the learning objective $\mathcal{L}_\pi$ in Eq.(17)
8: **end while**

A.2. Implementation Details

In this paper, all experiments are implemented with Tensorflow and executed on NVIDIA V100 GPUs. For all function approximators, we use fully connected neural networks with RELU activations. For policy networks, we use tanh (Gaussian) on outputs. We use Adam for all optimizers. The batch size is 256 and γ is 0.99. We rescale the reward to $[0, 1]$ as $r' = (r - r_{\min}) / (r_{\max} - r_{\min})$, where $r_{\max}$ and $r_{\min}$ is the maximum and the minimum reward in the dataset. Note that any affine transformation of the reward function does not change the optimal policy of the MDP.

The policy network is 3-layer MLP with 256 hidden units in each layer. The structure of our discriminator differs at the first hidden layer which has two input streams and each of them has 128 units, as illustrated in Figure 4. The learning rate for the policy is $1e-5$ and the learning rate for the discriminator network is $1e-4$. We search α in $\{1, 2, 5, 10\}$ for best model performance. We clip the output of d to $[0.1, 0.9]$. We set η to 0.5 across all tasks, which is the same as the ORIL paper (Zolna et al., 2020a).

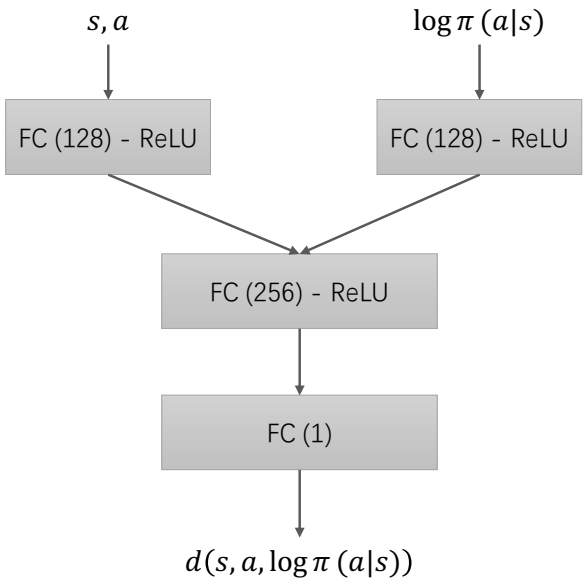


Figure 4. Structure of the discriminator network d.

B. Derivation Details

In this section, we provide the detailed design intuition and theorectical derivation of DWBC.

B.1. Decomposition and Reformulation of Learning Tasks

As discussed in Section 3.2, learning from both the expert dataset D_e and suboptimal dataset D_o implies the need of jointly solving two tasks: BC Task and Discriminating Task. A straightfoward solution is to learn the two tasks separately, which solves BC task by imitating the expert demonstrations in D_e and learn the discriminator via PU learning using data from both D_e and D_o:

$$\begin{aligned} \text{BC Task:} \quad & \pi(a|s) \leftarrow \arg\min_{\pi} \mathcal{L}_{BC} \\ \text{Discriminating Task:} \quad & d(s,a) \leftarrow \arg\min_{d} \mathcal{L}_{d} \end{aligned}$$

where $\mathcal{L}_{BC}$ and $\mathcal{L}_d$ are discussed and given in objectives (2) and (4) in the main article as follows:

$$\begin{aligned} \mathcal{L}_{BC} &= \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log \pi(a|s)] \\ \mathcal{L}_d &= \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log d(s,a)] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_o}[-\log(1-d(s,a))] - \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log(1-d(s,a))] \end{aligned}$$

Naïvely solving above two tasks separately is insufficient. First, the BC task only learn from the expert dataset D_e, fails to utilize the potential valuable information in the supoptimal dataset D_o. Second, as discussed in Section 3.2, both tasks lack sufficient information to improve their own performance. For example, a good discriminator could provide important information to distinguish the potential expert samples in the suboptimal dataset D_e, which are valuable for the learning of policy π; a well-performed policy $\pi(a|s)$ will assing large probabilities to expert actions under expert states, which could provide additional learning signal for the discriminator d to more easily contrast expert and non-expert transitions in D_o.

There are two existing approaches can be used to jointly solve above two tasks, however, both of them have some drawbacks. One approach is to cast the problem into a multi-task style multi-objective optimziation problem, by optimizing an augmented loss $\beta\mathcal{L}_{BC} + (1-\beta)\mathcal{L}_d, \beta \in (0,1)$ for both π and d. The problem is that the BC Task and the Discriminating Task are different tasks, the potential contradiction of the two tasks in certain settings may impede both tasks from achieving the best perforamnce. Moreover, properly selecting the hyperparameter β is very tricky. Another approach is to adopt a GAN-style model (Goodfellow et al., 2014) which treats the policy as the generator and optimize it implicitly through solving a min-max optimization problem with the discriminator loss $\mathcal{L}_d$. However, this is very costly and is known to suffer from training instability and issues such as mode collapse (Arjovsky et al., 2017). Moreover, although we have an explict loss function $\mathcal{L}_{BC}$ for π, it is not used in such a GAN-style model, which results in potential loss of information.

In this paper, we design an new cooperative learning mechansim to address above issues, which also results in a computationally efficient practical algorithm. It includes three key ingredients: 1) sharing information between the BC Task and the Discriminating Task to achieve cooperative learning; 2) enabling the BC Task to learn on both expert and suboptimal data by introduce an additional corrective loss $\mathcal{L}_w$ impacted by the discriminator outputs; 3) solving both tasks in fully supervised learning manner to maintain computational efficiency. In our approach, we consider following alternative formulation to establish information sharing across the two tasks and enable cooperative learning:

$$\begin{aligned} \text{New BC Task:} \quad & \pi(a|s) \leftarrow \arg\min_{\pi} \alpha\mathcal{L}_{BC} + \mathcal{L}_w, \quad \alpha > 1 \\ \text{New Discriminating Task:} \quad & d(s,a,\log\pi(a|s)) \leftarrow \arg\min_{d} \mathcal{L}_d \end{aligned} \tag{7}$$

with the new $\mathcal{L}_d$ given in objective (4) as follows:

$$\begin{aligned} \mathcal{L}_d =& \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log d(s,a,\log\pi(a|s))] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_o}[-\log(1-d(s,a,\log\pi(a|s)))] \\ & - \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log(1-d(s,a,\log\pi(a|s)))] \end{aligned}$$

In above reformulation, we design the information provided by the policy to discriminator as the element-wise imitation loss value $\log\pi(a|s)$, and the information provided to the policy as the addional corrective loss term $\mathcal{L}_w$ computed using

output values of the discriminator d on samples from both D_e and D_o (i.e., D_b). With the involvement of $\mathcal{L}_w$, we can allow the policy π learning from sub-optimal data under the guidance of the discriminator. Moreover, to more robustly learn the discriminator d, we borrow the idea of adversairal training (Lowd & Meek, 2005) and make the policy π challenge d by maximizing $\mathcal{L}_d$.

We will show in the next section that with the choice of element-wise imitation loss $\log \pi(a|s)$ as the form of information and the advesarial behavior of policy π, an exact form of $\mathcal{L}_w$ can be derived, and eventually, transforms the original BC task into a cost sensitive learning problem.

B.2. Drivation of the Corrective Loss Term $\mathcal{L}_w$

In this section, we resort to functional analysis and calculus of variation to derive the exact form of $\mathcal{L}_w$. Under the reformulated problem (7), both the discriminator d and its loss $\mathcal{L}_d$ are impacted by the information provided by policy π ($\log \pi(a|s)$). Hence they are now functional of π (i.e., function of a function). For simplicity, we can express functional d and $\mathcal{L}_d$ as $d(s, a, \log \pi(a|s))$ and $\mathcal{L}_d(d, \log \pi)$. We are interested to see how the variation of π impacts $\mathcal{L}_d$, and further influence d. Moreover, we can use a specific form of $\mathcal{L}_w$ to alter the behavior of the learned π to achieve the desired adversarial behavior.

By inspecting the form of $\mathcal{L}_d$, note that we can equivalently write it as following integral form:

$$\begin{aligned}
\mathcal{L}_d(d, \log \pi) =& \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e} [-\log d(s, a, \log \pi(a|s))] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_o} [-\log(1 - d(s, a, \log \pi(a|s)))] \\
& - \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e} [-\log(1 - d(s, a, \log \pi(a|s)))] \qquad (8) \\
=& \int_{\Omega_s} \int_{\Omega_a} \Big[P_{\mathcal{D}_e}(s, a) \cdot \eta [-\log d(s, a, \log \pi(a|s))] + P_{\mathcal{D}_o}(s, a) [-\log(1 - d(s, a, \log \pi(a|s)))] \\
& - P_{\mathcal{D}_e}(s, a) \cdot \eta [-\log(1 - d(s, a, \log \pi(a|s)))] \Big] \mathrm{d}s\mathrm{d}a \\
\triangleq& \int_{\Omega_s} \int_{\Omega_a} F(s, a, d, \log \pi(a|s)) \mathrm{d}s\mathrm{d}a \qquad (9)
\end{aligned}$$

where $P_{\mathcal{D}_e}(s, a)$, $P_{\mathcal{D}_o}(s, a)$ is the prabability distribution for state-action pair (s,a) in dataset $\mathcal{D}_e$ and $\mathcal{D}_o$, and Ω_s, Ω_a are the domain for state s and action a under $\mathcal{D}_b$. Observe that the functional $\mathcal{L}_d(d, \log \pi)$ has a natural integral form given the following functional $F(s, a, d, \log \pi(a|s))$:

$$\begin{aligned}
F(s, a, d, \log \pi(a|s)) =& P_{\mathcal{D}_e}(s, a) \cdot \eta [-\log d(s, a, \log \pi(a|s))] + P_{\mathcal{D}_o}(s, a) [-\log(1 - d(s, a, \log \pi(a|s)))] \\
& - P_{\mathcal{D}_e}(s, a) \cdot \eta [-\log(1 - d(s, a, \log \pi(a|s)))]
\end{aligned} \qquad (10)$$

A functional defined in an integral form like Eq. (9) is commonly studied in functional analysis and calculus of variation (Gelfand et al., 2000). To enforce the adversarial behavior of π, we want to make π challenge the discriminator d by maximizing $\mathcal{L}_d(d, \log \pi)$. By doing so, the policy is finding "adversarial attacks" for $\mathcal{L}_d$ such that minimizing $\mathcal{L}_d(d, \log \pi)$ becomes harder for the discriminator. This essentially lead to the following min-max optimization problem for $\mathcal{L}_d(d, \log \pi)$, and can be seen as minimizing the worst-case error, which makes the robustness of the discriminator significantly improved (Carlini et al., 2019; Fawzi et al., 2016; Goodfellow et al., 2015).

$$\min_d \max_\pi \mathcal{L}_d(d, \log \pi) \qquad (11)$$

Directly solving above min-max optimization problem can be highly complex. To simplify the analysis, we focus on the inner maximization problem for π and derive an necessary condition that leads to a tractable form of the corrective loss term $\mathcal{L}_w$. Consider d as an unknown external functional decided by the outer minimization problem, maximizing $\mathcal{L}_d(d, \log \pi)$ with respect to π requries to find the maxima of functional $\mathcal{L}_d(d, \log \pi)$. We can show with following proposition that a relaxed condition is needed to be satisfied.

Proposition B.1. *With a given discriminator d decided by the outer maximization problem of (11), and functional $F(s, a, d, \log \pi(a|s))$ defined in Eq.(10), if continuity of both F and d and its derivatives are satisfied, a relaxed neccessary condition for $\mathcal{L}_d(d, \log \pi)$ attaining its extrema with respect to π is:*

$$\int_{\Omega_s} \int_{\Omega_a} \frac{\partial F(s, a, d, \log \pi(a|s))}{\partial d(s, a, \log \pi(a|s))} \cdot \nabla_{\theta_\pi} \log \pi(a|s) \mathrm{d}s\mathrm{d}a = 0 \qquad (12)$$

where θ_π is the model parameters of π.

Proof. According to the calculus of variations (Gelfand et al., 2000), the extrema (maxima or minima) of functional $\mathcal{L}_d(d, \log \pi)$ with respect to π (d is a given function and considered as fixed) can be obtained by solving the associate Euler-Langrangian equation as follows:

$$F_\pi - \frac{\partial}{\partial s} F_{\frac{\partial \pi}{\partial s}} - \frac{\partial}{\partial a} F_{\frac{\partial \pi}{\partial a}} = 0$$

where F_f represents $\frac{\partial F}{\partial f}$. In our case, $\frac{\partial \pi}{\partial s}$ and $\frac{\partial \pi}{\partial a}$ does not appear in the form of F, hence the later two terms are zero. Therefore, it is necessary that the following functional equation holds:

$$F_\pi = \frac{\partial F}{\partial d} \cdot \frac{\partial d}{\partial \log \pi} \cdot \frac{\partial \log \pi}{\pi} = 0$$

Consider θ_π as the model parameter of π, above equation also suggests that

$$\frac{\partial F}{\partial d} \cdot \frac{\partial d}{\partial \log \pi} \cdot \frac{\partial \log \pi}{\pi} \cdot \frac{\partial \pi}{\partial \theta_\pi} = \frac{\partial F}{\partial d} \cdot \frac{\partial d}{\partial \log \pi} \cdot \nabla_{\theta_\pi} \log \pi = 0$$

As both d and F are real-valued functions, hence the same with their derivatives $\partial F / \partial d$ and $\partial d / \partial \log \pi$. Moreover, by assumption, the continuity of $\partial F / \partial d$ and $\partial d / \partial \log \pi$ is satisfied, as the set of real-valued continuous functions is a commutative ring (Hewitt, 1948), thus their order in above equation can be swapped. We have

$$\frac{\partial d}{\partial \log \pi} \cdot \frac{\partial F}{\partial d} \cdot \nabla_{\theta_\pi} \log \pi = 0 \tag{13}$$

Note that d is determined by the outer minimization problem of (11), thus $\partial d / \partial \log \pi$ is unknown and not obtainable by solely inspecting the inner maximization problem. To ensure above functional equation holds for any (s, a) in $\Omega_s \times \Omega_a$, we instead consider another solution of the functional equation Eq. (13) by letting $\frac{\partial F}{\partial d} \cdot \nabla_{\theta_\pi} \log \pi = 0$. Directly solving the new equation is still intractable, since both F and $\nabla_{\theta_\pi} \log \pi$ can be complicated functions, and the data distribution $P_{\mathcal{D}_e}(s, a)$ and $P_{\mathcal{D}_o}(s, a)$ is typically unknown. However, we can obtain a relaxed and tractable condition by computing the integration of $\frac{\partial F}{\partial d} \cdot \nabla_{\theta_\pi} \log \pi$. Since both the datasets $\mathcal{D}_e$ and $\mathcal{D}_o$ are finite and fixed, the domain of s and a, Ω_s and Ω_a, are also closed and bounded, hence the final integration is still 0, that is

$$\int_{\Omega_s} \int_{\Omega_a} \frac{\partial F}{\partial d} \cdot \nabla_{\theta_\pi} \log \pi \mathrm{d}s \mathrm{d}a = 0 \tag{14}$$

□

We are interested in the relaxed condition (14) because it is computational feasible and we can use it to derive the exact form of $\mathcal{L}_w$.

Corollary B.2. *The relaxed condition (14) can be satisfied by minimizing the corrective loss term $\mathcal{L}_w$ of the following form with respect to θ_π:*

$$\mathcal{L}_w = \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e} \left[\log \pi(a|s) \cdot \frac{\eta}{d} \right] - \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_o} \left[\log \pi(a|s) \cdot \frac{1}{1-d} \right] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e} \left[\log \pi(a|s) \cdot \frac{\eta}{1-d} \right] \tag{15}$$

where d represents $d(s, a, \log \pi(a|s))$ for simplicity.

Proof. Observe that

$$
\begin{aligned}
0 &= \int_{\Omega_s}\int_{\Omega_a} \frac{\partial F(s,a,d,\log\pi(a|s))}{\partial d(s,a,\log\pi(a|s))} \cdot \nabla_{\theta_\pi}\log\pi(a|s)\mathrm{d}s\mathrm{d}a \\
&= \int_{\Omega_s}\int_{\Omega_a} \Bigg[- P_{\mathcal{D}_e}(s,a)\cdot\left[\frac{\eta}{d(s,a,\log\pi(a|s))}\right] + P_{\mathcal{D}_o}(s,a)\left[\frac{1}{1-d(s,a,\log\pi(a|s))}\right] \\
&\quad - P_{\mathcal{D}_e}(s,a)\cdot\left[\frac{\eta}{1-d(s,a,\log\pi(a|s))}\right]\Bigg]\cdot\nabla_{\theta_\pi}\log\pi(a|s)\mathrm{d}s\mathrm{d}a \\
&= -\mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}\left[\frac{\eta}{d}\cdot\nabla_{\theta_\pi}\log\pi(a|s)\right] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_o}\left[\frac{1}{1-d}\cdot\nabla_{\theta_\pi}\log\pi(a|s)\right] - \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}\left[\frac{\eta}{1-d}\cdot\nabla_{\theta_\pi}\log\pi(a|s)\right]
\end{aligned}
\tag{16}
$$

where in the last equation, we slightly abuse the notations and write the output value of $d(s,a,\log\pi(a|s))$ as d for simplicity. Above condition can be equivalently perceived as the first-order optimality condition of the loss term $\mathcal{L}_w$ specified in Eq. (15), i.e., derivative equal to zero.

Comparing $\partial\mathcal{L}_w/\partial\theta_\pi$ and the final form of Eq. (16), note that we introduce a minus sign on $\mathcal{L}_w$. This is to ensure that by minimizing $\mathcal{L}_w$, we are updating in the gradient ascent direction of $\mathcal{L}_d(d,\log\pi)$ and find its maxima rather than minima. Hence minimizing $\mathcal{L}_w$ with respect to π (make $\partial\mathcal{L}_w/\partial\theta_\pi = 0$) satisfies the relaxed condition (14), which is derived from the neccessary condition of solving the inner maximization problem of $\mathcal{L}_d(d,\log\pi)$ specified in (11). □

Adding the new corrective loss term $\mathcal{L}_w$ back to our reformulated problem (7), we obtain the final learning objective of π for our BC task (Eq.(5) in the main article):

$$
\min_\pi \alpha \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}[-\log\pi(a|s)] - \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_e}\left[-\log\pi(a|s)\cdot\frac{\eta}{d\,(1-d)}\right] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_o}\left[-\log\pi(a|s)\cdot\frac{1}{1-d}\right], \quad \alpha > 1 \tag{17}
$$

Note that the derivation of $\mathcal{L}_w$ requires continuity to be satisfied in $\partial F/\partial d$. The involvement of discriminator output values $1/d(s,a,\log\pi(a|s))$ and $1/(1-d(s,a,\log\pi(a|s))$ may violate the continuity assumption. We thus clip the discriminator output values to the range of $[0.1, 0.9]$ in our practical algorithm.

C. Additional results

C.1. Datasets details

In Table 2, we list all datasets used in our paper and the number of trajectories and transitions in $\mathcal{D}_e$ and $\mathcal{D}_o$, where different X is labeled after the dataset name.

C.2. Learning curves

In Figure 5, we provide the learning curves of experiments conducted in Section 5.3. As the learning procedure of BC is quite fast and stable, for more clearly presentation, we plot the results of BC-exp and BC-all as a horizon bar with the shaded area as the standard deviation across different seeds. The average return in $\mathcal{D}_e$ is plotted as the red dashed line in all plots.

C.3. More comparison of DWBC and DWBC-old-d

To more clearly see the comparison of DWBC and DWBC-old-d, we first normalize the results presented in our paper to values that lie between 0 and 100 according to D4RL (Fu et al., 2020), where a score of 0 corresponds to a random policy and 100 corresponds to an expert. We then compute the mean value by dataset types, shown as follows.

It can be seen from Table 3 that DWBC outperform DWBC-old-d by at least **10%** on all type of datasets., we also find that DWBC achieves close to **20%** improvement when $\mathcal{D}_o$ contains a large number of expert data.

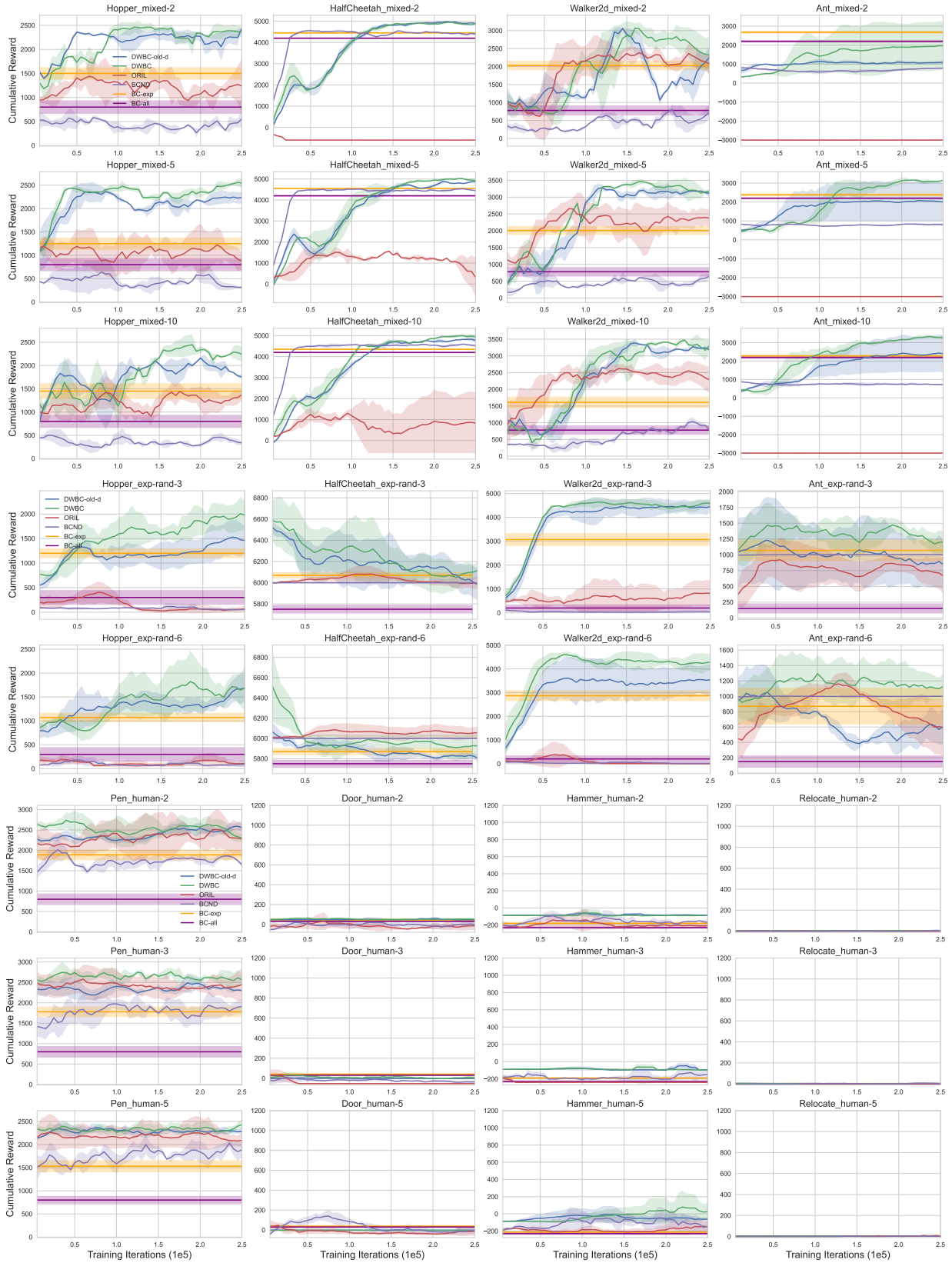


Figure 5. Learning curves of compared algorithms on different datasets.

Table 2. Dataset details.

Dataset-X	$\#\mathcal{D}_e$		$\#\mathcal{D}_o$	
	Trajectories	Transitions	Trajectories	Transitions
Hopper_mixed-2	204	96,222	1,835	303,737
Hopper_mixed-5	82	39,590	1,957	360,369
Hopper_mixed-10	41	19,176	1,998	380,783
Hopper_exp-rand-3	7	6,993	1,003	23,720
Hopper_exp-rand-6	4	3,996	1,006	26,717
Halfcheetah_mixed-2	20	19,980	182	181,818
Halfcheetah_mixed-5	8	7,992	194	193,806
Halfcheetah_mixed-10	4	3,996	198	197,802
Halfcheetah_exp-rand-3	7	6,993	1,003	1001,997
Halfcheetah_exp-rand-6	4	3,996	1,006	1004,994
Walker2d_mixed-2	109	74,857	984	226,050
Walker2d_mixed-5	44	31,010	1,049	269,897
Walker2d_mixed-10	22	15,569	1,071	285,338
Walker2d_exp-rand-3	7	6,993	1,003	21,874
Walker2d_exp-rand-6	4	3,996	1,006	24,871
Ant_mixed-2	49	46,646	436	254,869
Ant_mixed-5	20	19,209	465	282,306
Ant_mixed-10	10	9,866	475	29,1649
Ant_exp-rand-3	7	6,458	1,003	182,909
Ant_exp-rand-6	4	3,996	1,006	185,371
Pen_human-2	3	597	22	4,378
Pen_human-3	2	398	23	4,577
Pen_human-5	1	199	24	4,776
Door_human-2	3	770	22	5,934
Door_human-3	2	479	23	6,225
Door_human-5	1	255	24	6,449
Hammer_human-2	3	1,485	22	9,800
Hammer_human-3	2	844	23	10,441
Hammer_hunman-5	1	483	24	10,802
Relocate_human-2	3	1,328	22	8,589
Relocate_human-3	2	862	23	9,055
Relocate_human-5	1	511	24	9,406

Table 3. More comparison results.

	mixed-2 mean	**mixed-2 mean**	**mixed-10 mean**	**exp-rand-3 mean**	**exp-rand-6 mean**
DWBC-old-d	50.5	58.8	57.8	56.4	50.3
DWBC	57.3	67.0	67.9	62.3	61.6
Improvement (compared to DWBC-old-d)	13.4%	13.8%	17.3%	10.4%	22.4%