# Single Step Policy Alignment for Imitation Learning with Auxiliary Imperfect Demonstration

**Anonymous authors**
**Paper under double-blind review**

## Abstract

We propose a novel one-step supervised imitation learning (IL) framework called Adversarial Density Regression (ADR). This imitation learning (IL) framework seeks to utilize a single-step re-weighted behavioral cloning (BC) objective to rectify the policy acquired under conditions of unknown quality by aligning it with the expert distribution using demonstrations. Specifically, ADR is designed to address several limitations in previous IL algorithms: First, existing off-policy IL algorithms are based on the Bellman operator, which inevitably suffers from cumulative offsets from sub-optimal multi-step rewards. Additionally, these off-policy frameworks suffer from out-of-distribution (OOD) state-actions. Second, the conservative terms that help solve the OOD issue require nuanced and delicate balancing. To address these limitations, we fully integrate a one-step density-weighted BC objective for IL with auxiliary imperfect demonstration. Theoretically, we demonstrate that this adaptation can effectively correct the distribution of policies trained on unknown-quality datasets to align with the expert policy's distribution. The difference between the empirical and the optimal value function is proportional to the upper bound of ADR's objective, indicating that minimizing ADR's objective is akin to approaching the optimal value. Empirically, we conduct extensive evaluations and find that ADR outperforms all of the selected IL algorithms on tasks from the Gym-Mujoco domain. Meanwhile, ADR achieves about **90%** improvement over IQL when utilizing ground truth rewards on tasks from the Adroit and Kitchen domains.

## 1 Introduction

Reinforcement Learning (RL) has revolutionized various fields, including robot learning (Brohan et al., 2023a;b; Bhargava et al., 2020), language modeling (Ouyang et al., 2022; Meta, 2023), and natural science (Gómez-Bombarelli et al., 2018). Despite its success, RL requires extensive interactions with the environment to obtain the optimal policy, which poses challenges for sample efficiency. One way to address this limitation is by leveraging static RL datasets in offline settings. However, this approach often faces the issue of overestimation of out-of-distribution (OOD) states-actions (Levine et al., 2020). To mitigate this, prior research has introduced conservative methods, such as incorporating regularization terms (Fujimoto et al., 2019a; Wu et al., 2022) in the policy learning objective, or pessimism terms in value function learning objective (Kumar et al., 2020a). These methods have been found to effectively alleviate OOD issues and enable the learning of policies that outperform the behavior policy (Kostrikov et al., 2021). However, offline RL algorithms generally assume that offline datasets contain reward labels. Moreover, striking a balance with conservative terms in offline RL remains difficult, particularly for tasks with sparse rewards (Cen et al., 2024).

On the other hand, when the dataset does not contain rewards, we can utilize Imitation Learning (IL) algorithms to learn near-expert policy by leveraging a large amount of unknown-quality datasets and a small number of demonstrations (Argall et al., 2009). In particular, one of the most common methods is to train a discriminator through generative Adversarial learning (AL) to represent the reward or value functions (Ho and Ermon, 2016), followed by updating within RL frameworks. However, it is difficult for the discriminator to converge to its optimal value (Kostrikov et al., 2019). Furthermore, sub-optimal reward or value functions can lead to unstable training.

To address these limitations, we propose ADR, an efficient, single-stage supervised learning framework that employs a density-weighted behavioral cloning objective to optimize policy learning. The key objective of ADR is to closely align the policy distribution with that of the demonstrations while diverging from the distributions of datasets with unknown quality. Theoretically, this method effectively shifts the empirical distribution toward the expert distribution in a direct and corrective manner (Proposition 5.2). Moreover, we demonstrate in Proposition 5.3 that the value bound is proportional to the lower bound of ADR's objective. Thus, minimizing ADR's objective leads to convergence towards the optimal policy. In particular, ADR is a one-step supervised IL framework, where all training samples are in-sample, effectively eliminating the challenges of OOD issues. Furthermore, ADR is a simple and feasible solution, as most RL studies frame the offline RL problem within a Markov Decision Process (MDP) (Kumar et al., 2019; Kostrikov et al., 2021; Haarnoja et al., 2018; Fujimoto et al., 2019b; van Hasselt et al., 2015). Under the MDP setting, decision-making depends solely on the current observation and policy, independent of historical information. Thus, if the action support is adequately relocated, the policy's performance can be ensured. To validate ADR's effectiveness, we evaluated it across various tasks from the Adroit and Gym-Mujoco domains under the Learning from Demonstration (LfD) setting, where it demonstrated competitive results. Notably, ADR outperforms Implicit Q Learning (IQL) with rewards by 89.5% on tasks from the Adroit and Kitchen domains, and performs better than selected diffusion policy on the Adroit domain. *In summary, our main contribution is ADR, a single-step supervised IL method, specifically:*

- ADR operates as a single-step supervised learning paradigm, rendering it immune to the accumulated offsets resulting from suboptimal rewards. Meanwhile, compared to traditional single-step IL paradigms such as Behavioral Cloning (BC), ADR can achieve better performance with fewer demos based on adversarial density-weighted regression.
- ADR neither requires the addition of conservative terms nor extensive hyperparameter parameter tuning during the training process.
- Moreover, ADR poses a theoretical guarantee and empirical improvements. We prove and empirically show that optimizing ADR's objective is akin to approaching the demo policy. ADR also outperforms the majority of RL-combined approaches across diverse domains.

## 2 Related Work

**Behavior Policy Modeling.** Previously, estimating the support of the behavior policy has been approached using various methods, including Gaussian (Kumar et al., 2019; Wu et al., 2019) or Gaussian mixture (Kostrikov et al., 2021) sampling approaches, Variational Auto-Encoder (VAE) based techniques (Kingma and Welling, 2022; Debbagh, 2023), or accurate sampling via auto-regressive language models (Germain et al., 2015). Specifically, the most relevant research to our study involves utilizing VAE to estimate the density-based definition of action support (behavior density) (Fujimoto et al., 2019b; Wu et al., 2022). On the other hand, behavior density is utilized to regularize the offline training policy (Fujimoto and Gu, 2021), reducing the extrapolation error of offline RL algorithms, it has also been utilized in offline-to-online setting (Wu et al., 2022; Fujimoto and Gu, 2021; Nair et al., 2021) to ensure the stable online fine-tuning. Different from the previous study, Our focus is on using the estimated density from the demonstrations and datasets to optimize the policy with the ADR objective.

**Imitation Learning.** Modern Imitation Learning (IL) are generally RL-combined type. Primarily, RL combined-type IL can be categorized nto Learning from Demonstration (LfD) (Argall et al., 2009; Judah et al., 2014; Ho and Ermon, 2016; Brown et al., 2020; Ravichandar et al., 2020; Boborzi et al., 2022) and Learning from Observation (LfO) (Ross et al., 2011a; Liu et al., 2018; Torabi et al., 2019; Boborzi et al., 2022). Different from these approaches, our ADR does not require the estimation of reward/value functions.

## 3 Preliminaries

**Reinforcement Learning (RL).** We consider RL can be represented by a Markov Decision Process (MDP) tuple *i.e.*, $\mathcal{M} := (\mathcal{S}, \mathcal{A}, p_0, r, d_{\mathcal{M}}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ separately denotes observation and action space, $\mathbf{a} \in \mathcal{A}$

and $\mathbf{s} \in \mathcal{S}$ separately denotes state (observation) and action (decision making). $\mathbf{s}_0$ denotes initial observation, $p_0$ denotes initial distribution, $r(\mathbf{s}_t, \mathbf{a}_t) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes reward function. $d_{\mathcal{M}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition function, $\gamma \in [0, 1]$ denotes the discounted factor. The goal of RL is to obtain the optimal policy $\pi^*$ that can maximize the accumulated Return *i.e.*, $\pi^* := \arg\max_\pi \sum_{t=0}^{t=T} \gamma^t \cdot r(\mathbf{s}_t, \mathbf{a}_t)$, where $\tau = \{\mathbf{s}_0, \mathbf{a}_0, r(\mathbf{s}_0, \mathbf{a}_0), \cdots, \mathbf{s}_T, \mathbf{a}_T, r(\mathbf{s}_T, \mathbf{a}_T)|\mathbf{s}_0 \sim p_0, \mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t), \mathbf{s}_{t+1} \sim d_{\mathcal{M}}(\cdot|\mathbf{s}_t, \mathbf{a}_t)\}$, and $T$ denotes time horizon.

**Imitation Learning (IL).** In IL problem setting, $r(\mathbf{s}, \mathbf{a})$ is inaccessible, but we have access to a limited number of demonstrations $\mathcal{D}^* = \{\tau^* = \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{s}_k, \mathbf{a}_k, \cdots \mathbf{s}_T, \mathbf{a}_T|\mathbf{a}_t \sim \pi^*(\cdot|\mathbf{s}), \mathbf{s}_0 \sim p_0, \mathbf{s}_{t+1} \sim d_{\mathcal{M}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\}\}$, and large amount of unknown-quality dataset $\hat{\mathcal{D}} = \{\hat{\tau}|\hat{\tau} \sim \hat{\pi}\}$. In particular, one of the classical IL methods is BC, where the objective is to maximize the likelihood of expert decision-making, as follows:

$$\pi_\theta := \arg\max_{\pi_\theta} \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}^*}[\log \pi_\theta(\mathbf{a}|\mathbf{s})], \tag{1}$$

however, BC's performance is brittle when $\mathcal{D}^*$ is scarcity (Ross et al., 2011b). Another approach is to recover a policy $\pi(\cdot|\mathbf{s})$ by matching the distribution of the expert policy. Since $\pi^*$ cannot be directly accessed, previous studies frame IL as a distribution-matching problem. Specifically, the process initiates by estimating a reward or Q-function $c(\mathbf{s}, \mathbf{a})$ through Adversarial learning (Kostrikov et al., 2019), as below:

$$c(\mathbf{s}, \mathbf{a}) : = \arg\min_c \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\hat{\mathcal{D}}}[\log(\sigma(c(\mathbf{s}, \mathbf{a})))] + \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}^*}[\log(1 - \sigma(c(\mathbf{s}, \mathbf{a})))], \tag{2}$$

where $\sigma$ denotes the *Sigmoid* function. The empirical policy $\pi_\theta$ is then optimized within a RL framework. However, most of these distribution-matching approaches rely on Adversarial learning, which often suffers from unstable training caused by sub-optimal reward or value functions. Different from these IL methods, our approach is purely a supervised IL framework that entirely utilizes a density-weighted BC objective to align the empirical policy with epxert distributions. Nevertheless, our method necessitates the estimation of behavior density as a prerequisite.

**Behavior density estimation via Variational Auto-Encoder (VAE).** Typically, action support constrain *i.e.*, $D_{\mathrm{KL}}[\pi_\theta||\pi_\beta] \leq \epsilon$ has been utilized to confine the training policy to the support set of the behavior policy $\pi_\beta$ (Kumar et al., 2019; Fujimoto et al., 2019b), aiming to mitigate extrapolation error. Another efficient approach estimates behavior density via diffusion policy, however, diffusion suffers from higher computing complexity and are not coincide with our problem setting (details see Appendix 8). In this research, we propose leveraging existing datasets and demonstrations to separately learn the data and demo behavior densities, which are then utilized for ADR. In particular, we follow Wu et al. to estimate the density of action support with Linear Variational Auto-Encoder (VAE) (as demonstrated VAE-1 in Damm et al.) by Empirical Variational Lower Bound (ELBO) :

$$\log p_\Theta(\mathbf{a}|\mathbf{s}) \geq \mathbb{E}_{q_\Phi(\mathbf{z}|\mathbf{a},\mathbf{s})}[\log p_\Theta(\mathbf{a}, \mathbf{z}|\mathbf{s})] - D_{\mathrm{KL}}[q_\Phi(\mathbf{a}|\mathbf{s}, \mathbf{a})||p(\mathbf{s}|\mathbf{z})] \stackrel{\text{def}}{=} -\mathcal{L}_{\mathrm{ELBO}}(\mathbf{s}, \mathbf{a}; \Theta, \Phi), \tag{3}$$

and computing the policy likelihood through importance sampling during evaluation:

$$\log p_\Theta(\mathbf{a}|\mathbf{s}) \approx \mathbb{E}_{\mathbf{z}^l\sim q_\Phi(\mathbf{z}|\mathbf{s},\mathbf{a})}\left[\frac{1}{L}\sum_L \frac{p_\Theta(\mathbf{a}, \mathbf{z}^l|\mathbf{s})}{q_\Phi(\mathbf{z}^l|\mathbf{a}, \mathbf{s})}\right] \stackrel{\text{def}}{=} \mathcal{L}_{\pi_\beta}(\mathbf{s}, \mathbf{a}; \Theta, \Phi, L), \tag{4}$$

where $\mathbf{z}^l \sim q_\Phi(\mathbf{z}|\mathbf{s}, \mathbf{a})$ is the $l_{th}$ sampled VAE embedding, $\Theta$ and $\Phi$ are separately encoder's and decoder's parameters, $l$ and $L$ respectively denote the $l_{th}$ sampling index and the total sampling times.

## 4 Problem Formulation

Previous IL algorithms have several limitations: 1) Utilizing multi-step sub-optimal reward or value functions will cause accumulated offsets. Additionally, off-policy frameworks are suffered from OOD state actions. 2) Some off-policy offline frameworks necessitate tuning of hyperparameters to strike a balance between conservatism, and overly conservatism constrains the exploratory capacity of policies, limiting their ability to

adapt and improve beyond the demonstrations provided. To overcome these issues, we completely adapt a supervised learning objective ADR to correct the policy distribution on unknown-quality datasets using a small number of demonstrations. Meanwhile, the policy we employ is a Gaussian policy, which adheres to a Gaussian distribution *i.e.* $\pi \sim \mathcal{N}(\mu, \sigma)$. Our theoretical proofs are also grounded on the assumption that the policy used is a Gaussian one.

**Notations.** Before formulating our objective, we first define $P^*(\mathbf{a}|\mathbf{s})$ as the expert behavior density[1], and define the sub-optimal behavior density as $\hat{P}(\mathbf{a}|\mathbf{s})$. Meanwhile, we define the training policy as $\pi_\theta(\cdot|\mathbf{s}) : \mathcal{S} \to \mathcal{A}$. Additionally, we denote the stationary distributions of the empirical policy, datasets, and expert policy by $d^\pi$, $d^\mathcal{D}$ and $d^{\pi^*}$, respectively. And, we denote Kullback-Leibler (KL) divergence as $D_{\text{KL}}$.

**Definition 1.** *(Stationary Distribution) We separately define the $\gamma$ discounted stationary distribution (state-action occupancy) of expert and non-expert behavior as $d^{\pi^*}(\mathbf{s}, \mathbf{a})$ and $d^\pi(\mathbf{s}, \mathbf{a})$. In particular, $d^\pi(\mathbf{s}, \mathbf{a})$ can be formulated as:*

$$d^\pi(\mathbf{s}, \mathbf{a}) := (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \cdot \Pr(\mathbf{s} = \mathbf{s}_t, \mathbf{a} = \mathbf{a}_t | \mathbf{s}_0 \sim \mu_0, \mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t), \mathbf{s}_{t+1} \sim d_\mathcal{M}(\cdot|\mathbf{s}_t, \mathbf{a}_t)),$$

*and, $d^{\pi^*}(\mathbf{s}, \mathbf{a})$ can be formulated by replacing $\pi$ with $\pi^*$*

**Remark 4.1.** *$d^\pi(\mathbf{s}) > 0$ whenever $d^\mathcal{D}(\mathbf{s}) > 0$ is a guarantee that the on-policy samples $\mathcal{D}$ has coverage over the expert state-marginal, and is necessary for IL to succeed. (This remark has been extensively deliberated by Ma et al.)*

**Policy Distillation via KL Divergence.** Rusu et al. demonstrates the effectiveness of policy distillation by minimizing the KL divergence between the training policy $\pi_\theta$ and the likelihood of teacher policy set $\pi_i \in \Pi$, *i.e.*, $\pi := \arg\min_{\pi_\theta} D_{\text{KL}}[\pi_\theta||\pi_i]\big|_{\pi_i \in \Pi}$. Meanwhile, if the condition mentioned in Remark 4.1 is held, we can directly achieve expert behavior through distillation, *i.e.*,

$$\pi := \arg\min_{\pi_\theta} D_{\text{KL}}[\pi_\theta||P^*]. \tag{5}$$

however, it's insufficient to mimic the expert behavior by minimizing the KL divergence between $\pi_\theta(\mathbf{a}|\mathbf{s})$ and $P^*(\mathbf{a}|\mathbf{s})$, since the limited demonstrations aren't sufficient to help to estimate a good $P^*(\cdot|\mathbf{s})$. To address this limitation, we propose Adversarial Density Regression (ADR), a supervised learning algorithm that utilizes a limited number of demonstrations to correct the distribution learned by the policy on datasets of unknown quality, thereby bringing it closer to the expert distribution.

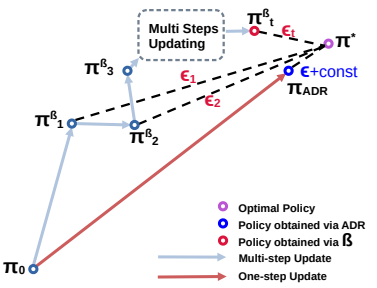


Figure 1: Blue path represents Bellman operator $\mathcal{B}$. Red path represents the optimal policy.

**Adversarial Density Regression (ADR).** In particular, beyond aligning $\pi_\theta$ with the expert distribution $P^*$, we also push $\pi_\theta$ away from the empirical distribution $\hat{P}$, as formulated in Equation 6. This approach is formalized as Adversarial Density Regression (ADR) in Definition 2. The primary advantage of ADR lies in its independence from the Bellman Operator, and ADR is one-step supervised learning paradigm. Therefore, ADR isn't impacted by the cumulative offsets that are introduced during multi-step updates (demonstrated in Figure 1), ensuring a more stable and reliable learning process.

**Definition 2** (Adversarial Density Regression (ADR))**.** *Given expert behavior density $P^*(\mathbf{a}|\mathbf{s})$ and sub-optimal behavior density $\hat{P}(\mathbf{a}|\mathbf{s})$, we formulate the objective of ADR, where $\pi_\theta$ approaches the expert behavior while diverging from the sub-optimal behavior, as follows:*

$$\pi_\theta := \arg\min_{\pi_\theta} \mathbb{E}_\mathcal{D}[D_{\text{KL}}\Big[\pi_\theta||P^*] - D_{\text{KL}}[\pi_\theta||\hat{P}]\Big], \tag{6}$$

[1]The conception of behavior density is proposed by Wu et al., representing the density probability of the given action $\mathbf{a}$ within the action support

**Density Weighted Regression (DWR).** However, it's computing in-efficient to directly compute the objective formulated in Definition 2. But, according to Theorem 4.2, we can instead computing:

$$\pi_\theta := \arg\min_{\pi_\theta} \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[\mathcal{W}(\hat{P}, P^*) \cdot ||\pi_\theta(\cdot|\mathbf{s}) - \mathbf{a}||^2\big]. \tag{7}$$

to replace Equation 6, where $\mathcal{W}(\hat{P}, P^*) = \log \frac{P^*(\mathbf{a}|\mathbf{s})}{\hat{P}(\mathbf{a}|\mathbf{s})}$ termed **density weight**.

**Theorem 4.2** (ADR can be solved by linear weighted regression form)**.** *Given expert log behavior density* $\log P^*(\mathbf{a}|\mathbf{s}) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$*, sub-optimal log behavior density* $\log \hat{P}(\mathbf{a}|\mathbf{s}) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$*, and the empirical policy* $\pi_\theta : \mathcal{S} \to \mathcal{A}$*, offline dataset* $\mathcal{D}$*. Minimizing Equation 6. is equivalent to:*

$$\min J(\pi_\theta) \equiv \min \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[\mathcal{W}(\hat{P}, P^*) \cdot ||\pi_\theta(\cdot|\mathbf{s}) - \mathbf{a}||^2\big]. \tag{8}$$

**Proof**. of Theorem 4.2, see Appendix F.1.

Furthermore, to address the limitations of BC's tendency to overestimate given state-action pairs, we propose alternately minimizing the upper bound of Equation 7, mitigating the overestimation issues:

$$J(\pi_\theta) \leq \mathbb{E}_{\beta_\mathcal{D}\sim\mathcal{D}}\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\beta_\mathcal{D}}\big[\mathcal{W}(\hat{P}, P^*)] \cdot \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\beta_\mathcal{D}}[||\pi_\theta(\cdot|\mathbf{s}) - \mathbf{a}||^2]. \tag{9}$$

**Proof**. of Equation 9, see Appendix F.1.

Where $\beta_\mathcal{D} \in \mathcal{D}$ denotes a batch of samples drawn from the offline dataset during the offline training process.

## 5 Theoretical Analysis of Adversarial Density Regression

In this section, we further conduct a theoretical analysis to demonstrate the convergence and advantage of ADR.

**Assumption 5.1.** *Suppose the policy extracted from Equation 9 is* $\pi$*, we separately define the state marginal of the dataset, empirical policy, and expert policy as* $d^\mathcal{D}$*,* $d^\pi$ *and* $d^{\pi^*}$*, they satisfy this relationship:*

$$D_{KL}[d^\pi||d^{\pi^*}] \leq D_{KL}[d^\mathcal{D}||d^{\pi^*}]. \tag{10}$$

**Proposition 5.2** (Policy Convergence of ADR)**.** *Assuming Equation 6 can finally converge to* $\epsilon$ *via minimizing Equation 8, meanwhile, assuming Assumption F.2 is held. Then* $\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\hat{\mathcal{D}}}[D_{KL}(\pi||\pi^*)] \to \frac{M}{2n} \cdot \sqrt{\log\frac{2}{\delta}} + \Delta C + \epsilon$*.*

**Proposition 5.3.** *(Value Bound of ADR) Given the empirical policy* $\pi$ *and the optimal policy* $\pi^*$*, let* $V^\pi(\rho_0)$ *and* $V^{\pi^*}(\rho_0)$ *separately denote the value network of* $\pi$ *and* $\pi^*$*, and given the discount factor* $\gamma$*. Meanwhile, let* $R_{max}$ *as the upper bound of the reward function i.e.,* $R_{max} = \max||r(\mathbf{s},\mathbf{a})||$*. Based on the Assumption 5.1, Assumption F.2, Lemma F.8, and Proposition 5.2, we can obtain:*

$$|V^\pi(\rho_0) - V^{\pi^*}(\rho_0)| \leq \underbrace{\frac{R_{max}}{1-\gamma} D_{TV}[d^*(\mathbf{s})||d^\mathcal{D}(\mathbf{s})]}_{w.l.o.g} + \frac{2 \cdot R_{max}}{1-\gamma} \cdot \sqrt{2 \cdot (\frac{M}{2n} \cdot \sqrt{\log\frac{2}{\delta}} + \Delta C + \epsilon)}, \tag{11}$$

*where,* $\Delta C = C_1 - C_2$ *is a constant term, dependent on the state distribution.* $\delta$ *originates from Assumption F.2,* $n = |\mathcal{D}^*|$*,* $M := \arg\max_{X_i}\{X_i = \pi^*(\mathbf{a}_t|\mathbf{s}_t) \log\frac{\pi^*(\mathbf{a}_t|\mathbf{s}_t)}{\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t)} | (\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}^*\}$*.*

**Proof**. of Proposition 5.2 and Proposition 5.3, see Appendix F.4 and Appendix F.9.

From Proposition 5.2, we can infer that if Equation 6 converges to a small threshold $\epsilon$, the KL divergence between the likelihood of $\pi$ and $\pi^*$ on unknown-quality data will converge to the same order of magnitude

*i.e.*, $O(\epsilon)$. This implies that the action distribution learned by the $\pi$ will become closer to the $\pi^*$, as long as the states in the unknown-quality data sufficiently cover the states of the $\pi^*$, $\pi$ will learn as many expert decisions as possible. At the same time, in Proposition 5.3, we further prove that the regret of policy $\pi$ is proportional to the convergence upper bound of Equation 6. Therefore, minimizing Equation 6 implies that $V^{\pi}(\rho_0)$ will converge to the $V^{\pi^*}(\rho_0)$ considering the current dataset. Specifically, the first term on the left-hand side of Equation 11 is determined by the quality of the dataset, which is generally applicable to all algorithms (**w.l.o.g**). However, the second term is unique to ADR, as the supervised optimization objective of ADR aligns with maximizing $V^{\pi}(\rho_0)$. Therefore, minimizing ADR's objective can bring $\pi$ closer to $\pi^*$.

**Further analysis regard the multiplicative terms.** In our Proposition 5.3, there are some multiplicative terms such as $\Delta C$ and $M$. Intuitively, the presence of these multiplicative terms makes it that the bound of Proposition 5.3 does not converge to zero. However, these terms are correlated with the quality of the dataset. When the dataset's quality approaches that of the demonstration, these multiplicative terms will converge to zero. We have provided the corresponding analysis in Appendix F.

**The superiority of density weighted form compared to KL form.** The KL divergence is not symmetric, so only convergence to a very small threshold during the training process can prove that the policy has fully converged to the target domain. Equation 9 is in the form of weighted MSE, thus it is symmetrical. During the training process, as long as the value of Equation 9 is sufficiently small, it can indicate that the policy is close to the target domain. Furthermore, Equation 9 has lower computational complexity (Apendix D), requiring less time during the training process.

**Policy Distribution Analysis.** To validate the near-optimal policy convergence, we visualize the policy distribution of both the behavior learned by ADR and expert behavior (sampled from dataset) in Figure 2. Remarkably, utilizing solely the `medium-replay` dataset, ADR is able to comprehensively cover the expert behavior, demonstrating its efficacy in mimicking the expert policy, thus validating our claim in Proposition 5.2.

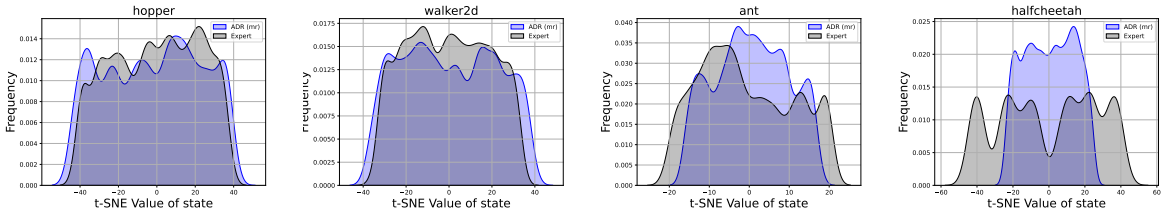


Figure 2: Policy Distribution. We sequentially sampled 500 samples $\tau_{\texttt{sampled}} = \{(\mathbf{s}_t, \mathbf{a}_t)|(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}_{\texttt{exp}}\}_{t=0}^{t=500}$ from the expert dataset $\mathcal{D}_{\texttt{exp}}$. At the same time, we generated 500 actions based on the policy learned from ADR *i.e.*, $\tau_{\texttt{generate}} = \{\mathbf{a}_t := \pi_\theta(\cdot|\mathbf{s}_t)|\mathbf{s}_t \in \tau_{\texttt{sampled}}\}$. Then, we reduced the dimensions of actions from all $\tau_{\texttt{sampled}}$ and $\tau_{\texttt{generate}}$ using t-SNE and plot the KDE curve.

## 6 Methods

To alleviate the impact from the scarcity of demonstrations, we introduce Adversarial Density Estimation (ADE).

**Adversarial Density Estimation (ADE).** Specifically, during the training stage, we utilize the ELBO of VAE to estimate the density probability of state-action pair in action support *i.e.*, Equation 4. Additionally, to alleviate the limitation of demonstrations' scarcity, we utilize adversarial learning (AL) in density estimation. This involves maximizing the density probability of expert offline samples while minimizing the density probability of sub-optimal offline samples to improve the estimation of expert behavior density. ($\Theta^*$ *doesn't mean the optimal parameter, instead, it means the parameters of VAE model utilized to estimate on expert*

*samples*):

$$\mathcal{J}_{\texttt{ADE}}(\Theta^*) = \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\pi^*}\left[\sigma(P_{\Theta^*}(\mathbf{a}|\mathbf{s}))\right] - \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\hat{\pi}}\left[\sigma(P_{\Theta^*}(\mathbf{a}|\mathbf{s}))\right], \tag{12}$$

Therefore, the expert density's objective can be formulated as :

$$\mathcal{J}(\Theta^*) = \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\pi^*}\left[\mathcal{L}_{\text{ELBO}}(\mathbf{s},\mathbf{a};\Theta^*,\Phi^*)\right] + \lambda\cdot\mathcal{J}_{\texttt{ADE}}(\Theta^*). \tag{13}$$

Accordingly, the objective for non-expert density can be formulated by substituting $\Theta^*$ and $\Phi^*$ in Equation 13 with $\hat{\Theta}$ and $\hat{\Phi}$. However, in practical implementations, we find that setting $\lambda = 0$ is sufficient to achieve good performance for sub-optimal behavior density.

**Density Weighted Regression (DWR).** After using ADE and obtaining the converged VAE estimators $P_{\Theta^*}(\mathbf{a}|\mathbf{s})$ and $P_{\hat{\Theta}}(\mathbf{a}|\mathbf{s})$. We freeze the parameter of these estimators, then approximate the **density weight** $W(\hat{P}, P^*) = \log\frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}$ using importance sampling:

$$\log\frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})} \approx \log p_{\hat{\Theta}}(\mathbf{a}|\mathbf{s}) - \log p_{\Theta^*}(\mathbf{a}|\mathbf{s}) \approx \mathcal{L}_{\pi_\beta}(\mathbf{s},\mathbf{a};\hat{\Theta},\hat{\Phi},L) - \mathcal{L}_{\pi_\beta}(\mathbf{s},\mathbf{a};\Theta^*,\Phi^*,L), \tag{14}$$

and then bring density weight into Equation 8 or 9, optimizing policy via gradient decent *i.e.*, $\theta \leftarrow \theta - \eta\cdot\nabla_\theta\mathcal{J}(\pi_\theta)$, where $\eta$ denotes learning rate (lr).

**Practical Implementation.** ADR comprises VAE Pre-training (Algorithm 1) and policy training (Algorithm 2) stages. During the VAE pre-training stage, we utilize VQ-VAE to separately estimate the target density $P^*(\mathbf{a}|\mathbf{s})$ and the suboptimal density $\hat{P}(\mathbf{a}|\mathbf{s})$ by minimizing Equation 9 (or Equation 13) and the VQ loss (van den Oord et al., 2018). During the policy training stage, we optimize the Multiple Layer Perception (MLP) policy $\pi_\theta$ by using Equation 7. For more details about our model architecture and more hyper-parameter settings, please refer to the Appendix. In terms of evaluation. We compute the normalized D4RL (normalized) score with the same method as Fu et al., and our experimental result is obtained by averaging the highest score in multiple runs.

**Algorithm 1** VAE Pretraining

**Require:** VAE (density estimator) parameterized by $(\Theta^*, \Phi^*)$ for expert dataset, VAE parameterized by $(\hat{\Theta}, \hat{\Phi})$ for unknown-quality dataset. Empirical policy $\pi_\theta(\cdot|\mathbf{s})$, unknown-quality offline datasets $\hat{\mathcal{D}}$, demonstrations $\mathcal{D}^*$; VAE training epochs $N_{VAE\ train}$ and policy training epochs $N_{policy\ train}$.

1: **while** $t_1 \leq N_{VAE\ train}$ **do**
2: Sample batch sub-optimal trajectory $\hat{\tau}$ from $\hat{\mathcal{D}}$, and sampling batch expert trajectory $\tau^*$ from $\mathcal{D}^*$.
3: update $(\Theta^*, \Phi^*)$ by Equation 13. Replace $(\Theta^*, \Phi^*)$ in Equation 13 with $(\hat{\Theta}, \hat{\Phi})$, and update $(\hat{\Theta}, \hat{\Phi})$.
4: **end while**

**Algorithm 2** Training Policy

**Require:** pre-trained density estimators $\hat{P}$, $P^*$, and datasets $\mathcal{D} = \hat{\mathcal{D}}\cup\mathcal{D}^*$

1: **while** $t_2 \leq N_{policy\ train}$ **do**
2: Computing $\mathcal{W}(\hat{P}, P^*) = \log\frac{P_{\hat{\Phi}}(\mathbf{a}|\mathbf{s})}{P_{\Phi^*}(\mathbf{a}|\mathbf{s})}$.
3: Bring $\mathcal{W}(\hat{P}, P^*)$ to Equation 9 or 7 and updating $\pi_\theta$.
4: **end while**

## 7 Evaluation

Our experiments are designed to answer: 1) Does ADR outperform previous IL approaches? We additionally encompass 2) Is it necessary to use an adversarial approach to assist in estimating the target behavior density? 3) Is it necessary to use the density-weighted form to optimize the policy? 4) Performance Comparison between ADR and a Highly Effective Diffusion Policy.

**Datasets.** The majority of our experimental setups are centered around Learning from Demonstration (LfD). For convenience, we denote using n demonstrations to conduct experiments under the LfD setting as LfD (n). We test our method on various domains, including Gym-Mujoco, Androit, and Kitchen domains (Fu

et al., 2021). Specifically, the datasets from the Gym-Mujoco domain include `medium` (`m`), `medium-replay` (`mr`), and `medium-expert` (`me`) collected from environments including `Ant`, `Hopper(hop)`, `Walker2d(wal)`, and `HalfCheetah(che)`, and the demonstrations are 5 expert trails from the respective environments. For the `kitchen` and `Adroit` domains, we rank and sort all trials by their return, and sample the trial with the highest return as demonstration. *The content inside the parentheses* `()` *represents an abbreviation.*

Table 1: **Previous IL approaches**. We summarize the majority of previous IL/DICE approaches here. Specifically, most of these methods involve estimating the reward or value function and are followed by optimizing with the weighted BC objective.

| Algorithm | Optimizing framework | estimating Target | Weighted BC |
|---|---|---|---|
| OTR (Luo et al., 2023) | IQL | $r(\mathbf{s}, \mathbf{a})$ | ✔ |
| SQIL (Reddy et al., 2019) | IQL | $r(\mathbf{s}, \mathbf{a})$ | ✔ |
| CLUE (Liu et al., 2023a) | IQL | $r(\mathbf{s}, \mathbf{a})$ | ✔ |
| IQ-Learn (Garg et al., 2022) | Inverse SAC | $r(\mathbf{s}, \mathbf{a})$ | ✗ |
| OIRL (Zolna et al., 2020) | Q-weighted BC | $r(\mathbf{s}, \mathbf{a})$ | ✔ |
| ValueDice (Kostrikov et al., 2019) | DICE | - | ✔ |
| Demodice (Kim et al., 2022) | DICE | - | ✔ |
| SMODICE (Ma et al., 2022a) | DICE | - | ✔ |
| ABC (Sasaki and Yamashina, 2021) | AL | - | ✗ |
| Noisy BC (Sasaki and Yamashina, 2021) | BC | - | ✗ |
| CEIL (Liu et al., 2023b) | HIM | $\mathbf{z}^*$ | ✗ |
| **ADR** (ours) | Density Weighted BC | $\hat{P}(\mathbf{a}\|\mathbf{s})$ and $P^*(\mathbf{a}\|\mathbf{s})$ | ✔ |

**Baselines.** The majority of selected baselines are shown in Table 1. Specifically, when assessing the Gym-Mujoco domain, the baselines encompass ORIL, SQIL, IQ-Learn, ValueDICE, DemoDICE, SMODICE utilized RL-based weighted BC approaches to update. Additionally, we also compare ADR with previous competitive contextualized BC framework CEIL. *In particular*, we selected DICE as one of our baselines due to its similar data setting to ADE and its high data efficiency.. When testing on kitchen or Adroit domains, we compare our methods with IL algorithms including OTR and CLUE that utilize reward relabeling approach, and policy optimization via Implicit Q Learning (IQL) (Kostrikov et al., 2021), besides, we also compare ADR with Conservative Q Learning (CQL) (Kumar et al., 2020b) and IQL utilizing ground truth reward separately denoted CQL (oracle) and IQL (oracle), where oracle denotes ground truth reward. However, we do not compare ADR with ABC and Noisy BC because our ablations (`Max ADE`, Noisy Test) have included settings with similar objectives. Furthermore, diffusion policy demonstrates strong performance in behavioral cloning. Therefore, we simultaneously chose recent latent-guided (Li, 2023) and return-guided (Ajay et al., 2023) diffusion policies for comparison on the Andriot task.

### 7.1 Majority experimental results

**LfD on Adroit and kitchen domains.** We test ADR on tasks sourced from Adroit and Kitchen domains. In particular, during the training process, we utilize single trajectory with the highest Return as a demonstration. The experimental results are summarized in Table 8, ADR achieves an impressive summed score of **526.6** points, representing an improvement of **89.5%** compared to IQL (oracle), **121.1%** compared to CQL (oracle), and surpassing all IL baselines, thus showcasing its competitive performance in long-horizon IL tasks. Meanwhile, these competitive experimental results also validate our claim that ADR, which optimizes policy in a single-step manner, can avoid the cumulative bias associated with multi-step updates using biased reward/Q functions within the RL framework. Moreover, this experiment also indicates the feasibility of utilizing ADR to conduct LfD without introducing extra datasets as demonstrations.

**LfD on Gym-Mujoco domain.** The majority of the experimental results on the tasks sourced from the Gym-Mujoco domain are displayed in Table 3. We utilized 5 expert trajectories as demonstrations and conduct ILD on all selected tasks. ADR achieves a total of **1008.7** points, surpassing most reward estimating and Q function estimating approaches. Therefore, the performance of our approach to continuous control has been validated. In particular, 1) ADR performs better than ORIL, IQL-Learn demonstrating the advantage of ADR over reward estimating+RL approaches. 2) The superior performance of ADR compared to SQIL, DemoDice, SMODICE, and ValueDice highlights the density weights over other regressive forms.

Table 2: **Experimental results of Kitchen and Adroit domains.** We test ADR on Adroit and kitchen domains and average the normalized D4RL score across 4 seeds. In particular, the experimental results of BC, CQL (oracle), and IQL (oracle) are directly quoted from Kostrikov et al. (2021), and results of IQL (OTR) on adroit domain are directly quoted from Luo et al., where oracle denotes ground truth reward.

| LfD (1) | BC | CQL (oracle) | IQL (oracle) | IQL (OTR) | IQL (CLUE) | ADR |
|---|---|---|---|---|---|---|
| door-cloned | 0.0 | 0.4 | 1.6 | 0.01 | 0.02 | **4.8±1.1** |
| door-human | 2 | 9.9 | 4.3 | 5.92 | 7.7 | **12.6±3.9** |
| hammer-cloned | 0.6 | 2.1 | 2.1 | 0.88 | 1.4 | **17.6±3.3** |
| hammer-human | 1.2 | 4.4 | 1.4 | 1.79 | 1.9 | **21.7±11.8** |
| pen-cloned | 37 | 39.2 | 37.3 | 46.87 | 59.4 | **84.4±19.2** |
| pen-human | 63.9 | 37.5 | 71.5 | 66.82 | 82.9 | **120.6±10.3** |
| relocate-cloned | -0.3 | -0.1 | -0.2 | -0.24 | -0.23 | **-0.2±0.0** |
| relocate-human | 0.1 | 0.2 | 0.1 | 0.11 | 0.2 | **2.0±1.4** |
| *Total (Androit)* | 104.5 | 93.6 | 118.1 | 122.2 | 153.3 | **263.5** |
| kitchen-mixed | 51.5 | 51.0 | 51.0 | 50.0 | - | **87.5±1.8** |
| kitchen-partial | 38.0 | 49.8 | 46.3 | 50.0 | - | **80.6±2.7** |
| kitchen-completed | 65.0 | 43.8 | 62.5 | 50.0 | - | **95.0±0.0** |
| *Total (Kitchen)* | 104.5 | 144.6 | 159.8 | 150.0 | - | **263.1** |
| **Total (Kitchen&Androit)** | 259 | 238.2 | 277.9 | 272.2 | - | **526.6** |

Table 3: **Experimental results of Gym-Mujoco domain.** We utilize 5 expert trajectories as a demonstration to conduct LfD setting IL experiment, our experimental results are averaged multiple times of runs. In particular, `m` denotes `medium`, `mr` denotes `medium-replay`, `me` denotes `medium-expert`.

| LfD (5) | ORIL (TD3+BC) | SQIL (TD3+BC) | IQ-Learn | ValueDICE | DemoDICE | SMODICE | CEIL | ADR |
|---|---|---|---|---|---|---|---|---|
| hopper-me | 51.2 | 5.9 | 21.7 | 72.6 | 63.7 | 64.7 | 80.8 | **109.1±3.2** |
| halfcheetah-me | 79.6 | 11.8 | 6.2 | 1.2 | 59.5 | 63.8 | 33.9 | **74.3±2.1** |
| walker2d-me | 38.3 | 13.6 | 5.2 | 7.4 | 101.6 | 55.4 | 99.4 | **110.1±0.2** |
| Ant-me | 6.0 | -5.7 | 18.7 | 30.2 | 112.4 | 112.4 | 85.0 | **132.7±0.3** |
| hopper-m | 42.1 | 45.2 | 17.2 | 59.8 | 50.2 | 54.1 | **94.5** | 69.0±1.1 |
| halfcheetah-m | **45.1** | 14.5 | 6.4 | 2 | 41.9 | 42.6 | **45.1** | 44.0±0.1 |
| walker2d-m | 44.1 | 12.2 | 13.1 | 2.8 | 66.3 | 62.2 | **103.1** | 86.3±1.7 |
| Ant-m | 25.6 | 20.6 | 22.8 | 27.3 | 82.8 | 86.0 | 99.8 | **106.6±0.5** |
| hopper-mr | 26.7 | 27.4 | 15.4 | 80.1 | 26.5 | 34.9 | 45.1 | **74.7±1.7** |
| halfcheetah-mr | 2.7 | 15.7 | 4.8 | 0.9 | 38.7 | 38.4 | **43.3** | 39.2±0.1 |
| walker2d-mr | 22.9 | 7.2 | 10.6 | 0 | 38.8 | 40.6 | 81.1 | **67.3±4.7** |
| Ant-mr | 24.5 | 23.6 | 27.2 | 32.7 | 68.8 | 69.7 | **101.4** | 95.4±1.1 |
| **Total (Gym-Mujoco)** | 408.8 | 192 | 169.2 | 316.9 | 751.2 | 724.7 | 912.5 | **1008.7** |

**Performance comparison with diffusion policy.** The conditioned diffusion policy is also an exceptionally powerful trajectory generation paradigm based on supervised learning. Therefore, we compare the effectiveness of ADR with Latent Diffuser (LD) (Li, 2023) that use latent space representation and Decision Diffuser (DD) (Ajay et al., 2023) that use Return as the conditioning factor to guided trajectory generation (Our dataset originates from the robotic arm domain and includes `cloned` and `human` datasets for four tasks: `door`, `hammer`, `pen`, and `relocate`.). The experimental results of ADR are better than those of DD and PD.

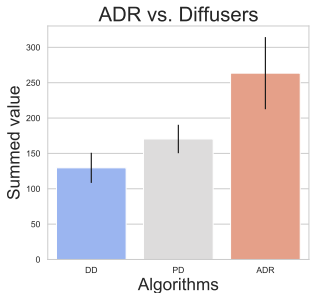


Figure 3: Total test scores on the Androit domain.

### 7.2 Ablations

**Ablation of ADE and DWR.** To demonstrate the effectiveness of ADE, we excluded ADE *i.e.*, $J_{\texttt{ADE}}(\Theta^*)$ from ADR during the VAE training process. Subsequently, we optimize policy by maximizing the target behavior density and minimizing the sub-optimal behavior density, and we named this experimental setting ADR (wo ADE). As shown in Figure 4 (a). ADR (wo ADE) performs better than ADR with over 50% confidence, validating the improvement brought by ADE. Meanwhile, in order to demonstrate the necessity of DWR, we 1) conduct an ablation by removing DWR, denoted as ADR (wo DWR), and find that ADR performs better than ADR (wo DWR) over 95% confidence. This indicates that DWR is necessary for ADR. 2) Optimizing the policy by solely maximizing the expression $\mathcal{L}_{\pi_\beta}(\mathbf{s}, \pi_\theta(\cdot|\mathbf{s}); \Theta^*, \Phi^*, L)|_{\mathbf{s}\sim\mathcal{D}}$, which is termed as `max ADE`, as shown in Figure 4 (a). According to the results, ADR performs better than `max ADE` with over 90% confidence. Therefore, we can't optimize the policy solely by utilizing ADE and maximizing likelihood. Besides, we observe that it won't bring an overwhelming decrease by removing ADE, therefore, we further

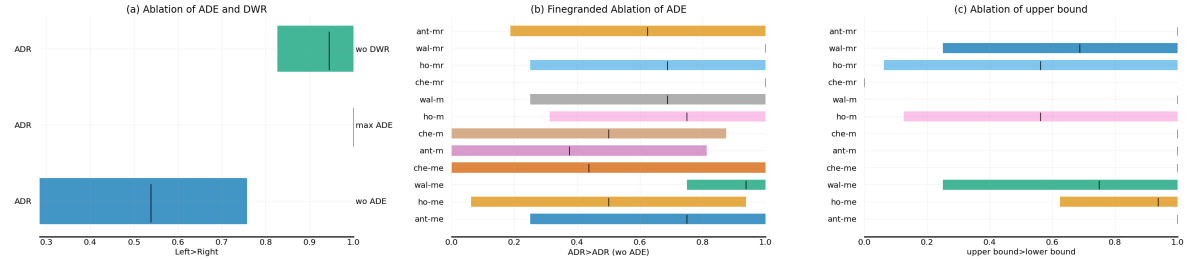


Figure 4: Ablation Results. We utilized the reliable library proposed by Agarwal et al. to conduct our experiments. The results show that the experimental setting on the left side performed better with a higher probability. Specifically, in (a) we removed part of modules *i.e.*, ADE or DWR from ADR and observed a reduction in performance. In (b), we further conduct comparisons among all tasks. Regarding (c), we carried out a fine-grained comparison of the upper and lower bounds of Equation 8 among all tasks. Note, (a) The left and right y-axes represent the selected algorithms A and B, respectively, while the x-axis represents the confidence in A>B. (b, c) involve comparisons between two algorithms, and left y-axis are selected tasks.

conduct fine-grand comparison across all tasks from Gym-mujoco domain, and we observe that ADR performs better than ADR (wo ADE) across all selected `mr` tasks, but lower than 50% confidence across several `m` or `me` tasks. Therefore, ADE is essential for training with lower-quality $\hat{\mathcal{D}}$, and won't bring too much improvement for training with higher-quality $\hat{\mathcal{D}}$.

**Robustness to demo' noisy.** In order to validate that ADR is robust to the demonstrations' noise, we choose `hop-m`, `wal-m`, `ant-m`, and `che-m`, then add Gaussian noisy $\Delta(\mathbf{a}) \sim \mathcal{N}(0, 1)$ to demonstrations with weight $w \in \{0.1, 0.3, 0.6, 0.9\}$ *i.e.*, $\hat{\mathbf{a}} \leftarrow \mathbf{a} + w \cdot \Delta(\mathbf{a})$, and utilize the Gaussian noised action to train our policy, further observing the performance decreasing. As shown in Figure 5. ADR can be well adapted to the demonstrations' noise. As the noise ratio increases, our method shows only a slight decline in performance on `ant-m`. However, there is no significant drop in performance on other tasks such as `wal-m`, `hop-m`, and `che-m`. Therefore, ADR has a certain level of noise resistance and can still maintain relatively good performance even in the presence of noise within demonstrations. (Extended discussion in Appendix 8)

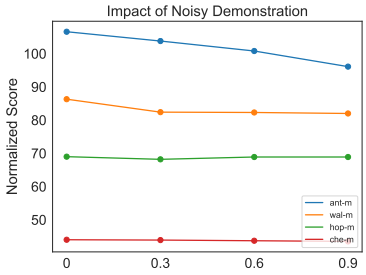


Figure 5: ADR's performance changes as the noise in the demonstrations increases.

**Ablation of the upper bound of ADR.** As shown in Figure 4 (c), optimizing the upper bound achieved better performance across 11 out of 12 tasks (except for `che-mr`) from the Gym-mujoco domain with over 50% confidence. Therefore, it is much more effective to optimize Equation 9 rather than Equation 7.

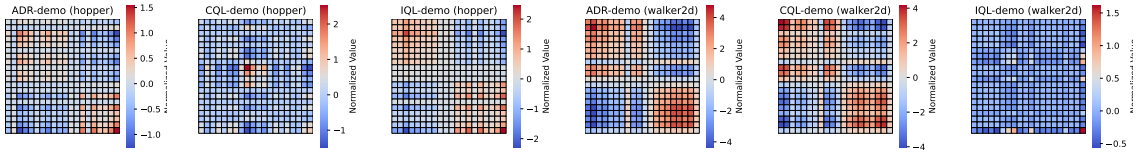


Figure 6: Heatmap of policy distributions. We stack the model's predictions alongside the samples in the dataset (details see Appendix G).

**Comparison of different methods' OOD risky.** To validate our claim that ADR is a supervised in-sample IL approach and therefore ADR does not suffer from OOD samples, we compare three different offline algorithms, including CQL (oracle), IQL (oracle), all using the same offline datasets. Specifically, we

first train policies using three different algorithms: ADR, CQL (oracle), IQL (oracle), each with the same datasets. For example, when training ADR with $\mathcal{D}^*$ and $\hat{\mathcal{D}}$, we simultaneously train CQL (oracle), IQL (oracle) using $\mathcal{D}^* \cup \hat{\mathcal{D}}$. After obtaining the pre-trained models, we sample states from the expert dataset and input them into these pre-trained models. We then plot heatmaps comparing the logits obtained from these models with the expert policy. As shown in Figure 6. ADR maintains its decision mode as a demonstration while being less susceptible to OOD scenarios (Note: the more similar the top-left and bottom-right corners of the heatmap are, the closer the algorithm is to the demo).

## 8 Extended Discussion

**Intuition behind ADR's robustness to noise.** As illustrated in Figure 4, we evaluated the robustness of ADR to noise across various Gaussian noise weights. The experimental results demonstrate that ADR maintains its performance across tasks, with the exception of the `ant` task. This stability can be attributed to ADR's density-weighted BC objective, which benefits from a substantial amount of suboptimal data during training. Consequently, ADR exhibits relative stability in the presence of noise. Then, we further analyze the noise robustness of ADR and propose solutions to enhance its noise robustness. We conduct training sessions using different weights under LfD (20) and LfD (5). Our findings reveals that LfD (20) is more susceptible to noise than LfD (5), particularly in the `ant` task. Conversely, LfD (5) exhibits relatively stable results, with no notable decline in performance. It is worth noting that the noise weight assigned to LfD (5) is actually higher than that of LfD (20). Based on above analysis, we will consider appropriately reducing the number of demos as one of the methods to enhance the effectiveness of ADR.

Table 4: Test with 20 demos.

| **LfD (20)** | $w = 0$ | $w = 0.3$ | $w = 0.9$ |
|---|---|---|---|
| `hopper-m` | 69.0±1.1 | 68.5±0.6 | 68.9±2.0 |
| `walker2d-m` | 87.9±0.7 | 82.3±0.2 | 82.0±0.7 |
| `Ant-m` | 106.6±0.5 | 103.d5±0.5 | 103.5±1.0 |
| `halfcheetah-m` | 44.0±0.1 | 43.8±0.2 | 43.5±0.2 |

Table 5: Test with 5 demos.

| **LfD (5)** | $w = 0$ | $w = 0.5$ | $w = 1.5$ |
|---|---|---|---|
| `hopper-m` | 69.0±1.1 | 65.9±0.2 | 68.5±2.4 |
| `walker2d-m` | 86.3±1.7 | 82.9±1.2 | 81.7±0.2 |
| `Ant-m` | 106.6±0.5 | 104.6±0.1 | 104.1±0.8 |
| `halfcheetah-m` | 44.0±0.1 | 43.6±0.1 | 43.2±0.1 |

**Why do we choose VAE as the density estimator?** Common density estimators encompass Gaussian models, Variational Autoencoders (VAEs), transformers, and diffusion-based models. Among these, diffusion models stand out for their superior representation capabilities. However, they entail a multi-step sampling process, which makes them computationally more demanding compared to VAEs and Gaussian models. When considering the application of these estimators in reinforcement learning (RL), although Gaussian models may offer simplicity in terms of fewer parameters, VAEs prove to be more suitable as density estimators in RL contexts. Our method, leveraging VAEs, demonstrates robust performance and offers reduced computational complexity compared to diffusion models. Notably, our research is situated within the MDP setting. In contrast, diffusion models and transformers are predominantly sequence models utilized in non-MDP settings, which can pose implementation challenges within the constraints of MDPs. For instance, given a current state-action pair $\{\mathbf{s}_0, \mathbf{a}_0\}$, it is impractical to construct a sequence $\{\mathbf{s}_{0-k}, \mathbf{a}_{0-k}, \cdots \mathbf{s}_0, \mathbf{a}_0\}$ for sequence models within an MDP context. However, to access performance difference between ADR and diffusers, we conduct comparison in Figure 3 of the main text. Our experimental results demonstrate that ADR performs better than the selected diffusion polices (results shown in Figure 3.). Additionally, there is currently another type of model called Flow(Lipman et al., 2023) that has been garnering increasing attention. Although the Flow model exhibits lower computational complexity during inference compared to diffusion models, estimating density with Flow requires tailored label design for the samples. In contrast, VAE doesn't necessitate task-specific label design.

## 9 Conclusion

We propose ADR, a density-weighted BC objective that utilizes a single-step update paradigm to align the empirical policy with the expert distribution. Benefits from ADR's single-step updates, ADR is not affected by the accumulated offset caused by multi-step rewards. And, ADR outperforms selected IL/DICE algorithms, diffusion policies.

## References


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

Anthony Brohan, Noah Brown, Justice Carbajal, and etc. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023b.

Aarushi Bhargava, Vamsi C. Meesala, Muhammad R. Hajj, and Shima Shahab. Nonlinear effects in high-intensity focused ultrasound power transfer systems. *arXiv preprint arXiv:2006.12691*, 2020.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Meta. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2307.09288*, 2023.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 09–15 Jun 2019a. URL https://proceedings.mlr.press/v97/fujimoto19a.html.

Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *arXiv preprint arXiv:2202.06239*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/0d2b2061826a5df3221116a5085a6052-Paper.pdf.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. 2021.

Zhepeng Cen, Zuxin Liu, Zitong Wang, Yihang Yao, Henry Lam, and Ding Zhao. Learning from sparse offline datasets via conservative density estimation. *arXiv preprint arXiv:2401.08819*, 2024.

Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2008.10.024. URL https://www.sciencedirect.com/science/article/pii/S0921889008001772.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*, 2016.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*, 2019.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

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

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2019b.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*, 2015.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2022.

Mohamed Debbagh. Learning structured output representations from attributes using deep conditional generative models. *arXiv preprint arXiv:2305.00980*, 2023.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. *arXiv preprint arXiv:1502.03509*, 2015.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2021.

Kshitij Judah, Alan Fern, Prasad Tadepalli, and Robby Goetschalckx. Imitation learning with demonstrations and shaping rewards. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), Jun. 2014. doi: 10.1609/aaai.v28i1.9024. URL `https://ojs.aaai.org/index.php/AAAI/article/view/9024`.

Daniel S. Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 330–359. PMLR, 30 Oct–01 Nov 2020. URL `https://proceedings.mlr.press/v100/brown20a.html`.

Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020. doi: 10.1146/annurev-control-100819-063206. URL `https://doi.org/10.1146/annurev-control-100819-063206`.

Damian Boborzi, Christoph-Nikolas Straehle, Jens S. Buchner, and Lars Mikelsons. Imitation learning by state-only distribution matching. *arXiv preprint arXiv:2202.04332*, 2022.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011a. PMLR. URL `https://proceedings.mlr.press/v15/ross11a.html`.

YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125, 2018. doi: 10.1109/ICRA.2018.8462901.

Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011b.

Simon Damm, Dennis Forster, Dmytro Velychko, Zhenwen Dai, Asja Fischer, and Jörg Lücke. The elbo of variational autoencoders converges to a sum of three entropies. *arXiv preprint arXiv:2010.14860*, 2023.

Yecheng Jason Ma, Andrew Shen, Dinesh Jayaraman, and Osbert Bastani. Versatile offline imitation from observations and examples via regularized state-occupancy matching. *arXiv preprint arXiv:2202.02433*, 2022a.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2016.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2018.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2021.

Yicheng Luo, Zhengyao Jiang, Samuel Cohen, Edward Grefenstette, and Marc Peter Deisenroth. Optimal transport for offline imitation learning. *arXiv preprint arXiv:2303.13971*, 2023.

Siddharth Reddy, Anca D. Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019.

Jinxin Liu, Lipeng Zu, Li He, and Donglin Wang. Clue: Calibrated latent guidance for offline reinforcement learning. *arXiv preprint arXiv:2306.13412*, 2023a.

Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, Matthieu Geist, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. *arXiv preprint arXiv:2106.12142*, 2022.

Konrad Zolna, Alexander Novikov, Ksenia Konyushkova, Caglar Gulcehre, Ziyu Wang, Yusuf Aytar, Misha Denil, Nando de Freitas, and Scott Reed. Offline learning from demonstrations and unlabeled experience. *arXiv preprint arXiv:2011.13885*, 2020.

Geon-Hyeong Kim, Seokin Seo, Jongmin Lee, Wonseok Jeon, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. DemoDICE: Offline imitation learning with supplementary imperfect demonstrations. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=BrPdX1bDZkQ`.

Fumihiro Sasaki and Ryota Yamashina. Behavioral cloning from noisy demonstrations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=zrT3HcsWSAt`.

Jinxin Liu, Li He, Yachen Kang, Zifeng Zhuang, Donglin Wang, and Huazhe Xu. Ceil: Generalized contextual imitation learning. *arXiv preprint arXiv:2306.14534*, 2023b.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020b.

Wenhao Li. Efficient planning with latent diffusion. *arXiv Preprint:2310.00311*, 2023. URL `https://arxiv.org/abs/2310.00311`.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv Preprint:2211.15657*, 2023. URL `https://arxiv.org/abs/2211.15657`.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *arXiv preprint arXiv:2108.13264*, 2022.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. 2023.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL `https://openreview.net/forum?id=SyAS49bBcv`.

Yecheng Jason Ma, Andrew Shen, Dinesh Jayaraman, and Osbert Bastani. Versatile offline imitation from observations and examples via regularized state-occupancy matching. *arXiv preprint arXiv:2202.02433*, 2022b.

## Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

*Our exploration of this topic is not aimed at entirely replacing the RL-combined IL (Reinforcement Learning-combined Imitation Learning) paradigm. RL has evolved to possess unique strengths; for instance, it enables end-to-end integration from dataset collection to model training. Consequently, when an accessible environment is available, RL allows for the gradual improvement of policies—a advantage that is hard to substitute. Our research is merely an attempt to find an alternative solution for scenarios where reward functions are challenging to estimate. However, our approach still has certain limitations. For example, implementing our solution in online environments may not be particularly straightforward, which warrants further investigation. Additionally, our method does not effectively synergize with sequential models, and this incompatibility represents a limitation of our proposed approach.*

## A Limitations

We have currently attempted to extend ADR to sequential models, such as the Decision Transformer (DT) (Chen et al., 2021) (Remove the Return token and use transformer as a fully supervised policy), but we have find that the experimental results are not as impressive as those under the MDP setting. We will further explore the possibility of extending ADR to sequential models.

## B Social Impacts

We propose a new supervised IL framework, ADR. Meanwhile, we point out that the advantage of ADR lies in that it can effectively avoid the cumulative offset sourced from sub-optimal Reward/Value functions. In addition, the effect of ADR exceeds all previous IL frameworks and even achieves better performance than IQL on robotic arm/kitchen tasks, which will greatly promote the development of IL frameworks under supervised learning.

## C Hyper parameters and Implementation details

Our method is slightly dependent on hyper-parameters. We introduce the core hyperparameters here:

Table 6: Crucial hyper-parameters of ADR.

| Hyperparameter | Value |
|---|---|
| VAE training iterations | $1e^5$ |
| policy training iterations | $1e^6$ |
| batch size | 64 |
| learning rate (lr) of $\pi$ | $1e^{-4}$ |
| lr of VQ-VAE | $1e^{-3}$ |
| evaluation frequency | $1e^3$ |
| L in Equation 4 | 1 |
| $\lambda$ in Equation 13 | 1 |
| Optimizing Equation 9 | All selected tasks except for `che-mr` |
| Random Seeds | {0,2,4,6} |
| Optimizing Equation 7 | `che-mr` |
| Model Architecture | |
| MLP Policy | 4× Layers MLP (hidden dim 256) |
| VQVAE (encoder and decoder) | 3× Layers MLP (hidden dim: 2× action dim; latent dim: 750)<br>4096 tabular embeddings |

## D Experiments

**Evaluation setting.** We run each task multiple times, recording all evaluated results and taking the highest score from each run as the outcome. We then average these highest scores. For score computation, we use the same metric as D4RL *i.e.*, $\frac{\text{output}-\text{expert}}{\text{expert}-\text{random}} \times 100$. Our experiment are running on computing clusters with 16×4 core cpu (Intel(R) Xeon(R) CPU E5-2637 v4 @ 3.50GHz), and 16×RTX2080 Ti GPUs

**Implementation details.** Our code is based on CORL (Tarasov et al., 2022). Specifically, in terms of a training framework, we adapted the offline training framework of Supported Policy Optimization (SPOT) (Wu et al., 2022), decomposing it into multiple modules and modifying it to implement our algorithm. Regarding the model architecture, we implemented the VQVAE ourselves, while the MLP policy architecture is based on CORL. Some general details such as warm-up, a discount of lr, e.g. are implemented by CORL. *We have appended our source code in the supplement materials.*

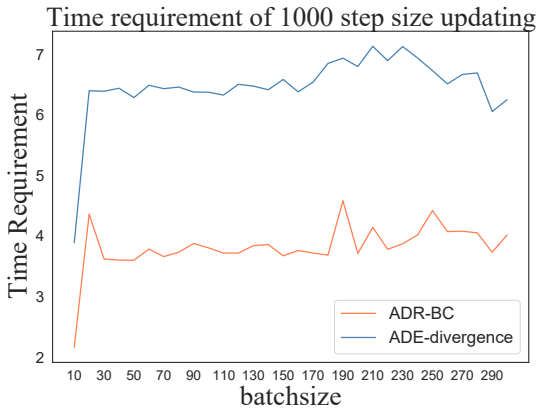


Figure 7: Comparison of training time.

**Computing efficiency of DWR.** To further showcase the computational efficiency of DWR, we selected the `che-mr` environment as the benchmark and systematically varied the batch size from 10 to 300 while measuring the training time (using a 1000-step size in the policy updating stage). As depicted in Figure 7, it's evident that the training time of ADR is significantly lower compared to ADE-divergence (which shares the same conceptual framework as Equation 6), and such advantage becomes especially pronounced with larger batch sizes. Therefore, the computational efficiency of ADR has been convincingly demonstrated.

**Ablation of the upper bound of ADR.** In order to demonstrate the effectiveness of minimizing Equation 9 (upper-bound) over minimizing Equation 7 (objective), we conduct fine-grained comparisons. Specifically, we compare minimizing Equation 9, Equation 7 on all selected tasks sourced from Gym-Mujoco domain (hop denotes hopper, wal denotes walker2d, che denotes halfcheetah), minimizing Equation 9 achieve overall better performance (8 out of 12), indicating the necessity of Equation 9.

## E Experimental Details of baselines

Our baselines on Gym-Mujoco domain mainly includes: ORIL (Zolna et al., 2020), SQIL (Reddy et al., 2019), IQ-Learn (Garg et al., 2022), ValueDICE (Kostrikov et al., 2019), DemoDICE (Kim et al., 2022), SMODICE (Ma et al., 2022a), and CEIL (Liu et al., 2023b). The majority of experimental results of these baselines are cited from CEIL (Liu et al., 2023b).

In terms of evaluation on kitchen or Adroit domains. The majority baselines include OTR (Luo et al., 2023) and CLUE (Liu et al., 2023a) that utilize reward estimating via IL approaches, and policy optimization

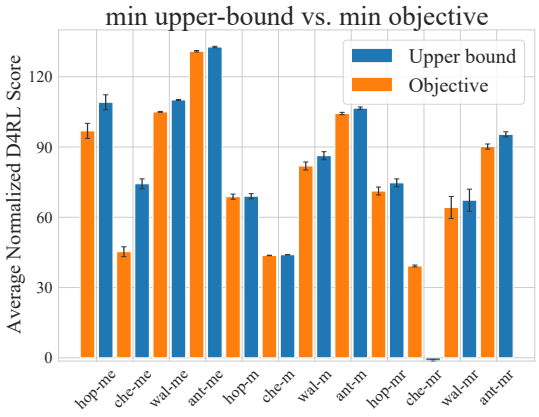


Figure 8: Abaltion of upper bound.

via Implicit Q Learning (IQL) (Kostrikov et al., 2021). We also encompass Conservative Q Learning (CQL) (Kumar et al., 2020b) and IQL for comparison. Specifically, these experimental results are from:

- The experiment results of OTR and CLUE are directly cited from Luo et al. and Liu et al.
- The experimental results of CQL (oracle) and IQL (oracle) are separately cited from Kumar et al. and Kostrikov et al., and the experimental results of OTR on kitchen domain is obtained by running the official codebase `https://github.com/ethanluoyc/optimal_transport_reward`.
- The experimental results of DD and LD in Figure 3 are cited from Li.

## F Theoretical Analysis

**Theorem F.1** (ADR can be solved by linear weighted regression form)**.** *Given expert log behavior density* $\log P^*(\mathbf{a}|\mathbf{s}) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$*, sub-optimal log behavior density* $\log \hat{P}(\mathbf{a}|\mathbf{s}) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$*, and the empirical policy* $\pi_\theta : \mathcal{S} \to \mathcal{A}$*, offline dataset* $\mathcal{D}$*. Minimizing the KL divergence between* $\pi_\theta$ *and* $P^*$*, while maximizing the KL divergence between* $\pi_\theta$ *and* $\hat{P}$*, i.e., Equation 6. is equivalent to:* $\min_{\pi_\theta} \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[\log \frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})} \cdot ||\pi_\theta(\cdot|\mathbf{s}) - \mathbf{a}||^2\big]$*,*

**Proof**.

$$
\begin{aligned}
J(\pi_\theta) &= \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}[D_{\mathrm{KL}}[\pi_\theta||P^*] - D_{\mathrm{KL}}[\pi_\theta||\hat{P}]] \\
&= \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\left[\pi_\theta(\mathbf{a}|\mathbf{s}) \cdot \log \frac{\pi_\theta(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\right] - \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\left[\pi_\theta(\mathbf{a}|\mathbf{s}) \cdot \log \frac{\pi_\theta(\mathbf{a}|\mathbf{s})}{\hat{P}(\mathbf{a}|\mathbf{s})}\right] \\
&= \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\left[\pi_\theta(\mathbf{a}|\mathbf{s}) \cdot \left(\log \frac{\pi_\theta(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})} - \log \frac{\pi_\theta(\mathbf{a}|\mathbf{s})}{\hat{P}(\mathbf{a}|\mathbf{s})}\right)\right] \\
&= \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[\pi_\theta(\mathbf{a}|\mathbf{s}) \cdot \log \frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\big].
\end{aligned}
\tag{15}
$$

Then we further derivative $\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[\pi_\theta(\mathbf{a}|\mathbf{s}) \cdot \log \frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\big]$:

According to our assumption in section of **Problem formulation**, the policy we employ is a Gaussian policy, which adheres to a Gaussian distribution $\pi \sim \mathcal{N}(\mu, \sigma)$, specifically, given i.d.d variables $\pi_0, \pi_1, \cdots, \pi_n$, it has $\mu = \frac{\sum \pi_i}{n}$ and $\sigma = \frac{\sum |\pi - \mu|}{n}$

Furthermore, we consider the process of maximizing the likelihood $\pi_\theta(\mathbf{a}|\mathbf{s})$ rather than only maximizing $\pi_\theta(\mathbf{a}|\mathbf{s})$ *i.e.*

$$\begin{aligned}\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[\pi_\theta(\mathbf{a}|\mathbf{s})\cdot\log\frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\big] &\equiv \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[\log\pi_\theta(\mathbf{a}|\mathbf{s})\cdot\log\frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\big]\\ &= \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[\log\frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{(\pi_\theta(\mathbf{a}|\mathbf{s})-\mu)^2}{2\sigma^2})\cdot\log\frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\big]\\ &= \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[(\log\frac{1}{\sigma\sqrt{2\pi}}-\frac{(\pi_\theta(\mathbf{a}|\mathbf{s})-\mu)^2}{2\sigma^2})\cdot\log\frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\big]\\ &= \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[const-\frac{(\pi(\mathbf{a}|\mathbf{s})-\mu)^2}{2\sigma^2}\cdot\log\frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\big].\end{aligned} \tag{16}$$

Subsequently, we substitute $\mu$ with $\frac{\sum\mathbf{a}_i}{n}$ when $\pi(\mathbf{a}|\mathbf{s})$ is given. In particular, the index $i$ does not represent a time index but rather identifies the variable $\mathbf{a}$. Subsequently, we obtain:

$$\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[const-\frac{(\pi_\theta(\mathbf{a}|\mathbf{s})-\frac{\sum\mathbf{a}_i}{n})^2}{2\sigma^2}\cdot\log\frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\big]=\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[const-\frac{(\frac{\sum(\pi_\theta(\mathbf{a}|\mathbf{s})-\mathbf{a}_i)}{n})^2}{2\sigma^2}\cdot\log\frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\big], \tag{17}$$

where *const* represents a constant.

Furthermore, since the offline data do not vary with the time index, the $\sigma$ is a fixed value, minimizing $\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[const-\frac{(\frac{\sum(\pi_\theta(\mathbf{a}|\mathbf{s})-\mathbf{a}_i)}{n})^2}{2\sigma^2}\cdot\log\frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\big]$ is equivalent to minimizing $\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[(\frac{(\sum(\pi_\theta(\mathbf{a}|\mathbf{s})-\mathbf{a}_i)^2}{n})\cdot\log\frac{P^*(\mathbf{a}|\mathbf{s})}{\hat{P}(\mathbf{a}|\mathbf{s})}\big]$.

Therefore, minimizing $J(\pi_\theta)$ is equivalent to minimizing $\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[(\pi-\mathbf{a})^2\cdot\log\frac{P^*(\mathbf{a}|\mathbf{s})}{\hat{P}(\mathbf{a}|\mathbf{s})}\big]$.

**Assumption F.2.** *Assuming* $\max\{D_{KL}[\pi^*||\hat{\pi}], D_{KL}[\hat{\pi}||\pi^*]\}\leq\delta$.

**Theorem F.3.** *Given $\mathcal{D}^*$, based on Assumption F.2, we have:*

$$\mathbb{E}_{\mathcal{D}^*}[\pi^*\log\frac{\pi^*}{\hat{\pi}}]\leq\frac{M}{2n}\cdot\sqrt{\log\frac{2}{\delta}}, \tag{18}$$

*with probability* $1-\delta$. *Where* $n=|\mathcal{D}^*|$, $M=\max_{(\mathbf{s}_t,\mathbf{a}_t)}\pi^*(\mathbf{a}_t|\mathbf{s}_t)\log\frac{\pi^*(\mathbf{a}_t|\mathbf{s}_t)}{\hat{\pi}(\mathbf{a}|\mathbf{s})}|_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}^*}$

**Proof**.

Our derivation is based on Hoeffding in-equality, and We first let $X_i=\pi^*(\mathbf{a}_i|\mathbf{s}_i)\log\frac{\pi^*(\mathbf{a}_i|\mathbf{s}_i)}{\hat{\pi}(\mathbf{a}|\mathbf{s})}$, $\bar{X}=\frac{\sum_t X_t}{n}$, then we have:

$$P(|\bar{X}_i-\mathbb{E}_{\pi^*}[D_{KL}[\pi||\pi^*]]|\geq m)\leq 2\cdot e^{-\frac{2n^2\cdot m^2}{M^2}}. \tag{19}$$

Then let $2\cdot e^{-\frac{2n^2\cdot m^2}{M^2}}=\delta$, we obtain $t=\frac{M}{2n}\sqrt{\log\frac{2}{\delta}}$. Furthermore, with $1-\delta$ probability we have:

$$|\bar{X}_i-\mathbb{E}_{\pi^*}[D_{KL}[\pi||\pi^*]]|\leq 2\cdot e^{-\frac{2n^2\cdot m^2}{M^2}}. \tag{20}$$

Meanwhile, we have assumed that $D_{KL}[\pi^*||\hat{\pi}]\leq\delta$, and thus we obtain $\mathbb{E}_{\mathcal{D}^*}[\pi^*\log\frac{\pi^*}{\hat{\pi}}]\leq\frac{M}{2n}\cdot\sqrt{\log\frac{2}{\delta}}$.

**Proposition F.4** (Policy Convergence of ADR)**.** *Assuming Equation 6 can finally converge to $\epsilon$ via minimizing Eq 8, meanwhile, assuming Assumption F.2 is held. Then* $\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\hat{\mathcal{D}}}[D_{KL}(\pi||\pi^*)]\to\frac{M}{2n}\cdot\sqrt{\log\frac{2}{\delta}}+\Delta C+\epsilon$. *Where* $n=|\mathcal{D}^*|$,$M:=\arg\max_{X_i}\{X_i=\pi^*(\mathbf{a}_t|\mathbf{s}_t)\log\frac{\pi^*(\mathbf{a}_t|\mathbf{s}_t)}{\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t)}|(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}^*\}$ *with probability* $1-\delta$.

**Proof**.

Using Bayes' rule, we have: $P^*(\mathbf{a}|\mathbf{s}) = \frac{\pi^*(\mathbf{a}|\mathbf{s})P(\mathbf{s})}{P^*(\mathbf{s})}, \quad \hat{P}(\mathbf{a}|\mathbf{s}) = \frac{\hat{\pi}(\mathbf{a}|\mathbf{s})P(\mathbf{s})}{\hat{P}(\mathbf{s})}$

Substitute it into the KL divergence terms in the objective function.$D_{KL}[\pi||P^*]$, $D_{KL}[\pi||\hat{P}]$, we have:

$$\mathbb{E}_{\mathcal{D}}[D_{KL}[\pi||P^*]] = \mathbb{E}_{\mathcal{D}}\left[\pi(\mathbf{a}|\mathbf{s}) \cdot \log \frac{\pi(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\right] = \mathbb{E}_{\mathcal{D}}\left[D_{KL}[\pi||\pi^*]\right] + C_1, \tag{21}$$

$$\mathbb{E}_{\mathcal{D}}[D_{KL}[\pi||\hat{P}]] = \mathbb{E}_{\mathcal{D}}\left[\pi(\mathbf{a}|\mathbf{s}) \cdot \log \frac{\pi(\mathbf{a}|\mathbf{s})}{\hat{P}(\mathbf{a}|\mathbf{s})}\right] = \mathbb{E}_{\mathcal{D}}\left[D_{KL}[\pi||\hat{\pi}]\right] + C_2, \tag{22}$$

Here, $C_1$ and $C_2$ are constants related to the marginal distribution of the state $P(s)$, $\hat{P}(s)$ and $P^*(s)$, and they do not change with the policy $\pi$

Then, we bring Equation 21 and Equation 22 to Equation 6. Then we have:

$$\mathbb{E}_{\mathcal{D}}\left[D_{KL}[\pi||\pi^*]\right] + C_1 - \left(\mathbb{E}_{\mathcal{D}}\left[D_{KL}[\pi||\hat{\pi}]\right] + C_2\right) \leq \epsilon. \tag{23}$$

**Case 1** Meanwhile, we can observe from Equation F.1 that it's a weighted BC objective, and we assume this objective can well estimate the offline dataset *i.e.*, $\mathbb{E}_{\hat{\mathcal{D}}}[D_{KL}[\pi||\hat{\pi}]] \to 0$, therefore $\mathbb{E}_{\mathcal{D}}[D_{KL}[\pi||\hat{\pi}]] = \mathbb{E}_{\hat{\mathcal{D}}\cup\mathcal{D}^*}[D_{KL}[\pi||\hat{\pi}]] \approx \mathbb{E}_{\hat{\mathcal{D}}}[D_{KL}[\pi||\hat{\pi}]]$.

**Case 2** Similar to **Case 1**, we can also obtain: $\mathbb{E}_{\mathcal{D}^*}[D_{KL}[\pi||\hat{\pi}]] \approx \mathbb{E}_{\mathcal{D}^*}[D_{KL}[\pi^*||\hat{\pi}]]]$.

Assign Equation 23, we have:

$$\mathbb{E}_{\mathcal{D}}\left[D_{KL}[\pi||\pi^*]\right] - \mathbb{E}_{\mathcal{D}}\left[D_{KL}[\pi||\hat{\pi}]\right] \leq \epsilon + C_2 - C_1 \tag{24}$$

$$\mathbb{E}_{\mathcal{D}}\left[D_{KL}[\pi||\pi^*]\right] \leq \mathbb{E}_{\mathcal{D}}\left[D_{KL}[\pi||\hat{\pi}]\right] + \Delta C + \epsilon \tag{25}$$

$$(\textbf{Case 1})\ \mathbb{E}_{\mathcal{D}}\left[D_{KL}[\pi||\pi^*]\right] \leq \mathbb{E}_{\mathcal{D}^*}\left[D_{KL}[\pi||\hat{\pi}]\right] + \Delta C + \epsilon \tag{26}$$

$$(\textbf{Case 2})\ \mathbb{E}_{\mathcal{D}}\left[D_{KL}[\pi||\pi^*]\right] \leq \mathbb{E}_{\mathcal{D}^*}\left[D_{KL}[\pi^*||\hat{\pi}]\right] + \Delta C + \epsilon \tag{27}$$

$$(\textbf{Theorem } F.3)\ \mathbb{E}_{\mathcal{D}}\left[D_{KL}[\pi||\pi^*]\right] \leq \frac{M}{2n} \cdot \sqrt{\log \frac{2}{\delta}} + \Delta C + \epsilon, \tag{28}$$

where, $\Delta C = C_1 - C_2$ is a constant term, dependent on the state distribution. $\delta$ originates from Assumption F.2, $n = |\mathcal{D}^*|$, $M := \arg\max_{X_i}\{X_i = \pi^*(\mathbf{a}_t|\mathbf{s}_t) \log \frac{\pi^*(\mathbf{a}_t|\mathbf{s}_t)}{\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t)} | (\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}^*\}$.

**Lemma F.5.** *Given the state distribution of empirical and expert policy $d(\mathbf{s})$, $d^{\pi^*}(\mathbf{s})$. Meanwhile, given the state-action distribution of empirical and expert policy $d^{\pi}(\mathbf{s}, \mathbf{a})$, $d^{\pi^*}(\mathbf{s}, \mathbf{a})$ we have:*

$$D_{KL}[d^{\pi}(\mathbf{s})||d^{\pi^*}(\mathbf{s})] \leq D_{KL}[d^{\pi}(\mathbf{s}, \mathbf{a})||d^{\pi^*}(\mathbf{s}, \mathbf{a})]. \tag{29}$$

**Lemma F.6.** *Given the distribution of empirical and expert transitions $d^{\pi}(\mathbf{s}, \mathbf{a}, \mathbf{s}')$, $d^{\pi^*}(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ we have following relationship:*

$$D_{KL}[d^{\pi}(\mathbf{s}, \mathbf{a}, \mathbf{s}')||d^{\pi^*}(\mathbf{s}, \mathbf{a}, \mathbf{s}')] = D_{KL}[d^{\pi}(\mathbf{s}, \mathbf{a})||d^{\pi^*}(\mathbf{s}, \mathbf{a})]. \tag{30}$$

**Proof**. of Lemma F.5 and Lemma F.6 see Lemma 1 and Lemma 2 from Ma et al.

**Assumption F.7.** *Suppose the policy extracted from Equation is $\pi$, we separately define the state marginal of the dataset, empirical policy, and expert policy as $d^{\mathcal{D}}$, $d^{\pi}$ and $d^{\pi^*}$, they satisfy this relationship:*

$$D_{KL}[d^{\pi}||d^{\pi^*}] \leq D_{KL}[d^{\mathcal{D}}||d^{\pi^*}]. \tag{31}$$

**Lemma F.8** (lemma 2 from Cen et al. (2024))**.** *Suppose the maximum reward is $R_{max} = \max ||r(\mathbf{s}, \mathbf{a})||$, and $V(\rho_0) = \mathbb{E}_{\mathbf{s}_0}[V(\mathbf{s}_0)]$ denote the performance given a policy $\pi$, then with Assumption F.7:*

$$|V^{\pi}(\rho_0) - V^{\pi^*}(\rho_0)| \leq \frac{R_{max}}{1-\gamma} D_{TV}[d^*(\mathbf{s})||d^{\mathcal{D}}(\mathbf{s})] + \frac{2 \cdot R_{max}}{1-\gamma} E_{d^{\mathcal{D}}}[D_{TV}[\pi(\cdot|\mathbf{s})||\pi^*(\cdot||\mathbf{s})]]. \tag{32}$$

Proof of Lemma F.8 see Lemma 2 from Cen et al.

**Proposition F.9.** *(Value Bound of ADR) Given the empirical policy $\pi$ and the optimal policy $\pi^*$, let $V^{\pi}(\rho_0)$ and $V^{\pi^*}(\rho_0)$ separately denote the value network of $\pi$ and $\pi^*$, and given the discount factor $\gamma$. Meanwhile, let $R_{max}$ as the upper bound of the reward function i.e., $R_{max} = \max ||r(\mathbf{s}, \mathbf{a})||$. Based on the Assumption F.7, Assumption F.2, Lemma F.8, and Proposition 5.2, we can obtain:*

$$|V^{\pi}(\rho_0) - V^{\pi^*}(\rho_0)| \leq \frac{R_{max}}{1-\gamma} D_{TV}[d^*(\mathbf{s})||d^{\mathcal{D}}(\mathbf{s})] + \frac{2 \cdot R_{max}}{1-\gamma} \cdot \sqrt{2 \cdot (\frac{M}{2n} \cdot \sqrt{\log \frac{2}{\delta}} + \Delta C + \epsilon)}, \tag{33}$$

*Where, $\Delta C = C_1 - C_2$ is a constant term, typically dependent on the state distribution. The $\delta$ originates from Assumption F.2, $n = |\mathcal{D}^*|$, $M := \arg\max_{X_i}\{X_i = \pi^*(\mathbf{a}_t|\mathbf{s}_t) \log \frac{\pi^*(\mathbf{a}_t|\mathbf{s}_t)}{\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t)} | (\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}^*\}$.*

**Proof**.

In Proposition 5.2, we have proved that if $\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\big[\pi_\theta(\mathbf{a}|\mathbf{s}) \cdot \log \frac{\hat{P}(\mathbf{a}|\mathbf{s})}{P^*(\mathbf{a}|\mathbf{s})}\big]$ can finally converge to $\epsilon$. Then $\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\hat{\mathcal{D}}}[D_{KL}(\pi||\pi^*)] \to \frac{M}{2n} \cdot \sqrt{\log \frac{2}{\delta}} + \Delta C + \epsilon$

Subsequently, based on Lemma F.8, we derivative:

$$\begin{aligned}
|V^{\pi}(\rho_0) - V^{\pi^*}(\rho_0)| \leq & \frac{R_{max}}{1-\gamma} D_{TV}[d^*(\mathbf{s})||d^{\mathcal{D}}(\mathbf{s})] + \frac{2 \cdot R_{max}}{1-\gamma} E_{d^{\mathcal{D}}}[D_{TV}[\pi(\cdot|\mathbf{s})||\pi^*(\cdot||\mathbf{s})]] && (34)\\
\leq & \frac{R_{max}}{1-\gamma} D_{TV}[d^*(\mathbf{s})||d^{\mathcal{D}}(\mathbf{s})] + \frac{2 \cdot R_{max}}{1-\gamma} E_{d^{\mathcal{D}}}[\sqrt{2 \cdot D_{KL}[\pi(\cdot|\mathbf{s})||\pi^*(\cdot||\mathbf{s})]}] && (35)\\
= & \frac{R_{max}}{1-\gamma} D_{TV}[d^*(\mathbf{s})||d^{\mathcal{D}}(\mathbf{s})] + \frac{2 \cdot R_{max}}{1-\gamma} \cdot \sqrt{2 \cdot (\frac{M}{2n} \cdot \sqrt{\log \frac{2}{\delta}} + \Delta C + \epsilon)}. && (36)
\end{aligned}$$

**Proof** of Equation 9.

First, we define: $X = \log \frac{P^*}{\hat{P}}$, $Y = ||\pi_\theta - \mathbf{a}||^2$. And,we try to prove Equation 9.

- case1: If X is independent with Y, and then $\mathbb{E}[XY] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$. Furthermore, Equation 9 is held.
- case2: If X is not independent with Y, *i.e.* $Cov(X, Y) < 0$, we can apply the in-equability of covariance, *i.e.*: $Cov(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$. Since X and Y are two different distributions. Therefore, $Cov(X, Y) \leq 0$, and then Equation 9 is held.

**Proof** of the multiplicative terms' convergency.

- Regard $M := \arg\max_{X_i}\{X_i = \pi^*(\mathbf{a}_t|\mathbf{s}_t) \log \frac{\pi^*(\mathbf{a}_t|\mathbf{s}_t)}{\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t)} | (\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}^*\}$. Obviously, $M \to 0$ when $D_{KL}(\hat{\pi}||\pi^*) \to 0$.

- In terms of $n$, it is the capacity of $|\mathcal{D}^*|$, and therefore, $n$ is a fixed integer.
- Regard $C_1 - C_2$, it has $C_1 - C_2 = \pi(\mathbf{a}|\mathbf{s}) \cdot \log \frac{P^*}{\hat{P}} \leq \log \frac{P^*}{\hat{P}}$. And $\log \frac{P^*}{\hat{P}} \to 0$ when $D_{KL}(\hat{\pi}||\pi^*) \to 0$. Therefore, the conclusion of Proposition 5.3 is related to dataset quality. Specifically, as $\pi^*$ becomes closer to $\hat{\pi}$, the multiplicative terms will approach zero.

## G Supplement Experimental Results

**Experimental Results from Every seeds.** We present the experimental results for each run, which can be accessed through the logging data in supply materials we have provided in table 7. Meanwhile, we provide the comparision between ADR and diffusions in table 8

Table 7: Experimental results from All seeds. Includes 5 demonstrations for learning from demonstration (Lfd) on the Gym-mujoco domain, and 1 demonstration for Lfd on the Kitchen and Adroit domain. Our seeds are 0, 2, 4, 6. The training data is included in the appendix, and the value of each seed is obtained by returning the maximum value.

| **Tasks** | **Seed 1** | **Seed 2** | **Seed 3** | **Seed 4** | **Avg.** |
|---|---|---|---|---|---|
| `hopper-me` | 108.73135306 | 112.36561301 | 104.13708473 | 111.21583144 | 109.1± 3.2 |
| `halfcheetah-me` | 76.91686914 | 73.34520366 | 71.3600813 | 75.65439524 | 74.3± 2.1 |
| `walker2d-me` | 110.01480035 | 110.15162557 | 110.41349757 | 109.86814345 | 110.1± 0.2 |
| `Ant-me` | 132.47422373 | 132.43903581 | 132.87375784 | 133.18474616 | 132.7± 0.3 |
| `hopper-m` | 67.43902685 | 68.53755386 | 69.49494087 | 70.39486176 | 69.0± 1.1 |
| `halfcheetah-m` | 44.26977365 | 43.96688663 | 43.96063228 | 44.002488 | 44.0± 0.1 |
| `walker2d-m` | 89.01287452 | 84.82661744 | 84.96199657 | 86.20352661 | 86.3± 1.7 |
| `Ant-m` | 107.18757783 | 105.82195401 | 106.37078241 | 106.89800012 | 106.6± 0.5 |
| `hopper-mr` | 76.28604245 | 75.62349403 | 75.23570126 | 71.8023475 | 74.7± 1.7 |
| `halfcheetah-mr` | 39.04827579 | 39.08606318 | 39.24549748 | 39.34331542 | 39.2± 0.1 |
| `walker2d-mr` | 69.91171614 | 60.40786853 | 72.87922707 | 65.9015982 | 67.3± 4.7 |
| `Ant-mr` | 95.29014082 | 97.260068 | 94.74996758 | 94.31474188 | 95.4± 1.1 |
| `door-cloned` | 3.3699566 | 4.83888018 | 4.5226364 | 6.33812655 | 4.8± 1.1 |
| `door-human` | 9.35201591 | 13.05773712 | 9.10674378 | 18.71432687 | 12.6± 3.9 |
| `hammer-cloned` | 12.26944958 | 19.06662599 | 18.08395955 | 21.09296431 | 17.6± 3.3 |
| `hammer-human` | 9.37490127 | 13.78847087 | 40.01083644 | 23.73657046 | 21.7± 11.8 |
| `pen-cloned` | 110.88785576 | 92.09658 | 75.64396931 | 59.05532153 | 84.4± 19.2 |
| `pen-human` | 118.47072952 | 136.50561455 | 107.8325132 | 119.68575723 | 120.6± 10.3 |
| `relocate-cloned` | -0.19486202 | -0.18540353 | -0.25482428 | -0.23930115 | -0.2± 0.0 |
| `relocate-human` | 0.92621742 | 3.62704217 | 3.07594114 | 0.2939339 | 2.0± 1.4 |
| `kitchen-mixed` | 87.5 | 90.0 | 87.5 | 85.0 | 87.5± 1.8 |
| `kitchen-partial` | 80.0 | 77.5 | 85.0 | 80.0 | 80.6± 2.7 |
| `kitchen-completed` | 95.0 | - | - | - | 95.0 |

Table 8: ADR and Diffusion policy on Androit tasks.

| **Tasks** | **DD** | **LD** | **ADR** |
|---|---|---|---|
| `door-cloned` | 9.0±1.6 | 12.0±1.6 | **4.8±1.1** |
| `door-human` | 6.9±1.2 | 9.8±1.0 | **12.6±3.9** |
| `hammer-cloned` | 0.9±0.1 | 4.2±0.1 | **17.6±3.3** |
| `hammer-human` | 1.0±0.1 | 4.6±0.1 | **21.7±11.8** |
| `pen-cloned` | 47.7±9.2 | 60.7±9.1 | **84.4±19.2** |
| `pen-human` | 64.1±9.0 | 79.0±8.1 | **120.6±10.3** |
| `relocate-cloned` | -0.2±0.0 | -0.1±0.0 | **-0.2±0.0** |
| `relocate-human` | 0.2±0.1 | 0.2±0.1 | **2.0±1.4** |
| *Total (Androit)* | 129.6 | 170.4 | **263.5** |

**Training stability of ADR.** Despite behavior cloning not being theoretically monotonic, we still present the training curve of ADR. As shown in Figure 9 and Figure 10, we averaged multiple runs and plotted the training curve, demonstrating that ADR exhibits stable training performance.

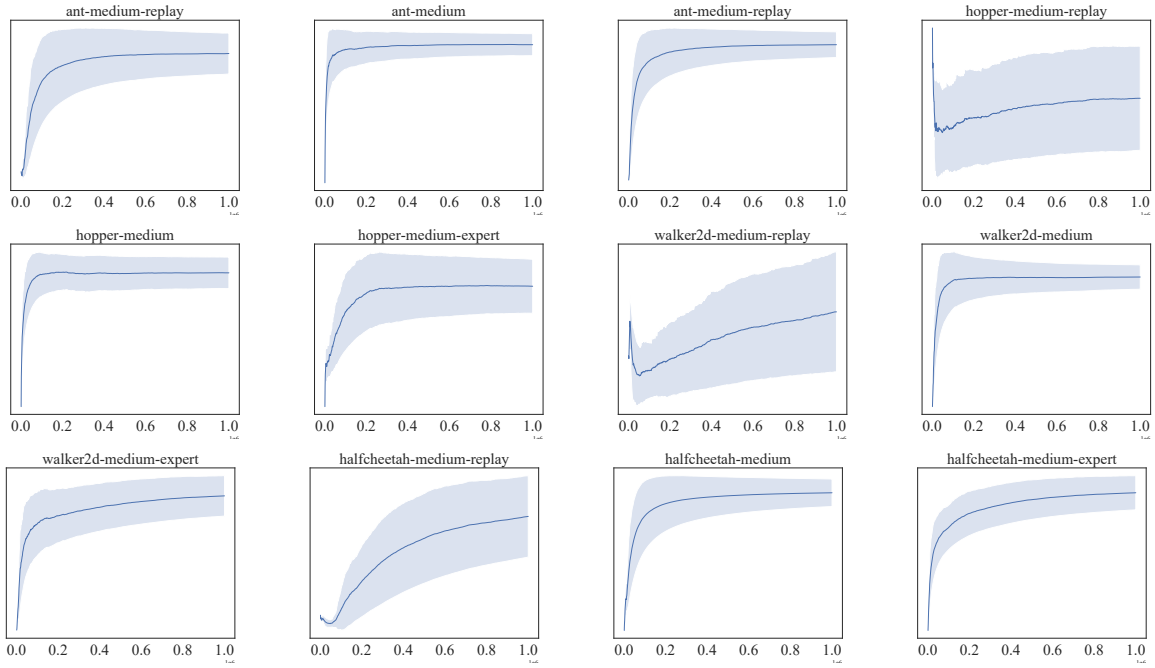


Figure 9: Training curves of ADR on all tasks sourced from Gym-Mujoco domain.

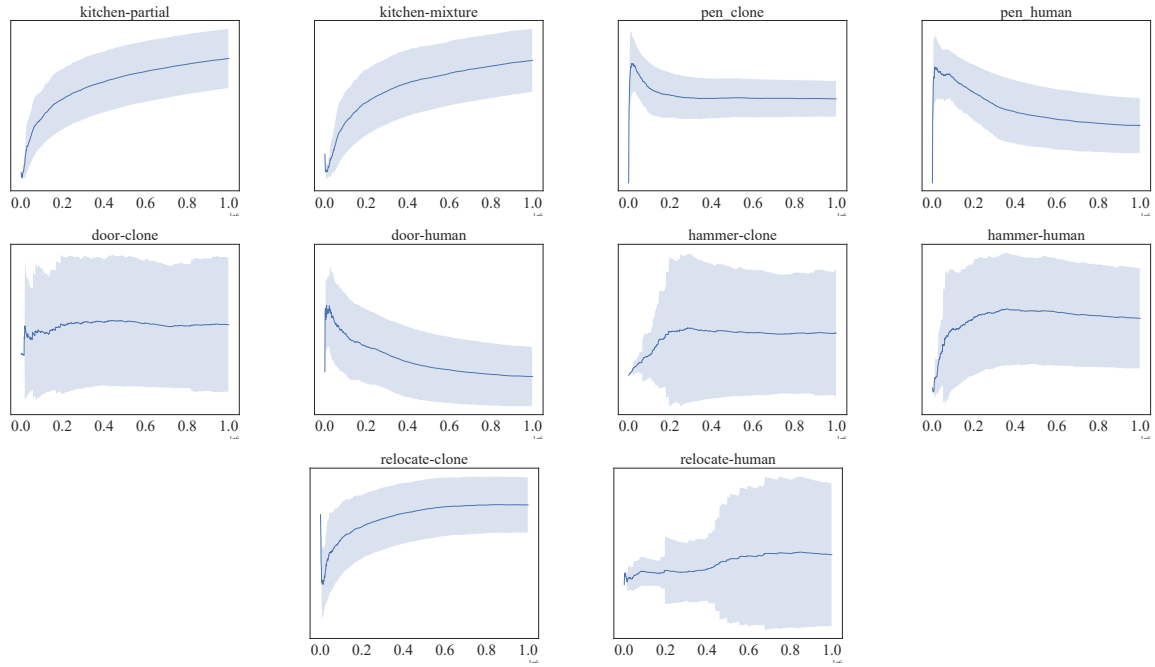


Figure 10: Training curves of ADR on tasks sourced from kitchen and Adroit domain.

**OOD Risky Analysis.** We further elaborate on the process of collecting experimental results related to Figure 11. Firstly, we need to train policys on chosen datasets. Specifically, our ADR is trained on five expert trajectories as demonstrations $\mathcal{D}^*$ and the complete medium dataset $\hat{\mathcal{D}}$, which serves as the unknown-quality dataset mentioned in the paper, while retaining the best-performing model. Additionally, when training IQL and CQL, we mix the demonstrations $\mathcal{D}^* \cup \hat{\mathcal{D}}$ with the unknown-quality dataset and use both IQL and CQL algorithms for training. After obtaining the models, we collect the logits from different models using the following specific method: we sample the states $\{\mathbf{s}_{-20}, \mathbf{s}_{-19}, \cdots, \mathbf{s}_{-1}\} \sim \pi^*$ of the last 20 steps from a trajectory in the expert dataset and use them as inputs for ADR, IQL, and CQL. Simultaneously, we retain the actions $\{\mathbf{a}_{-20}, \mathbf{a}_{-19}, \cdots, \mathbf{a}_{-1}\} \sim \pi^*$ corresponding to these states to create heatmaps.

We collect action prediction by inputting the sampled states into three models obtained by train (ADR, IQL and CQL) respectively. And after obtaining the actions, we reduce them to one dimension using PCA. Subsequently, we stack the collected actions together with the actions from the same time steps in the sampled expert dataset, calculate the covariance matrix, and then plot a heatmap to obtain Figure 11. Specifically, since the format of the dataset is `[model prediction, demo]` , only the top-left and bottom-right quarters of the heatmap have higher correlation values, which are higher than the correlations in the remaining positions of the heatmap. For convenience, we name each heatmap plot as "Algorithm-Demo". From the plots, we can observe that ADR learns relatively good patterns on both the hopper and walker2d tasks, while CQL and IQL can only learn specific patterns respectively.

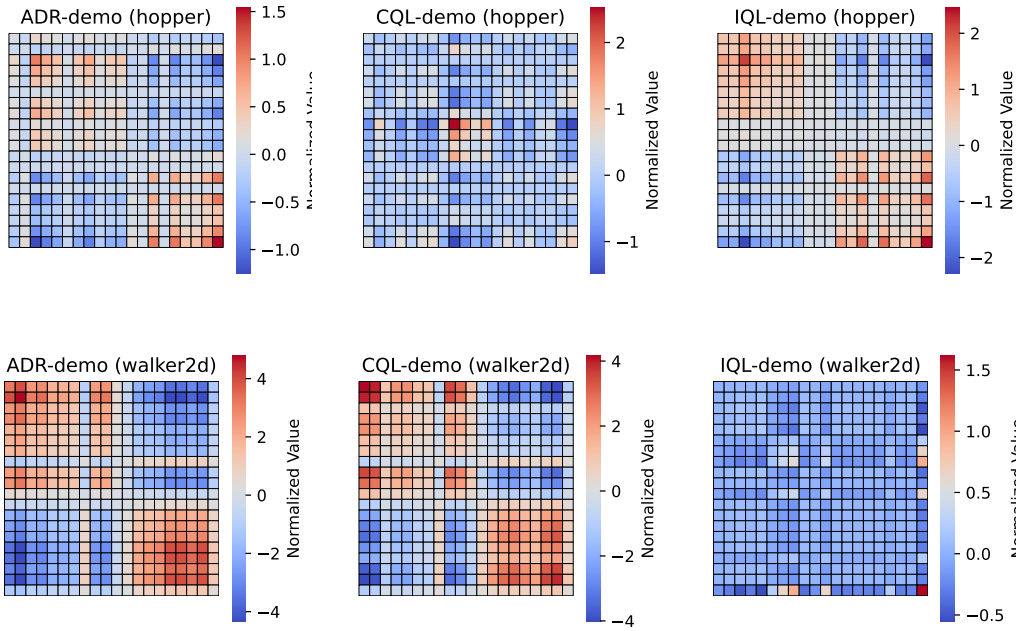


Figure 11: Heatmap of policy distributions. Higher values along the diagonal indicate a better fit of the policy to the expert policy, while lower values outside the diagonal indicate lower OOD risk for the policy.