A Appendix

A.1 Additional Discussion on Imbalanced Datasets

We consider imbalanced datasets that mixes the trajectories logged by high- and low- performing behavior policies, where trajectories from low-performing behavior policies predominate the dataset. When dealing with such imbalanced datasets, the regularized offline RL objective (Equation 2) tends to constrain the learned policy π to stay close to the overall mediocre behavior policy rather than the high-performing one, which is an unnecessary form of conservativeness. We illustrate this issue from intuitive and analytical aspects in the following.

Intuitively, for example, when considering a dataset $\mathcal{D}$ that mix the trajectories from two behavior policies $\pi^H_\mathcal{D}$ and $\pi^L_\mathcal{D}$ where $\pi^H_\mathcal{D}$ significantly outperforms $\pi^L_\mathcal{D}$ (i.e., $J(\pi^H_\mathcal{D}) \geq J(\pi^L_\mathcal{D})$), if the dataset is imbalanced such that state-action pairs (s_t, a_t) logged by $\pi^L_\mathcal{D}$ predominate, the regularized offline RL objective will penalize deviation from $\pi^L_\mathcal{D}$ more than deviation from $\pi^H_\mathcal{D}$. This is because uniform sampling on the imbalanced dataset oversamples state-action pairs (s_t, a_t) from $\pi^L_\mathcal{D}$, leading to an over-weighting of the regularization term on those state-action pairs (s_t, a_t).

Analytically, the issue of unnecessary conservativeness (Section 3) can be explained by how the performance of offline RL $J(\pi)$ is dependent on the performance of the behavior policy $J(\pi_\mathcal{D})$, as suggested in [13]. In the case of an imbalanced dataset where the behavior policy can be regarded a mixture of $\pi^H_\mathcal{D}$ and $\pi^L_\mathcal{D}$, the performance of the behavior policy $J(\pi_\mathcal{D})$ can be estimated by the expected return of the distribution of trajectory in the dataset $\mathcal{D}$, as shown below:

$$J(\pi_\mathcal{D}) \approx \mathbb{E}_{\tau_i \sim \mathcal{D}}\left[G(\tau_i)\right] = \sum_{i=1}^{N} \mathcal{D}(\tau_i) G(\tau_i), \tag{16}$$

where $\mathcal{D}(\tau_i)$ denotes the probability mass of trajectory τ_i in $\mathcal{D}$, $G(\tau_i) := \sum_{t=0}^{T_i - 1} r(s^i_t, a^i_t)$ is the return (i.e., sum of rewards) of trajectory τ_i, and T_i is its length. Since the trajectories from $\pi^L_\mathcal{D}$ dominate the dataset, the average return is mostly determined by low-return trajectories, leading to low $J(\pi_\mathcal{D})$. This prevents pessimistic and conservative algorithms [22, 8, 19, 21, 26] from achieving high $J(\pi)$ with low $J(\pi_\mathcal{D})$. Both types of algorithms aim to learn a policy π that outperforms $\pi_\mathcal{D}$ (i.e., $J(\pi) \geq J(\pi_\mathcal{D})$), while constraining the policy π to stay close to $\pi_\mathcal{D}$ [4]. Recent theoretical analysis by Singh et al. [39] on conservative Q-learning (CQL) [22] (an offline RL algorithm) shows that the performance improvements of the learned policy π over the behavior policy is upper-bounded by the deviation of π from $\pi_\mathcal{D}$ as shown in: $J(\pi) - J(\pi_\mathcal{D}) \leq C\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\mathcal{C}(s,a)\right]$, where C is a constant and $\mathcal{C}$ denotes the regularization penalty (see Section 2). This implies that regularized objective would constrain the policy π from deviating from $\pi_\mathcal{D}$, thus corroborating the findings in [13] despite the absence general theoretical guarantees for all offline RL algorithms.

Why our method improves performance on imbalanced datasets? Our importance reweighting method can be seen as a modification of the performance of the behavior policy, which represents the lower bound in conservative and pessimistic offline RL algorithms. By adjusting the weights of state-action pairs sampled from the dataset, we simulate the sampling of data from an alternative dataset $\mathcal{D}_w$ that would be generated by an alternative behavior policy $\pi_{\mathcal{D}_w}$. By maximizing the expected return $J(\pi_{\mathcal{D}_w})$ of this alternative behavior policy $\pi_{\mathcal{D}_w}$, we can obtain a $\pi_{\mathcal{D}_w}$ that outperforms the original behavior policy $\pi_\mathcal{D}$, i.e., $J(\pi_{\mathcal{D}_w}) \geq J(\pi_\mathcal{D})$. As mentioned earlier, most offline RL algorithms aim to achieve a policy that performs at least as well as the behavior policy that collected the dataset. By training an offline RL algorithm using a dataset $\mathcal{D}_w$ induced by reweighting with w, we obtain a higher lower bound on the policy's performance, as $J(\pi_{\mathcal{D}_w}) \geq J(\pi_\mathcal{D})$.

A.2 Implementation details

A.2.1 Offline RL algorithms

We implement our method and other baselines on top of two offline RL algorithms: Conservative Q-Learning (CQL) [22] and Implicit Q-Learning (IQL) [19]. Our baselines, advantage-filtering (AW) and percentage-filtering (PF), are weighted-sampling methods (see Section 5.1). Therefore, they do not require changing the implementation of offline RL algorithms. In the following sections, we explain the details of our importance weighting method for each algorithm.

CQL. We reweight both the actor (policy) and the critic (Q-function) in CQL as follows:

$$\max_{\pi} \mathbb{E}_{s\sim\mathcal{D},a\sim\pi(.|s)}\left[w_\phi(s)\left(Q(s,a)-\log\pi(a|s)\right)\right] \quad \text{(Actor)} \tag{17}$$

$$\min_{Q} \alpha\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[w_{\phi,\psi}(s,a)\left(\log\sum_{a'\in\mathcal{A}}\exp Q(s,a')-Q(s,a)\right)\right]+ \quad \text{(Critic)} \tag{18}$$

$$\mathbb{E}_{(s,a,s')\sim\mathcal{D},a'\sim\pi(.|s')}\left[w_{\phi,\psi}(s,a)\left(r(s,a)+\gamma Q(s',a')-Q(s,a)\right)^2\right]$$

$$\min_{\alpha} \mathbb{E}_{s\sim\mathcal{D},a\sim\pi(.|s)}\left[w_\phi(s)\left(-\log\pi(a|s)-\bar{\mathcal{H}}\right)\right] \quad \text{(Entropy coefficient [12])}, \tag{19}$$

where $\bar{\mathcal{H}}$ denotes the target entropy used in soft actor critic (SAC) [12] and w_ϕ and $w_{\phi,\psi}$ denote the weights predicted by our method (see Section 4). We follow the notations in CQL paper [22]. The implementation details of computing Equation 17 can be found in [22]. Our implementation is adpated from open-sourced implementation: `JaxCQL`[3].

IQL. We reweight state-value function (V), state-action value function (i.e,. Q-function, Q), and policy (π) in IQL as follows:

$$\min_{V} \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[w_\phi(s)L_2^\tau(Q(s,a)-V(s))\right] \tag{20}$$

$$\min_{Q} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[w_{\phi,\psi}(s,a)\left(r(s,a)+\gamma V(s')-V(s)\right)^2\right] \tag{21}$$

$$\max_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[w_{\phi,\psi}(s,a)\exp\left(\beta\left(Q(s,a)-V(s)\right)\right)\log\pi(a|s)\right], \tag{22}$$

where L_2^τ denotes the upper expectile loss [19] and β denotes the temperature parameter for IQL. Our implementation is adapted from the official implementation[4] for implicit Q-learning (IQL) [19].

We attach our implementation in the supplementary material, where CQL and IQL implementations are in `JaxCQL` and `implicit_q_learning`, respectively.

A.2.2 Density-weighting function

The following is the training objective of the importance weighting function $w_{\phi,\psi}$ and w_ϕ in our method, as described in Equation 10 (Section 4.2):

$$\max_{\phi,\psi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\underbrace{w_{\phi,\psi}(s,a)r(s,a)}_{\text{Return}}-\lambda_F\underbrace{\left(w_\phi(s')-w_{\phi,\psi}(s,a)\right)^2}_{\text{Bellman flow conservation penalty}}\right]-\lambda_K\underbrace{D_{KL}(\mathcal{D}_w||\mathcal{D})}_{\text{KL regularization}}.$$

The KL regularization term can be expressed as follows:

$$\begin{aligned} D_{KL}(\mathcal{D}_w||\mathcal{D}) &= \sum_{s,a}\mathcal{D}_w(s,a)\log\frac{\mathcal{D}_w(s,a)}{\mathcal{D}(s,a)} \\ &= \sum_{s,a}\mathcal{D}(s,a)\frac{\mathcal{D}_w(s,a)}{\mathcal{D}(s,a)}\log\frac{\mathcal{D}_w(s,a)}{\mathcal{D}(s,a)} \\ &= \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\frac{\mathcal{D}_w(s,a)}{\mathcal{D}(s,a)}\log\frac{\mathcal{D}_w(s,a)}{\mathcal{D}(s,a)}\right] \\ &= \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[w(s,a)\log w(s,a)\right]. \end{aligned} \tag{23}$$

Thus, we can rewrite the training objective (Equation 10) to the follows:

$$\max_{\phi,\psi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\underbrace{w_{\phi,\psi}(s,a)r(s,a)}_{\text{Return}}-\lambda_F\underbrace{\left(w_\phi(s')-w_{\phi,\psi}(s,a)\right)^2}_{\text{Bellman flow conservation penalty}}-\underbrace{\lambda_K w_{\phi,\psi}(s,a)\log w_{\phi,\psi}(s,a)}_{\text{KL regularization}}\right]. \tag{24}$$

[3] https://github.com/young-geng/JaxCQL

[4] https://github.com/ikostrikov/implicit_q_learning

The objective in Equation 24 can be optimized by minimizing the loss function $L(\phi, \psi)$ using stochastic gradient descent shown below:

$$L(\phi, \psi) := L_R(\phi, \psi) + \lambda_F L_F(\phi, \psi) + \lambda_K L_K(\phi, \psi) \tag{25}$$

$$L_R(\phi, \psi) := -\sum_{i=1}^{B} \bar{w}_{\phi,\psi}(s_i, a_i) r(s_i, a_i) \tag{26}$$

$$L_F(\phi, \psi) := \frac{1}{B} \sum_{i=1}^{B} \left(w_{\phi,\psi}(s'_i) - w_{\phi,\psi}(s_i, a_i)\right)^2. \tag{27}$$

$$L_K(\phi, \psi) := \sum_{i=1}^{B} \bar{w}_{\phi,\psi}(s, a) \log \bar{w}_{\phi,\psi}(s, a), \tag{28}$$

where B denote batch size and (s_i, a_i) denotes the i^{th} state-action pair in the batch. $\bar{w}_{\phi,\psi}(s_i, a_i)$ denotes the batch normalized weights predictions [32] that is defined as follows:

$$\bar{w}_{\phi,\psi}(s_i, a_i) := \frac{w_{\phi,\psi}(s_i, a_i)}{\sum_{j=1}^{B} w_{\phi,\psi}(s_j, a_j)}, \tag{29}$$

where $w_{\phi,\psi}(s_i, a_i)$ denotes the importance weights predictions from the network ϕ and ψ (see Section 4.2). As [32] suggests, applying batch-level normalization on the importance weights predictions can make the normalization requirement of importance weights, i.e., $\mathbb{E}[w] = 1$, more likely being satisfied during optimization than other approaches (e.g., adding normalization penalty term) (see [32] for details).

A.3 Evaluation detail

A.3.1 Training details

For both the baselines and our method, we train CQL and IQL for a total of one million gradient steps. The hyperparameters used for CQL and IQL are set to the optimal values recommended in their publicly available implementations. As for the training of our density weighting functions ϕ, ψ, we employ the same network architecture as the Q-value function architectures in CQL and IQL, which consists of a two-layer Multilayer Perceptron (MLP) with 256 neurons and ReLU activation in each layer. To minimize the objective defined in Equation 25, we train ϕ and ψ using the Adam optimizer [17] with a learning rate of 0.0001 and a batch size of 256. In each gradient step of CQL and IQL, we train ϕ and ψ for one gradient step as well.

We conducted hyperparameter search in the datasets with diverse trajectories. For AW with CQL we searched temperatue η: 0.01 (L), 0.1 (M, the best from the original paper), 1.0 (H), 5.0 (XH). For AW with IQL, we searched temperature η: 0.01 (L), 0.2 (M) (the best from the original paper), 1.0 (H), 5.0 (XH). For PF in CQL and IQL, we searched K over 0.1, 0.2, and 0.5. The hyperparameter search results are all already presented in Figures 2a. For DW-AW, we use the temperature η used in AW-M for CQL and IQL, except for small datasets. In small datasets (Figure 2b), we use the temperature used in AW-XH for DW-AW. For DW-AW and DW-Uniform, we searched $\lambda_K \in \{0.2, 1.0\}$ and $\lambda_F \in \{0.1, 1.0, 5.0\}$. We use the best found hyperparameter by the time we started the large scale experiments: $(\lambda_K, \lambda_F) = (0.2, 0.1)$ for CQL and $(\lambda_K, \lambda_F) = (1.0, 1.0)$ for IQL. We present the hyperparameter search results of DW-Uniform in Figure 2.

A.3.2 Evaluation details

We follow the evaluation protocol used in most offline RL research [22, 8, 19]. Each offline RL algorithm and sampling method (including our DW approach) combination is trained for one million gradient steps with three random seeds in each dataset. We evaluate the policy learned by each method in the environment corresponding to the dataset for 20 episodes every 1000 gradient step. The main performance metric reported is the interquartile mean (IQM) [1] of the normalized mean return over the last 10 rounds of evaluation across multiple datasets, along with its 95% confidence interval calculated using the bootstrapping method. As suggested in [1], IQM is a robust measure of central tendency by discarding the top and bottom 25% of samples, making it less sensitive to outliers.

Compute. We ran all the experiments using workstations with two RTX 3090 GPUs, AMD Ryzen Threadripper PRO 3995WX 64-Cores CPU, and 256GB RAM.

A.3.3 Dataset curation.

Following the protocol in prior offline RL benchmarking [7, 13], we develop representative datasets for Scenarios (i) and (ii) using the locomotion tasks from the D4RL Gym suite.

Scenario (i): Datasets with trajectories starting from similar initial states. This type of datasets was proposed in [13], mixing trajectories gathered by high- and low-performing policies, as described in Section 3. Each trajectory is collected by rolling out a policy starting from similar initial states until reaching timelimit or terminal states. As suggested in [13], these datasets are generated by combining $1 - \sigma\%$ of trajectories from the `random-v2` dataset (low-performing) and $\sigma\%$ of trajectories from the `medium-v2` or `expert-v2` dataset (high-performing) for each locomotion environment in the D4RL benchmark. For instance, a dataset that combines $1 - \sigma\%$ of random and $\sigma\%$ of medium trajectories is denoted as `random-medium-`$\sigma\%$. We evaluate our method and the baselines on these imbalanced datasets across four $\sigma \in \{1, 5, 10, 50\}$, four environments. We construct 32 of this type of datasets in all the combinations of {`ant`, `halfcheetah`, `hopper`, `walker2d`} $\times$ {`random-medium`, `random-expert`} $\times$ $\{\sigma \in \{1, 5, 10, 50\}\}$. Also, we consider a variant of smaller versions of these datasets that have small number of trajectories, where each dataset contains $50,000$ state-action pairs, which is 20 times smaller. These smaller datasets can test if a method overfits to small amounts of data from high-performing policies. As every method fails to suprpass random policy when $\sigma = 1$ and $\sigma = 5$, we only evaluate all the methods in $\sigma = 10$. Note that the results in $\sigma = 50$ is not different from the results in larger version of mixed dataset with $\sigma = 50$ since the amount of high-performing trajectories is still sufficient when $\sigma = 50$.

Scenario (ii): Datasets with trajectories starting from diverse initial states. Trajectories in this type of dataset start from a wider range of initial states and have varying lengths. This characteristic is designed to simulate scenarios where trajectories from different parts of a task are available in the dataset. One real-world example of this type of dataset is a collection of driving behaviors obtained from a fleet of self-driving cars. The dataset might encompass partial trajectories capturing diverse driving behaviors, such as merging onto a highway, changing lanes, navigating through intersections, and parking, although not every trajectory accomplishes the desired driving task of going from one specific location to the other. As not all kinds of driving behaviors occur with equal frequency, such a dataset is likely to be imbalanced, with certain driving behaviors being underrepresented. We curate this type of datasets by adapting the datasets from Scenario (i). These datasets are created by combining $1 - \sigma\%$ of trajectory segments from the `random-v2` dataset (representing low-performing policies) and $\sigma\%$ of trajectory segments from either the `medium-v2` or `expert-v2` dataset (representing high-performing policies) for each locomotion environment in the D4RL benchmark. Each trajectory segment is a subsequence of a trajectory in the dataset of Scenario (i). Specifically, for each trajectory τ_i, a trajectory segment is selected, ranging in length from 10 to 50 timesteps within τ_i. We chose the minimum and maximum lengths to be 10 and 50 timesteps, respectively, based on our observation that most locomotion behaviors exhibit intervals lasting between 10 and 50 timesteps. Since locomotion behaviors are periodic in nature, these datasets can simulate locomotion recordings from various initial conditions, such as different poses and velocities.

The datasets in Scenario (ii) capture a different form of imbalance that is overlooked in the datasets of Scenario (i). The imbalanced datasets in Scenario (i) combine full trajectories from low- and high-performing policies, starting from similar initial states. These datasets capture the imbalance resulting from the policies themselves, while overlooking the imbalance caused by different initial conditions. Even when using the same behavior policy, the initial conditions, i.e., the initial states of trajectories, can lead to an imbalance in the dataset, as certain initial conditions tend to yield higher returns compared to others. For example, an agent starting near the goal and another starting far away from the goal may achieve significantly different returns, even when following the same policy. This type of imbalance can exist in real-world datasets if certain initial conditions are oversampled during the dataset collection process.

Figure 4 presents the distribution of normalized returns for both types of datasets in both scenarios. The trajectory returns are normalized to a range of 0 to 1 using max-min normalization, specifically

$(G(\tau_i) - G_{\text{min}})/(G_{\text{max}} - G_{\text{min}})$, where G_{min} and G_{max} denote the minimum and maximum trajectory returns in the dataset, respectively. From Figure 4, we observe that the two types of datasets exhibit different return distributions. The return distribution in Scenario (i) (i.e., mixed) is closer to a bimodal distribution, where a trajectory is either from a high- or low-performing policy. On the other hand, the return distribution in Scenario (ii) (i.e., mixed (diverse)) follows a heavy-tailed distribution, where high-return trajectories are located in the long tails. This indicates that in addition to the imbalance resulting from the combination of trajectories from low- and high-performing policies, the presence of diverse initial states introduces another type of imbalance in the dataset, where initial states that easily lead to high returns are much less prevalent than others.

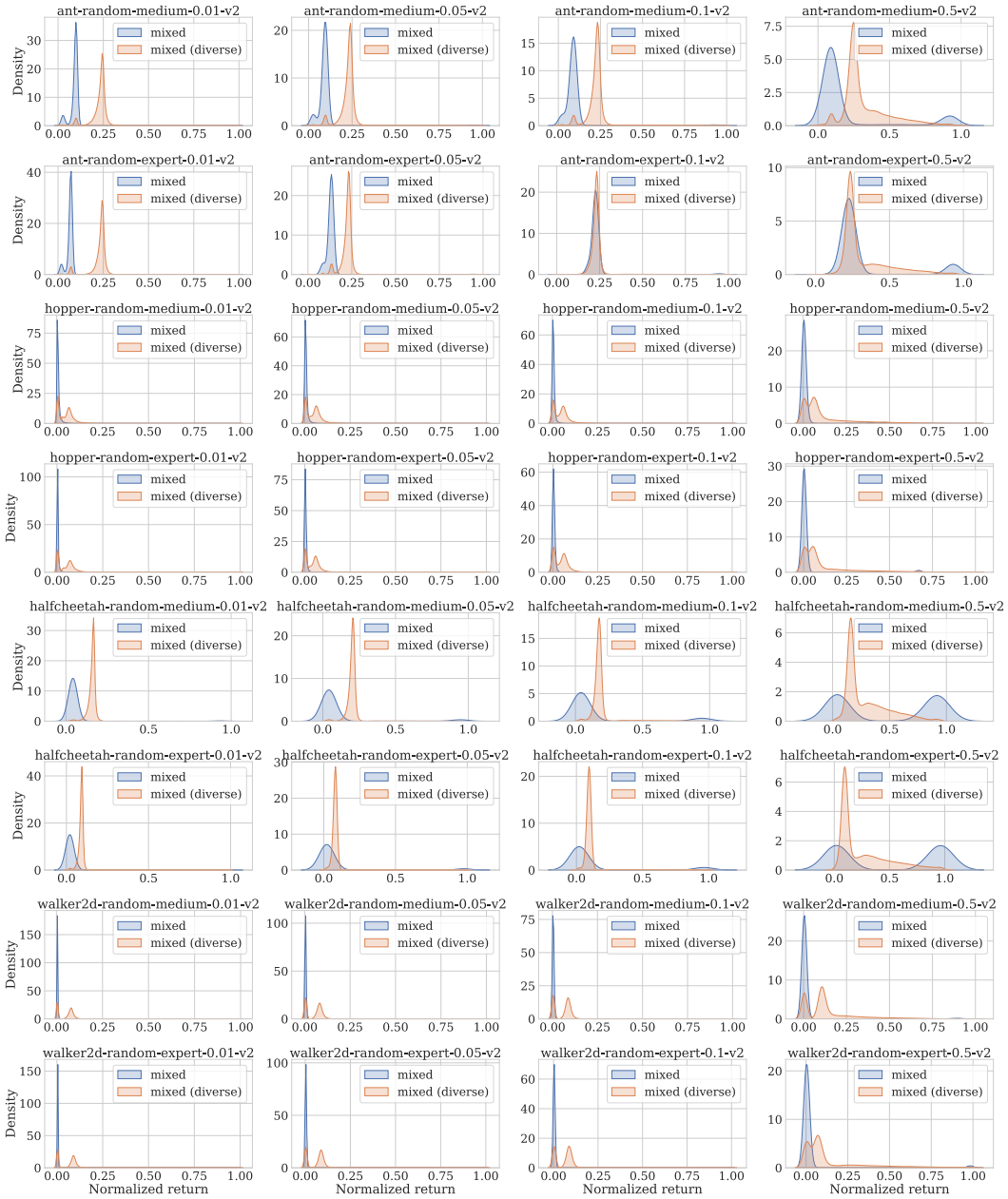

Figure 4: Return distributions of mixed datasets with similar initial states (blue, Scenario (i)) and diverse initial states (orange, Scenario (ii)). Both types of datasets lead to different kinds of imbalance and return distributions. See Section A.3.3 for details.

A.4 Additional results

We present the full results in total of 113 datasets in Tables 4 and 5, including original D4RL datasets [7], mixed datasets (Figure 2a, denoted as Mixed), mixed datasets with diverse initial states (Figure 3, denoted as Mixed (diverse)), and small datasets (Figure 2b, denoted as Mixed (small)). As [13] suggested, the original D4RL datasets are not imbalanced and thus weighted sampling and importance weighting methods perform on par with uniform sampling.

A.4.1 Comparison with OptDICE

OptDiCE [27] and AlgaeDiCE [34], as well as our method, all involve learning importance weights for policy optimization. However, the usage and learning of importance weights vary across these methods. We compare our method with OptDiCE as it is the state-of-the-art method on policy optimization using DiCE. In the following, we illustrate the difference between our method and OptDiCE.

Learning importance weights. OptDiCE learns the importance weights through solving the optimization problem same as Equation 10 in an approach different from ours. OptDiCE learns the importance weights through optimizing the following primal-dual objective:

$$\min_{\nu} \max_{w \geq 0} \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[e_\nu(s,a)w(s,a) - \alpha f\big(w(s,a)\big)\right] + (1-\gamma)\mathbb{E}_{s_0\sim p_0}[\nu(s_0)] \approx \tag{30}$$

$$\min_{\nu} \ \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\hat{e}_\nu(s,a,s')(f')^{-1}\big(\tfrac{1}{\alpha}\hat{e}_\nu(s,a,s')\big)_+ - \alpha f\big((f')^{-1}(\tfrac{1}{\alpha}\hat{e}_\nu(s,a,s'))_+\big)\right] + \tag{31}$$
$$(1-\gamma)\mathbb{E}_{s_0\sim p_0}[\nu(s_0)],$$

where f can be any convex function (e.g., $f(x) := x\log x$), $\hat{e}_\nu(s,a,s') := r(s,a) + \gamma\nu(s') - \nu(s)$, and $x_+ := \max(0,x)$. The coefficient α denotes the strength of regularization: the higher the α is, the uniform the importance weights are. The learned importance weights $w(s,a)$ are expressed in terms of ν as follows:

$$w(s,a) = \max\left(0, (f)^{-1}\left(\frac{\hat{e}_\nu(s,a,s')}{\alpha}\right)\right). \tag{32}$$

On the other hand, our method learns the importance weights without solving the min-max optimization.

Applying importance weights. OptDiCE extracts the policy from the learned importance weights using information projection (I-projection) [27]. The implementation of I-projection is non-trivial, while its idea is close to weighted behavior cloning objective shown as follows:

$$\max_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}_w}\left[\log\pi(a|s)\right] = \max_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[w(s,a)\log\pi(a|s)\right]. \tag{33}$$

Differing from OptDiCE, we apply the importance weights to actor-critic offline RL algoritmhs like CQL [22] and IQL [19].

We present the performance of our methods in CQL and IQL, and OptDiCE in imbalanced datasets used in Scenario (i) in Figure 5, showing that OptDiCE underpeform all the other methods on imbalanced datasets and even underperforms uniform sampling approach. This result indicates that even though OptDiCE learns the importance weights by solving a similar objective with our method, OptDiCE is not effective on imbalanced datasets.

A.4.2 Comparison with weights learned by OptDiCE

While the policy learned using OptDiCE may not be effective on imbalanced datasets, it remains uncertain whether the importance weights learned with OptDiCE can enhance the performance of offline RL algorithms. To investigate this hypothesis, we reweight the training objective of CQL (Equation 17) using the importance weights learned with OptDiCE (Equation 32). We present a comparison with our method in Table 1, where OptDiCEW refers to CQL trained with OptDiCE weights, and α represents the regularization strength (Equation 30). We examine different values of α ranging from 0.1 to 5.0, but none of them consistently outperforms our method. It is worth noting that each configuration results in a significant performance loss in certain datasets. While this observation suggests that the importance weights learned using OptDiCE are not effective in improving the performance of offline RL algorithms, it is important to note that our work does not

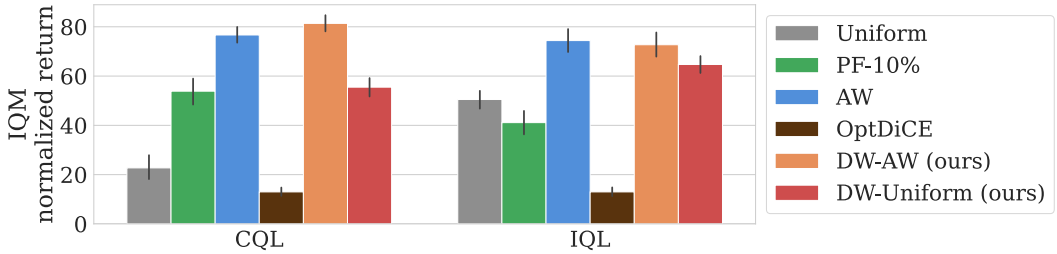

Figure 5: Results on imbalanced datasets of trajectories with similar initial states (Scenario (i) in Section 5.1). OptDiCE [27] fails to achieve high return on imbalanced datasets and even performs worse than CQL and IQL with uniform sampling. Note that OptDiCE is a density-ratio importance correction estimation (DiCE) based offline RL algorithm rather than a sampling or importance weighting method. While both bars of OptDiCE in this figure denote the same result, we plot the performance of OptDiCE alongside CQL and IQL with different sampling (or weighting) methods for comparison.

	OptDiCEW ($\alpha = 0.1$)	OptDiCEW ($\alpha = 1.0$)	OptDiCEW ($\alpha = 5.0$)
hopper-random-expert-diverse-5%-v2	3.3 (-17.5)	39.1 (+18.3)	4.0 (-16.8)
hopper-random-expert-diverse-10%-v2	1.5 (-56.4)	29.7 (-28.2)	11.9 (-46.0)
hopper-random-medium-diverse-5%-v2	-*	6.4 (-58.5)	57.3 (-7.6)
hopper-random-medium-diverse-10%-v2	2.2 (-52.9)	30.1 (-25.0)	41.0 (-14.1)

(*: training gets terminated due to NaN values)

Table 1: Comparison of the weights learned by our method and that learned by OptDiCE approach. Each cell in the table denotes the mean return obtained by CQL trained with OptDiCE weights and the number in the parenthesis indicates the improvements (i.e., $\text{score}_{\text{OptDiCE}} - \text{score}_{\text{Ours}}$) over the performance of DW-Uniform reported in Figure 3. It shows that OptDiCE underperforms our method in most datasets (i.e,. negative improvements), and is sensitive to the regularization strength parameter α (see Section A.4.2).

aim to propose a new off-policy evaluation method based on DiCE. Therefore, a comprehensive comparison between different methods for learning importance weights is left for future research endeavors.

A.4.3 Hyperparameter studies

In this section, we investigate the hyperparameters of KL regularization strength (λ_K) and Bellman flow conservation strength (λ_F). We present the average returns obtained in four specific imbalanced datasets in Table 2. These datasets are chosen because they exhibit significant performance differences between uniform sampling and other methods. Our observations indicate that higher penalties for flow conservation and lower strengths of KL regularization tend to improve performance in both CQL and IQL.

A.4.4 Comparison of re-weighting both objectives and only regularization objective

In theory, the regularization term $\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[C(s,a)\right]$ (Equation 2) is the only component that can potentially harm performance on imbalanced datasets, as it encourages the policy to imitate the suboptimal actions that dominate the dataset. However, our findings suggest that reweighting all the training objectives in offline RL algorithms leads to improved performance. In this section, we compare the performance of CQL trained with reweighted regularization only (referred to as "Reg. only") and reweighting all objectives (referred to as "All"). For the "All" approach, we train CQL using Equation 17. On the other hand, for the "Reg. only" approach, we remove the term $w_{\phi,\psi}(s,a)$ from the α training and the term $w_\phi(s)$ from the π training in Equation 17. Table 3 presents the mean return of both approaches, indicating that "All" exhibits slightly better performance compared to "Reg. only".

(λ_K, λ_F)	$(0.2, 0.1)$	$(0.2, 1.0)$	$(0.2, 5.0)$	$(1.0, 0.1)$	$(1.0, 1.0)$	$(1.0, 5.0)$
hopper-random-expert-diverse-5%-v2	20.4	13.8	22.1	9.6	7.5	7.9
hopper-random-expert-diverse-10%-v2	51.7	27.8	42.1	10.9	9.8	5.3
hopper-random-medium-diverse-5%-v2	64.7	66.1	60.3	16.2	5.5	25.4
hopper-random-medium-diverse-10%-v2	55.4	58.9	65.3	3.6	7.2	7.3

(a) CQL

(λ_K, λ_F)	$(0.2, 0.1)$	$(0.2, 1.0)$	$(0.2, 5.0)$	$(1.0, 0.1)$	$(1.0, 1.0)$	$(1.0, 5.0)$
hopper-random-expert-diverse-5%-v2	14.7	56.4	54.2	16.8	68.4	75.1
hopper-random-expert-diverse-10%-v2	16.2	67.8	51.0	78.5	63.0	67.5
hopper-random-medium-diverse-5%-v2	53.8	50.8	51.5	50.9	49.5	52.6
hopper-random-medium-diverse-10%-v2	43.4	48.8	53.0	42.5	53.2	50.8

(b) IQL

Table 2: Hyperparameter studies of our DW method in **(a)** CQL and **(b)** IQL. λ_K and λ_F denote the strength of KL-regularization and Bellman flow conservation penatly, respectively (see Section A.2). Lower KL-regularization strength λ_F and higher flow conservation penalty strength λ_F lead to better performance.

	Reg. only	All
hopper-random-expert-diverse-10%-v2	57.1	64.9
walker2d-random-expert-diverse-10%-v2	1.4	8.2

Table 3: Reweighting all terms in Equation 17 shows better performance than reweighting only the regularization term. See Section A.4.4.

		Uniform	AW	PF	DW+AW (ours)	DW+Uniform (ours)
D4RL MuJoCo	hopper-random-v2	9.0	7.7	7.5	5.8	9.5
	hopper-medium-expert-v2	97.2	103.7	105.7	104.7	86.9
	hopper-medium-replay-v2	85.5	95.8	90.4	97.2	86.0
	hopper-full-replay-v2	100.4	101.2	101.5	101.4	99.7
	hopper-medium-v2	61.8	67.5	65.0	65.2	66.1
	hopper-expert-v2	107.0	108.2	107.0	106.7	105.4
	halfcheetah-random-v2	21.0	15.6	2.7	10.9	12.0
	halfcheetah-medium-expert-v2	71.2	87.4	72.1	88.8	86.1
	halfcheetah-medium-replay-v2	45.3	44.7	42.1	44.3	45.0
	halfcheetah-full-replay-v2	75.1	76.7	75.1	78.3	77.0
	halfcheetah-medium-v2	46.5	46.5	45.4	46.6	46.5
	halfcheetah-expert-v2	81.0	87.6	60.9	20.7	44.7
	ant-random-v2	7.5	7.7	6.8	29.4	32.4
	ant-medium-expert-v2	127.4	128.7	127.5	129.8	118.1
	ant-medium-replay-v2	96.0	88.6	80.7	84.9	95.0
	ant-full-replay-v2	128.4	124.5	125.3	127.5	128.6
	ant-medium-v2	100.0	92.3	92.1	91.9	97.9
	ant-expert-v2	124.5	131.4	127.0	127.2	127.6
	walker2d-random-v2	6.0	4.5	11.1	7.6	8.6
	walker2d-medium-expert-v2	109.6	109.3	108.5	109.4	109.7
	walker2d-medium-replay-v2	74.3	78.1	69.9	77.8	74.4
	walker2d-full-replay-v2	90.9	88.5	88.1	92.4	90.8
	walker2d-medium-v2	82.1	81.3	77.8	81.7	82.1
	walker2d-expert-v2	108.8	108.5	108.8	108.8	108.1
D4RL Antmaze	antmaze-umaze-v0	73.7	77.3	56.3	76.7	71.0
	antmaze-umaze-diverse-v0	48.0	36.0	23.4	33.7	34.0
	antmaze-medium-diverse-v0	0.0	6.0	0.0	5.3	3.1
	antmaze-medium-play-v0	2.2	10.7	0.0	8.7	1.3
	antmaze-large-diverse-v0	0.0	2.0	2.7	1.3	1.7
	antmaze-large-play-v0	0.4	0.7	0.0	0.3	0.0
D4RL Kitchen	kitchen-complete-v0	24.2	30.2	6.4	9.5	16.5
	kitchen-partial-v0	38.4	36.0	39.9	54.0	30.0
	kitchen-mixed-v0	30.4	50.5	48.1	47.5	43.8
D4RL Adroit	pen-human-v1	1.6	-3.6	-0.2	-7.4	28.5
	pen-cloned-v1	-1.3	-2.5	8.6	-4.2	9.1
	hammer-human-v1	-7.0	-7.0	-6.9	-7.0	-7.0
	hammer-cloned-v1	-7.0	-7.0	-7.0	-6.9	-6.9
	door-human-v1	-9.4	-9.4	-5.1	-9.4	-9.4
	door-cloned-v1	-9.4	-9.4	21.6	-9.4	-9.4
	relocate-human-v1	1.1	-2.1	-0.8	-2.1	-2.1
	relocate-cloned-v1	-2.3	-2.4	0.2	-2.1	-2.1
Mixed	ant-random-medium-1%-v2	33.8	73.6	5.9	70.4	61.2
	ant-random-medium-5%-v2	53.1	86.1	72.0	86.1	87.0
	ant-random-medium-10%-v2	81.3	88.1	92.2	86.8	86.5
	ant-random-medium-50%-v2	97.4	93.3	92.0	94.0	96.1
	ant-random-expert-1%-v2	10.0	73.1	4.2	59.7	37.9
	ant-random-expert-5%-v2	35.2	113.8	62.9	112.2	90.7
	ant-random-expert-10%-v2	48.3	119.6	102.7	123.1	109.1
	ant-random-expert-50%-v2	117.3	130.4	122.4	131.3	113.7
	hopper-random-medium-1%-v2	0.6	55.1	58.2	56.0	60.1
	hopper-random-medium-5%-v2	1.5	61.4	42.6	60.3	59.4
	hopper-random-medium-10%-v2	1.6	56.3	66.4	67.0	59.9
	hopper-random-medium-50%-v2	22.6	37.2	65.5	41.3	54.3
	hopper-random-expert-1%-v2	17.5	59.6	14.7	61.7	21.3
	hopper-random-expert-5%-v2	13.7	99.7	31.8	104.2	52.0
	hopper-random-expert-10%-v2	14.9	99.8	36.8	107.9	51.9
	hopper-random-expert-50%-v2	100.6	108.9	105.4	104.4	83.3
	halfcheetah-random-medium-1%-v2	37.1	39.8	18.1	38.4	36.6
	halfcheetah-random-medium-5%-v2	41.1	45.4	42.3	45.3	43.9
	halfcheetah-random-medium-10%-v2	44.6	45.8	44.9	45.7	41.9
	halfcheetah-random-medium-50%-v2	46.6	46.5	44.9	46.5	46.2
	halfcheetah-random-expert-1%-v2	21.3	21.7	4.7	13.5	14.5
	halfcheetah-random-expert-5%-v2	24.8	62.2	7.4	67.5	17.6
	halfcheetah-random-expert-10%-v2	30.9	71.0	70.3	74.0	27.1
	halfcheetah-random-expert-50%-v2	58.7	80.7	55.1	83.4	37.8
	walker2d-random-medium-1%-v2	2.9	40.9	2.3	47.1	0.6
	walker2d-random-medium-5%-v2	0.0	74.2	44.6	74.2	9.9
	walker2d-random-medium-10%-v2	0.6	74.6	73.5	78.7	58.8
	walker2d-random-medium-50%-v2	76.9	81.7	81.2	80.8	78.0
	walker2d-random-expert-1%-v2	4.0	64.6	3.0	83.4	-0.1
	walker2d-random-expert-5%-v2	0.2	107.6	32.7	106.9	12.2
	walker2d-random-expert-10%-v2	3.1	108.1	17.0	108.3	0.8
	walker2d-random-expert-50%-v2	0.3	108.5	108.1	108.7	92.5
Mixed (diverse)	ant-random-medium-diverse-1%-v2	9.6	20.7	5.9	67.4	29.3
	ant-random-medium-diverse-5%-v2	53.3	84.2	39.5	82.0	72.4
	ant-random-medium-diverse-10%-v2	78.0	88.2	92.7	91.8	87.6
	ant-random-medium-diverse-50%-v2	101.9	91.3	92.5	94.6	100.5
	ant-random-expert-diverse-1%-v2	7.5	9.5	6.2	27.6	9.8
	ant-random-expert-diverse-5%-v2	12.7	29.1	7.6	96.1	54.4
	ant-random-expert-diverse-10%-v2	23.4	75.3	57.3	109.6	84.7
	ant-random-expert-diverse-50%-v2	112.5	116.1	121.3	123.7	121.1
	hopper-random-medium-diverse-1%-v2	3.1	13.8	50.7	55.7	9.5
	hopper-random-medium-diverse-5%-v2	19.0	24.2	55.5	61.9	66.4
	hopper-random-medium-diverse-10%-v2	5.8	4.3	35.9	61.1	54.2
	hopper-random-medium-diverse-50%-v2	31.8	56.0	58.5	58.2	59.6
	hopper-random-expert-diverse-1%-v2	10.2	13.7	3.0	16.3	2.3
	hopper-random-expert-diverse-5%-v2	5.2	10.0	27.6	44.4	20.2
	hopper-random-expert-diverse-10%-v2	5.4	58.4	24.5	54.5	64.9
	hopper-random-expert-diverse-50%-v2	96.1	100.4	81.0	102.6	96.7
	halfcheetah-random-medium-diverse-1%-v2	39.6	41.0	32.2	22.0	40.4
	halfcheetah-random-medium-diverse-5%-v2	44.5	44.1	44.6	34.8	44.6
	halfcheetah-random-medium-diverse-10%-v2	43.7	45.3	44.7	42.7	45.5
	halfcheetah-random-medium-diverse-50%-v2	46.8	46.8	45.7	43.4	46.4
	halfcheetah-random-expert-diverse-1%-v2	11.8	22.0	7.5	2.5	16.3
	halfcheetah-random-expert-diverse-5%-v2	29.4	35.4	10.0	6.1	24.5
	halfcheetah-random-expert-diverse-10%-v2	24.1	39.5	16.6	10.9	16.2
	halfcheetah-random-expert-diverse-50%-v2	53.3	69.1	7.1	41.4	63.9
	walker2d-random-medium-diverse-1%-v2	1.9	1.3	26.8	1.0	1.1
	walker2d-random-medium-diverse-5%-v2	0.5	4.6	37.2	38.6	1.0
	walker2d-random-medium-diverse-10%-v2	0.3	16.2	55.5	71.6	35.5
	walker2d-random-medium-diverse-50%-v2	76.8	79.5	77.4	75.0	78.4
	walker2d-random-expert-diverse-1%-v2	12.3	5.0	-0.1	5.3	3.5
	walker2d-random-expert-diverse-5%-v2	0.6	2.0	61.8	0.1	0.7
	walker2d-random-expert-diverse-10%-v2	1.7	1.6	49.9	0.3	8.2
	walker2d-random-expert-diverse-50%-v2	7.2	0.9	67.4	89.4	106.2
Mixed (small)	ant-random-medium-10%-small-v2	5.9	29.9	6.4	17.6	25.1
	ant-random-expert-10%-small-v2	5.9	34.9	5.3	10.4	12.1
	hopper-random-medium-10%-small-v2	39.7	3.4	44.6	50.7	49.7
	hopper-random-expert-10%-small-v2	20.4	6.9	17.9	47.8	43.2
	halfcheetah-random-medium-10%-small-v2	10.2	24.6	28.3	24.4	19.7
	halfcheetah-random-expert-10%-small-v2	2.1	3.5	2.2	4.3	3.5
	walker2d-random-medium-10%-small-v2	1.6	0.2	0.4	34.6	31.2
	walker2d-random-expert-10%-small-v2	13.6	-0.0	0.1	24.4	44.2

Table 4: Full results of average returns of CQL in total of 113 datasets.

		Uniform	AW	PF	DW+AW (ours)	DW+Uniform (ours)
D4RL MuJoCo	hopper-random-v2	7.6	6.8	8.0	6.4	8.5
	hopper-medium-expert-v2	85.4	111.1	111.8	110.8	81.0
	hopper-medium-replay-v2	86.7	98.1	96.0	99.9	79.7
	hopper-full-replay-v2	108.1	102.1	88.2	107.5	99.8
	hopper-medium-v2	65.7	58.3	64.5	61.7	62.5
	hopper-expert-v2	109.7	111.0	110.3	105.7	108.2
	halfcheetah-random-v2	12.7	7.3	4.2	10.7	10.8
	halfcheetah-medium-expert-v2	90.6	94.7	94.2	93.9	93.7
	halfcheetah-medium-replay-v2	44.0	44.0	29.4	44.1	44.6
	halfcheetah-full-replay-v2	73.5	76.3	72.3	76.4	75.9
	halfcheetah-medium-v2	47.5	47.8	45.4	47.9	47.7
	halfcheetah-expert-v2	94.9	95.3	73.8	95.2	95.1
	ant-random-v2	11.9	12.2	8.3	15.8	16.3
	ant-medium-expert-v2	133.3	131.9	133.2	129.3	130.1
	ant-medium-replay-v2	93.8	82.9	71.4	86.8	89.8
	ant-full-replay-v2	130.1	129.9	128.9	131.4	130.2
	ant-medium-v2	100.0	98.9	96.2	98.1	99.6
	ant-expert-v2	126.2	131.4	119.6	128.6	127.3
	walker2d-random-v2	6.7	2.7	10.4	3.7	7.0
	walker2d-medium-expert-v2	110.1	109.7	109.8	109.8	109.7
	walker2d-medium-replay-v2	61.3	47.0	42.2	62.6	65.1
	walker2d-full-replay-v2	86.8	84.5	85.6	80.6	95.0
	walker2d-medium-v2	77.9	70.0	65.3	75.8	80.8
	walker2d-expert-v2	109.9	109.9	109.6	109.5	109.4
D4RL Antmaze	antmaze-umaze-v0	88.0	90.7	0.0	89.3	81.3
	antmaze-umaze-diverse-v0	67.3	75.3	0.0	72.0	61.0
	antmaze-medium-diverse-v0	76.0	61.3	0.0	70.0	78.7
	antmaze-medium-play-v0	72.0	22.0	0.0	30.0	64.7
	antmaze-large-diverse-v0	36.7	23.3	0.0	20.7	40.0
	antmaze-large-play-v0	43.3	9.3	0.0	10.0	42.0
D4RL Kitchen	kitchen-complete-v0	62.8	26.3	10.0	19.8	60.0
	kitchen-partial-v0	47.7	73.2	72.3	66.3	57.0
	kitchen-mixed-v0	49.8	47.8	52.2	24.3	36.7
D4RL Adroit	pen-human-v1	80.4	83.2	36.3	88.3	74.9
	pen-cloned-v1	82.9	89.2	53.8	84.4	91.5
	hammer-human-v1	3.1	0.5	3.2	0.8	1.2
	hammer-cloned-v1	1.1	1.4	1.0	2.3	1.4
	door-human-v1	2.5	0.6	0.1	0.0	1.4
	door-cloned-v1	0.0	0.6	2.4	-0.0	1.5
	relocate-human-v1	0.5	0.0	-0.0	-0.0	0.1
	relocate-cloned-v1	-0.0	0.1	0.0	0.0	-0.0
Mixed	ant-random-medium-1%-v2	17.5	56.0	5.1	58.5	55.3
	ant-random-medium-5%-v2	68.1	83.3	15.4	87.6	89.3
	ant-random-medium-10%-v2	82.0	88.8	40.2	91.3	88.6
	ant-random-medium-50%-v2	93.7	101.4	96.8	94.3	98.9
	ant-random-expert-1%-v2	13.7	28.5	5.5	31.5	43.5
	ant-random-expert-5%-v2	36.3	100.9	5.5	95.2	105.4
	ant-random-expert-10%-v2	73.7	126.0	14.0	125.4	115.0
	ant-random-expert-50%-v2	122.5	128.2	127.7	130.3	125.4
	hopper-random-medium-1%-v2	52.2	56.1	42.4	56.5	51.7
	hopper-random-medium-5%-v2	59.0	57.1	63.4	46.2	59.8
	hopper-random-medium-10%-v2	63.2	57.1	65.3	63.6	62.8
	hopper-random-medium-50%-v2	50.6	56.2	57.2	61.2	61.9
	hopper-random-expert-1%-v2	11.1	74.8	16.4	64.8	22.2
	hopper-random-expert-5%-v2	22.7	111.3	24.9	110.0	22.4
	hopper-random-expert-10%-v2	46.7	111.5	33.9	110.6	64.3
	hopper-random-expert-50%-v2	88.0	111.7	92.2	109.4	105.1
	halfcheetah-random-medium-1%-v2	31.0	13.9	3.0	22.0	7.2
	halfcheetah-random-medium-5%-v2	39.1	41.7	25.9	42.2	11.6
	halfcheetah-random-medium-10%-v2	40.3	43.1	45.3	45.0	45.1
	halfcheetah-random-medium-50%-v2	45.3	47.3	43.4	47.1	46.6
	halfcheetah-random-expert-1%-v2	4.2	3.8	2.3	2.8	3.6
	halfcheetah-random-expert-5%-v2	9.1	74.0	4.4	48.7	55.4
	halfcheetah-random-expert-10%-v2	17.0	91.3	81.5	87.1	70.6
	halfcheetah-random-expert-50%-v2	83.7	94.8	32.0	94.4	93.8
	walker2d-random-medium-1%-v2	54.6	45.4	39.4	49.4	61.8
	walker2d-random-medium-5%-v2	66.2	62.8	47.3	67.8	67.9
	walker2d-random-medium-10%-v2	63.4	65.8	62.6	62.8	74.3
	walker2d-random-medium-50%-v2	70.7	70.0	69.6	75.6	74.1
	walker2d-random-expert-1%-v2	20.0	9.6	11.4	9.8	35.9
	walker2d-random-expert-5%-v2	25.3	108.6	93.5	104.3	65.0
	walker2d-random-expert-10%-v2	64.4	109.3	107.2	109.1	58.1
	walker2d-random-expert-50%-v2	109.2	109.4	109.6	109.3	109.4
Mixed (diverse)	ant-random-medium-diverse-1%-v2	12.6	29.4	11.7	40.8	24.1
	ant-random-medium-diverse-5%-v2	29.1	83.7	24.8	88.1	73.3
	ant-random-medium-diverse-10%-v2	61.0	91.8	54.6	89.3	89.3
	ant-random-medium-diverse-50%-v2	91.2	98.0	89.0	97.0	96.8
	ant-random-expert-diverse-1%-v2	12.4	20.0	10.1	30.4	20.7
	ant-random-expert-diverse-5%-v2	17.1	77.4	10.9	86.2	71.5
	ant-random-expert-diverse-10%-v2	27.9	101.1	22.9	100.5	93.8
	ant-random-expert-diverse-50%-v2	101.9	123.3	122.8	127.7	126.9
	hopper-random-medium-diverse-1%-v2	37.0	48.2	47.4	54.9	63.0
	hopper-random-medium-diverse-5%-v2	46.7	44.7	49.3	48.6	49.0
	hopper-random-medium-diverse-10%-v2	47.6	47.8	47.2	51.9	53.2
	hopper-random-medium-diverse-50%-v2	51.2	48.7	53.6	54.6	52.1
	hopper-random-expert-diverse-1%-v2	10.5	14.6	6.5	22.3	17.4
	hopper-random-expert-diverse-5%-v2	17.1	59.0	31.4	78.7	41.0
	hopper-random-expert-diverse-10%-v2	33.7	92.2	52.3	103.1	63.0
	hopper-random-expert-diverse-50%-v2	90.1	106.8	22.7	98.9	109.4
	halfcheetah-random-medium-diverse-1%-v2	14.4	2.7	8.0	2.2	14.4
	halfcheetah-random-medium-diverse-5%-v2	34.9	20.1	17.1	2.3	15.2
	halfcheetah-random-medium-diverse-10%-v2	39.7	39.7	24.0	30.9	24.3
	halfcheetah-random-medium-diverse-50%-v2	44.4	22.5	46.6	8.8	35.1
	halfcheetah-random-expert-diverse-1%-v2	5.6	4.5	5.7	3.2	4.6
	halfcheetah-random-expert-diverse-5%-v2	4.8	8.5	3.6	9.9	10.3
	halfcheetah-random-expert-diverse-10%-v2	9.6	16.4	4.3	13.2	28.0
	halfcheetah-random-expert-diverse-50%-v2	66.4	70.7	-0.9	72.9	85.0
	walker2d-random-medium-diverse-1%-v2	36.0	33.3	14.6	61.3	59.8
	walker2d-random-medium-diverse-5%-v2	67.5	61.7	64.8	67.9	55.3
	walker2d-random-medium-diverse-10%-v2	58.8	58.9	59.2	66.2	60.5
	walker2d-random-medium-diverse-50%-v2	66.8	49.4	74.6	63.7	64.0
	walker2d-random-expert-diverse-1%-v2	10.5	13.0	1.6	31.0	26.4
	walker2d-random-expert-diverse-5%-v2	19.2	33.1	7.7	66.4	76.4
	walker2d-random-expert-diverse-10%-v2	49.9	61.2	8.5	73.1	86.4
	walker2d-random-expert-diverse-50%-v2	64.1	108.8	15.3	108.1	108.6
Mixed (small)	ant-random-medium-10%-small-v2	8.5	53.2	4.5	52.7	21.7
	ant-random-expert-10%-small-v2	8.1	27.8	4.0	24.0	13.9
	hopper-random-medium-10%-small-v2	11.0	51.7	9.5	49.5	39.0
	hopper-random-expert-10%-small-v2	3.5	20.6	4.0	29.9	8.0
	halfcheetah-random-medium-10%-small-v2	3.7	15.0	22.6	14.6	7.3
	halfcheetah-random-expert-10%-small-v2	2.4	3.1	-1.9	3.2	2.2
	walker2d-random-medium-10%-small-v2	1.6	24.1	0.3	37.6	8.4
	walker2d-random-expert-10%-small-v2	0.5	2.8	0.2	4.8	1.7

Table 5: Full results of average returns of IQL in total of 113 datasets.