Cross-Domain Offline Reinforcement Learning with Nearest-Neighbor Guided Diffusion Model

Linh Le Pham Van[1] **Minh Hoang Nguyen**[1], **Duc Kieu**[1],
Hung Le [1], **Hung The Tran**[2], **Sunil Gupta**[1]
[1]Applied AI Institute, Deakin University, Geelong, Australia
[2]Hanoi University of Science and Technology
l.le@deakin.edu.au, s223669184@deakin.edu.au, v.kieu@deakin.edu.au,
thai.le@deakin.edu.au, Hungtt@soict.hust.edu.vn, sunil.gupta@deakin.edu.au

Abstract

Cross-domain offline reinforcement learning (RL) seeks to enhance sample efficiency in offline RL by utilizing additional offline source datasets. Existing approaches address this problem by measuring domain gaps through domain classifiers, target transition dynamics modeling, or mutual information estimation using contrastive loss. However, these methods often require large target datasets, which are impractical in many real-world scenarios. In this work, we address cross-domain offline RL under a limited target data setting, identifying two primary challenges: (1) Dataset imbalance, which leads to overfitting in neural network-based domain gap estimators, resulting in uninformative measurements; and (2) Partial domain overlap, where only a subset of the source data is closely aligned with the target domain. To overcome these issues, we propose Guided-Diffusion Cross-Domain (DnD), a novel framework for cross-domain offline RL with limited target samples. Specifically, DnD utilizes k-nearest neighbor (k-NN) estimation to measure domain proximity without neural network training, effectively mitigating overfitting. Furthermore, DnD introduces a nearest-neighbor-guided diffusion model to generate additional source samples that are better aligned with the target domain, thus enhancing policy learning with more effective source samples. Through theoretical analysis and extensive experiments in diverse MuJoCo environments, we demonstrate that DnD significantly outperforms state-of-the-art cross-domain offline RL methods, achieving substantial performance gains.

Introduction

Reinforcement Learning (RL) has demonstrated its ability to solve complex real-world problems (Mnih et al. 2015; Schrittwieser et al. 2020). However, RL typically requires extensive trial-and-error interactions with the environment, which can be infeasible in scenarios where data collection is costly or safety is a concern, such as autonomous driving or healthcare. A common solution is to train policies in a safer, faster source environment (e.g., a simulator) while leveraging a limited amount of real-world target data. This paradigm is known as cross-domain RL (Eysenbach et al. 2021; Xu et al. 2023; Lyu et al. 2024b).

Previous research has tackled cross-domain RL in various settings, including online where both domains are online (Eysenbach et al. 2021; Lyu et al.), or offline target with online source (hybrid) (Niu et al. 2022, 2023). In this work, we focus on the cross-domain offline setting (Liu, Hongyin, and Wang 2022; Liu et al. 2024; Xue et al. 2024), where both source and target domains are offline. This setting is crucial for enhancing the sample efficiency of offline RL methods (Kumar et al. 2020; Kostrikov, Nair, and Levine 2022). Existing works on cross-domain offline reinforcement learning (RL) have introduced various methods to measure the dynamics gap between domains, such as training domain classifiers (Liu, Hongyin, and Wang 2022), using conditional variational autoencoder (CVAE) models to approximate target dynamics (Liu et al. 2024), and estimating mutual information with contrastive loss (Wen et al. 2024). However, these approaches typically rely on large target datasets, which are impractical in many real-world applications, such as healthcare. To address this limitation, we focus on cross-domain offline RL in the limited target data setting, which brings several challenges that require novel solutions.

We begin by carefully analyzing the challenges in the cross-domain offline RL with *limited* target data setting. Specifically, we identify two major challenges: (1) Source-target dataset imbalance, which can cause neural network-based domain gap estimators to overfit or be biased toward a large amount of source samples, leading to uninformative dynamics gap measurements; and (2) Partial domain overlap, where only a partial subset of the source data is closely aligned with the target domain. To address the dataset imbalance challenge, we propose a novel estimation to quantify the proximity of source samples to the target domain via k-nearest neighbor (k-NN) without neural network training. To tackle the partial domain overlap challenge, we propose to generate more source samples close to the target domain to enhance policy learning. Leveraging the effectiveness and flexibility of diffusion models as demonstrated in (Janner et al. 2022; Lu et al. 2023), we introduce a novel nearest-neighbor-guided diffusion model. Specifically, we utilize a classifier-free guidance diffusion model, which learns the distribution of the source dataset conditioning with the $k-$NN based proximity score. Thus, our method provides an accurate and effective sampling method for generating the desired source samples. We name our approach Guided-**D**iffusio**n** Cross-**D**omain Offline RL (DnD). In addition to our methodological contributions, we provide a theoretical analysis of DnD and empirically

validate its effectiveness across various Gym-MuJoCo environments (Todorov, Erez, and Tassa 2012; Brockman et al. 2016). Our results demonstrate that DnD significantly outperforms state-of-the-art cross-domain offline RL methods, achieving substantial performance improvements.

Related Works

Cross-Domain Offline RL

Cross-domain RL aims to improve sample efficiency in the target domain by leveraging data from additional source environments. Several approaches have been proposed to address this challenge, including system identification (Werbos 1989; Zhu et al. 2018; Chebotar et al. 2019), domain randomization (Sadeghi and Levine 2017; Tobin et al. 2017; Peng et al. 2018), and meta-RL (Finn, Abbeel, and Levine 2017; Nagabandi et al. 2018; Wu et al. 2023). However, these methods often require environment models or domain knowledge to carefully select randomized parameters. Recently, several methods have attempted to measure dynamics discrepancy for various settings, including purely online (Eysenbach et al. 2021; Le Pham Van, The Tran, and Gupta 2024; Xu et al. 2023), purely offline (Liu et al. 2024; Wen et al. 2024; Xue et al. 2024), or hybrid setting (Niu et al. 2022). In this work, we focus on the cross-domain *offline* setting, where both the source and target domains are offline. Previous approaches tackle this problem through reward modification (Liu, Hongyin, and Wang 2022), support constraints (Liu et al. 2024), or data filtering via mutual information (Wen et al. 2024). However, these methods require training neural networks, which can be challenging when only limited target data is available. To overcome this limitation, we propose using $k-$NN estimation to measure the divergence between source and target domains, avoiding the need for neural network training. The concurrent work from Anonymous (2025) introduced OTDF, which leverages optimal transport to address neural network training challenges. While both optimal transport and $k-$NN estimation avoid reliance on neural networks, OTDF does not explicitly handle partial domain overlap. In contrast, we employ a guided-diffusion model to augment the source dataset with samples that are close to the target domain. These strategies enable our method to perform effectively, even in settings with limited target data.

Diffusion Model in RL

Diffusion models (Ho, Jain, and Abbeel 2020; Song et al. 2021) have been successfully applied as policy models (Kang et al. 2023; Wang, Hunt, and Zhou 2023), planners (Janner et al. 2022; Liang et al. 2023; Li 2024), and data synthesizers (Lu et al. 2023), showcasing their effectiveness across various RL tasks, including offline RL (Lu et al. 2023; Wang, Kulkarni, and Verdú 2009), multi-task learning (He et al. 2023), and meta RL (Ni et al. 2023). In this work, we propose a novel nearest-neighbor-guided diffusion model tailored for cross-domain offline RL, introducing a unique approach to address domain adaptation challenges.

Nearest Neighbor in RL

The nearest neighbor approach has been applied to various RL problems, such as enforcing policy constraints in offline setting (Ran et al. 2023), quantifying uncertainty (Qiao et al. 2024), performing MixUp-based data augmentation (Sander et al. 2022), and imitation learning (Lyu et al. 2024a). In contrast, our paper studies the cross-domain RL with limited target samples. Furthermore, we leverage k-NN estimation to guide sampling from a diffusion model, enabling the generation of additional source data closely aligned with the target domain.

Preliminaries

Reinforcement Learning

We first introduce Markov Decision Processes (MDP) which is denoted as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \gamma, r, d_0, P)$, where $\mathcal{S}, \mathcal{A}$ are the state and action spaces. The parameter $\gamma \in (0, 1)$ is the discounted factor, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, d_0 is the initial state distribution and P is the transition dynamics. We assume the rewards are bounded, which means $|r(s, a)| \leq r_{\max}, \forall s, a$. We denote a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ as a map from state space $\mathcal{S}$ to a probability distribution over actions space $\mathcal{A}$. Given a policy π and a transition dynamics (model) P, we denote discounted state-action occupancy as $d^\pi_P(s, a) = (1 - \gamma)\mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}(s_t = s, a_t = a)\right]$. Given a policy π and the model P, we define state-action value function Q^π_P as $Q^\pi_P(s, a) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a\right]$, and the value function V^π_P as $V^\pi_P(s) = \mathbb{E}_{a\sim\pi}\left[Q^\pi_P(s, a)\right]$. The objective of RL is to find a policy that maximizes the expected return $J^P(\pi) = \mathbb{E}_{s\sim d_0}\left[V^\pi_P(s)\right]$.

We address the cross-domain offline setting, where the source domain is defined as $\mathcal{M}_{src} = (\mathcal{S}, \mathcal{A}, \gamma, r, d_0, P_{src})$ and the target domain as $\mathcal{M}_{tar} = (\mathcal{S}, \mathcal{A}, \gamma, r, d_0, P_{tar})$. We assume that the two domains share the same state space $\mathcal{S}$, action space $\mathcal{A}$, discount factor γ, reward function r, and initial state distribution d_0, differing only in their dynamics models. In this setting, we have access to offline datasets collected from both domains: D_{src} from the source domain and D_{tar} from the target domain. The objective is to utilize the additional source dataset D_{src} to improve policy learning for the target domain, using the target dataset D_{tar}. We further denote the empirical MDP induced by a dataset D as $\widehat{\mathcal{M}} = \left(\mathcal{S}, \mathcal{A}, r, d_0, \gamma, \widehat{P}\right)$, where $\widehat{P}$ represents the empirical transition dynamics derived from D. In the cross-domain offline setting, this results in two empirical MDPs: $\widehat{\mathcal{M}}_{src}$ with transitions $\widehat{P}_{src}$ and $\widehat{\mathcal{M}}_{tar}$ with transitions $\widehat{P}_{tar}$.

Diffusion Model

Diffusion models (Ho, Jain, and Abbeel 2020) are a class of generative models that learn to denoise and generate data from noise iteratively. As both the diffusion model and RL involve time steps, we use *superscripts* to denote diffusion steps and *subscripts* for RL time steps. Specifically, diffusion models corrupt clean data samples $x^0 \sim q\left(x^0\right)$ by progressively adding Gaussian noise, following a Markov process:

$$q\left(x^{1:T} \mid x^0\right) = \prod_{t=1}^{T} q\left(x^t \mid x^{t-1}\right) = \prod_{t=1}^{T} \mathcal{N}\left(\sqrt{\alpha^t} x^{t-1}, \beta^t \mathrm{I}\right) \quad (1)$$

where β^t represents the noise schedule, chosen such that $q\left(x^T \mid x^0\right) \approx \mathcal{N}(0, \mathrm{I})$. Notably, we can directly sample at an arbitrary timestep t given data samples using $q\left(x^t \mid x^0\right) = \mathcal{N}\left(\sqrt{\overline{\alpha}^t} x^0, \left(1-\overline{\alpha}^t\right) \mathrm{I}\right)$ with $\overline{\alpha}^t = \prod_{i=1}^{t} \alpha^i$. Then, diffusion models learn to reverse the corruption process for generating data, where the reverse transition $q\left(x^{t-1} \mid x^t\right)$ is approximated by a parameterized model $p_\theta\left(x^{t-1} \mid x^t\right) = \mathcal{N}\left(\mu_\theta\left(x^t\right), \sigma_t \mathrm{I}\right)$. The training objective of diffusion models is to maximize the variational bound of $\log p_\theta(x^0)$, resulting in matching the parameterized model $p_\theta\left(x^{t-1} \mid x^t\right)$ with the distribution $q\left(x^{t-1} \mid x^t, x^0\right)$, which is given by:

$$q\left(x^{t-1} \mid x^t, x^0\right) = \mathcal{N}\left(\frac{1}{\sqrt{\alpha^t}}\left(x^t - \frac{1-\alpha^t}{\sqrt{1-\overline{\alpha}^t}}\epsilon\right), \beta^t \mathrm{I}\right) \quad (2)$$

Here, ϵ represents the Gaussian noise added to x^0 to form x^t. Instead of directly matching the mean of $q\left(x^{t-1} \mid x^t, x^0\right)$ with the mean of $p_\theta\left(x^{t-1} \mid x^t\right)$, i.e., $\mu_\theta\left(x^t\right)$, a noise network ϵ_θ is adopted to estimate ϵ from x^t. This network is trained by minimizing the regression loss:

$$\min_\theta \mathbb{E}_{x^0, \epsilon, t}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\overline{\alpha}^t} x^0 + \sqrt{1-\overline{\alpha}^t}\epsilon, t\right)\right\|^2\right] \quad (3)$$

where $x^0 \sim q(x^0)$, $\epsilon \sim \mathcal{N}(0, \mathrm{I})$, and $t \sim \mathcal{U}[0, T]$.

Guided sampling from diffusion models generates data conditioned on specific attributes, denoted as y. In our setting, y represents a score function that quantifies the proximity of a sample to the target domain. The objective is to sample from the conditional distribution $p(x|y)$. There are two primary techniques: classifier-guidance and classifier-free guidance. In this work, we consider the classifier-free guidance approach (Ho and Salimans 2022). Classifier-free sampling incorporates an additional conditional noise model, $\epsilon_\theta(x^t, y, t)$. During sampling, the noise for classifier-free guidance is computed as:

$$\hat{\epsilon}_w(x^t, y, t) = w\epsilon_\theta(x^t, y, t) + (1-w)\epsilon_{\theta'}(x^t, t), \quad (4)$$

where w is the guidance coefficient that controls the strength of the conditioning. In practice, the models $\epsilon_\theta(x^t, y, t)$ and $\epsilon_{\theta'}(x^t, t)$ share the same parameters, i.e., $\theta = \theta'$, and the unconditional model is implemented by setting y to an empty value, i.e. $y = \emptyset$.

Cross-Domain Offline RL with Limited Target Samples

In this section, we present the major challenges in cross-domain offline RL with limited target samples. Due to space limits, we defer more detailed discussions in Appendix .

Datasets Imbalanced Problem

In cross-domain offline RL, a critical challenge is effectively quantifying the domain gap between the source and target domains. Accurate measurement of this gap enables subsequent strategies to leverage source samples to enhance policy learning for the target domain. Prior works have proposed various approaches for this purpose, including training domain classifiers (Liu, Hongyin, and Wang 2022), or estimating target models (Liu et al. 2024). However, these methods often rely on training parametric models such as neural networks, which are prone to overfitting in scenarios with limited target data and significant dataset imbalances. As shown in Figure 1, limited target samples can cause domain classifiers to develop biases, favoring data that appears more frequently while failing to capture the true dynamics gaps. In addition to the overfitting issues, the imbalance between source and target data can bias the learned policy, potentially causing it to overfit the source data during training.

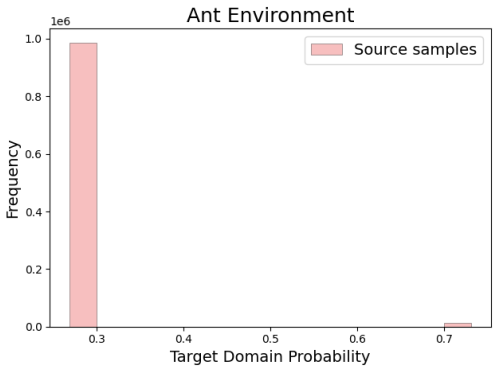


Figure 1: Predicted target domain probabilities for input source samples using trained domain classifiers (DARA) under the Ant environment. The predictions exhibit low diversity and are biased toward a value of 0.3, offering limited information about the domain gaps.

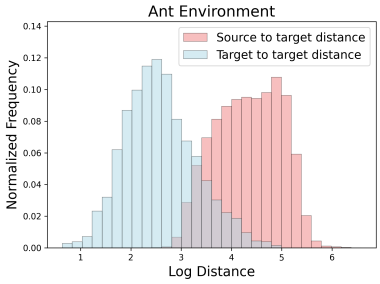


Figure 2: Nearest Neighbor Distance histograms. Blue shows the distance of source samples to their nearest target samples. Pink shows the distance of target samples to their nearest target samples.

Partial Overlapping between Domains

Leveraging source samples is critical for enhancing policy learning and improving sample efficiency in cross-domain offline RL. However, discrepancies in dynamics between the source and target domains, along with potential policy shifts during data collection processes, not all source data is close to the target data. To investigate how the source data is close to the target data, we compute the distances from each source transition $(s, a, s')_{src}$ to its nearest transition in the target dataset D_{tar} and the distances from each target

transition $(s, a, s')_{tar}$ to its nearest neighbor within D_{tar}. The histograms in Figure 5 reveal only partial overlap between two histograms, highlighting that only a subset of the source dataset is beneficial for target policy learning. Previous methods (Liu et al. 2024; Wen et al. 2024) addressed this by filtering out source samples far from the target domain. However, this approach reduces the number of usable source samples, ultimately impacting the sample efficiency of the algorithms. This raises an important question: "Can we further improve sample efficiency in cross-domain offline RL by generating additional source data that closely aligns with the target domain?"

Diffusion-Guided Cross-Domain Offline RL

In this section, we introduce our proposed method, illustrated in Figure 3. We first present the $k-$NN estimation for measuring domain gaps between the source and target. We then leverage a guided diffusion model, conditioned on $k-$NN scores, to augment the dataset with source samples that closely resemble the target domain. Next, we formulate a practical algorithm that integrates $k-$NN estimation and guided diffusion for policy learning. Finally, we summarize our approach, DnD, and provide its theoretical analysis.

k-NN Based Domain Gap Estimation

Training neural networks often encounter challenges caused by the limited and imbalanced datasets, raising a critical question: "Can we measure domain gaps between source and target datasets without relying on neural network training?" To address this, we propose using k-NN estimation as an alternative approach to effectively quantify domain gaps, bypassing the limitations of prior methods.

In the following, we denote $\oplus$ as the concatenation operator, and $x_{tar} = s_{tar} \oplus a_{tar} \oplus s'_{tar}$ and $x_{src} = s_{src} \oplus a_{src} \oplus s'_{src}$, where $s_{tar}, a_{tar}, s'_{tar} \sim D_{tar}$ and $s_{src}, a_{src}, s'_{src} \sim D_{src}$. Thus we have $D_{src} = \{x_{src,i}\}_{i=1}^{i=N}$ and $D_{tar} = \{x_{tar,i}\}_{i=1}^{i=M}$, where N, M are the sizes of the source dataset and target dataset respectively. We denote $\nu_{k,\text{src}}(i)$ is the Euclidean distance of the kth nearest neighbor of $x_{src,i}$ in the source dataset D_{src}, and $\nu_{k,\text{tar}}(i)$ is the Euclidean distance of the kth nearest neighbor of $x_{src,i}$ in the target dataset D_{tar}. Let $\mathcal{B}(x, R)$ denote a closed ball around $x \in \mathbb{R}^d$ with the radius R, and $\mathcal{V}(\mathcal{B}(x, R)) = cR^d$ is its volume, where $c = \frac{\pi^{d/2}}{\Gamma(d/2+1)}$ is the volume of a d-dimensional unit ball, and Γ is the Gamma function. Following the k-NN based density estimators (Wang, Kulkarni, and Verdú 2009; Póczos and Schneider 2011), given a dataset D, we have the following estimator:

$$P_{D,k}(x^i) = \frac{k/|D|}{\mathcal{V}(\mathcal{B}(x, \nu_{k,D}(x)))} = \frac{k}{|D| c \nu_{k,D}(x)}, \quad (5)$$

where $|D|$ is the size of the dataset D and $\nu_{k,D}(x)$ is the distance between x and its kth nearest neighbor in D.

We quantify the domain gaps between the source and target domains as the k-NN estimator for the Kullback-Leibler divergence $\mathcal{D}_{KL}$. Specifically, we measure the estimation of $\mathcal{D}_{KL}$ between source and target domains as follows:

$$\begin{aligned} \widehat{\mathcal{D}}_{KL}(P_{src}||P_{tar}) &= \frac{1}{N}\sum_{i=1}^{N}\left(\log \frac{M\nu_{k,\text{tar}}^d(i)}{(N-1)\nu_{k,\text{src}}^d(i)}\right) \\ &\propto 1/N \sum_{i=1}^{N} \left(\log(\nu_{k,\text{tar}}^d(i)) - \log(\nu_{k,\text{src}}^d(i))\right) \end{aligned} \quad (6)$$

Thus, given a particular source data $x_{src,i} \in D_{src}$, we consider the discrepancy of it to the target dataset as follows:

$$\begin{aligned} &\rho_k(x_{src,i}) \\ &= \log(\|x_{src,i} - x_{tar}^{k,i}\|_2) - \log\left(\|x_{src,i} - x_{src,i,k}\|_2\right), \end{aligned} \quad (7)$$

where $x_{src,i,k}$ and $x_{tar,i,k}$ are the k nearest neighbors of $x_{src,i}$ in the source dataset D_{src} and the target dataset D_{tar} respectively. Intuitively, $\rho_k(x_{src,i})$ quantifies the discrepancy of the source sample $x_{src,i}$ to the target dataset, and is small if the distance from $x_{src,i}$ to its k-nearest neighbors in the target dataset is smaller than the distance to its k-nearest neighbors in the source dataset. The critical advantage of k-NN estimation lies in its consistency and simplicity: it is stable and avoids the need for neural network training, thus mitigating the risk of overfitting when estimating the domain gap. Furthermore, it is computationally efficient and easy to implement, leveraging the KD-tree data structure. In practice, we use FAISS library (Douze et al. 2024) in our implementation and compute all scores for 1 million source samples with 5000 target samples within 1 minute.

Nearest-Neighbor Guided Diffusion Model

To overcome the partial overlapping challenge, and bring more sample efficiency for cross-domain offline RL, we propose to upsample the source dataset with generated samples close to the target domain using the diffusion model. To ensure the generated source data is close to the target domain, we opt to use the classifier-free guidance diffusion model with scores computed leveraging the k-NN estimation.

Given the source dataset D_{src}, for each source transition $x_{src,i} = (s_{src}, a_{src}, s'_{src})_i$ in D_{src}, we leverage the k-NN estimation and compute its $\rho_k(x_{src,i})$. The k-NN estimation provides a reliable measurement of how close a source sample is to the target domain. Notably, we train a diffusion model on the source dataset D_{src} and the corresponding k-NN estimation scores as the conditional context y. We adopt the design of EDM (Karras et al. 2022) for our diffusion model, as it has demonstrated superior empirical performance in data modeling compared to earlier designs (Ho, Jain, and Abbeel 2020; Song et al. 2021). Specifically, we employ a fixed noise schedule $\sigma_{\max} = \sigma^T > \cdots > \sigma^1 > \sigma^0 = 0$ and diffuse clean data samples using the transition $q(x^t|x_{src,i}) = \mathcal{N}\left(x_{src,i}, (\sigma^t)^2 \mathrm{I}\right)$. Then, we train a denoiser $G_\theta(\cdot)$ to directly predict the clean samples using classifier-free training (Ho and Salimans 2022):

$$\min_\theta \mathbb{E}\left[\left\|x_{src,i} - G_\theta\left(x_{src,i} + \sigma^t\epsilon, m \cdot \rho_k(x_{src,i}), \sigma^t\right)\right\|^2\right] \quad (8)$$

where $x_{src,i} \sim D_{src}$, $\epsilon \sim \mathcal{N}(0, \mathrm{I})$, and $t \sim \mathcal{U}[0, T]$. The score $\rho_k(x_{src,i})$ is randomly masked during training via the null token m is sampled from the Bernoulli distribution

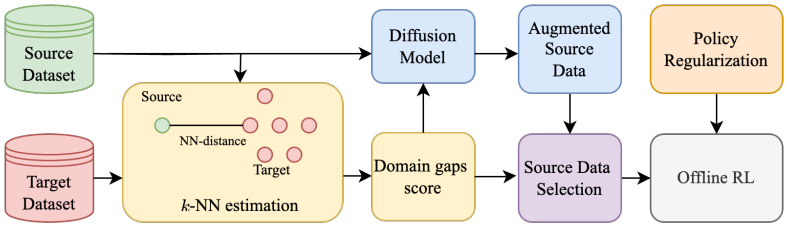


Figure 3: Illustration of our method. First, we use k-NN estimation to quantify the domain gap score. Next, we leverage a diffusion model to upsample the source data, generating samples close to the target domain. The datasets are then utilized in an offline RL framework, incorporating a regularization term to ensure the learned policy remains within the support region of the target dataset.

with $p\left(m=\emptyset\right)=0.25$. Notably, the trained denoiser's output can be reformulated into the noise model's objective as $\epsilon=\frac{G_\theta\left(x_{src,i}+\sigma^t\epsilon,\sigma^t\right)-x^t}{\sigma^t}$. Therefore, we can use the trained denoiser to perform guided sampling as introduced in Eq. (4) for the conditional generation.

The training process for the diffusion model is done before the training of the RL policy. Once the diffusion model is trained, we generate source samples that are close to the target domain by setting the value of the conditional context y to be higher than the top $\kappa\%$ of the source sample scores in the source dataset D_{src}. Specifically, we uniformly sample a value χ from the range $[\kappa, 100]$ and set y to the χ-th quantile of the source dataset's score distribution. Finally, the generated samples are combined with the source dataset for later policy learning.

Practical Algorithm

Based on the k-NN estimation as the domain gaps measurement and the guided-diffusion model for upsampling source dataset with generated samples close to the target domain, we obtain a practical policy adaptation algorithm, named as DnD (Guided-**D**iffusio**n** Cross-**D**omain Offline RL). We summarize our proposed algorithm in Algorithm 1. In practice, we normalize the value of ρ_k to $[0,1]$. Specifically, we first subtract each value with the minimum value of ρ_k in the source dataset D_{src} to adjust the value to the range $[0,+\infty]$, and then scale it as follows: $w_k(x_{src,i})=1/(1+\hat{\rho}_k(x_{src,i}))$, where the input $\hat{\rho}_k(x_{src,i})$ is the adjusted non-negative value of $\rho_k(x_{src,i})$. Intuitively, the larger the w_k value, the closer the source sample $x_{src,i}$ to the target domain. We select the source data close to the target domain to reduce the dynamics gap while training. Specifically, we formulate our objective function for training the value function $Q_\phi(s,a)$ as follows:

$$\mathcal{L}_Q=\mathbb{E}_{D_{tar}}\left[(Q_\phi-\mathcal{T}Q_\phi)^2\right]+\mathbb{E}_{D_{src}}\left[\omega(s,a,s')(Q_\phi-\mathcal{T}Q_\phi)^2\right], \tag{9}$$

where $\mathcal{T}$ is the Bellman operator, $\omega(s,a,s'):=w_k\mathbb{1}(w_k\geq w_{k,\xi\%})$, $\mathbb{1}$ is the indicator function, and $w_{k,\xi\%}$ denotes the top ξ-quantile score used for source data selection. Eq (9) ensures the policy adaptively emphasizes source data close to the target domain, improving adaptation performance. As we mentioned, the imbalanced dataset could bias the policy to source samples. Thus, we employ a policy regularization to ensure the learned policy is close to the support areas of the target dataset. Similar to Wu et al. (2022), we learn a CVAE, denoted as $\hat{\pi}^b_{tar}(a|s)$, to model the behavior target policy. Thus, we optimize the policy as follows:

$$\mathcal{L}_\pi^{\text{reg}}=\mathcal{L}_\pi-\lambda\mathbb{E}_{s\sim D_{src}\bigcup D_{tar}}\left[\log\hat{\pi}^b_{tar}\left(\pi(.|s)|s\right)\right], \tag{10}$$

where $\mathcal{L}_\pi$ is the policy loss of the offline RL method, λ is the coefficient controlling the strength of the additional policy regularization. We note that DnD can be integrated with any offline RL method. In our implementation, we choose IQL (Kostrikov, Nair, and Levine 2022) as our backbone of DnD.

Theoretical Analysis

We provide a theoretical guarantee for using the source dataset to improve the performance in the target domain under the cross-domain offline RL setting. Specifically, we have the following performance bound for any policy π:

Theorem 0.1. *Denote D_{src} as the offline source dataset from source domain $\mathcal{M}_{src}$ and D_{tar} as the offline target dataset from target domain $\mathcal{M}_{tar}$. Let the empirical policy in the offline target dataset D_{tar} be $\pi_{D_{tar}}=\frac{\sum_{D_{tar}}\mathbb{1}(s,a)}{\sum_{D_{tar}}\mathbb{1}(s)}$. Given a policy π, we have the following:*

$$\begin{aligned}
&J^{P_{tar}}(\pi)-J^{\widehat{P}_{src}}(\pi)\\
&\geq-\frac{2r_{max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi}_{P_{tar}},s'\sim P_{tar}}\left[D_{TV}\left(\pi(.|s')||\pi_{D_{tar}}(.|s')\right)\right]\\
&-\frac{2r_{max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi_{D_{tar}}}_{P_{tar}},s'\sim P_{tar}}\left[D_{TV}\left(\pi(.|s')||\pi_{D_{tar}}(.|s')\right)\right]\\
&-\frac{2r_{max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi_{D_{tar}}}_{P_{tar}}}\left[\sqrt{1/2D_{KL}(P_{tar}(.|s,a)||\widehat{P}_{src}(.|s,a))}\right].
\end{aligned} \tag{11}$$

We provide detailed proof in Appendix . The first and the second terms of the divergence in Eq. (11) measure the deviation between the current learned policy and the target behavior policy; the third term measures the dynamics mismatch between the actual target dynamics model and the estimated source dynamics model. Our proposed method aims to reduce the third term by effectively measuring the dynamics gaps between two domains and generating more source samples close to the target domain. Specifically, our k-NN estimation ρ_k estimates the KL divergence between the source dynamics and the target dynamics as shown in Eq. (6). Additionally, we select the nearest source samples

to update the Q-function (Eq. 9, Figure 3), further reducing the dynamics gap. Furthermore, our algorithm incorporates a policy regularization to address policy deviation between the learned policy and the target behavior policy.

Algorithm 1: Guided-**D**iffusio**n** Cross-**D**omain Offline RL – DnD

Input: Offline source D_{src}, offline target D_{tar} datasets.
Initialize: Value function Q, policy π, diffusion model G_θ, k NN value, coefficient λ, source score buffer SB.
for each source sample $x_{src,i} \in D_{src}$ **do**
    Compute its k-NN score $\rho_k(x_{src,i})$ via Eq. (7).
    $SB \leftarrow SB \bigcup \{\rho_k(x_{src,i})\}$.
end for
Train G_θ with D_{src} and SB via Eq. (8).
Generate samples from the diffusion model and add to the source dataset D_{src} and source score buffer SB.
for each iteration **do**
    Sample a batch $b_{src} = \{s, a, s', r\}$ and the corresponding score batch w_{src} from D_{src} and SB.
    Sample a batch $b_{tar} = \{s, a, s', r\}$ from D_{tar}.
    Update Q using b_{src}, w_{src} and b_{tar} via Eq. (9).
    Update π using b_{src} and b_{tar} via Eq. (10).
end for
return Q, π.

Experiments

In this section, we present empirical evaluations of our method, focusing on the following questions: (1) Does DnD improve sample efficiency for the base method and outperform strong baselines in cross-domain offline RL under limited target data settings? (2) How does k-NN estimation impact DnD's performance? (3) How does the guided diffusion model influence DnD's performance? Furthermore, we conduct parameter studies to provide deeper insights into the behavior and effectiveness of DnD. We defer more experiment results in Appendix due to space limits.

Tasks and Baselines

To evaluate the policy adaptation performance, we conducted experiments on four environments from the Gym-Mujoco framework (Brockman et al. 2016; Todorov, Erez, and Tassa 2012)(Ant, Halfcheetah, Hopper, Walker) and consider the gravity and kinematic shifts as the dynamics shifts between the source and the target environments. Specifically, we modify gravity strength to induce the gravity shift and restrict the rotation range of certain joints to create the kinematic shift. For offline source datasets, we use D4RL (Fu et al. 2020) and consider target datasets of varying quality—*medium*, *medium-expert*, and *expert*—following D4RL standards. Each target dataset contains 5000 transitions to enforce a limited target data setting. Please see Appendix for more details about the environment settings.

We evaluate our method DnD against the state-of-the-art cross-domain offline RL methods: **DARA** (Liu, Hongyin, and Wang 2022), which trains the domain classifiers to measure domain gaps; **BOSA** (Liu et al. 2024), which uses support-constrained regularization to ensure the value function and policy are optimized using in-support samples; **IGDF** (Wen et al. 2024), which quantifies domain gaps using a mutual information score function; **SRPO** (Xue et al. 2024) that does reward modification via the stationary state distribution; and **OTDF** (Anonymous 2025) that leverages optimal transport to estimate the domain gaps. We also compare DnD with the state-of-the-art offline RL algorithm, **IQL** (Kostrikov, Nair, and Levine 2022), trained on a mixture of the source and target datasets. Details of baselines are provided in Appendix . We evaluate the performance on the target domain in offline settings using the normalized score. All methods are trained for 1 million steps across 5 random seeds. Table 1 shows the performance of DnD and other baselines under the *gravity shift* setting. Due to space constraints, results for the *kinematic shift* setting are included in Table 9 in Appendix .

Adaptation Performance Evaluation

Answering question 1): Table 1 and Table 9 in Appendix demonstrate that DnD consistently outperforms IQL, winning in **36** out of 36 tasks under the gravity shift and **35** out of 36 tasks under the kinematic shift. Notably, DnD achieves a **97.9%** improvement in target normalized scores under gravity shift tasks and a **59.4%** improvement under kinematic shift tasks. Compared to other baselines, DnD achieves a significant performance gain. Under the gravity shift, DnD outperforms in **33** out of 36 tasks, attaining a total normalized score of **1632.7**, surpassing the second-best baseline, OTDF, by **40.7%**. Under the kinematic shift, DnD excels in **29** out of 36 tasks, achieving a total normalized score of **1902.2**, compared to 1547.6 for OTDF. These results validate the effectiveness of our approach.

We observe that cross-domain methods relying on neural network-based dynamics gap estimation (IGDF, BOSA, DARA) perform similarly to IQL across many tasks, indicating their limitations in effective offline policy adaptation. This is likely due to challenges in training neural networks on imbalanced datasets with limited target data, as discussed in Section and Appendix . In contrast, DnD employs $k-$NN estimation that does not require training neural networks. While OTDF leverages optimal transport to mitigate issues arising from limited target data and shows improvements over other baselines, it does not augment the dataset. DnD further enhances sample efficiency by leveraging a diffusion model to generate target-aligned data, leading to improved performance in the target domain.

Ablation Studies

DnD with other domain gaps measurement **Answering question 2):** To assess the importance of $k-$NN estimation in DnD, we replace the $k-$NN score with the score computed via the score estimated by domain classifiers as in DARA. We compare DnD with this variant on *medium* source datasets of Ant and Walker under gravity shifts. Table 2 shows that this modification leads to a notable performance drop, suggesting that domain classifiers struggle to accurately estimate domain gaps. These results highlight the effectiveness of $k-$NN estimation in DnD.

Table 1: Results in gravity shift tasks. We report normalized scores and their standard deviations in the target domain, averaged over five random seeds. The best score is bold. Half=Halfcheetah, Hopp=Hopper, m=medium, me=medium-expert, mr=medium-replay, e=expert.

Source	Target	IQL	DARA	BOSA	SRPO	IGDF	OTDF	DnD
Ant-m	m	10.2±1.8	9.4±0.9	12.4±2.0	11.7±1.0	11.3±1.3	45.1±12.4	**56.9±2.2**
Ant-m	me	9.4±1.2	10.0±0.9	11.6±1.3	10.2±1.2	9.4±1.4	33.9±5.4	**47.5±3.9**
Ant-m	e	10.2±0.3	9.8±0.6	11.8±0.4	9.5±0.6	9.7±1.6	33.2±9.0	**36.1±7.8**
Ant-me	m	9.8±2.4	8.1±1.8	8.1±3.0	8.4±2.1	8.9±1.5	18.6±11.9	**55.7±8.1**
Ant-me	me	9.0±0.8	6.4±1.4	6.2±1.5	6.1±3.5	7.2±2.9	34.0±9.4	**46.3±4.9**
Ant-me	e	9.1±2.6	10.4±2.9	4.2±3.9	8.8±1.0	9.2±1.5	23.2±2.9	**42.7±13.0**
Ant-mr	m	18.9±2.6	21.7±2.1	13.9±1.5	18.7±1.7	19.6±1.0	29.6±10.7	**41.8±4.7**
Ant-mr	me	19.1±3.0	18.3±2.1	15.9±2.7	18.7±1.8	20.3±1.6	25.4±2.1	**27.6±0.1**
Ant-mr	e	18.5±0.9	20.0±1.3	14.5±1.7	19.9±2.1	18.8±2.1	24.5±2.8	**28.0±0.3**
Half-m	m	39.6±3.3	41.2±3.9	38.9±4.0	36.9±4.5	36.6±5.5	40.7±7.7	**48.0±0.6**
Half-m	me	39.6±3.7	40.7±2.8	40.4±3.0	40.7±2.3	38.7±6.2	28.6±3.2	**48.9±0.7**
Half-m	e	42.4±3.8	39.8±4.4	40.5±3.9	39.4±1.6	39.6±4.6	36.1±5.3	**48.8±1.0**
Half-me	m	38.6±6.0	37.8±3.3	41.8±5.1	42.5±2.3	37.7±7.3	39.5±3.5	**49.9±1.4**
Half-me	me	39.6±3.0	39.4±4.4	38.7±3.7	43.3±2.7	40.7±3.2	32.4±5.5	**48.1±1.9**
Half-me	e	43.4±0.9	45.3±1.3	39.9±2.7	43.3±3.0	41.1±4.1	26.5±9.1	**51.0±0.8**
Half-mr	m	20.1±5.0	17.6±6.2	20.0±4.9	17.5±5.2	14.4±2.2	21.5±6.5	**32.2±0.8**
Half-mr	me	17.2±1.6	20.2±5.2	16.7±4.2	16.3±1.7	10.0±2.5	14.7±4.1	**30.3±5.1**
Half-mr	e	20.7±5.5	22.4±1.7	15.4±4.2	23.1±4.0	15.3±3.7	11.4±1.9	**32.7±2.6**
Hopp-m	m	11.2±1.1	17.3±3.8	15.2±3.3	12.4±1.0	15.3±3.5	**32.4±8.0**	30.6±5.5
Hopp-m	me	14.7±3.6	15.4±2.5	21.1±9.3	14.2±1.8	15.1±3.6	24.2±3.6	**35.7±1.4**
Hopp-m	e	12.5±1.6	19.3±10.5	12.7±1.7	11.8±0.9	14.8±4.0	33.7±7.8	**51.3±10.6**
Hopp-me	m	19.1±6.6	18.5±12.3	15.9±5.9	19.7±8.5	22.3±5.4	26.4±10.1	**52.2±6.6**
Hopp-me	me	16.8±2.7	16.0±6.1	17.3±2.5	15.8±3.3	16.6±7.7	28.3±6.7	**50.4±11.3**
Hopp-me	e	20.9±4.1	23.9±14.8	23.2±7.9	21.4±1.9	26.0±9.2	44.9±10.6	**52.8±10.3**
Hopp-mr	m	13.9±2.9	10.7±4.3	3.3±1.9	14.0±2.6	15.3±4.4	31.1±13.4	**35.4±0.7**
Hopp-mr	me	13.3±6.3	12.5±5.6	4.6±1.7	14.4±4.2	15.4±5.5	24.2±6.1	**41.0±2.0**
Hopp-mr	e	11.0±2.6	14.3±6.0	3.2±0.8	16.4±5.0	16.1±4.0	**31.0±9.8**	29.9±7.8
Walker-m	m	28.1±12.9	28.4±13.7	38.0±11.2	21.4±7.0	22.1±8.4	36.6±2.3	**52.5±2.0**
Walker-m	me	35.7±4.7	30.7±9.7	40.9±7.2	34.0±9.9	35.4±9.1	44.8±7.5	**59.2±2.7**
Walker-m	e	37.3±8.0	36.0±7.0	41.3±8.6	39.5±3.8	36.2±13.6	44.0±4.0	**63.8±2.7**
Walker-me	m	39.9±13.1	41.6±13.0	32.3±7.2	46.4±3.5	33.8±3.1	30.2±9.8	**57.5±3.3**
Walker-me	me	49.1±6.9	45.8±9.4	40.1±4.5	36.4±3.4	44.7±2.9	53.3±7.1	**67.8±4.0**
Walker-me	e	40.4±11.9	56.4±3.5	43.7±4.4	45.8±8.0	45.3±10.4	61.1±3.4	**67.1±4.8**
Walker-mr	m	14.6±2.5	14.1±6.1	7.6±5.8	17.9±3.8	11.6±4.6	32.7±7.0	**42.2±5.9**
Walker-mr	me	15.3±1.9	15.9±5.8	4.8±5.8	15.3±4.5	13.9±6.5	**31.6±6.1**	31.5±5.2
Walker-mr	e	15.8±7.2	15.7±4.5	7.1±4.6	13.7±8.1	15.2±5.3	31.3±5.3	**39.3±6.2**
Total Score		825.0	851.0	763.2	825.5	803.6	1160.7	**1632.7**

Table 2: Performance comparison between DnD and its variant using classifiers score to measure the domain gaps.

Source	Target	Classifier score	k-NN score (DnD)
Ant-m	m	31.5±5.3	**56.9±2.2**
Ant-m	me	15.0±0.8	**47.5±3.9**
Ant-m	e	19.2±1.9	**36.1±7.8**
Walker-m	m	52.1±3.5	**52.5±2.0**
Walker-m	me	**59.2±2.7**	**52.9±2.7**
Walker-m	e	61.9±2.6	**63.8±2.7**

Effect of Guided-Diffusion Model **Answering question 3):** We compare DnD against three variants: (1) DnD without upsampling via the diffusion model (*w/o diffusion*), (2) DnD where the target dataset is upsampled using a diffusion model (*upsample target*), and (3) DnD with naive diffusion-based generation without guidance (*w/o guidance*). We conduct experiments on Ant *medium* source dataset under gravity shift and present results in Table 3. We observe that *upsample target* slightly improves performance over *w/o diffusion*. However, the *w/o guidance* variant, which naively upsam-

Table 3: Performance comparison between DnD and its variants on using the diffusion model. We bold the highest scores.m=medium, e=expert, me=medium-expert.

Method	m-m	m-me	m-e
w/o diffusion	23.6±2.9	14.4±0.8	19.7±1.4
upsample target	25.4±3.1	23.7±3.7	24.2±2.7
w/o guidance	46.8±11.5	31.6±11.5	22.43.6
w guidance (DnD)	**56.9±2.2**	**47.5±3.9**	**36.1±7.8**

Table 4: Performance comparison with different values of k-NN.

Source	Target	k-Nearest Neighbor		
		1	5	10
Half-gravity-m	me	48.8±0.7	**48.9±0.7**	48.5±0.5
Half-gravity-m	e	**49.1±0.6**	48.8±1	49.0±0.3
Half-kinematic-m	me	**19.2±5**	19.1±1	14.6±1.5
Half-kinematic-m	e	13.4±1.2	13.1±0.8	**13.5±0.9**
Hopp-gravity-m	me	36.9±12	40.6±5.5	**44.6±10.5**
Hopp-gravity-m	e	40.2±8	**51.3±10.6**	34.4±9.8
Hopp-kinematic-m	me	73.6±3.1	**78.2±5.1**	76.6±4.2
Hopp-kinematic-m	e	57.9±11.7	59.8±21.8	**61.9±17.5**

ples the source dataset, achieves significantly better results than *upsample target*. We hypothesize that the limited target data setting hinders the training diffusion model in the target domain, thus affecting adaptation performance. Among all variants, DnD achieves the highest performance, demonstrating the effectiveness of our proposed method of using a guided-diffusion model to upsample the source dataset with generated data close to the target domain.

Nearest Neighbor k We evaluate the impact of k in DnD by testing different values ($k = 1, 5, 10$). As shown in Table 4, DnD remains robust across varying k values. In our experiments, we set $k = 5$ by default and do not tune it.

Conclusion

In this paper, we tackle the challenge of cross-domain offline RL in scenarios with limited target data, which is widely encountered in many real-world applications. We systematically analyze the limitations of existing methods under this setting, identifying key shortcomings. Building on these insights, we propose DnD, a novel algorithm that leverages k-NN estimation to effectively quantify domain gaps and uses this score in a guided diffusion model to generate source samples closer to the target domain. Our approach is compatible with any offline RL method, offering broad applicability. Extensive experiments demonstrate the superior performance of DnD across various benchmarks. A limitation of DnD is that it only adopts a diffusion model for the single-step dynamics model. Extending it to model full trajectory distributions presents an exciting avenue for further exploration and in-depth study.

References

Anonymous. 2025. Cross-Domain Offline Policy Adaptation with Optimal Transport and Dataset Constraint. In *The Thirteenth International Conference on Learning Representations*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Chebotar, Y.; Handa, A.; Makoviychuk, V.; Macklin, M.; Issac, J.; Ratliff, N.; and Fox, D. 2019. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, 8973–8979. IEEE.

Dong, Z.; Yuan, Y.; Hao, J.; Ni, F.; Ma, Y.; Li, P.; and Zheng, Y. 2024. CleanDiffuser: An Easy-to-use Modularized Library for Diffusion Models in Decision Making. *arXiv preprint arXiv:2406.09509*.

Douze, M.; Guzhva, A.; Deng, C.; Johnson, J.; Szilvasy, G.; Mazaré, P.-E.; Lomeli, M.; Hosseini, L.; and Jégou, H. 2024. The Faiss library.

Eysenbach, B.; Chaudhari, S.; Asawa, S.; Levine, S.; and Salakhutdinov, R. 2021. Off-Dynamics Reinforcement Learning: Training for Transfer with Domain Classifiers. In *International Conference on Learning Representations*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135. PMLR.

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

He, H.; Bai, C.; Xu, K.; Yang, Z.; Zhang, W.; Wang, D.; Zhao, B.; and Li, X. 2023. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in neural information processing systems*, 36: 64896–64917.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Ho, J.; and Salimans, T. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Janner, M.; Du, Y.; Tenenbaum, J.; and Levine, S. 2022. Planning with Diffusion for Flexible Behavior Synthesis. In *International Conference on Machine Learning*, 9902–9915. PMLR.

Johnson, J.; Douze, M.; and Jégou, H. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3): 535–547.

Kang, B.; Ma, X.; Du, C.; Pang, T.; and Yan, S. 2023. Efficient diffusion policies for offline reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 67195–67212.

Karras, T.; Aittala, M.; Aila, T.; and Laine, S. 2022. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing*.

Kostrikov, I.; Nair, A.; and Levine, S. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *International Conference on Learning Representations*.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191.

Le Pham Van, L.; The Tran, H.; and Gupta, S. 2024. Policy Learning for Off-Dynamics RL with Deficient Support. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 1093–1100.

Li, W. 2024. Efficient Planning with Latent Diffusion. In *The Twelfth International Conference on Learning Representations*.

Liang, Z.; Mu, Y.; Ding, M.; Ni, F.; Tomizuka, M.; and Luo, P. 2023. AdaptDiffuser: diffusion models as adaptive self-evolving planners. In *Proceedings of the 40th International Conference on Machine Learning*, 20725–20745.

Liu, J.; Hongyin, Z.; and Wang, D. 2022. DARA: Dynamics-Aware Reward Augmentation in Offline Reinforcement Learning. In *International Conference on Learning Representations*.

Liu, J.; Zhang, Z.; Wei, Z.; Zhuang, Z.; Kang, Y.; Gai, S.; and Wang, D. 2024. Beyond ood state actions: Supported cross-domain offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 13945–13953.

Lu, C.; Ball, P. J.; Teh, Y. W.; and Parker-Holder, J. 2023. Synthetic experience replay. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 46323–46344.

Luo, Y.; Xu, H.; Li, Y.; Tian, Y.; Darrell, T.; and Ma, T. 2018. Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees. In *International Conference on Learning Representations*.

Lyu, J.; Bai, C.; Yang, J.-W.; Lu, Z.; and Li, X. ???? Cross-Domain Policy Adaptation by Capturing Representation Mismatch. In *Forty-first International Conference on Machine Learning*.

Lyu, J.; Ma, X.; Wan, L.; Liu, R.; Li, X.; and Lu, Z. 2024a. SEABO: A Simple Search-Based Method for Offline Imitation Learning. In *The Twelfth International Conference on Learning Representations*.

Lyu, J.; Xu, K.; Xu, J.; Yan, M.; Yang, J.; Zhang, Z.; Bai, C.; Lu, Z.; and Li, X. 2024b. ODRL: A Benchmark for Off-Dynamics Reinforcement Learning. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Nagabandi, A.; Clavera, I.; Liu, S.; Fearing, R. S.; Abbeel, P.; Levine, S.; and Finn, C. 2018. Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning. In *International Conference on Learning Representations*.

Ni, F.; Hao, J.; Mu, Y.; Yuan, Y.; Zheng, Y.; Wang, B.; and Liang, Z. 2023. Metadiffuser: Diffusion model as conditional planner for offline meta-rl. In *International Conference on Machine Learning*, 26087–26105. PMLR.

Niu, H.; Ji, T.; Liu, B.; Zhao, H.; Zhu, X.; Zheng, J.; Huang, P.; Zhou, G.; Hu, J.; and Zhan, X. 2023. H2O+: An Improved Framework for Hybrid Offline-and-Online RL with Dynamics Gaps. arXiv:2309.12716.

Niu, H.; Qiu, Y.; Li, M.; Zhou, G.; Hu, J.; Zhan, X.; et al. 2022. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 36599–36612.

Peng, X. B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2018. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, 3803–3810. IEEE.

Póczos, B.; and Schneider, J. 2011. On the Estimation of α-Divergences. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 609–617. JMLR Workshop and Conference Proceedings.

Qiao, Z.; Lyu, J.; Jiao, K.; Liu, Q.; and Li, X. 2024. Sumo: Search-based uncertainty estimation for model-based offline reinforcement learning. *arXiv preprint arXiv:2408.12970*.

Ran, Y.; Li, Y.-C.; Zhang, F.; Zhang, Z.; and Yu, Y. 2023. Policy regularization with dataset constraint for offline reinforcement learning. In *International Conference on Machine Learning*, 28701–28717. PMLR.

Sadeghi, F.; and Levine, S. 2017. CAD2RL: Real Single-Image Flight Without a Single Real Image. *Robotics: Science and Systems XIII*.

Sander, R.; Schwarting, W.; Seyde, T.; Gilitschenski, I.; Karaman, S.; and Rus, D. 2022. Neighborhood mixup experience replay: Local convex interpolation for improved sample efficiency in continuous control tasks. In *Learning for Dynamics and Control Conference*, 954–967. PMLR.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609.

Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 23–30. IEEE.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, 5026–5033. IEEE.

Wang, Q.; Kulkarni, S. R.; and Verdú, S. 2009. Divergence estimation for multidimensional densities via k-Nearest-Neighbor distances. *IEEE Transactions on Information Theory*, 55(5): 2392–2405.

Wang, Z.; Hunt, J. J.; and Zhou, M. 2023. Diffusion Policies as an Expressive Policy Class for Offline Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*.

Wen, X.; Bai, C.; Xu, K.; Yu, X.; Zhang, Y.; Li, X.; and Wang, Z. 2024. Contrastive Representation for Data Filtering in Cross-Domain Offline Reinforcement Learning. In *Forty-first International Conference on Machine Learning*.

Werbos, P. J. 1989. Neural networks for control and system identification. In *Proceedings of the 28th IEEE Conference on Decision and Control,*, 260–265. IEEE.

Wu, J.; Wu, H.; Qiu, Z.; Wang, J.; and Long, M. 2022. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 31278–31291.

Wu, Z.; Xie, Y.; Lian, W.; Wang, C.; Guo, Y.; Chen, J.; Schaal, S.; and Tomizuka, M. 2023. Zero-shot policy transfer with disentangled task representation of meta-reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 7169–7175. IEEE.

Xu, K.; Bai, C.; Ma, X.; Wang, D.; Zhao, B.; Wang, Z.; Li, X.; and Li, W. 2023. Cross-domain policy adaptation via value-guided data filtering. *Advances in Neural Information Processing Systems*, 36: 73395–73421.

Xue, Z.; Cai, Q.; Liu, S.; Zheng, D.; Jiang, P.; Gai, K.; and An, B. 2024. State regularized policy optimization on data with dynamics shift. *Advances in neural information processing systems*, 36.

Zhu, S.; Kimmel, A.; Bekris, K. E.; and Boularias, A. 2018. Fast model identification via physics engines for data-efficient policy search. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 3249–3256.

Missing Theoretical Proofs

In this section, we provide formal proofs omitted from the main paper.

Useful Lemmas

Lemma .2. *[Extended telescoping lemma.] Let $\mathcal{M}_1 = (\mathcal{S}, \mathcal{A}, P_1, r, \gamma)$ and $\mathcal{M}_2 = (\mathcal{S}, \mathcal{A}, P_2, r, \gamma)$ be two MDPs with different transition dynamics P_1 and P_2. Given two policies π_1 and π_2, we have the following:*

$$J^{P_1}(\pi_1) - J^{P_2}(\pi_2) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi_1}_{P_1}}\left[\mathbb{E}_{s'\sim P_1, a'\sim\pi_1}\left[Q^{\pi_2}_{P_2}(s',a')\right] - \mathbb{E}_{s'\sim P_2, a'\sim\pi_2}\left[Q^{\pi_2}_{P_2}(s',a')\right]\right]. \tag{12}$$

This lemma is Lemma C.2 in (Xu et al. 2023) and is the extended version of the telescoping lemma in (Luo et al. 2018). Here, we provide the brief proof.

Proof. We define W_j as the expected return when follow the policy π_1 in $\mathcal{M}_1$ for the first j steps, then switching to the policy π_1 and $\mathcal{M}_2$ for the remainder. We have:

$$W_j := \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{\substack{t<j: s_t,a_t\sim P_1,\pi_1 \\ t\geq j: s_t,a_t\sim P_2,\pi_2}}\left[r\left(s_t,a_t\right)\right] = \mathbb{E}_{\substack{t<j: s_t,a_t\sim P_1,\pi_1 \\ t\geq j: s_t,a_t\sim P_2,\pi_2}}\left[\sum_{t=0}^{\infty}\gamma^t r\left(s_t,a_t\right)\right]. \tag{13}$$

Note that we have:

$$\begin{aligned} W_0 &= \mathbb{E}_{s,a\sim d^{\pi_2}_{P_2}}\left[r(s_t,a_t)\right] = J^{P_2}(\pi_2), \\ \text{and } W_\infty &= \mathbb{E}_{s,a\sim d^{\pi_1}_{P_1}}\left[r(s_t,a_t)\right] = J^{P_1}(\pi_1). \end{aligned} \tag{14}$$

We then have the following:

$$J^{P_1}(\pi_1) - J^{P_2}(\pi_2) = \sum_{j=0}^{\infty}(W_{j+1} - W_j). \tag{15}$$

Write W_j and W_{j+1} as the following:

$$\begin{aligned} W_j &= R_j + \mathbb{E}_{s_j,a_j\sim P_1,\pi_1}\left[\mathbb{E}_{s_{j+1},a_{j+1}\sim P_2,\pi_2}\left[\gamma^{j+1}Q^{\pi_2}_{P_2}(s_{j+1},a_{j+1})\right]\right], \\ W_{j+1} &= R_j + \mathbb{E}_{s_j,a_j\sim P_1,\pi_1}\left[\mathbb{E}_{s_{j+1},a_{j+1}\sim P_1,\pi_1}\left[\gamma^{j+1}Q^{\pi_2}_{P_2}(s_{j+1},a_{j+1})\right]\right] \end{aligned} \tag{16}$$

Thus, we have:

$$\begin{aligned} J^{P_1}(\pi_1) - J^{P_2}(\pi_2) &= \sum_{j=0}^{\infty}(W_{j+1} - W_j) \\ &= \sum_{j=0}^{\infty}\gamma^{j+1}\mathbb{E}_{s_j,a_j\sim P_1,\pi_1}\left[\mathbb{E}_{s_{j+1},a_{j+1}\sim P_1,\pi_1}\left[Q^{\pi_2}_{P_2}(s_{j+1},a_{j+1})\right] - \mathbb{E}_{s_{j+1},a_{j+1}\sim P_2,\pi_2}\left[Q^{\pi_2}_{P_2}(s_{j+1},a_{j+1})\right]\right] \\ &= \frac{\gamma}{1-\gamma}\mathbb{E}_{s_j,a_j\sim P_1,\pi_1}\left[\mathbb{E}_{s_{j+1},a_{j+1}\sim P_1,\pi_1}\left[Q^{\pi_2}_{P_2}(s_{j+1},a_{j+1})\right] - \mathbb{E}_{s_{j+1},a_{j+1}\sim P_2,\pi_2}\left[Q^{\pi_2}_{P_2}(s_{j+1},a_{j+1})\right]\right], \end{aligned} \tag{17}$$

which concludes the proof.

Lemma .3. *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$ be the MDP. Given two policies π_1 and π_2, we have the following:*

$$J^{P}(\pi_1) - J^{P}(\pi_2) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi_1,P},s'\sim P}\left[\mathbb{E}_{a'\sim\pi_1}\left[Q^{\pi_2}_{P}(s',a')\right] - \mathbb{E}_{a'\sim\pi_2}\left[Q^{\pi_2}_{P}(s',a')\right]\right]. \tag{18}$$

This is Lemma B.3 in (Lyu et al.). We provide the brief proof here.

Proof. Using Lemma .2 and replace P_1 and P_2 by P, we have the following:

$$J^{P}(\pi_1) - J^{P}(\pi_2) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s_j,a_j\sim P,\pi_1}\left[\mathbb{E}_{s_{j+1},a_{j+1}\sim P,\pi_1}\left[Q^{\pi_2}_{P}(s_{j+1},a_{j+1})\right] - \mathbb{E}_{s_{j+1},a_{j+1}\sim P,\pi_2}\left[Q^{\pi_2}_{P}(s_{j+1},a_{j+1})\right]\right], \tag{19}$$

which completes the proof.

Proof of Theorems 0.1

Theorem .4 (Performance Bound)**.** *Denote D_{src} as the offline source dataset from source domain $\mathcal{M}_{src}$ and D_{tar} as the offline target dataset from target domain $\mathcal{M}_{tar}$. Let the empirical policy in the offline target dataset D_{tar} be $\pi_{D_{tar}} = \frac{\sum_{D_{tar}} \mathbb{1}(s,a)}{\sum_{D_{tar}} \mathbb{1}(s)}$. Given a policy π, we have the following:*

$$
\begin{aligned}
&J^{P_{tar}}(\pi) - J^{\widehat{P}_{src}}(\pi) \\
&\geq -\frac{2r_{max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi}_{P_{tar}},s'\sim P_{tar}}\left[D_{TV}\left(\pi(.|s')||\pi_{D_{tar}}(.|s')\right)\right] - \frac{2r_{max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi_{D_{tar}}}_{P_{tar}},s'\sim P_{tar}}\left[D_{TV}\left(\pi(.|s')||\pi_{D_{tar}}(.|s')\right)\right] \\
&- \frac{2r_{max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi_{D_{tar}}}_{P_{tar}}}\left[\sqrt{1/2 D_{KL}(P_{tar}(.|s,a)||\widehat{P}_{src}(.|s,a))}\right],
\end{aligned}
\tag{20}
$$

Proof. We start by converting the performance difference in the following form:

$$
J^{P_{tar}}(\pi) - J^{\widehat{P}_{src}}(\pi) = \underbrace{\left(J^{P_{tar}}(\pi) - J^{P_{tar}}(\pi_{D_{tar}})\right)}_{\text{(a)}} + \underbrace{\left(J^{P_{tar}}(\pi_{D_{tar}}) - J^{\widehat{P}_{src}}(\pi)\right)}_{(b)}. \tag{21}
$$

For term (a) in the RHS, based on Lemma .3, we have:

$$
\begin{aligned}
J^{P_{tar}}(\pi) - J^{P_{tar}}(\pi_{D_{tar}}) &= \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{P_{tar}},s'\sim P_{tar}}\left[\mathbb{E}_{a'\sim\pi}\left[Q^{\pi_{D_{tar}}}_{P_{tar}}(s',a')\right] - \mathbb{E}_{a'\sim\pi_{D_{tar}}}\left[Q^{\pi_{D_{tar}}}_{P_{tar}}(s',a')\right]\right] \\
&\geq -\frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{P_{tar}},s'\sim P_{tar}}\left|\mathbb{E}_{a'\sim\pi}\left[Q^{\pi_{D_{tar}}}_{P_{tar}}(s',a')\right] - \mathbb{E}_{a'\sim\pi_{D_{tar}}}\left[Q^{\pi_{D_{tar}}}_{P_{tar}}(s',a')\right]\right| \\
&= -\frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{P_{tar}},s'\sim P_{tar}}\left|\sum_{a'\in\mathcal{A}}\left(\pi(a'|s') - \pi_{D_{tar}}(a'|s')\right)Q^{\pi_{D_{tar}}}_{P_{tar}}(s',a')\right| \\
&\geq -\frac{r_{\max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi}_{P_{tar}},s'\sim P_{tar}}\left|\sum_{a'\in\mathcal{A}}\left(\pi(a'|s') - \pi_{D_{tar}}(a'|s')\right)\right| \\
&= -\frac{2r_{\max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi}_{P_{tar}},s'\sim P_{tar}}\left[D_{TV}\left(\pi(.|s')||\pi_{D_{tar}}(.|s')\right)\right].
\end{aligned}
\tag{22}
$$

For term (b) in the RHS, based on Lemma .2, we have:

$$
\begin{aligned}
&J^{P_{tar}}(\pi_{D_{tar}}) - J^{\widehat{P}_{src}}(\pi) \\
&= \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi_{D_{tar}}}_{P_{tar}}}\left[\mathbb{E}_{s'\sim P_{tar},a'\sim\pi_{D_{tar}}}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right] - \mathbb{E}_{s'\sim\widehat{P}_{src},a'\sim\pi}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right]\right] \\
&= \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi_{D_{tar}}}_{P_{tar}}}\left[\underbrace{\left(\mathbb{E}_{s'\sim P_{tar},a'\sim\pi_{D_{tar}}}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right] - \mathbb{E}_{s'\sim P_{tar},a'\sim\pi}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right]\right)}_{(c)}\right. \\
&\left. + \underbrace{\left(\mathbb{E}_{s'\sim P_{tar},a'\sim\pi}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right] - \mathbb{E}_{s'\sim\widehat{P}_{src},a'\sim\pi}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right]\right)}_{(d)}\right].
\end{aligned}
\tag{23}
$$

We first bound term (c) as follows:

$$
\begin{aligned}
&\mathbb{E}_{s'\sim P_{tar},a'\sim\pi_{D_{tar}}}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right] - \mathbb{E}_{s'\sim P_{tar},a'\sim\pi}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right] \\
&= \mathbb{E}_{s'\sim P_{tar}}\left[\sum_{a'\in\mathcal{A}}\left(\pi_{D_{tar}}(a'|s') - \pi(a'|s')\right)Q^{\pi}_{\widehat{P}_{src}}(s',a')\right] \\
&\geq -\mathbb{E}_{s'\sim P_{tar}}\left[\sum_{a'\in\mathcal{A}}\left|\pi_{D_{tar}}(a'|s') - \pi(a'|s')\right|\left|Q^{\pi}_{\widehat{P}_{src}}(s',a')\right|\right] \\
&\geq -\frac{2r_{\max}}{1-\gamma}\mathbb{E}_{s'\sim P_{tar}}\left[D_{TV}\left(\pi(.|s')||\pi_{D_{tar}}(.|s')\right)\right].
\end{aligned}
\tag{24}
$$

For term (d), we have:

$$\begin{aligned}
&\mathbb{E}_{s'\sim P_{tar},a'\sim\pi}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right]-\mathbb{E}_{s'\sim\widehat{P}_{src},a'\sim\pi}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right]\\
&=\mathbb{E}_{a'\sim\pi}\left[\sum_{s'\in\mathcal{S}}\left(P_{tar}(s'|s,a)-\widehat{P}_{src}(s'|s,a)\right)Q^{\pi}_{\widehat{P}_{src}}(s',a')\right]\\
&\geq-\frac{2r_{\max}}{1-\gamma}D_{TV}(P_{tar}(.|s,a)||\widehat{P}_{src}(.|s,a))\\
&\geq-\frac{2r_{\max}}{1-\gamma}\sqrt{1/2D_{KL}(P_{tar}(.|s,a)||\widehat{P}_{src}(.|s,a))}.
\end{aligned}\tag{25}$$

Thus we can bound term (b) as follows:

$$\begin{aligned}
&J^{P_{tar}}(\pi_{D_{tar}})-J^{\widehat{P}_{src}}(\pi)\\
&=\frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi_{D_{tar}}}_{P_{tar}}}\Bigg[\underbrace{\left(\mathbb{E}_{s'\sim P_{tar},a'\sim\pi_{D_{tar}}}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right]-\mathbb{E}_{s'\sim P_{tar},a'\sim\pi}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right]\right)}_{(c)}\\
&+\underbrace{\left(\mathbb{E}_{s'\sim P_{tar},a'\sim\pi}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right]-\mathbb{E}_{s'\sim\widehat{P}_{src},a'\sim\pi}\left[Q^{\pi}_{\widehat{P}_{src}}(s',a')\right]\right)}_{(d)}\Bigg]\\
&\geq-\frac{2r_{\max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi_{D_{tar}}}_{P_{tar}}}\left[\mathbb{E}_{s'\sim P_{tar}}\left[D_{TV}\left(\pi(.|s')||\pi_{D_{tar}}(.|s')\right)\right]+\sqrt{1/2D_{KL}(P_{tar}(.|s,a)||\widehat{P}_{src}(.|s,a))}\right].
\end{aligned}\tag{26}$$

Thus, we have:

$$\begin{aligned}
&J^{P_{tar}}(\pi)-J^{\widehat{P}_{src}}(\pi)\\
&\geq-\frac{2r_{\max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi}_{P_{tar}},s'\sim P_{tar}}\left[D_{TV}\left(\pi(.|s')||\pi_{D_{tar}}(.|s')\right)\right]\\
&-\frac{2r_{\max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi_{D_{tar}}}_{P_{tar}}}\left[\mathbb{E}_{s'\sim P_{tar}}\left[D_{TV}\left(\pi(.|s')||\pi_{D_{tar}}(.|s')\right)\right]+\sqrt{1/2D_{KL}(P_{tar}(.|s,a)||\widehat{P}_{src}(.|s,a))}\right]\\
&=-\frac{2r_{\max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi}_{P_{tar}},s'\sim P_{tar}}\left[D_{TV}\left(\pi(.|s')||\pi_{D_{tar}}(.|s')\right)\right]-\frac{2r_{\max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi_{D_{tar}}}_{P_{tar}},s'\sim P_{tar}}\left[D_{TV}\left(\pi(.|s')||\pi_{D_{tar}}(.|s')\right)\right]\\
&-\frac{2r_{\max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi_{D_{tar}}}_{P_{tar}}}\left[\sqrt{1/2D_{KL}(P_{tar}(.|s,a)||\widehat{P}_{src}(.|s,a))}\right],
\end{aligned}\tag{27}$$

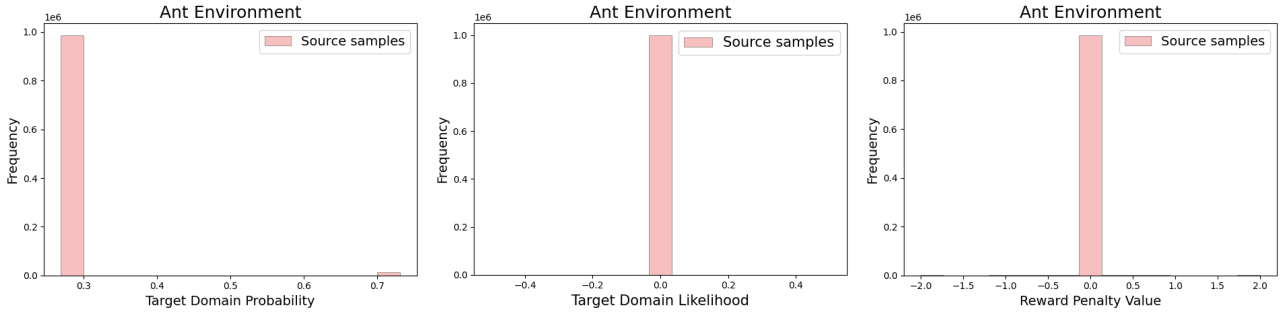
which completes the proof.

Cross-Domain Offline RL with Limited Target Samples

In this section, we provide detailed discussions about the challenge of the Cross-domain Offline RL with limited target data setting.

Datasets Imbalanced Problem

In cross-domain offline RL, a critical challenge is effectively quantifying the domain gap between the source and target domains. Accurate measurement of this gap enables subsequent strategies to leverage source samples to enhance policy learning for the target domain. Prior works have proposed various approaches for this purpose, including training domain classifiers (Liu, Hongyin, and Wang 2022), or estimating target models (Liu et al. 2024). However, these methods often rely on training parametric models such as neural networks, which are prone to overfitting in scenarios with limited target data and significant dataset imbalances. For instance, limited target samples can cause classifiers to develop biases, favoring data that appears more frequently while failing to capture the true dynamics gap. To investigate potential overfitting issues, we train the domain classifiers from DARA (Liu, Hongyin, and Wang 2022) and the CVAE dynamics model from BOSA (Liu et al. 2024) in Ant environment with *medium* source and *expert* target datasets under the gravity shift. We then evaluate the predicted target likelihood of these models on samples from the source dataset D_{src}. Figures 4a and 4b show the histograms of the predicted target likelihoods obtained from the learned domain classifiers and the dynamics model. Both exhibit significant overfitting, as they predominantly output uniform predictions across all source samples. Additionally, the reward penalties computed via DARA's domain classifiers are



(a) Predicted target likelihood via classifiers. (b) Predicted Target Likelihood via CVAE. (c) Reward penalties in DARA.

Figure 4: (a) Histogram of the predicted target likelihood of the domain classifiers in DARA for the source sample in the source dataset. (b) The histogram of the predicted target likelihood of the CVAE dynamics target model proposed in BOSA. (c) The histogram of the reward penalty values computed using domain classifiers in DARA.

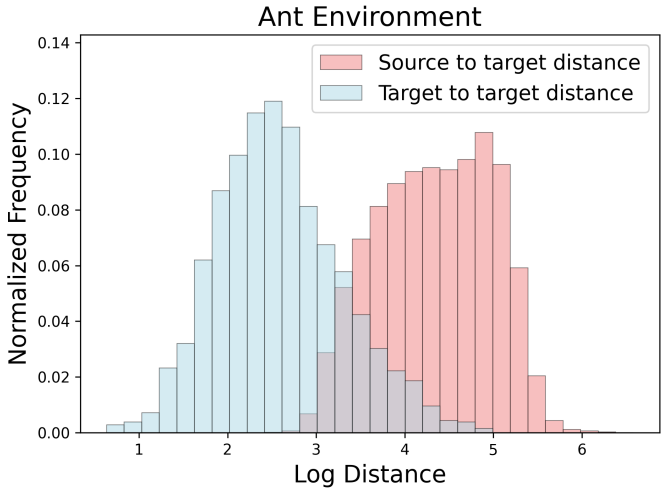


Figure 5: Nearest Neighbor Distance histograms. Blue shows the distance of source samples to their nearest target samples. Pink shows the distance of target samples to their nearest target samples.

near zero, providing little to no useful signal for policy learning, as shown in Figure 4c. In addition to the overfitting issues, the imbalance between source and target data can bias the learned policy, potentially causing it to overfit the source data during training.

Partial Overlapping between Domains

Leveraging source samples is critical for enhancing policy learning and improving sample efficiency in cross-domain offline RL. However, discrepancies in dynamics between the source and target domains, along with potential policy shifts during data collection processes, not all source data is close to the target data. To investigate how the source data is close to the target data, we conduct experiments in the Ant environment with *medium* source and *expert* target datasets under gravity shift. Specifically, we compute the distances from each source sample to its nearest neighbor in the target dataset D_{tar} and the distances from each target sample to its nearest neighbor within D_{tar}. The histograms in Figure 5 reveal only partial overlap between two histograms, highlighting that only a subset of the source dataset is beneficial for target policy learning. Previous methods (Liu et al. 2024; Wen et al. 2024) addressed this by filtering out source samples far from the target domain. However, this approach reduces the number of usable source samples, ultimately impacting the sample efficiency of the algorithms. This raises an important question: "Can we further improve sample efficiency in cross-domain offline RL by generating additional source data that closely aligns with the target domain?"

Environment Setting

This section provides a detailed description of the environment settings we use in our experiments.

Source Environments

We conduct our experiments on four Mujoco environments: Ant, Halfcheetah, Hopper, and Walker from the Openai-Gym libraries (Todorov, Erez, and Tassa 2012; Brockman et al. 2016). For our offline source, we use datasets from the D4RL benchmark (Fu et al. 2020), which provides three types of datasets for each environment: medium, medium-replay, and medium-expert. Specifically, the medium datasets consist of experiences collected from an SAC policy that was early-stopped after 1 million steps. The medium-replay datasets contain the replay buffer of a policy trained to achieve the medium agent's performance. The medium-expert datasets are constructed by combining medium and expert data in a 50-50 ratio. Notably, the sizes of these datasets vary significantly; for example, the medium datasets contain 1 million samples, while the medium-replay datasets may contain as few as 100,000 samples.

Target Environments

In this paper, we consider the gravity shift and kinematic shift as the dynamics shift between the source domain and the target domain. Gravity shift refers to differences in gravitational forces acting on the source and target robots. To simulate this shift, we modify the gravitational acceleration parameter in the environment while keeping its direction consistent with the default configuration in MuJoCo. Specifically, we retain the default gravitational acceleration value for the source domain and set the value in the target domain to be half of that in the source domain. On the other hand, when referring to kinematic shift, we are referring to altering some joints of the simulated robots. We provide the XML modifications in Section and Section , along with the XML files of the agents in our supplementary materials.

The target datasets are collected following a procedure similar to D4RL. We consider three dataset quality levels: *expert*, *medium*, and *medium-expert*. Each dataset consists of five trajectories, totaling 5,000 transitions, to reflect the limited target data setting. The *expert* datasets are collected using the last checkpoint of the trained policy, while the *medium* datasets use a policy performing at approximately 1/2 or 1/3 of the expert policy's performance. The *medium-expert* datasets are constructed by combining two trajectories from the *medium* dataset and three from the *expert* dataset. Finally, we use the target datasets collected by (Anonymous 2025) in our experiments as we consider the same target domain setting as them.

Gravity shift The XML files of Ant, Halfcheetah, Hopper, and Walker are modified as follows:

```
<option gravity="0 0 -4.905" timestep="0.01"/>
```

Kinematic shift The kinematic shifts of the robot occurred at different parts, detailed below:

- **Ant-kinematic**: The rotation angles of the joints on the hips of two legs in the ant robot are adjusted from $[-30, 30]$ to $[-0.3, 0.3]$.

```
# hip joints of the front legs
<joint axis="0 0 1" name="hip_1" pos="0.0 0.0 0.0" range="-0.30 0.30" type="hinge"/
    >
<joint axis="0 0 1" name="hip_2" pos="0.0 0.0 0.0" range="-0.30 0.30" type="hinge"/
    >
```

- **HalfCheetah-kinematic**: The rotation angle of the joint on the thigh of the robot's back leg is adjusted from $[-0.52, 1.05]$ to $[-0.0052, 0.0105]$.

```
# back thigh
<joint axis="0 1 0" damping="6" name="bthigh" pos="0 0 0" range="-.0052 .0105"
    stiffness="240" type="hinge"/>
```

- **Hopper-kinematic**: The rotation angle of the head joint is adjusted from $[150, 0]$ to $[0.15, 0]$ and the rotation angle of the joints on the robot's foot is modified from $[-45, 45]$ to $[-18, 18]$.

```
# head joint
<joint axis="0 -1 0" name="thigh_joint" pos="0 0 1.05" range="-0.150 0" type="hinge
    "/>
# foot joint
<joint axis="0 -1 0" name="foot_joint" pos="0 0 0.1" range="-18 18" type="hinge"/>
```

- **Walker-Kinematic**: The rotation angle of the foot joint on the robot's right leg is modified from $[-45, 45]$ to $[-0.45, 0.45]$.

```
# right foot
<joint axis="0 -1 0" name="foot_joint" pos="0 0 0.1" range="-0.45 0.45" type="hinge
    "/>
```

Evaluation Metric

To evaluate the adaptation performance of the learned policy in the target domain, we use the *normalized score* (NS) metric as similar in (Fu et al. 2020; Lyu et al. 2024b). The normalized score of a given policy in the target domain is computed as follows:

$$NS = \frac{J - J_r}{J_e - J_r} \times 100, \tag{28}$$

where J, J_e, J_r denotes the return of the given, expert, and random policies in the target domain. We list the reference scores in Table 5 and provide the minimum return, the maximum return, and the average return of the trajectories in each target dataset in Table 6.

Table 5: The reference scores of the Mujoco datasets under the gravity shifts. J_r denotes the performance of the random policy in the target domain, and J_e denotes the performance of the expert policy in the target domain. The reference scores are used to compute the *normalized score* for evaluating the performance of the learned policies.

Environment	Dynamic Shift Type	Reference score J_r	Reference score J_e
Ant	Gravity	-325.6	4317.065
Halfcheetah	Gravity	-280.18	9509.15
Hopper	Gravity	-26.336	3234.3
Walker	Gravity	10.079	5194.713
Ant	Kinematic	-325.6	5122.57
Halfcheetah	Kinematic	-280.18	7065.03
Hopper	Kinematic	-26.336	2842.73
Walker	Kinematic	10.079	3257.51

Table 6: Trajectories return information of the target datasets. We report the trajectory return statistics of the target domain datasets, including the minimum return (min return), maximum return (max return), and average return.

Task Name	Dynamics shift	Dataset type	Min return	Max return	Average return
Ant	Gravity	medium	377.10	3247.66	2314.45
Ant	Gravity	medium-expert	377.10	4511.55	2131.79
Ant	Gravity	expert	335.28	4584.53	3365.35
Ant	Kinematic	medium	2826.00	3111.98	3017.82
Ant	Kinematic	medium-expert	2826.00	5122.58	4240.99
Ant	Kinematic	expert	5009.82	5122.57	5072.50
Halfcheetah	Gravity	medium	4179.82	4383.32	4296.27
Halfcheetah	Gravity	medium-expert	4342.78	8243.03	6567.94
Halfcheetah	Gravity	expert	7846.18	8339.18	8131.54
Halfcheetah	Kinematic	medium	2709.52	2782.61	2755.50
Halfcheetah	Kinematic	medium-expert	2709.52	7065.04	5298.61
Halfcheetah	Kinematic	expert	6951.27	7065.04	6998.93
Hopper	Gravity	medium	1784.88	2885.13	2367.66
Hopper	Gravity	medium-expert	2416.82	4143.63	3297.79
Hopper	Gravity	expert	3745.59	4186.19	4051.07
Hopper	Kinematic	medium	1849.06	1886.89	1870.16
Hopper	Kinematic	medium-expert	1868.20	2842.17	2452.67
Hopper	Kinematic	expert	2840.97	2842.73	2841.83
Walker	Gravity	medium	2421.98	3444.63	2897.85
Walker	Gravity	medium-expert	3144.32	5166.62	4415.11
Walker	Gravity	expert	5159.51	5219.14	5174.51
Walker	Kinematic	medium	1415.69	2223.17	2026.49
Walker	Kinematic	medium-expert	1415.69	3257.51	2442.82
Walker	Kinematic	expert	2874.92	3257.51	3077.19

Algorithm's Implementations

In this section, we provide a details implementation of our method, DnD, and the baselines we use in the experiments. Specifically, we use the report We report the hyperparameter in Table 7. Our implementation is based on ODRL (Lyu et al. 2024b), Synther (Lu et al. 2023), and CleanDiffuser (Dong et al. 2024).

DnD

In our implementation, we use the FAISS library (Douze et al. 2024; Johnson, Douze, and Jégou 2019) to compute the k-NN estimation. We compute the k-NN based score for all source samples in the source dataset D_{src} at the beginning. The computation

time is only within 1 minute for 1 million source samples and 5 thousand target samples.

After computing the score for each source sample, we train our diffusion model using the offline source dataset with the corresponding scores as the conditional context y. We use the design of EDM (Karras et al. 2022) for our diffusion model, it has demonstrated its performance in previous offline RL works (Lu et al. 2023). Specifically, we employ a fixed noise schedule $\sigma_{\max} = \sigma^T > \cdots > \sigma^1 > \sigma^0 = 0$ and diffuse clean data samples using the transition $q(x^t|x_{src,i}) = \mathcal{N}\left(x_{src,i}, (\sigma^t)^2\mathrm{I}\right)$. Then, we train a denoiser $G_\theta(\cdot)$ to directly predict the clean samples using classifier-free training (Ho and Salimans 2022):

$$\min_\theta \mathbb{E}\left[\left\|x_{src,i} - G_\theta\left(x_{src,i} + \sigma^t\epsilon, m \cdot \rho_k(x_{src,i}), \sigma^t\right)\right\|^2\right] \tag{29}$$

where $x_{src,i} \sim D_{src}$, $\epsilon \sim \mathcal{N}(0, \mathrm{I})$, and $t \sim \mathcal{U}[0, T]$. The score $\rho_k(x_{src,i})$ is randomly masked during training via the null token m, which is sampled from the Bernoulli distribution with $p\left(m = \emptyset\right) = 0.25$. We then used the trained diffusion model to generate source samples that are close to the target domain by setting the value of the conditional context y to be higher than the top $\kappa\%$ of the source sample scores in the source dataset D_{src}. Specifically, we uniformly sample a value χ from the range $[\kappa, 100]$ and set y to the χ-th quantile of the source dataset's score distribution. The number of generated samples is 1 million and fixed for all environments. Finally, the generated samples are combined with the source dataset for later policy learning. We use the IQL (Kostrikov, Nair, and Levine 2022) as the backbone for DnD.

We summarize our proposed method in Algorithm 1. Based on the analysis in Section , to reduce the dynamics gaps and tighten the bound in Theorem 0.1, we select the source samples that have the highest scores during training to update the Q value function. Then, we further use the score as the weighting value for the source data during the training of the Q value function. This approach ensures the policy adaptively emphasizes source samples that closely align with the target domain, improving adaptation performance.

As we mentioned, the imbalanced dataset could bias the policy to the source samples. Thus, we employ additional policy regularization to ensure the learned policy is close to the support areas of the target dataset. Similar to (Wu et al. 2022), we learn a conditional VAE, denoted as $\hat{\pi}^b_{tar}(a|s)$, to model the behavior target policy $\pi^b_{tar}(a|s)$. Thus, we optimize the policy with the loss function as follows:

$$\mathcal{L}^{\text{reg}}_\pi = \mathcal{L}_\pi - \lambda\mathbb{E}_{s\sim D_{src}\bigcup D_{tar}}\left[\log\hat{\pi}^b_{tar}\left(\pi(.|s)|s\right)\right], \tag{30}$$

where $\mathcal{L}_\pi$ is the policy loss of the offline RL method, λ is the coefficient controlling the strength of the additional policy regularization. In our implementation, we use the default config for CVAE as presented in Wu et al. (2022) .

DARA

DARA (Liu, Hongyin, and Wang 2022) use the learned domain classifiers to compute the reward penalty Δ_r and correct the source samples using the following:

$$\hat{r} = r - \alpha\Delta_r, \tag{31}$$

where α is the penalty coefficient that controls the strength of the reward penalty term. The reward penalty is clipped to $[-10, 10]$ following the original paper (Liu, Hongyin, and Wang 2022). We report the results of DARA with IQL as the backbone s the fair comparison.

BOSA

BOSA (Liu et al. 2024) handles the dynamics shift problems using a supported policy and value optimization. The value function (critic) in BOSA is learned with the following objective:

$$\min_{Q_\phi}\mathcal{L}_{\text{mix}}\left(Q_\phi\right) := \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{D}_{\text{mix}},\mathbf{a}'\sim\pi_\theta(\mathbf{a}'|\mathbf{s}')}\left[\delta\left(Q_\phi\right)\cdot\mathbb{1}\left(\hat{P}_{\text{target}}\left(\mathbf{s}'\mid\mathbf{s},\mathbf{a}\right) > \epsilon'_{\text{th}}\right)\right] + \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}_{\text{source}}}\left[Q_\phi(\mathbf{s},\mathbf{a})\right], \tag{32}$$

where $\delta\left(Q_\phi\right) = \left(Q_\phi(s,a) - r - Q_{\bar{\phi}}(s,a)\right)^2$, $\mathbb{1}$ is the indicator function, $\hat{P}_{tar}$ is the learned target transition dynamics, ϵ_{th} is the selection threshold. CVAE is used to model the target dynamics model.

IGDF

IGDF (Wen et al. 2024) measure the dynamics mismatch between two domains via the mutual information estimated via contrastive learning. IGDF uses a score function h, using the target dataset as the positive sample and the source dataset as the negative sample. Then, based on the score function, IGDF updates the value function using the following objective:

$$\mathcal{L}_Q = \frac{1}{2}\mathbb{E}_{D_{\text{tar}}}\left[\left(Q_\theta - \mathcal{T}Q_\theta\right)^2\right] + \frac{1}{2}\alpha\cdot h\left(s,a,s'\right)\mathbb{E}_{(s,a,s')\sim D_{\text{src}}}\left[\mathbb{1}\left(h\left(s,a,s'\right) > h_{\xi\%}\right)\left(Q_\theta - \mathcal{T}Q_\theta\right)^2\right], \tag{33}$$

where α is the weighting coefficient for the TD in the source data, and ξ is the data selection ratio.

Table 7: Hyperparameter setup for DnD and the baselines.

	Hyperparams	Value
Shared	Actor network	(256,256)
	Critic network	(256,256)
	Learning rate	3e-4
	Discounted factor	0.99
	Buffer size	1e6
	Activation	ReLU
	Target update coefficient	5e-3
	Batch size for source and target	128
	Temperature coefficient	0.2
	Max log std	2
	Min log std	-20
DARA	Domain classifiers network	(256,256)
	Reward penalty coefficient α	0.1
BOSA	Policy regularization coefficient λ_π	0.1
	Transition coefficient $\lambda_{transition}$	0.1
	Threshold parameter ϵ, ϵ'	log(0.01)
	Value weight	0.1
IGDF	Representation dimension	{16,64}
	Contrastive encoder network	(256,256)
	Encoder training steps	7000
	Importance coefficient	1.0
	Data selection ratio $\xi\%$	75%
SRPO	Discriminator network	(256, 256)
	Data selection ratio	0.5
	Reward coefficient	{0.1, 0.3}
OTDF	CVAE training steps	10000
	CVAE learning rate	1e-3
	Cost function	cosin
	Data filtering ratio	80%
	Policy coefficient	{0.1, 0.5}
DnD	kth nearest neighbor	5
	Data selection ratio	50%
	Policy regularization coefficient	0.1
	Score threshold for guided sampling	90%

IQL

IQL (Kostrikov, Nair, and Levine 2022) is a state-of-the-art offline RL algorithm. It updates the state-value function and state-action value function via the following objectives:

$$\begin{aligned} \mathcal{L}_V &= \mathbb{E}_{(s,a)\sim\mathcal{D}_{\mathrm{src}}\cup\mathcal{D}_{\mathrm{tar}}}\left[L_2^\tau\left(Q_{\theta'}(s,a)-V_\psi(s)\right)\right], \\ \mathcal{L}_Q &= \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\mathrm{src}}\cup\mathcal{D}_{\mathrm{tar}}}\left[\left(r(s,a)+\gamma V_\psi\left(s'\right)-Q_\theta(s,a)\right)^2\right], \end{aligned} \tag{34}$$

where $L_2^\tau(u) = |\tau - \mathbb{1}(u<0)|u^2$. The policy is learned using the advantage-weighted behavior cloning objective:

$$\mathcal{L}_\pi = \mathbb{E}_{D_{src}\bigcup D_{tar}}\left[\exp\left(\beta\times A(s,a)\right)\log\pi(a|s)\right], \tag{35}$$

where $A(s,a) = Q(s,a) - V(s,a)$, and β is the inverse temperature coefficient.

SRPO

SRPO (Xue et al. 2024) proposes optimizing the policy by solving the following constrained optimization problem:

$$\max_{\pi} \mathbb{E}_{s_t,a_t\sim\tau_\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \quad \text{s.t.} \quad D_{KL}(d_\pi(\cdot)\parallel\zeta(\cdot))<\epsilon, \tag{36}$$

where τ_π is the trajectory induced by policy π, $d_\pi(\cdot)$ is the stationary state distribution of policy π, and $\zeta(\cdot)$ represents the optimal state distribution under different environment dynamics.

This problem can be reformulated into an unconstrained optimization problem using Lagrange multipliers, where the logarithm of the probability density ratio, $\lambda \log \frac{\zeta(s_t)}{d_\pi(s_t)}$, is added to the standard reward function. Based on this formulation, SRPO samples a batch of size N from two offline datasets, D_{src} and D_{tar}, and ranks the transitions based on state values. A proportion ρN (in the paper $\rho = 0.5$ is used) of samples with high state values is labeled as real data, while the remaining samples are labeled as fake data.

A discriminator $D_\delta(\cdot)$ is then trained to distinguish between these samples, and the rewards are modified as follows:

$$\hat{r}_{\text{SRPO}} = r + \lambda \cdot \frac{D_\delta(s)}{1 - D_\delta(s)}, \tag{37}$$

where λ is the reward coefficient.

OTDF

OTDF (Anonymous 2025) attempts to solve the problem of estimating domain gaps in the limited target data setting with the optimal transport. Specifically, they use cousin distance as the cost function for solving optimal transport problems between the datasets of two domains. After obtaining the optimal coupling $\mu*$, they measure the deviation of the source domain to the target domain as follows:

$$d(u_t) = -\sum_{t'=1}^{|D_{tar}|} C(u_t, u_{t'})\mu*_{t,t'}, u_t = (s^t_{src}, a^t_{src}, (s'_{src})^t) \sim D_{src}, \tag{38}$$

where C is the cost function. They then use $d(u_t)$ to select good source data for policy training via the source data filtering. Additionally, they also employ a policy regularization to ensure the learned policy is close to the support of the target dataset.

Hyperparamters

We adopt the hyperparameters reported in Anonymous (2025) for baseline methods. For DnD, we set $k = 5$ for k-NN estimation, use a 50% data selection ratio, a policy regularization coefficient of 0.1, and a 90% score threshold for guided sampling from the diffusion model. We note that without bothering the hyperparameter tuning, DnD achieves strong performance across diverse tasks with a single set of hyperparameters.

Computing Infrastructure

We use Python 3.9, Gym 0.23.1, Mujoco 2.3.2 and D4RL 1.1. All experiments are conducted on a Ubuntu 22.04 server with CUDA version 12.2. We report the computing infrastructure that we use to run our experiments in Table 8.

CPU	Number of CPU Cores	GPU	VRAM	RAM
Intel(R) Xeon(R) Gold 6248 CPU	10	V100	32 GB	377 GB

Table 8: Computing infrastructure.

More Experimental Results

In this section, we present additional experimental results omitted from the main text due to space constraints. We use the same number reported in Anonymous (2025) since we consider similar gravity and kinematic shift settings. We present the performance comparison between DnD and the other baselines under kinematic shift and report additional results on the impact of the guided-diffusion model. Furthermore, we conduct ablation studies on DnD's hyperparameters to provide deeper insights into its behavior and effectiveness.

Missing Results on Kinematic Shift Tasks

We present the performance comparison between DnD and other methods under kinematic shifts in Table 9. We observe DnD excels in **29** out of 36 tasks, surpassing IQL performance by **59.4%**, and achieves a total normalized score of **1902.2**, compared to 1547.6 of the second best method OTDF. Besides that, we observe the cross-domain RL methods that involve training neural networks to estimate the domain gaps fail to bring performance improvement compared to IQL. These results validate the effectiveness of DnD in cross-domain offline RL with limited target data.

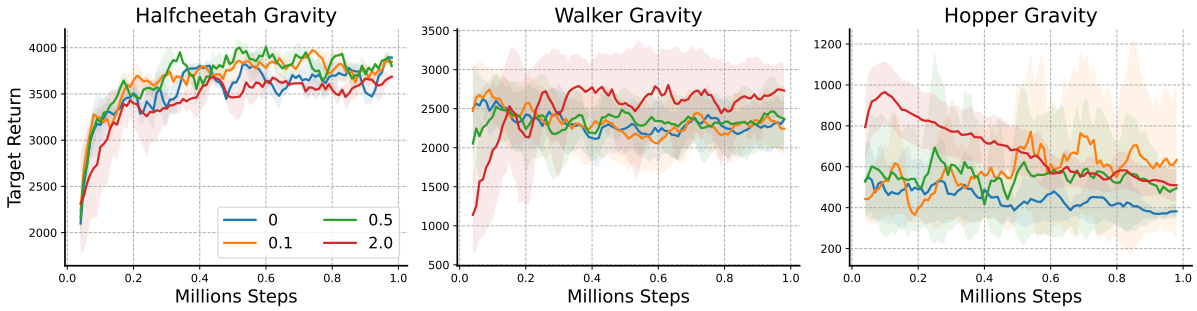


Figure 6: Ablation study on the effect of policy regularization weight λ. The solid curves are the average target returns, and the shaded areas represent the standard deviation over 5 random seeds.

Effect of Guided-Diffusion Model

In this section, we provide additional experiment results to answer question 3. Specifically, we compare DnD against three variants: (1) DnD without upsampling via the diffusion model (*w/o diffusion*), (2) DnD where the target dataset is upsampled using a diffusion model (*upsample target*), and (3) DnD with naive diffusion-based generation without guidance (*w/o guidance*). We conduct experiments on Ant *medium* and *medium-expert* source datasets under gravity shift and present results in Table 10. We observe that *upsample target* slightly improves performance over *w/o diffusion*. However, the *w/o guidance* variant, which naively upsamples the source dataset, achieves significantly better results than *upsample target*. We hypothesize that the limited target data setting hinders the training diffusion model in the target domain, thus affecting adaptation performance. Among all variants, DnD achieves the highest performance, demonstrating the effectiveness of our proposed method of using a guided-diffusion model to upsample the source dataset with generated data close to the target domain.

Guided Score κ

The parameter κ controls the conditional k-NN score used for generating source samples, effectively determining their proximity to the target domain. To evaluate its impact, we conduct experiments on HalfCheetah and Hopper under both gravity and kinematic shifts. The results, presented in Table 11, show that a lower κ value ($\kappa = 80$) leads to significant performance drops across multiple tasks. We hypothesize that this occurs because the generated source samples are not sufficiently beneficial for policy adaptation. These findings highlight the importance of generating source samples that closely align with the target domain to improve adaptation performance.

Effect of λ

λ controls the strength of the policy regularization in Eq (10). A small λ may lead to the policy being biased toward the source dataset, while a large λ limits knowledge transfer from the source. We evaluate different λ values ($\lambda \in 0, 0.1, 0.5, 2$) and present the results in Figure 6. Removing policy regularization, i.e. setting $\lambda = 0$, leads to suboptimal performance. We also observe that the optimal λ varies by task (e.g., Halfcheetah prefers $\lambda = 0.1$, while Walker performs best with $\lambda = 2.0$). To balance this trade-off, we set $\lambda = 0.1$ for all tasks.

Data Selection Ratio ξ

Parameter ξ controls how many source data we select to use in a batch at each training step for policy learning, with larger ξ indicating more source data will be rejected. We evaluate the impact of ξ on DnD's performance using medium source datasets under both gravity and kinematic shifts. Specifically, we sweep $\xi \in 0, 25, 50, 75$, where $\xi = 0$ means all source data is used for training. As shown in Figure 7, using $\xi = 0$ is suboptimal, confirming that naively combining source and target datasets is ineffective. While different tasks prefer different ξ values, we find that $\xi = 50$ provides a balanced trade-off across tasks.

Effect of Weighting Q-function

We conduct experiments to evaluate the impact of weighting the Q-function with the score as in Eq. (9). Specifically, we compare DnD with a variant that removes this weighting during value function updates, as follows:

$$\mathcal{L}_Q = \mathbb{E}_{D_{tar}}\left[(Q_\phi - \mathcal{T}Q_\phi)^2\right] + \mathbb{E}_{D_{src}}\left[\mathbb{1}(w_k \geq w_{k,\xi\%})(Q_\phi - \mathcal{T}Q_\phi)^2\right], \tag{39}$$

Since the source data selection ratio remains constant, Eq. (39) may suffer from bad source transitions, as low-quality source samples can still be used for training. Weighting the Q-function mitigates this issue by reducing the influence of source samples that deviate significantly from the target domain. As shown in Table 12, removing the weighting mechanism leads to performance drops in 3 out of 4 tasks, highlighting its importance.

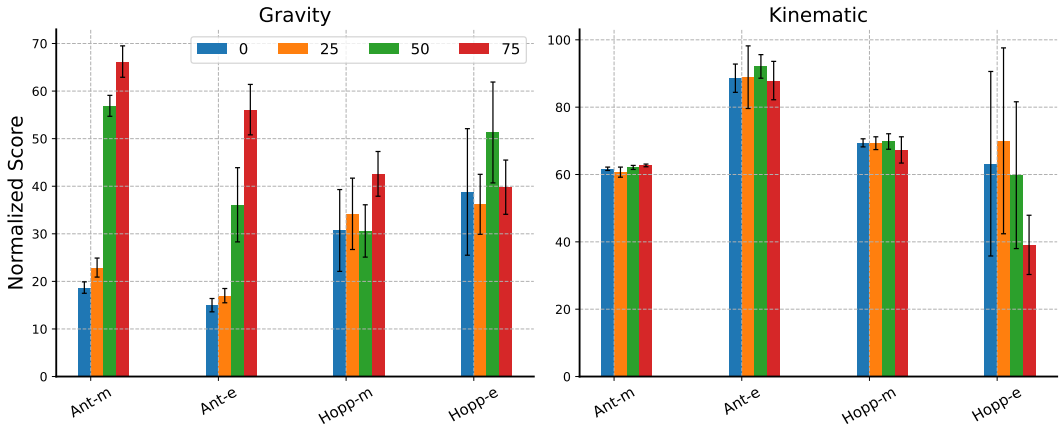


Figure 7: Parameter study of the source data selection ratio. We report the results of Ant and Hopper tasks.

Table 9: Results in kinematic shift tasks. We report normalized scores and their standard deviations in the target domain, averaged over five random seeds. The best score is bold. Half=Halfcheetah, Hopp=Hopper, m=medium, me=medium-expert, mr=medium-replay.

Source	Target	IQL	DARA	BOSA	SRPO	IGDF	OTDF	DnD
Ant-m	medium	50.0±5.6	42.3±7.6	20.9±2.6	50.5±6.7	54.5±1.3	55.4±0.0	**62.1±0.6**
Ant-m	medium-expert	57.8±7.2	54.1±3.8	31.7±7.0	54.9±1.3	54.5±4.6	60.7±3.6	**68.9±1.0**
Ant-m	expert	59.6±18.5	54.2±11.3	45.4±8.6	45.5±9.3	49.4±14.6	90.4±4.8	**92.1±3.5**
Ant-me	medium	49.5±4.1	44.7±4.3	19.0±8.0	41.3±8.1	41.8±8.8	50.2±4.3	**60.6±1.3**
Ant-me	medium-expert	37.2±2.0	33.3±7.0	6.4±2.5	32.8±8.0	41.5±4.9	48.8±2.7	**60.4±3.7**
Ant-me	expert	18.7±8.1	17.8±23.6	14.5±9.0	35.2±15.5	14.4±22.9	**78.4±12.2**	76.0±4.1
Ant-mr	medium	43.7±4.6	42.0±5.4	19.0±1.8	45.3±5.1	41.4±5.0	52.8±4.4	**61.9±0.5**
Ant-mr	medium-expert	36.5±5.9	36.0±6.7	19.1±1.6	36.2±6.6	37.2±4.7	54.2±5.2	**58.8±3.6**
Ant-mr	expert	24.4±4.8	22.1±0.4	19.5±0.8	27.1±3.7	24.3±2.8	**74.7±10.5**	43.8±2.6
Half-m	medium	12.3±1.2	10.6±1.2	8.3±1.2	16.8±4.2	23.6±5.7	**40.2±0.0**	38.5±1.4
Half-m	medium-expert	10.8±1.9	12.9±2.8	8.7±1.3	10.3±2.7	9.8±2.4	10.1±4.0	**19.1±1.0**
Half-m	expert	12.6±1.7	12.1±1.0	10.8±1.7	12.2±0.9	12.8±0.7	8.7±2.0	**13.1±0.8**
Half-me	medium	21.8±6.5	25.9±7.4	30.0±4.3	17.2±3.3	21.9±6.5	30.7±9.6	**38.4±1.4**
Half-me	medium-expert	7.6±1.4	9.5±4.2	6.8±2.9	9.6±2.4	8.9±3.3	10.9±4.2	**24.1±4.6**
Half-me	expert	9.1±2.4	10.4±1.3	4.9±3.2	11.2±1.0	10.7±1.4	3.2±0.6	**13.4±2.0**
Half-mr	medium	10.0±5.4	11.5±4.9	7.5±3.1	10.2±3.7	11.6±4.6	**37.8±2.1**	19.5±1.8
Half-mr	medium-expert	6.5±3.1	9.2±4.7	6.6±1.7	9.5±1.8	8.6±2.3	9.7±2.0	**11.4±2.1**
Half-mr	expert	13.6±1.4	14.8±2.0	10.4±4.9	14.8±2.2	13.9±2.2	7.2±1.4	**15.6±2.9**
Hopp-m	medium	58.7±8.4	43.9±15.2	12.3±6.6	65.4±1.5	65.3±1.4	65.6±1.9	**69.8±2.3**
Hopp-m	medium-expert	68.5±12.4	55.4±16.9	15.6±10.8	43.9±30.8	51.1±18.5	55.4±25.1	**78.2±5.1**
Hopp-m	expert	79.9±35.5	83.7±19.6	14.8±5.5	53.1±39.8	**87.4±25.4**	35.0±19.4	59.8±21.8
Hopp-me	medium	66.0±0.5	61.1±4.0	35.0±20.1	64.6±2.6	65.2±1.5	65.3±2.4	**69.6±1.3**
Hopp-me	medium-expert	45.1±15.7	61.9±16.9	13.9±4.9	54.7±17.0	62.9±15.6	38.6±15.9	**75.5±9.6**
Hopp-me	expert	44.9±19.8	84.2±21.1	12.0±4.3	57.6±40.6	52.8±19.7	29.9±11.3	**64.5±24.2**
Hopp-mr	medium	36.0±0.1	39.4±7.2	3.2±2.6	36.1±0.2	35.9±2.4	35.5±12.2	**64.8±2.4**
Hopp-mr	medium-expert	36.1±0.1	34.1±3.6	4.4±2.8	36.0±0.1	36.1±0.1	47.5±14.6	**69.7±7.5**
Hopp-mr	expert	36.0±0.1	36.1±0.2	3.7±2.5	36.1±0.1	36.1±0.3	49.9±30.5	**69.9±18.0**
Walker-m	medium	34.3±9.8	35.2±22.5	14.3±11.2	39.0±6.7	41.9±11.2	49.6±18.0	**63.2±4.2**
Walker-m	medium-expert	30.2±12.5	51.9±11.5	13.6±7.7	38.6±6.5	42.3±19.3	43.5±16.4	**53.5±7.0**
Walker-m	expert	56.4±18.2	40.7±14.4	15.3±2.5	57.3±12.2	60.4±17.5	46.7±13.6	**70.5±12.0**
Walker-me	medium	41.8±8.8	38.1±14.4	21.4±8.3	36.9±4.3	41.2±13.0	44.6±6.0	**59.4±6.8**
Walker-me	medium-expert	22.2±8.7	23.6±8.1	15.9±4.1	23.2±7.9	28.1±4.0	16.5±7.2	**53.2±7.3**
Walker-me	expert	26.3±10.4	36.0±9.2	18.5±3.6	40.9±9.6	46.2±19.4	42.4±9.1	**69.2±7.0**
Walker-mr	medium	11.5±7.1	12.5±4.3	1.9±2.1	14.3±3.1	22.2±5.2	49.7±9.7	**52.9±8.4**
Walker-mr	medium-expert	9.7±3.8	11.2±5.0	4.6±3.0	4.2±5.1	7.6±4.9	**55.9±17.1**	36.4±5.4
Walker-mr	expert	7.7±4.8	7.4±2.4	3.6±1.5	13.2±8.5	7.5±2.1	**51.9±7.9**	44.4±8.5
Total Score		1193.0	1219.8	513.5	1195.7	1271.0	1547.6	**1902.2**

Table 10: Performance comparison between DnD and its variants on using the diffusion model. We bold the highest scores.m=medium, e=expert, me=medium-expert.

Method	m-m	m-me	m-e
w/o diffusion	23.6±2.9	14.4±0.8	19.7±1.4
up sample target	25.4±3.1	23.7±3.7	24.2±2.7
w/o guidance	46.8±11.5	31.6±11.5	22.43.6
w guidance (DnD)	**56.9±2.2**	**47.5±3.9**	**36.1±7.8**
Method	me-m	me-me	me-e
w/o diffusion	22.5±3.7	16.1±1.3	18.4±1.7
up sample target	22.6±2.8	27.1±1.1	22.6±2.6
w/o guidance	40.7±6.5	35.8±7.2	30.2±4.1
w guidance (DnD)	**55.7±8.1**	**46.3±4.9**	**42.7±13.0**

Table 11: Performance comparison of DnD between different guided score κ values. We highlight the highest score in bold. Half=Halfcheetah, Hopp=Hopper, m=medium, me=medium-expert, e=expert.

Source	Target	κ		
		80	90	99
Half-gravity-m	me	48.2±0.8	**48.9±0.7**	47.8±1.8
Half-gravity-m	e	48.8±0.4	48.8±1	**49.5±1**
Half-kinematic-m	me	17.7±4.1	**19.1±1**	16.7±3.6
Half-kinematic-m	e	**13.7±1.3**	13.1±0.8	13.2±1.4
Hopp-gravity-m	me	33.6±8	**40.6±5.5**	35±4.9
Hopp-gravity-m	e	32.8±8.5	**51.3±10.6**	40.5±12.2
Hopp-kinematic-m	me	74.6±4.2	78.2±5.1	**79.3±8.5**
Hopp-kinematic-m	e	53.4±20.6	**59.8±21.8**	53.5±18.2

Table 12: Ablation study on the effect of weighting q-function. We report the normalized score on the target domain. The highest scores are bold.

Task	w/o weighting q	w weighting q
Ant-kinematic	87.5±6.6	**92.1±3.5**
Hopper-kinematic	51.1±18.9	**59.8±21.8**
Halfcheetah-gravity	**49.4±0.8**	48.8±1
Walker-gravity	62.4±1.9	**63.8±2.7**