DADIFF: Diffusion-Driven Cross-Domain Policy Adaptation for Reinforcement Learning

Abstract— **Transferring policies across domains poses a vital challenge in reinforcement learning, due to the dynamics mismatch between the source and target domains. In this paper, we consider the setting of online dynamics adaptation, where policies are trained in the source domain with sufficient data, while only limited interactions with the target domain are allowed. There are a few existing works that address the dynamics mismatch by employing domain classifiers, value-guided data filtering, or representation learning. Instead, we study the domain adaptation problem from a generative modeling perspective. Specifically, we introduce DADIFF, a diffusion-based framework that leverages the discrepancy between source and target domain generative trajectories in the generation process of the next state to estimate the dynamics mismatch. Both reward modification and data selection variants are developed to adapt the policy to the target domain. We also provide a theoretical analysis to show that the performance difference of a given policy between the two domains is bounded by the generative trajectory deviation. More discussions on the applicability of the variants and the connection between our theoretical analysis and the prior work are further provided. We conduct extensive experiments in environments with various shifts to validate the effectiveness of our method. The results demonstrate that our method provides superior performance compared to existing approaches, effectively addressing the dynamics mismatch. We provide the code of our method at `https://anonymous.4open.science/r/DADiff-release-DB61`.**

I. INTRODUCTION

Reinforcement learning (RL) has shown strong potential in complex decision-making tasks, but training directly in the real-world environment (*target domain*) is often restricted by safety, cost, and limited interaction budgets. An alternative strategy is to train policies in a surrogate environment (*source domain*), such as a simulator, and then transfer them to the target domain. But due to the dynamics mismatch between the source and target domains, directly transferring the policy often leads to performance degradation, which is a critical challenge in the sim-to-real problem [1], [2]. One solution to this transfer problem is known as *online dynamics adaptation* [3], [4], where policies are trained with abundant source-domain data and only limited interactions in the target domain. In this setting, the state space, action space, and reward function remain consistent across domains, while the transition dynamics differ. Compared with solutions such as domain randomization [5]–[7] or simulator calibration [8], online dynamics adaptation does not require access to high-fidelity simulators or prior knowledge of target dynamics, and can therefore be applied in situations where such information is unavailable.

Existing online dynamics adaptation methods, including classifier-based approaches [9], value-guided filtering [3], and representation learning [10], capture dynamics discrepancy from different perspectives: classifiers provide coarse distinctions between domains, value-guided methods depend on the modeling of forward predictions, and representation learning relies on assumptions of invariant latent structures across domains. When the domains are complex or stochastic, a key challenge that remains is to develop an approach capable of capturing dynamics discrepancy in a more fine-grained and distributional manner.

The generative modeling perspective provides a potential direction. Generative models, such as diffusion models [11], [12] and flow matching methods [13], have demonstrated strong capability in representing complex distributions. When state transitions are viewed as a conditional generative process, the mismatch between source and target domains can be interpreted as a discrepancy between their respective generative processes. Specifically, the multi-step sampling procedure in diffusion models and flow matching methods produces several latent states, which construct a generative trajectory, serving as structured signals of source–target dynamics deviation. These latent states allow the discrepancy to be captured not only at the next-state level but also along the entire trajectory. Intuitively, if the source and target domains follow different dynamics, their trajectories will diverge at multiple steps, a phenomenon we term *generative trajectory deviation*. This notion provides a fine-grained view of dynamics discrepancy by revealing how divergence accumulates along the trajectory, rather than relying solely on local or aggregated comparisons. Our theoretical analysis further connects trajectory deviation to performance guarantees, providing motivation for algorithmic design.

Building on this perspective, we introduce DADIFF, a diffusion-based framework for online dynamics adaptation. DADIFF leverages latent states in diffusion models to measure generative trajectory deviation between source and target domains, and exploits this deviation in two complementary ways: (i) DADIFF-modify, which adjusts source-domain rewards with deviation-based penalties, and (ii) DADIFF-select, which filters source-domain data based on deviation before value function updates. We further discuss the applicability of these variants to different tasks, highlight the advantages of our method compared to prior work, and establish a connection between our analysis and the theoretical guarantee of prior work. Empirical results in environments with various shifts show the superior performance of our method compared to existing algorithms.

II. Related Works

a) Domain Adaptation in RL: Generalizing RL policies to diverse environments is critical for real-world deployment, where transition dynamics [9], [14], [15], state or action spaces [16], [17] may be different. To address domain adaptation, prior work falls under three categories: (i) domain randomization that randomizes transition dynamics to expose agents to many environment configurations [18], [19], (ii) meta-learning to few-shot adapt to many environments [20], [21], and (iii) expert demonstrations of target environments through imitation learning [22], [23]. However, these approaches are either computationally expensive or require hard-to-obtain demonstrations. With only limited target-domain data, some works perform reward modifications to transition to the target domain by using transition classifiers [9], [24] or reward augmentations [4], [25]. Data selection methods [3], [26] have also been used to filter out part of the source-domain transitions and train policies on both source and target domain data. When the domains are complex or stochastic, a key challenge that remains is to develop an approach capable of capturing the dynamics discrepancy. Our method explores this challenge from a generative modeling perspective by measuring the generative trajectory deviation between the source and target domains.

b) Diffusion Models in RL: Diffusion models [11], [12] have been extensively used for generating effective decision-making policies in several domains, such as reinforcement learning [27] and robotics [28]. Specifically, they are widely leveraged to synthesize data for offline RL [29], facilitate planning and action generation in multi-task scenarios [30], and enhance the representational capacity of learned RL policies [31]. In addition, diffusion models have also been extended to the multi-agent settings [32]. In the field of domain adaptation, they are utilized to augment the target-domain data in order to boost the performance of offline RL policies [33]. However, the introduction of synthesizers may lead to extra computational costs, and the quality of synthesized data is hard to guarantee. In contrast, we choose to directly estimate the dynamics discrepancy by multiple latent states from diffusion models instead of generating more synthetic data.

III. Preliminaries

a) Online Dynamics Adaptation: We consider two Markov Decision Processes (MDPs), denoted as $\mathcal{M}_{\text{src}} = (\mathcal{S}, \mathcal{A}, P_{\text{src}}, r, \gamma)$ and $\mathcal{M}_{\text{tar}} = (\mathcal{S}, \mathcal{A}, P_{\text{tar}}, r, \gamma)$ for the source domain and target domain, respectively. The state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and discount factor $\gamma \in [0, 1]$ are consistent across both domains, while the transition dynamics P_{src} and P_{tar} differ. The goal of online dynamics adaptation is to learn a policy π that achieves high performance in the target domain $\mathcal{M}_{\text{tar}}$, utilizing sufficient data from the source domain and only limited interactions from the target domain. In addition, we specify a domain $\mathcal{M}$ and define the probability that a policy π encounters a state s at time step t as $P^{\pi}_{\mathcal{M},t}(s)$. Therefore, the normalized probability that a policy π visits a state-action pair (s, a) in the domain $\mathcal{M}$ can be represented as $\rho^{\pi}_{\mathcal{M}}(s, a) := (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P^{\pi}_{\mathcal{M},t}(s)\pi(a|s)$. The expected return of a policy π in $\mathcal{M}$ is defined as $\eta_{\mathcal{M}}(\pi) = \mathbb{E}_{(s,a)\sim\rho^{\pi}_{\mathcal{M}}}[r(s, a)]$. We assume the reward are bounded by $|r(s, a)| \leq r_{\max}, \forall s \in \mathcal{S}, a \in \mathcal{A}$.

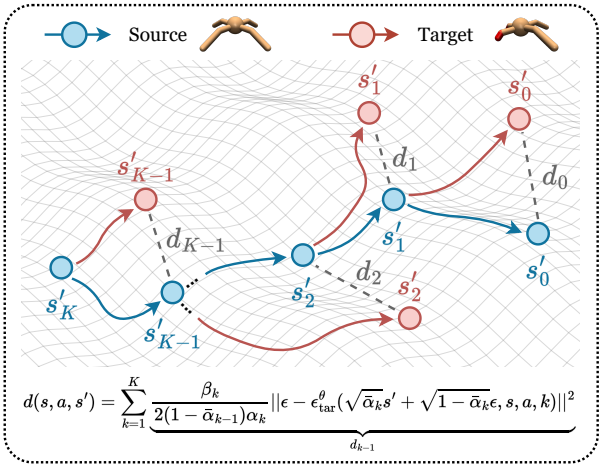


Fig. 1: Illustration of DADiff. This figure visualizes the generative trajectories in the source and target domains. The deviation $d(s, a, s')$ is measured by the discrepancy d_k of each latent state s'_k in the source and target domain generative trajectories.

b) Diffusion Models: Diffusion models [11], [12] are a family of generative models that learn to generate samples from a target distribution. We mainly focus on the denoising diffusion probabilistic model (DDPM) [12] in this paper. DDPM consists of a forward process and a reverse process. The forward process is regarded as a Markov chain that gradually adds noise to data, transforming a clean data point x_0 into Gaussian noise, which is formulated as follows,

$$x_k = \sqrt{1-\beta_k}x_{k-1} + \sqrt{\beta_k}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{1}$$

where x_k is the noisy data at diffusion timestep k, β_k is the noise schedule, and ϵ is Gaussian noise. To simplify the forward process, we can directly sample the noisy data at diffusion timestep k as follows,

$$x_k = \sqrt{\bar{\alpha}_k}x_0 + \sqrt{1-\bar{\alpha}_k}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{2}$$

where $\alpha_k = 1 - \beta_k$ and $\bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i$. The reverse process learns to denoise the noisy data step by step, which is formulated as follows,

$$x_{k-1} = \frac{1}{\sqrt{\alpha_k}}\left(x_k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta(x_k, k)\right) + \sqrt{\frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k}\beta_k}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{3}$$

where $\epsilon_\theta(x_k, k)$ is a noise model that estimates the noise from the noisy data point x_k. The noisy data points $\{x_k\}_{k=0}^{K}$ form a generative trajectory from the initial noisy data x_K to the clean data x_0. The training objective of the noise model is formulated as follows,

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{x_0,\epsilon,k}\left[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_k}x_0 + \sqrt{1-\bar{\alpha}_k}\epsilon, k)||^2\right]. \tag{4}$$

IV. Methodology

In this section, we first introduce a theoretical analysis to demonstrate the connection between the dynamics mismatch

and the generative trajectory mismatch. Then, we present our diffusion-based method, DADIFF, which measures the generative trajectory deviation from the perspective of diffusion models and adapts the learned policy to the target domain. The overview of our method is shown in Figure 1.

A. Theoretical Analysis

Before introducing the theoretical analysis, we first provide the definition of a generative trajectory, which is crucial for the analysis. For clarity, we denote the next state s' as s'_0.

Definition 4.1: (Generative trajectory.) Specify a domain $\mathcal{M}$ with transition dynamics $P_{\mathcal{M}}(s'_0|s,a)$. There is a generative trajectory for the next state s'_0 consisting of K auxiliary variables $\{s'_k\}_{k=1}^K$, referred to as latent states. These latent states form a Markov chain from the initial latent state s'_K to the next state s'_0 conditioned on the state–action pair (s,a).

Remark. The Markov-chain definition enables the transition dynamics to be decomposed into multiple conditional probabilities, *i.e.*, $P_{\mathcal{M}}(s'_0|s,a) = \int P_{\mathcal{M}}(s'_K|s,a)\prod_{k=1}^K P_{\mathcal{M}}(s'_{k-1}|s'_k,s,a)ds'_{1:K}$. In this way, the next state s'_0 can be viewed as being generated step by step with latent states, forming a generative trajectory. The discrepancy of such generative trajectories across domains provides a natural estimation of the dynamics discrepancy.

We construct generative trajectories in both source and target domains, starting from the same initial latent state s'_K, and derive Theorem 4.2 to establish the connection between the dynamics mismatch and the generative trajectory mismatch. The detailed proof is provided in Appendix VII-B.

Theorem 4.2: (Performance bound controlled by generative trajectory discrepancy.) Denote $\mathcal{M}_{\text{src}}$ and $\mathcal{M}_{\text{tar}}$ as the source and target domains with different dynamics, respectively. The performance difference of any policy π evaluated in $\mathcal{M}_{\text{src}}$ and $\mathcal{M}_{\text{tar}}$ can be bounded as below,

$$\begin{aligned}&\eta_{\mathcal{M}_{\text{src}}}(\pi) - \eta_{\mathcal{M}_{\text{tar}}}(\pi) \le \\ &\frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2}\underbrace{\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\sqrt{\mathbb{E}_{P_{\text{src}}}[D_{\text{KL}}(P_{\text{src}}(s'_K \mid s,a) \,\|\, P_{\text{tar}}(s'_K \mid s,a))]}\right]}_{(a)\text{: initial latent state deviation}} + \\ &\frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2}\underbrace{\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\sqrt{\mathbb{E}_{P_{\text{src}}}\left[\sum_{k=1}^K D_{\text{KL}}\big(P_{\text{src}}(s'_{k-1} \mid s'_k,s,a) \,\|\, P_{\text{tar}}(s'_{k-1} \mid s'_k,s,a)\big)\right]}\right]}_{(b)\text{: latent state transition mismatch}}.\end{aligned} \tag{5}$$

Remark. This bound indicates that the performance difference of a policy π between the source and target domains is controlled by the initial latent state deviation term (a) and the latent state transition mismatch term (b). Since the generative trajectories in both the source and target domains share the same initial latent state s'_K, term (a) vanishes, leaving term (b) as the sole determinant of the performance difference. In other words, as long as the generative trajectories are similar in the source and target domains, the performance difference is small, and vice versa. We note that PAR [10] can be considered as a special case of Theorem 4.2 when $K=1$. A discussion on the connection between our analysis and the theoretical guarantee of PAR is provided in Section VI-A.

B. Domain Adaptation with Diffusion

Theorem 4.2 provides a theoretical guarantee linking the performance difference of a policy π to the generative trajectory, thereby motivating a careful design of latent states in the trajectory. In this section, we adopt the formulation of DDPM to better characterize the dynamics discrepancy.

We first redeclare the reverse process of DDPM in a reparameterized form to describe the latent state transition in domain $\mathcal{M}$ as follows,

$$s'_{k-1} = \frac{1}{\sqrt{\alpha_k}}\left(s'_k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_{\mathcal{M}}(s'_k,s,a,k)\right) + \sqrt{\frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k}\beta_k}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0,I), \tag{6}$$

where $\epsilon_{\mathcal{M}}(s'_k,s,a,k)$ is the noise from the latent state s'_k in domain $\mathcal{M}$. It indicates that the latent state transition follows a Gaussian distribution, *i.e.*,

$$P_{\mathcal{M}}(s'_{k-1} \mid s'_k,s,a) \sim \mathcal{N}\left(\frac{1}{\sqrt{\alpha_k}}\Big(s'_k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_{\mathcal{M}}(s'_k,s,a,k)\Big), \frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k}\beta_k I\right). \tag{7}$$

According to Theorem 4.2, the performance difference of a policy π across domains is determined by the latent state transition mismatch term (b). Therefore, we can estimate the generative trajectory deviation $d(s,a,s')$ with the defined distribution of latent state transition in Equation 7 as follows,

$$\begin{aligned}d(s,a,s') &= \sum_{k=1}^K D_{\text{KL}}(P_{\text{src}}(s'_{k-1}|s'_k,s,a)||P_{\text{tar}}(s'_{k-1}|s'_k,s,a)) \\ &= \sum_{k=1}^K \frac{\beta_k}{2(1-\bar{\alpha}_{k-1})\alpha_k}\left\|\epsilon_{\text{src}}(s'_k,s,a,k) - \epsilon_{\text{tar}}(s'_k,s,a,k)\right\|^2.\end{aligned} \tag{8}$$

We derive this equation by computing the KL divergence between two Gaussian distributions. Notably, as the state transition tuple (s,a,s') comes from the source domain, the noise $\epsilon_{\text{src}}(s'_k,s,a,k)$ estimated in the reverse process must be consistent with the noise used in the forward process to generate the latent state s'_k, which indicates $\epsilon_{\text{src}}(s'_k,s,a,k) = \epsilon$ with $\epsilon \sim \mathcal{N}(0,I)$. Besides, we introduce a noise model $\epsilon^\theta_{\text{tar}}(s'_k,s,a,k)$, trained with target-domain data, to estimate the noise in the target domain. The training objective is formulated as follows,

$$\mathcal{L}_{\text{noise}} = \mathbb{E}_{(s,a,s')\sim\mathcal{D}_{\text{tar}},\epsilon,k}\left[\left\|\epsilon - \epsilon^\theta_{\text{tar}}(\sqrt{\bar{\alpha}_k}s'_0 + \sqrt{1-\bar{\alpha}_k}\epsilon, s,a,k)\right\|^2\right]. \tag{9}$$

This objective mirrors the standard DDPM training loss, but conditions on (s,a) to capture dynamics in the target domain. For the latent state s'_k in Equation 8, there are two ways to obtain it: (i) by iteratively applying the reverse process in Equation 6, and (ii) by sampling directly from the forward process of DDPM, *i.e.*, $s'_k = \sqrt{\bar{\alpha}_k}s'_0 + \sqrt{1-\bar{\alpha}_k}\epsilon$ with $\epsilon \sim \mathcal{N}(0,I)$. Specifically, the first way requires sequential sampling across all steps to generate the entire generative trajectory, which is computationally expensive. In contrast, the second way can produce all latent states in parallel, yielding a much more efficient implementation. Therefore, we choose to obtain the latent state s'_k via the forward process in our method. Finally, the deviation $d(s,a,s')$ can

be practically estimated as follows,

$$d(s,a,s') = \sum_{k=1}^{K} \frac{\beta_k}{2(1-\bar{\alpha}_{k-1})\alpha_k} \left\| \epsilon - \epsilon_{\text{tar}}^{\theta}(\sqrt{\bar{\alpha}_k}s'_0 + \sqrt{1-\bar{\alpha}_k}\epsilon, s, a, k) \right\|^2, \quad \epsilon \sim \mathcal{N}(0, I). \tag{10}$$

We further introduce two variants based on SAC [34] to utilize the deviation $d(s,a,s')$, including reward modification and data selection, since we find that baselines adopting these two techniques exhibit complementary advantages in different tasks, which is shown in Section V-B. We analyze the possible reason for this phenomenon from the reward distribution aspect in Section VI-B. The details of DADIFF variants are provided as follows.

a) Reward modification.: We refer to this variant as DADIFF-modify. It adopts the deviation $d(s,a,s')$ as a reward penalty to modify the reward function in the source domain, *i.e.*,

$$r_{\text{mod}}(s,a,s') = r(s,a,s') - \lambda d(s,a,s'), \tag{11}$$

where λ is a penalty coefficient to balance the original reward and the penalty. The objective function for training the value function gives,

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_{(s,a,r_{\text{mod}},s')\sim\mathcal{D}_{\text{src}}\cup\mathcal{D}_{\text{tar}}} \left[(Q_\phi - \mathcal{T}Q_\phi)^2 \right], \tag{12}$$

where $\mathcal{D}_{\text{tar}}$ and $\mathcal{D}_{\text{src}}$ are the datasets from the target and source domains, respectively, Q_ϕ is the value function, and $\mathcal{T}$ is the Bellman operator.

b) Data selection.: We refer to this variant as DADIFF-select. We select fixed percentage data with the lowest deviation $d(s,a,s')$ from a batch of source domain data. The selected data is then used to update the value function. We formulate the objective function of the value function as follows,

$$\begin{aligned} \mathcal{L}_{\text{critic}} = {} & \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\text{tar}}} \left[(Q_\phi - \mathcal{T}Q_\phi)^2 \right] + \\ & \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\text{src}}} \left[\omega(s,a,s')(Q_\phi - \mathcal{T}Q_\phi)^2 \right], \end{aligned} \tag{13}$$

where $\omega(s,a,s') = \mathbb{1}(d(s,a,s') < d_{\xi\%})$, $\mathbb{1}$ is the indicator function, and $d_{\xi\%}$ denotes the lowest ξ-quantile deviation in the batch.

For both variants, the objective function of the policy π is formulated as:

$$\mathcal{L}_{\text{actor}} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\text{src}}\cup\mathcal{D}_{\text{tar}}} \left[-\min_{i=1,2} Q_{\phi_i}(s,a) + \tau \log \pi(a|s) \right], \tag{14}$$

where τ is the entropy temperature coefficient, and i denotes the value function index. We provide the pseudocode of DADIFF in Algorithm 1.

Algorithm 1: Domain Adaptation with DADIFF

Input: Source domain $\mathcal{M}_{\text{src}}$, target domain $\mathcal{M}_{\text{tar}}$, and target domain interaction frequency F
Initialization: Policy π, value function $\{Q_{\phi_i}\}_{i=1,2}$, target value function $\{Q_{\phi'_i}\}_{i=1,2}$, noise model $\epsilon_{\text{tar}}^{\theta}$, replay buffers $\{\mathcal{D}_{\text{src}}, \mathcal{D}_{\text{tar}}\}$, penalty coefficient λ, data selection ratio ξ, batch size N

for $i = 1, 2, \ldots$ **do**
  Collect $(s_{\text{src}}, a_{\text{src}}, r_{\text{src}}, s'_{\text{src}})$ from $\mathcal{M}_{\text{src}}$, store in $\mathcal{D}_{\text{src}}$
  if $i \bmod F = 0$ **then**
    Collect $(s_{\text{tar}}, a_{\text{tar}}, r_{\text{tar}}, s'_{\text{tar}})$ from $\mathcal{M}_{\text{tar}}$, store in $\mathcal{D}_{\text{tar}}$
  Sample N transitions from $\mathcal{D}_{\text{tar}}$, train model $\epsilon_{\text{tar}}^{\theta}$ via Eq. 9
  Sample N transitions from $\mathcal{D}_{\text{src}}$, compute $d(s_{\text{src}}, a_{\text{src}}, s'_{\text{src}})$ via Eq. 10
  if *using reward modification* **then** // reward modification
    Modify source domain rewards via Eq. 11
    Update value functions Q_{ϕ_i} by minimizing Eq. 12
  else // data selection
    Select ξ-quantile data from $\mathcal{D}_{\text{src}}$ by $d(s_{\text{src}}, a_{\text{src}}, s'_{\text{src}})$
    Update value functions Q_{ϕ_i} by minimizing Eq. 13
  Update actor π by minimizing Eq. 14
  Update target value functions $Q_{\phi'_i}$

V. EXPERIMENTS

A. Experimental Setup

We conduct experiments in four environments (*ant*, *hopper*, *halfcheetah*, *walker*) from Gym MuJoCo [35], [36]. The source domain is set as the original environment, while the target domain is set as the environment with shifts in kinematics, morphology, friction, or gravity. Kinematic shifts restrict joint rotation ranges, morphology shifts reduce limb sizes, friction shifts modify the friction coefficient, and gravity shifts adjust gravitational acceleration. Kinematic and morphology configurations follow PAR [10], while friction and gravity shifts follow ODRL [37] at a level of 0.5.

We compare our method with the following baselines: **DARC** [9], which trains domain classifiers to estimate the dynamics discrepancy and modifies the reward function in the source domain; **VGDF** [3], which uses a value-guided data filtering method to select data from the source domain; **PAR** [10], which trains encoders to estimate the representation discrepancy and modifies the reward function in the source domain; **SAC-IW**, which estimates the dynamics discrepancy as an importance sampling term for value function; **SAC-tune**, which fine-tunes the policy in the target domain for 10^5 environmental steps; **SAC-tar** [34], which is the vanilla SAC trained in the target domain with 10^5 environmental steps; **Oracle** [34], which is the vanilla SAC trained in the target domain with 1M environmental steps. We implement all algorithms based on the official code of ODRL [37] and follow the hyperparameters in the original paper. We allow all algorithms to interact with the source domain for 1M environmental steps and the target domain for 10^5 environmental steps, *i.e.*, the target domain interaction frequency $F = 10$. All algorithms are trained with five random seeds.

B. Adaptation Performance Evaluation

We conduct experiments on sixteen tasks with diverse shifts to evaluate the adaptation performance of DADIFF and baselines. The results are summarized in Figure 2. Overall, DADIFF exhibits consistently strong performance, demonstrating superior or competitive performance against all baselines in the majority of tasks. While some existing methods, such as VGDF, PAR, or SAC-tune, occasionally reach competitive results in specific tasks, their performance fluctuates significantly across different tasks. In contrast,

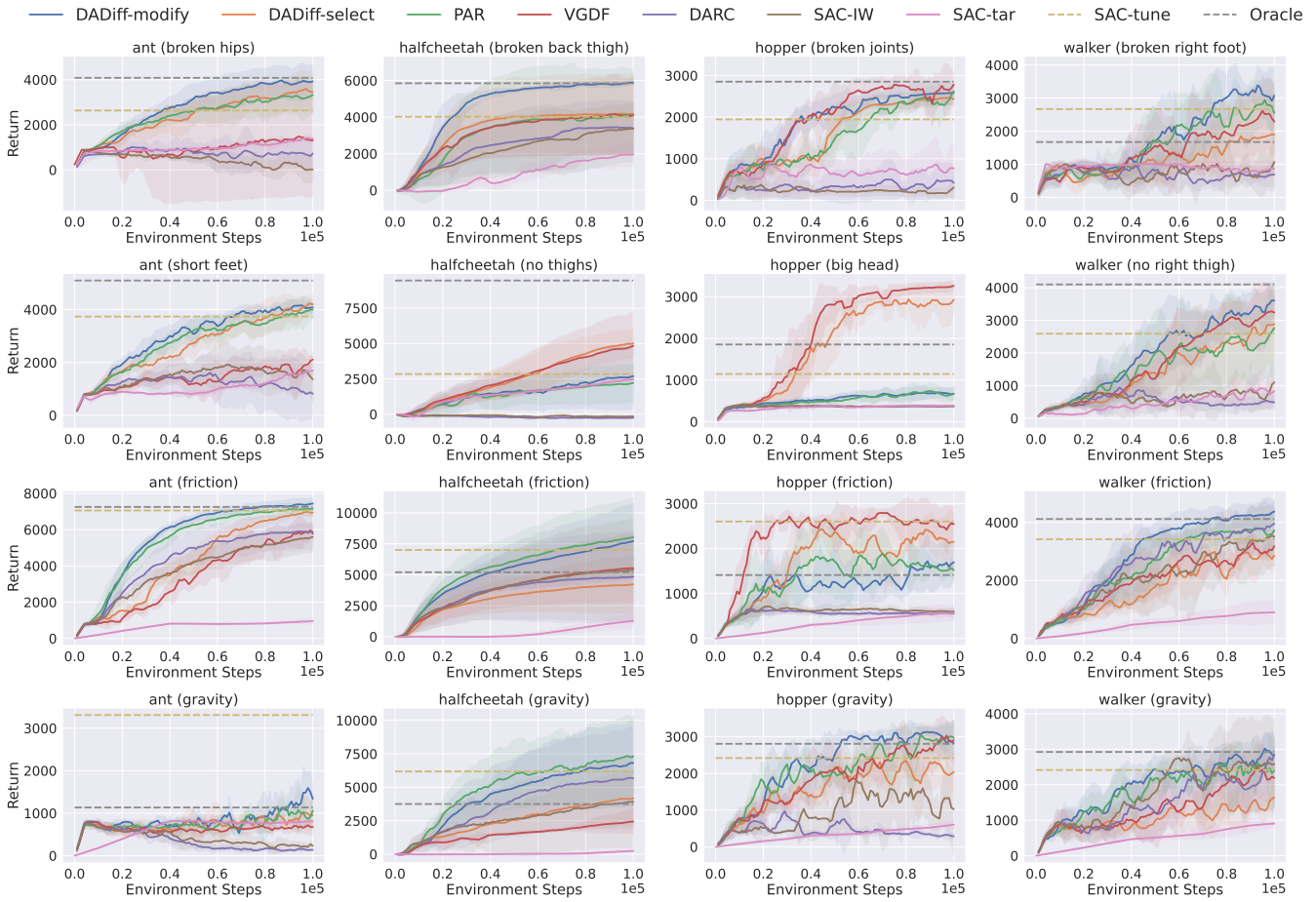


Fig. 2: Adaptation performance under kinematic, morphology, friction, and gravity shifts (from top to bottom). The solid curves and the shaded regions denote the mean and standard deviation over five random seeds, respectively. DADiff demonstrates superior or highly competitive performance against all baselines in the majority of tasks.

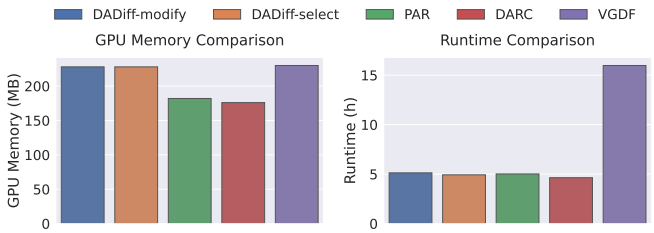


Fig. 3: GPU memory and runtime comparisons on the *halfcheetah (broken back thigh)* task. In the GPU memory comparison, DADIFF-modify and DADIFF-select exhibits slightly higher GPU memory cost compared to PAR and DARC. In the runtime comparison, VGDF requires 3× more training time than other methods due to its model-based approach.

DADIFF maintains stable and superior adaptation performance across a wide range of shift types. We further discuss the performance of two variants of DADIFF, DADIFF-modify and DADIFF-select, respectively.

a) Reward modification variant.: The reward modification variant of our method, DADIFF-modify, demonstrates strong and consistent performance across diverse tasks. As shown in Figure 2, it surpasses other reward modification baselines, including PAR, DARC, and SAC-IW, in most tasks and achieves performance comparable to oracle-level methods. On average, DADIFF-modify improves by 8.7% across all sixteen tasks, with the largest gain of 42.3% on the *halfcheetah (broken back thigh)*. In addition to its performance advantages, we observe that our method incurs a slight increase in GPU memory usage compared to PAR and DARC due to latent state generation, as shown in Figure 3. This modest increase, however, contributes positively to adaptation performance by enabling better discrepancy estimation, thus representing a favorable trade-off between computational cost and effectiveness. To further explore the performance of DADIFF-modify in stochastic environments, we provide an experiment in Section VI-A.

b) Data selection variant.: In Figure 2, the data selection variant, DADIFF-select, proves to be a highly effective alternative by achieving competitive performance against top baselines in tasks where reward modification methods falter. Specifically, in the *halfcheetah (no thighs)*, *hopper (big head)*, and *hopper (friction)* tasks, reward modification methods exhibit poor performance. In contrast, DADIFF-select achieves results that are highly competitive with the top-performing baseline, VGDF. This indicates that in certain tasks, directly filtering for transitions with low dynamics mismatch is a more effective strategy than modifying rewards. We analyze the possible reason in Section VI-B. Furthermore, while VGDF demonstrates top-tier performance in

these tasks, it carries significant trade-offs. Since VGDF is a model-based approach, it takes significantly longer to train by more than $3\times$, as shown in Figure 3. On the other hand, DADIFF-select is able to match or exceed the performance of VGDF on such environments while maintaining comparable efficiency to similar model-free baselines.

C. Parameter Study

The performance of DADIFF is influenced by several key hyperparameters. To better understand their roles, we conducted a series of experiments across different tasks. The results on *halfcheetah (broken back thigh)* and *walker (no right thigh)* are presented in Figure 4.

a) Penalty Coefficient λ.: λ controls the scale of reward penalty in DADIFF-modify. As shown in Figure 4a, we evaluate the performance of DADIFF-modify across multiple values of λ. We find that a worse performance is often shown in the setting $\lambda = 0$, where no penalty is adopted for rewards. It demonstrates the necessity of reward modification. Meanwhile, the results also indicate that the optimal value of λ is task-dependent, and there could be multiple values that yield good performance for a specific task. For instance, in the *halfcheetah (broken back thigh)* task, both $\lambda = 0.5$ and $\lambda = 5.0$ achieve the best performance. A poorly chosen λ can significantly degrade performance, highlighting the importance of tuning this coefficient.

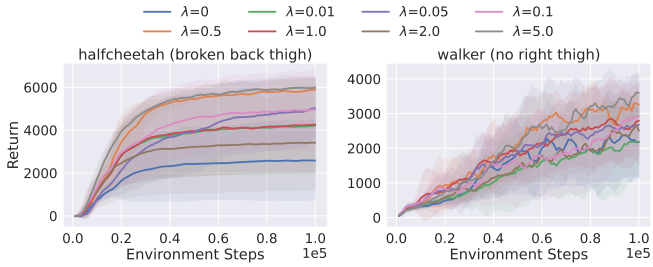
b) Data Selection Ratio $\xi\%$.: $\xi\%$ controls the ratio of source domain data to retain in DADIFF-select. As shown in Figure 4b, we evaluate the performance of DADIFF-select across multiple values of $\xi\%$. Similar to the penalty coefficient, the optimal value of $\xi\%$ is task-dependent. We find that both too much ($\xi\% = 100\%$) and too little (($\xi\% = 0\%$)) source data can lead to suboptimal performance. As retaining too much source data may introduce transitions with significant dynamics mismatch, while retaining too little may result in insufficient data for effective learning.

c) Diffusion Timesteps K.: K controls the number of diffusion timesteps used to measure the discrepancy in both DADIFF-modify and DADIFF-select. We provide the results of DADIFF-modify in Figure 4c. The results shows that performance improves up to $K = 100$. Increasing K further to 200 causes a decline, likely due to the limited capacity of the noise model, which may struggle to accurately estimate noise across too many timesteps.

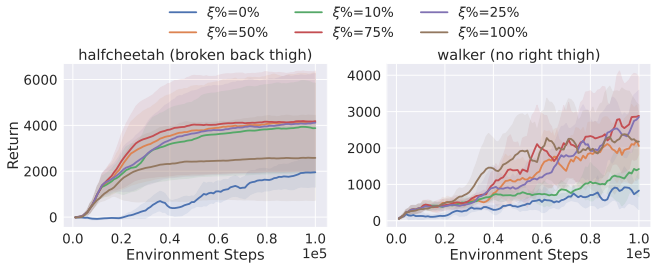
VI. DISCUSSIONS

A. Connection between DADIFF and PAR

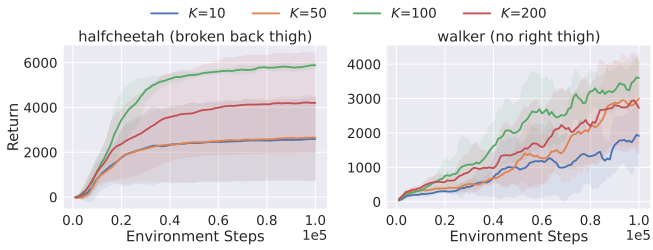
We explore the connection between PAR and our method from a theoretical perspective. The performance bound of our method is controlled by the generative trajectory discrepancy in Theorem 4.2. We consider a special case, where the number of latent states in the trajectory is $K = 1$. Instead of considering latent states in the generative trajectory, we take s'_1 as a latent representation and introduce the one-to-one representation mapping assumption in PAR [10], which assumes that there exists a one-to-one mapping for each state-action pair (s, a) and its latent representation s'_1. In



(a) Penalty coefficient λ.



(b) Data ratio $\xi\%$.



(c) Diffusion timesteps K.

Fig. 4: Parameter study. The solid curves and the shaded regions denote the mean and standard deviation over five random seeds, respectively.

this setting, the state-action pair (s, a) in Equation 5 can be all replaced by the corresponding latent representation s'_1. Therefore, the performance bound can be rewritten as follows,

$$\eta_{\mathcal{M}_{\text{src}}}(\pi) - \eta_{\mathcal{M}_{\text{tar}}}(\pi) \leq \frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2} \mathbb{E}_{\rho^{\pi}_{\text{src}}} \left[\sqrt{\mathbb{E}_{P_{\text{src}}} \left[D_{\text{KL}}(P_{\text{src}}(s'_0|s'_1) || P_{\text{tar}}(s'_0|s'_1)) \right]} \right]. \tag{15}$$

We further introduce a conclusion proven in PAR [10]:

$$D_{\text{KL}}(P_{\text{src}}(s'_1|s'_0) || P_{\text{tar}}(s'_1|s'_0)) = D_{\text{KL}}(P_{\text{src}}(s'_0|s'_1) || P_{\text{tar}}(s'_0|s'_1)) + \mathbb{H}(s'_{\text{src}}) - \mathbb{H}(s'_{\text{tar}}). \tag{16}$$

Therefore, the performance bound can be rewritten as follows,

$$\eta_{\mathcal{M}_{\text{src}}}(\pi) - \eta_{\mathcal{M}_{\text{tar}}}(\pi) \leq \frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2} \mathbb{E}_{\rho^{\pi}_{\text{src}}} \left[\sqrt{\mathbb{E}_{P_{\text{src}}} \left[D_{\text{KL}}(P_{\text{src}}(s'_1|s'_0) || P_{\text{tar}}(s'_1|s'_0)) \right]} \right] + \frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2} \mathbb{E}_{\rho^{\pi}_{\text{src}}} \left[\sqrt{\mathbb{E}_{P_{\text{src}}} \left[\mathbb{H}(s'_{\text{src}}) - \mathbb{H}(s'_{\text{tar}}) \right]} \right]. \tag{17}$$

This performance bound is consistent with the performance bound of PAR, which indicates that PAR can be considered as a special case of our method. However, the one-to-one representation mapping assumption may not hold in

TABLE I: Adaptation performance under stochastic dynamics controlled by the standard deviation parameter ς. Average return and standard deviation over five random seeds are reported. The best results are in **bold**, and performance change relative to the deterministic setting ($\varsigma = 0.0$) is shown in parentheses.

Environment	ς	DADIFF-modify	PAR
hopper (broken joints)	0.00	2582.1±251.6	**2623.1±105.2**
	0.01	**2591.0±159.2** (↑0.34%)	2398.3±297.8 (↓8.57%)
	0.02	**2515.9±101.8** (↓2.57%)	2328.7±302.9 (↓11.22%)
	0.03	**2574.2±280.6** (↓0.31%)	2406.1±455.7 (↓8.27%)
walker (broken right foot)	0.00	**3390.4±464.4**	2943.3±546.7
	0.01	**2879.3±688.9** (↓15.08%)	2373.8±1072.4 (↓19.35%)
	0.02	2812.5±934.6 (↓17.05%)	**2825.8±466.6** (↓3.99%)
	0.03	**3176.8±796.4** (↓6.30%)	1613.9±878.7 (↓45.17%)

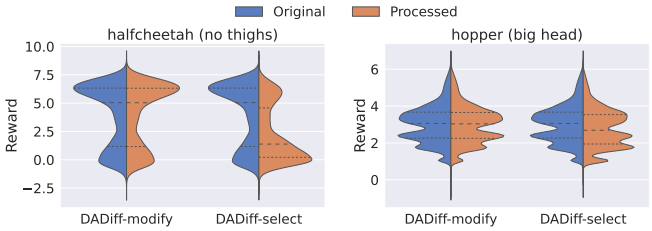


Fig. 5: Reward distribution comparison between the source-domain rewards before processing (Original) and after modification or selection (Processed).

practice, especially in stochastic environments, which limits the application of PAR. In contrast, our method does not rely on this assumption and can handle more general scenarios. We validate this point in environments with stochastic dynamics. Noises with different standard deviation ς are introduced to the actions to simulate stochastic dynamics, and two tasks with kinematic shifts, *hopper (broken joints)* and *walker (broken right foot)*, are considered. We evaluate the performance of DADIFF-modify and PAR, which is presented in Table I. Notably, our method maintains robust performance even as the standard deviation ς increases, while PAR's performance degrades significantly. We believe the decrease in PAR's performance is due to its reliance on one-to-one representation assumptions, which may not hold in stochastic settings.

B. Reward Distribution Analysis

We further examine the reasons behind the superior performance of DADIFF-select, in contrast to the severe failure of DADIFF-modify on *halfcheetah (no thighs)* and *hopper (big head)* tasks, as illustrated in Figure 2. Specifically, we analyze the reward distributions of source-domain data after modification or selection. The results are presented in Figure 5. We find that DADIFF-select generates a higher distribution in the low-reward region compared to DADIFF-modify on both tasks. This suggests that the low-reward data may play a crucial role in these tasks, which can effectively guide the policy to avoid undesirable states and actions.

VII. CONCLUSION

This work explores the problem of online dynamics adaptation in reinforcement learning from a generative modeling perspective. We first theoretically analyze the performance bound of a policy in the source and target domains, which is controlled by the generative trajectory discrepancy. Based on this analysis, we propose a novel method, DADIFF, which utilizes diffusion models to measure the dynamics discrepancy and performs either reward modification or data selection to adapt to the target domain. Extensive experiments demonstrate that our method outperforms existing baselines in tasks with various shifts.

APPENDIX

A. Useful Lemmas

Lemma 7.1: (Telescoping lemma.) Denote $\mathcal{M}_1 = (\mathcal{S}, \mathcal{A}, P_1, r, \gamma)$ and $\mathcal{M}_2 = (\mathcal{S}, \mathcal{A}, P_2, r, \gamma)$ as two MDPs with the same state and action spaces but different transition dynamics P_1 and P_2. The performance difference of a policy π evaluated in $\mathcal{M}_1$ and $\mathcal{M}_2$ can be expressed as:

$$\eta_{\mathcal{M}_1}(\pi) - \eta_{\mathcal{M}_2}(\pi) = \frac{\gamma}{1-\gamma}\mathbb{E}_{\rho^\pi_{\mathcal{M}_1}(s,a)}\left[\mathbb{E}_{s'\sim P_1}[V^\pi_{\mathcal{M}_2}(s')] - \mathbb{E}_{s'\sim P_2}[V^\pi_{\mathcal{M}_2}(s')]\right]$$

Proof. Please see Lemma 4.3 in SLBO [38] for a detailed proof.

B. Proof of Theorem 4.2

Theorem 7.2: (Performance bound controlled by generative trajectory discrepancy.) Denote $\mathcal{M}_{\text{src}}$ and $\mathcal{M}_{\text{tar}}$ as the source and target domains with different dynamics, respectively. The performance difference of any policy π evaluated in $\mathcal{M}_{\text{src}}$ and $\mathcal{M}_{\text{tar}}$ can be bounded as below,

$$\eta_{\mathcal{M}_{\text{src}}}(\pi) - \eta_{\mathcal{M}_{\text{tar}}}(\pi) \le \frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2}\underbrace{\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\sqrt{\mathbb{E}_{P_{\text{src}}}[D_{\text{KL}}(P_{\text{src}}(s'_K \mid s,a) \,\|\, P_{\text{tar}}(s'_K \mid s,a))]}\right]}_{(a)\text{: initial latent state deviation}} + \frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2}\underbrace{\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\sqrt{\mathbb{E}_{P_{\text{src}}}\left[\sum_{k=1}^{K} D_{\text{KL}}\big(P_{\text{src}}(s'_{k-1} \mid s'_k, s, a) \,\|\, P_{\text{tar}}(s'_{k-1} \mid s'_k, s, a)\big)\right]}\right]}_{(b)\text{: latent state transition mismatch}}.$$

Proof. As the value function $V^\pi_{\mathcal{M}}(s)$ estimates the expected return of a policy π starting from state s in domain $\mathcal{M}$, and the rewards are bounded, we have $|V^\pi_{\mathcal{M}}(s)| \le r_{\max}/(1-\gamma), \forall s$. By using Lemma 7.1, we have:

$$\begin{aligned}
&\eta_{\mathcal{M}_{\text{src}}}(\pi) - \eta_{\mathcal{M}_{\text{tar}}}(\pi) = \frac{\gamma}{1-\gamma}\mathbb{E}_{\rho^\pi_{\text{src}}}[\mathbb{E}_{P_{\text{src}}}[r(s,a)] - \mathbb{E}_{P_{\text{tar}}}[r(s,a)]] \\
&= \frac{\gamma}{1-\gamma}\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\int_{s'_0} P_{\text{src}}(s'_0|s,a)V^\pi_{\text{tar}}(s'_0) - \int_{s'_0} P_{\text{tar}}(s'_0|s,a)V^\pi_{\text{tar}}(s'_0)ds'_0\right] \\
&\le \frac{\gamma}{1-\gamma}\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\int_{s'_0} (P_{\text{src}}(s'_0|s,a) - P_{\text{tar}}(s'_0|s,a))\,|V^\pi_{\text{tar}}(s'_0)|\,ds'_0\right] \\
&\le \frac{\gamma r_{\max}}{(1-\gamma)^2}\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\int_{s'_0} P_{\text{src}}(s'_0|s,a) - P_{\text{tar}}(s'_0|s,a)ds'_0\right] \\
&= \frac{\gamma r_{\max}}{(1-\gamma)^2}\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\int_{s'_{0:K}} P_{\text{src}}(s'_{0:K}|s,a) - P_{\text{tar}}(s'_{0:K}|s,a)ds'_{0:K}\right] \\
&= \frac{2\gamma r_{\max}}{(1-\gamma)^2}\mathbb{E}_{\rho^\pi_{\text{src}}}[D_{\text{TV}}(P_{\text{src}}(s'_{0:K}|s,a)||P_{\text{tar}}(s'_{0:K}|s,a))] \\
&\le \frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2}\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\sqrt{D_{\text{KL}}(P_{\text{src}}(s'_{0:K}|s,a)\|P_{\text{tar}}(s'_{0:K}|s,a))}\right] \quad \text{(a)} \\
&= \frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2}\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\sqrt{\mathbb{E}_{P_{\text{src}}}\left[\log\frac{P_{\text{src}}(s'_{0:K}|s,a)}{P_{\text{tar}}(s'_{0:K}|s,a)}\right]}\right] \\
&= \frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2}\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\sqrt{\mathbb{E}_{P_{\text{src}}}\left[\log\frac{P_{\text{src}}(s'_K|s,a)}{P_{\text{tar}}(s'_K|s,a)} + \sum_{k=1}^{K}\log\frac{P_{\text{src}}(s'_{k-1}|s'_k,s,a)}{P_{\text{tar}}(s'_{k-1}|s'_k,s,a)}\right]}\right] \quad \text{(b)} \\
&\le \frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2}\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\sqrt{\mathbb{E}_{P_{\text{src}}}[D_{\text{KL}}(P_{\text{src}}(s'_K|s,a)||P_{\text{tar}}(s'_K|s,a))]}\right] + \\
&\quad \frac{\sqrt{2}\gamma r_{\max}}{(1-\gamma)^2}\mathbb{E}_{\rho^\pi_{\text{src}}}\left[\sqrt{\mathbb{E}_{P_{\text{src}}}\left[\sum_{k=1}^{K} D_{\text{KL}}(P_{\text{src}}(s'_{k-1}|s'_k,s,a)||P_{\text{tar}}(s'_{k-1}|s'_k,s,a))\right]}\right] \quad \text{(c)}
\end{aligned}$$

where $D_{\mathrm{TV}}(P||Q)$ is the total variation distance between two distributions P and Q, the step (a) holds by Pinsker's inequality [39], the step (b) holds by the Markov property, and the step (c) holds by the subadditivity of the square root function. The proof shows that the performance difference can be controlled by the distributional divergence of latent states in generative trajectories.

REFERENCES

[1] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[2] L. Da, J. Turnau, T. P. Kutralingam, A. Velasquez, P. Shakarian, and H. Wei, "A survey of sim-to-real methods in rl: Progress, prospects and challenges with foundation models," *arXiv preprint arXiv:2502.13187*, 2025.

[3] K. Xu, C. Bai, X. Ma, D. Wang, B. Zhao, Z. Wang, X. Li, and W. Li, "Cross-domain policy adaptation via value-guided data filtering," *Advances in Neural Information Processing Systems*, vol. 36, pp. 73 395–73 421, 2023.

[4] J. Lyu, C. Bai, J. Yang, Z. Lu, and X. Li, "Cross-domain policy adaptation by capturing representation mismatch," *arXiv preprint arXiv:2405.15369*, 2024.

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

[6] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Conference on Robot Learning*. PMLR, 2020, pp. 1162–1176.

[7] A. Curtis, E. Li, M. Noseworthy, N. Gothoskar, S. Chitta, H. Li, L. P. Kaelbling, and N. E. Carey, "Flow-based domain randomization for learning and sequencing robotic skills," in *Forty-second International Conference on Machine Learning*, 2025.

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

[9] B. Eysenbach, S. Asawa, S. Chaudhari, S. Levine, and R. Salakhutdinov, "Off-dynamics reinforcement learning: Training for transfer with domain classifiers," *arXiv preprint arXiv:2006.13916*, 2020.

[10] J. Lyu, C. Bai, J. Yang, Z. Lu, and X. Li, "Cross-domain policy adaptation by capturing representation mismatch," in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 33 638–33 663.

[11] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. pmlr, 2015, pp. 2256–2265.

[12] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[13] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[14] Z. Xue, Q. Cai, S. Liu, D. Zheng, P. Jiang, K. Gai, and B. An, "State regularized policy optimization on data with dynamics shift," *Advances in neural information processing systems*, vol. 36, pp. 32 926–32 937, 2023.

[15] L. Da, M. Gao, H. Mei, and H. Wei, "Prompt to transfer: Sim-to-real transfer for traffic signal control with prompt learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 82–90.

[16] Y. Ge, A. Macaluso, L. E. Li, P. Luo, and X. Wang, "Policy adaptation from foundation model feedback," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 19 059–19 069.

[17] K.-C. Pan, M. Chen, Y.-D. Huang, X. Liu, and P.-C. Hsieh, "Cross-domain reinforcement learning under distinct state-action spaces via hybrid q functions."

[18] R. B. Slaoui, W. R. Clements, J. N. Foerster, and S. Toth, "Robust visual domain randomization for reinforcement learning," *arXiv preprint arXiv:1910.10537*, 2019.

[19] Y. Jiang, C. Li, W. Dai, J. Zou, and H. Xiong, "Variance reduced domain randomization for reinforcement learning with policy gradient," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 2, pp. 1031–1048, 2023.

[20] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv preprint arXiv:1803.11347*, 2018.

[21] Z. Wu, Y. Xie, W. Lian, C. Wang, Y. Guo, J. Chen, S. Schaal, and M. Tomizuka, "Zero-shot policy transfer with disentangled task representation of meta-reinforcement learning," *arXiv preprint arXiv:2210.00350*, 2022.

[22] D. S. Raychaudhuri, S. Paul, J. Vanbaar, and A. K. Roy-Chowdhury, "Cross-domain imitation from observations," in *International conference on machine learning*. PMLR, 2021, pp. 8902–8912.

[23] A. Fickinger, S. Cohen, S. Russell, and B. Amos, "Cross-domain imitation learning via optimal transport," *arXiv preprint arXiv:2110.03684*, 2021.

[24] Y. Guo, Y. Wang, Y. Shi, P. Xu, and A. Liu, "Off-dynamics reinforcement learning via domain adaptation and reward augmented imitation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 136 326–136 360, 2024.

[25] L. L. P. Van, H. T. Tran, and S. Gupta, "Policy learning for off-dynamics rl with deficient support," *arXiv preprint arXiv:2402.10765*, 2024.

[26] X. Wen, C. Bai, K. Xu, X. Yu, Y. Zhang, X. Li, and Z. Wang, "Contrastive representation for data filtering in cross-domain offline reinforcement learning," *arXiv preprint arXiv:2405.06192*, 2024.

[27] B. Kang, X. Ma, C. Du, T. Pang, and S. Yan, "Efficient diffusion policies for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 67 195–67 212, 2023.

[28] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, vol. 44, no. 10-11, pp. 1684–1704, 2025.

[29] C. Lu, P. Ball, Y. W. Teh, and J. Parker-Holder, "Synthetic experience replay," *Advances in Neural Information Processing Systems*, vol. 36, pp. 46 323–46 344, 2023.

[30] H. He, C. Bai, K. Xu, Z. Yang, W. Zhang, D. Wang, B. Zhao, and X. Li, "Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning," *Advances in neural information processing systems*, vol. 36, pp. 64 896–64 917, 2023.

[31] Y. Wang, L. Wang, Y. Jiang, W. Zou, T. Liu, X. Song, W. Wang, L. Xiao, J. Wu, J. Duan, *et al.*, "Diffusion actor-critic with entropy regulator," *Advances in Neural Information Processing Systems*, vol. 37, pp. 54 183–54 204, 2024.

[32] Z. Zhu, M. Liu, L. Mao, B. Kang, M. Xu, Y. Yu, S. Ermon, and W. Zhang, "Madiff: Offline multi-agent learning with diffusion models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 4177–4206, 2024.

[33] L. L. P. Van, M. H. Nguyen, D. Kieu, H. Le, H. T. Tran, and S. Gupta, "Dmc: Nearest neighbor guidance diffusion model for offline cross-domain reinforcement learning," *arXiv preprint arXiv:2507.20499*, 2025.

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

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

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

[37] J. Lyu, K. Xu, J. Xu, J.-W. Yang, Z. Zhang, C. Bai, Z. Lu, X. Li, *et al.*, "Odrl: A benchmark for off-dynamics reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 59 859–59 911, 2024.

[38] Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma, "Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees," *arXiv preprint arXiv:1807.03858*, 2018.

[39] I. Csiszár and J. Körner, *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press, 2011.