CROSS-DOMAIN ENERGY-GUIDED DIFFUSION GENERATION FOR OFF-DYNAMICS REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Off-dynamics offline reinforcement learning seeks to learn a target-domain policy from a large source dataset and a limited target dataset under mismatched transition dynamics. Existing approaches such as reward augmentation and data filtering are constrained to the existing source dataset and thus have limited coverage beyond the collected dataset. Recent model-based methods address this issue by learning target-aware dynamics, but are limited to transition-level generation, which can lead to accumulated errors over long horizons. These limitations motivate trajectory-level generation for off-dynamics offline RL. We propose CEDGE, a Cross-domain Energy-guided Diffusion GEneration framework. CEDGE trains a trajectory diffusion model on source-domain trajectories and adapts the generated samples to the target domain through energy guidance. This guidance is derived from the mismatch between source and target trajectory distribution and decomposed into return, domain, and policy energies for high return, target-dynamics compatibility, and source behavior correction respectively. The resulting energy-guided trajectories are useful both for direct planning and as synthetic data for policy learning. Compared with previous methods, CEDGE performs target adaptation through energy guidance, and the source trained diffusion model can be reused across target dynamics without retraining. Our theoretical analysis studies generation error and downstream policy performance. Empirical results demonstrate that trajectory-level energy-guided generation improves diffusion planning under dynamics shifts and produces synthetic data that improves downstream target-domain policy learning.

1 INTRODUCTION

Off-dynamics offline reinforcement learning (RL) (Kaelbling et al., 1996; Eysenbach et al., 2020; Liu & Xu, 2024; Guo et al., 2024; Wang et al., 2026) considers a data regime where a policy has access to a large source dataset and a limited target dataset under shifted transition dynamics. The source dataset typically contains abundant data collected from the source environment, whose dynamics are different from the target environment. In contrast, the limited target dataset is often insufficient for learning a good policy in the target domain, as no additional interaction with the target environment is permitted during training. Therefore, the primary challenge is how to leverage the abundant source-domain experience to improve target-domain policy learning without introducing detrimental dynamics mismatch.

Existing off-dynamics offline RL methods offer partial solutions to this challenge. Reward augmentation techniques modify source rewards using domain classifiers that estimate the dynamics discrepancy between the source and target domains (Eysenbach et al., 2020; Liu et al., 2022; Lyu et al., 2024a). Data filtering selects source samples that resemble the target domain and discards samples prone to harmful transfer (Xu et al., 2023; Wen et al., 2024; Lyu et al., 2025; Xia et al., 2026). While these methods can mitigate negative and harmful effect that the highly mismatched source data induced for target policy training, both approaches are fundamentally limited to the trajectories that have already been collected in the source dataset. Neither method can synthesize novel target-domain behavior to improve data coverage. Recently, a model-based method, MOBODY (Guo et al., 2026), addresses this limitation by learning target-aware dynamics models and gener-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ating synthetic rollouts for policy learning. Thus, policy optimization can extend beyond the initial offline dataset via interaction with the target-aware dynamics model. Despite this advancement, the generated rollouts are still constructed at the transition level. As the rollout horizon increases, even minor one-step prediction errors can compound, causing later states to drift away from the true target-domain state distribution. Consequently, compounding prediction errors can limit transition-level generation over long horizons, which motivates our focus on trajectory-level generation. As shown in Figure 1, transition level rollout generation accumulates prediction errors and deviates from the target reference trajectory, while trajectory level generation is closely aligned with the target trajectory. This motivates trajectories generation with a generative model rather than generating transitions step by step.

Diffusion models (Ho et al., 2020; Song et al., 2021a) provide solutions for modeling RL trajectory distributions both for planning (Janner et al., 2022; Wang et al., 2022) and generating synthetic data for downstream policy learning (Lu et al., 2023b; Jackson et al., 2024). However, merely training a trajectory diffusion model on source-domain trajectories will not work, as the generated samples retain the source transition dynamics and behavior patterns. In contrast, training a trajectory diffusion model directly on the limited target dataset is challenging due to insufficient target-domain coverage. Several recent works leverage diffusion model solving off-dynamics RL problems by incorporating target data into diffusion model training (Niu et al., 2024; Ng et al., 2024). Target adaptation is tied to diffusion training, so adapting to new target environments requires additional expensive diffusion model optimization. This motivates our proposed method by learning a reusable diffusion model from abundant source data. Then, we can perform the planning and synthetic data generation on any new target environment through energy guidance without retraining diffusion models.

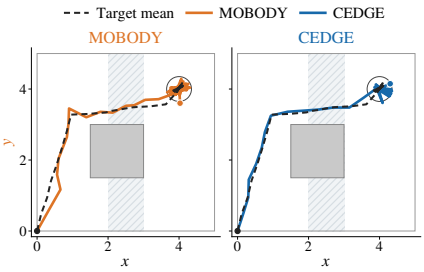

Figure 1: Controlled shifted example comparing transition-level rollout generation (MOBODY) and trajectory-level generation (CEDGE). MOBODY recursively predicts next states and deviates from the target reference due to accumulated errors, while CEDGE jointly generates the trajectory and remains closer to the target reference. More details are in Section 5.2.

In this paper, we propose CEDGE, a cross-domain energy-guided diffusion generation framework for off-dynamics offline RL. Our method first trains a reusable trajectory diffusion model using source-domain trajectories and then employs energy-guided sampling to steer the generated samples toward the target domain. The resulting energy-guided trajectories facilitate two downstream applications: direct diffusion planning and offline policy optimization. We formally derive the energy guidance from the distributional mismatch between the source trajectory distribution and the desired target-domain trajectory distribution. This guidance decomposes into return, domain, and policy energy terms, which encourage high-return trajectories, target-domain dynamic compatibility, and effective compensation for source behavior bias, respectively. By guiding full trajectories during the diffusion sampling process with energy guidance, we can generate trajectories more compatible with the target domain and effectively mitigate the accumulated errors inherent in transition-level generation with long-horizon step by step rollouts. Moreover, compared with previous methods, CEDGE performs target adaptation through energy guidance, and the source-trained diffusion model can be reused across target dynamics without retraining. We evaluate our approaches on the ODRL benchmark across environments and dynamics shifts, and demonstrate that CEDGE improves diffusion planning and downstream target-domain policy learning.

2 Most Relevant Work

We present the work most relevant to ours. A comprehensive overview can be found in Appendix C.

Diffusion models in RL. Diffusion models (Ho et al., 2020; Karras et al., 2022) have recently emerged as a powerful paradigm for decision-making and offline reinforcement learning. A significant line of work utilizes diffusion models for trajectory-level planning such as Diffuser (Janner et al., 2022) and Veteran (Lu et al., 2025). Another line of work uses diffusion models as expressive

policy classes for offline RL. Diffusion-QL (Wang et al., 2022) and IDQL (Hansen-Estruch et al., 2023) use diffusion-based action modeling combined with value-guided policy improvement. (Lu et al., 2023b; Jackson et al., 2024) have also explored using diffusion models as data generators for downstream policy learning. Compared to these works, our method leverages diffusion not just as a planner or a policy class, but as a source-domain trajectory prior that can be adapted to novel target dynamics through energy guidance and subsequently coupled with offline policy optimization.

Off-dynamics RL. Off-dynamics reinforcement learning (Eysenbach et al., 2020; Liu et al., 2022; 2024c; Tang et al., 2024) addresses the challenge of transferring policies across domains with mismatched transition dynamics. A common approach is reward augmentation based on domain classifiers such as DARC (Eysenbach et al., 2020) and DARA (Liu et al., 2022). Data filtering methods utilize contrastive representations (Wen et al., 2024), optimal transport (Lyu et al., 2025), or localized dynamics-aware filtering (Xia et al., 2026) to select data from the source dataset. More recently, a model-based approach, MOBODY (Guo et al., 2026), attempts to surpass the dataset support constraint by learning target-aware dynamics and generating synthetic transitions for policy learning. xTED (Niu et al., 2024) performs diffusion-based trajectory editing, while DmC (Van et al., 2025) uses nearest-neighbor guidance to steer diffusion generation toward the target domain. Our work differs from these methods by learning a reusable source trajectory distribution and adapting its generation to target dynamics through energy guidance.

3 PRELIMINARY

Off-dynamics Offline RL We begin by defining reinforcement learning with the standard Markov Decision Process (MDP) which is characterized by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma, \rho_0)$. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces respectively; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function; $p(s' \mid s, a)$ defines the transition dynamics; $\gamma \in (0, 1)$ is the discount factor; and ρ_0 is the initial-state distribution. In the off-dynamics setting, we consider two distinct operational domains: a source domain $\mathcal{M}_{\rm src} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, p_{\rm src}, \gamma, \rho_0)$ and a target domain $\mathcal{M}_{\rm trg} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, p_{\rm trg}, \gamma, \rho_0)$. These two environments share the same state space, action space, reward function, and discount factor, but crucially differ in their underlying transition dynamics, such that $p_{\rm src}(s' \mid s, a) \neq p_{\rm trg}(s' \mid s, a)$.

Our work specifically focuses on the challenging paradigm of off-dynamics offline RL. We assume access to a large, pre-collected offline dataset from the source domain $\mathcal{D}_{\rm src}$, and a limited offline dataset from the target domain $\mathcal{D}_{\rm trg}$. Both datasets are collected by corresponding behavior policies in their respective environments, and importantly, no further environment interactions are permitted during the training process. The objective is to learn a policy π that maximizes the expected discounted return within the target domain: $J_{\rm trg}(\pi) = \mathbb{E}_{\pi, p_{\rm trg}, \rho_0}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$.

Generative Modeling via Diffusion Models Diffusion models learn a data distribution through a two-stage framework: a forward noising process and a reverse denoising process (Ho et al., 2020). The forward process progressively perturbs the input data with Gaussian noise, while the reverse process is trained to iteratively remove this noise, ultimately recovering a sample from the original data distribution. In our setting, the data sample is defined as a trajectory segment. Let $\tau_0 = (s_0, a_0, \ldots, s_{H-1}, a_{H-1}, s_H)$ denote a trajectory segment of horizon H. We model this process across T diffusion steps. For each step $k \in \{1, \ldots, T\}$, the noisy trajectory follows: $q(\tau_k \mid \tau_0) = \mathcal{N}(\tau_k; \sqrt{\bar{\alpha}_k}\, \tau_0, (1 - \bar{\alpha}_k) I)$, where $\beta_k \in (0, 1)$ is the noise variance at step k, $\alpha_k = 1 - \beta_k$, and $\bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i$. Following existing diffusion planning methods (Janner et al., 2022; Lu et al., 2025), we train a denoiser $D_\theta(\tau_k, k)$ on the source-domain trajectories ($\mathcal{D}_{\rm src}$) to predict the clean trajectory τ_0 from a noisy sample τ_k. The training objective is:

$$\mathcal{L}_{\rm diff}(\theta) = \mathbb{E}_{\tau_0 \sim \mathcal{D}_{\rm src},\, k \sim {\rm Unif}(\{1,\ldots,T\}),\, \epsilon \sim \mathcal{N}(0,I)} \big[\, \| D_\theta(\tau_k, k) - \tau_0 \|_2^2 \,\big], \tag{3.1}$$

where $\tau_k = \sqrt{\bar{\alpha}_k}\tau_0 + \sqrt{1 - \bar{\alpha}_k}\epsilon$. After training, this denoiser defines a trajectory prior over the source-domain data distribution. At inference time, we iteratively denoise Gaussian noise to generate trajectories that approximate the source domain trajectory distribution.

4 CROSS-DOMAIN ENERGY-GUIDED DIFFUSION GENERATION

In this section, we introduce our CEDGE. CEDGE first trains a trajectory diffusion model on the source dataset $\mathcal{D}_{\rm src}$ using the diffusion objective in (3.1) and details can be found in Algorithm 1 in

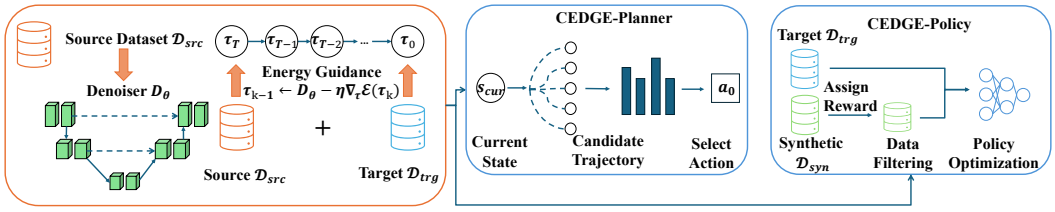

Figure 2: Overview of CEDGE. A source-domain trajectory diffusion model is adapted to target dynamics through learned energy guidance. The resulting guided trajectories can be utilized either for direct planning or as high-quality synthetic data for downstream policy optimization.

Appendix E. We then derive the trajectory-level energy guidance necessary to adapt a source-domain diffusion prior to the target domain. Subsequently, we detail how the three distinct energy modules are learned, and lastly, we explain how the guided sampler is employed for both policy planning and synthetic-data generation. The overall architecture is summarized in Figure 2.

4.1 Trajectory-Level Energy Guidance

We train the trajectory diffusion model on the source dataset $\mathcal{D}_{\text{src}}$ and guide its generated trajectories to each target domain using energy guidance. We derive this guidance from the difference between the source trajectory distribution and the desired target trajectory distribution.

Target and source trajectory distributions. Let $\tau = (s_0, a_0, \ldots, s_{H-1}, a_{H-1}, s_H)$ be a trajectory segment of horizon H. Following the probabilistic inference perspective in RL (Levine, 2018; Eysenbach et al., 2020), given the assumption that trajectory optimality is proportional to the exponential of the cumulative reward, the desired target-domain trajectory distribution is defined as:

$$p^\star(\tau) \propto \rho_0(s_0)\big(\textstyle\prod_{t=0}^{H-1} p_{\text{trg}}(s_{t+1} \mid s_t, a_t)\big) \exp\big(\textstyle\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t)\big), \tag{4.1}$$

where $\rho_0(s_0)$ is the initial-state distribution, $p_{\text{trg}}(s_{t+1} \mid s_t, a_t)$ is the target-domain transition dynamics, and the exponential return term weights trajectories proportionally to their discounted return. This distribution represents the *ideal* target trajectories, not merely the empirical distribution of the limited $\mathcal{D}_{\text{trg}}$. The source trajectory distribution $q_{\text{src}}(\tau)$ is induced by the source domain transition dynamics $p_{\text{src}}(s_{t+1} \mid s_t, a_t)$ and the behavior policy $\pi_\beta(a_t \mid s_t)$ in source dataset:

$$q_{\text{src}}(\tau) = \rho_0(s_0)\big(\textstyle\prod_{t=0}^{H-1} p_{\text{src}}(s_{t+1} \mid s_t, a_t)\pi_\beta(a_t \mid s_t)\big). \tag{4.2}$$

Energy decomposition. We assume a common initial-state distribution, ρ_0, for both domains. Since the trajectory diffusion model is trained on $\mathcal{D}_{\text{src}}$, it approximates $q_{\text{src}}(\tau)$. To adapt its generated samples to the target domain, we define $p^\star(\tau) \propto q_{\text{src}}(\tau)\exp\big(-\mathcal{E}(\tau)\big)$, where $\mathcal{E}(\tau)$ is the energy function on clean trajectories. Therefore, $\mathcal{E}(\tau) = \log q_{\text{src}}(\tau) - \log p^\star(\tau) + C$, where C is independent of τ. Substituting (4.1) and (4.2), the common initial-state distribution cancels, yielding

$$\mathcal{E}(\tau) = \textstyle\sum_{t=0}^{H-1}[\log p_{\text{src}}(s_{t+1} \mid s_t, a_t) - \log p_{\text{trg}}(s_{t+1} \mid s_t, a_t) - \gamma^t r(s_t, a_t) + \log \pi_\beta(a_t \mid s_t)] + C.$$

The energy can be decomposed into three additive terms: domain energy ($\mathcal{E}_1$), return energy ($\mathcal{E}_2$), and policy energy ($\mathcal{E}_3$): $\mathcal{E}_1(\tau) = \sum_{t=0}^{H-1}[\log p_{\text{src}}(s_{t+1} \mid s_t, a_t) - \log p_{\text{trg}}(s_{t+1} \mid s_t, a_t)]$, $\mathcal{E}_2(\tau) = -\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t)$, and $\mathcal{E}_3(\tau) = \sum_{t=0}^{H-1} \log \pi_\beta(a_t \mid s_t)$. Here, $\mathcal{E}_1$ favors transitions more consistent with the target dynamics, $\mathcal{E}_2$ favors higher-return trajectories, and $\mathcal{E}_3$ compensates for the source behavior policy in $q_{\text{src}}(\tau)$. The derivation is for clean trajectories $\tau = \tau_0$. During reverse diffusion, we approximate the required noisy-time guidance using three timestep-conditioned energy functions.

$$\mathcal{E}(\tau_k, k) = \lambda_1\mathcal{E}_1(\tau_k, k) + \lambda_2\mathcal{E}_2(\tau_k, k) + \lambda_3\mathcal{E}_3(\tau_k, k). \tag{4.3}$$

These practical guidance functions do not necessarily equal the exact noisy-time energy gradient; their discrepancy is captured by ϵ_g in Proposition 4.1.

4.2 Learning Energy Guidance Functions

We now detail how the energy functions in (4.3) are learned from offline data. For each clean training trajectory τ_0, we sample a diffusion timestep $k \sim \text{Uniform}(\{1, \ldots, T\})$ and Gaussian noise

$\epsilon \sim \mathcal{N}(0, I)$, and construct $\tau_k = \sqrt{\bar{\alpha}_k}\tau_0 + \sqrt{1-\bar{\alpha}_k}\epsilon$, using the same forward diffusion schedule as the source trajectory diffusion model. All three energy modules are trained on these noisy samples and conditioned on the corresponding timestep k. We denote the state and action components of the noisy trajectory τ_k by $s_{t,k}$ and $a_{t,k}$, where t is environment steps and k is diffusion steps. The energy function training and energy-guided sampling procedures are summarized in Algorithm 2 and Algorithm 3 in Appendix E.

Domain energy function. To implement $\mathcal{E}_1(\tau_k, k)$, which quantifies the transition dynamics mismatch, we adapt prior work (Eysenbach et al., 2020; Liu et al., 2022) and utilize two domain classifiers. The detailed derivation is provided in Appendix D. We can formulate the directional gradient of the energy guidance as:

$$\nabla_{\tau_k}\mathcal{E}_1(\tau_k, k) = \nabla_{\tau_k}\sum_{t=0}^{H-1}\Big[\log p_{\theta_{\mathrm{SAS}}}(\mathrm{src} \mid s_{t,k}, a_{t,k}, s_{t+1,k}, k) - \log p_{\theta_{\mathrm{SAS}}}(\mathrm{trg} \mid s_{t,k}, a_{t,k}, s_{t+1,k}, k) - \log p_{\theta_{\mathrm{SA}}}(\mathrm{src} \mid s_{t,k}, a_{t,k}, k) + \log p_{\theta_{\mathrm{SA}}}(\mathrm{trg} \mid s_{t,k}, a_{t,k}, k)\Big].$$

This gradient utilizes two domain classifiers: $p_{\theta_{\mathrm{SAS}}}$ classifies the noisy transition $(s_{t,k}, a_{t,k}, s_{t+1,k})$ as target or source, and $p_{\theta_{\mathrm{SA}}}$ classifies the noisy state-action pair $(s_{t,k}, a_{t,k})$ as target or source. The classifiers are trained by minimizing the following cross-entropy loss functions:

$$\begin{aligned}\mathcal{L}_{\mathrm{SAS}} = &-\mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}_{\mathrm{src}},\,k}\big[\log p_{\theta_{\mathrm{SAS}}}(\mathrm{src} \mid s_{t,k}, a_{t,k}, s_{t+1,k}, k)\big] \\ &- \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}_{\mathrm{trg}},\,k}\big[\log p_{\theta_{\mathrm{SAS}}}(\mathrm{trg} \mid s_{t,k}, a_{t,k}, s_{t+1,k}, k)\big], \qquad (4.4)\\ \mathcal{L}_{\mathrm{SA}} = &-\mathbb{E}_{(s_t,a_t)\sim\mathcal{D}_{\mathrm{src}},\,k}\big[\log p_{\theta_{\mathrm{SA}}}(\mathrm{src} \mid s_{t,k}, a_{t,k}, k)\big] \\ &- \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}_{\mathrm{trg}},\,k}\big[\log p_{\theta_{\mathrm{SA}}}(\mathrm{trg} \mid s_{t,k}, a_{t,k}, k)\big]. \qquad (4.5)\end{aligned}$$

where $\mathcal{D}_{\mathrm{src}}$ and $\mathcal{D}_{\mathrm{trg}}$ denote the source and target datasets, respectively.

Return energy function. Next, we implement the return guidance term $\mathcal{E}_2(\tau_k, k)$, which steers the trajectory generation towards high-return trajectories. Since $\mathcal{E}_2(\tau_k, k)$ must assign lower energy to trajectories with larger returns, we train a differentiable return predictor $J_\psi(\tau_k, k)$ that estimates the discounted cumulative reward for a trajectory segment τ_0. We train this predictor on the combined trajectory dataset $\mathcal{D}_{\mathrm{src}} \cup \mathcal{D}_{\mathrm{trg}}$ by minimizing the standard regression loss:

$$\mathcal{L}_R = \mathbb{E}_{\tau_0\sim\mathcal{D}_{\mathrm{src}}\cup\mathcal{D}_{\mathrm{trg}},\,k}\big[\big(J_\psi(\tau_k, k) - \textstyle\sum_{t=0}^{H-1}\gamma^t r(s_t, a_t)\big)^2\big]. \qquad (4.6)$$

The energy guidance is derived from the predictor's gradient: $\nabla_{\tau_k}\mathcal{E}_2(\tau_k, k) = -\nabla_{\tau_k}J_\psi(\tau_k, k)$. This ensures that the reverse diffusion process is guided towards trajectories with higher predicted return.

Policy energy function. Lastly, we address the third guidance term, $\mathcal{E}_3(\tau_k, k)$, which compensates for the source behavior policy bias embedded in the source diffusion model. Since the true source behavior policy π_β is unknown, we approximate it by learning a Gaussian policy model, $\pi_\omega(a_k \mid s_k, k)$, via behavior cloning on $\mathcal{D}_{\mathrm{src}}$. We train π_ω by maximizing the likelihood of source actions, equivalent to minimizing the negative log-likelihood:

$$\mathcal{L}_\pi = -\mathbb{E}_{(s_t,a_t)\sim\mathcal{D}_{\mathrm{src}},\,k}\left[\log \pi_\omega(a_{t,k} \mid s_{t,k}, k)\right]. \qquad (4.7)$$

Using the learned Gaussian behavior policy, we define the policy energy guidance as $\mathcal{E}_3(\tau_k, k) = \sum_{t=0}^{H-1}\log\pi_\omega(a_{t,k} \mid s_{t,k}, k)$, and its gradient with respect to the trajectory is given by: $\nabla_{\tau_k}\mathcal{E}_3(\tau_k, k) = \nabla_{\tau_k}\sum_{t=0}^{H-1}\log\pi_\omega(a_{t,k} \mid s_{t,k}, k)$. This gradient is applied during reverse diffusion to mitigate the bias toward source domain action preferences inherited from the offline data.

4.3 Guided Trajectories for Planning and Policy Learning

Diffusion as the planner. The trained source trajectory diffusion model can be directly employed as a planner at inference time (Janner et al., 2022; Lu et al., 2025). Given the current state s_{cur}, we generate multiple energy-guided trajectory candidates of horizon H by running the reverse diffusion process while fixing the first state to s_{cur}. At each diffusion step k, the current noisy trajectory τ_k is guided by the weighted energy $\mathcal{E}(\tau_k, k)$ defined in (4.3). After reverse diffusion, we obtain a set of clean trajectory candidates $\{\hat{\tau}_0^{(n)}\}_{n=1}^N$. We then select the trajectory with the lowest return energy, $\hat{\tau}_0^\star = \arg\min_{\hat{\tau}_0^{(n)}}\mathcal{E}_2(\hat{\tau}_0^{(n)})$, which corresponds to the highest predicted return. We execute its first action and replan from the next observed state.

Policy learning with synthetic data. Alternatively, we use the energy-guided trajectories as synthetic data for offline policy learning. We first generate synthetic trajectories using the learned diffusion model and energy guidance. To assign rewards to the generated transitions, we train a reward model $\hat{r}_\eta(s_t, a_t, s_{t+1})$ on the observed rewards from the source and target datasets by minimizing

$$\mathcal{L}_{\text{reward}} = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}_{\text{src}}\cup\mathcal{D}_{\text{trg}}}\left[\hat{r}_\eta(s_t, a_t, s_{t+1}) - r_t\right]^2. \tag{4.8}$$

The learned reward model is then used to annotate transitions from the generated trajectories. Synthetic trajectories may still contain transitions that are unlikely under the target dynamics or have low predicted returns. We therefore filter the synthetic trajectories in two steps. First, we keep trajectories with lower domain energy $\mathcal{E}_1(\hat{\tau}_0)$, which favors trajectories with larger estimated target-to-source transition-density ratios. We then keep trajectories with lower return energy $\mathcal{E}_2(\hat{\tau}_0)$, corresponding to higher predicted returns. The retained trajectories form the synthetic dataset $\mathcal{D}_{\text{syn}}$. Finally, we train the target domain policy with IQL (Kostrikov et al., 2021) on $\mathcal{D}_{\text{syn}} \cup \mathcal{D}_{\text{trg}}$.

4.4 Theoretical Analysis

In this section, we study two questions: how closely energy-guided samples follow the desired trajectory distribution and how their errors affect policy learning with synthetic and real target data. More details and proofs can be found in Appendix A.

Proposition 4.1 (Energy-guided generation accuracy)**.** Let P_{syn} be the generated trajectory distribution and let $p^\star$ be the desired distribution in (4.1). Under Assumption A.1,

$$D_{\text{TV}}(P_{\text{syn}}, p^\star) \le \epsilon_{\text{num}} + \sqrt{(\epsilon_T + \epsilon_s^2 + \epsilon_g^2)/2} =: B_{\text{gen}}. \tag{4.9}$$

Here ϵ_T measures the residual mismatch between the Gaussian initialization in sampling and the noised target distribution at the terminal noise level, which decreases as the terminal noise level increases. ϵ_s measures the error between the learned source score and the exact source score, and ϵ_g measures the error between the applied guidance and the exact noisy time guidance required for $p^\star$. The term ϵ_{num} captures the additional error from replacing the corresponding continuous reverse diffusion with the finite-step sampler.

We next study policy learning from a mixture of real target and synthetic transitions. Let n and m denote their respective sample sizes. We denote $N = n + m$ and $\zeta = m/N$. We assume bounded rewards and define $V_{\max} = R_{\max}/(1-\gamma)$. Let ϵ_r measure reward-prediction error on synthetic data and ϵ_{ref} measure the residual mismatch between the reference distribution $p^\star$ and the true target dynamics. We analyze the finite class minimax value learner of Chen & Jiang (2019), where $\mathcal{F}$ is the candidate Q-function class and $\mathcal{G} \supseteq \mathcal{F}$ is the regression function class. The learner, mixed Bellman operator, and coverage assumptions are defined in Appendix A.2.

Theorem 4.2 (Finite-sample policy performance)**.** Under the realizability, completeness, independence, and coverage conditions in Assumption A.1 and Assumption A.2, let $\pi^\star$ be an optimal target policy and $\widehat{\pi}$ be the greedy policy returned by the learner and let C_ζ denote the target policy coverage coefficient of the mixed dataset. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$J_{\text{trg}}(\pi^\star) - J_{\text{trg}}(\widehat{\pi}) \le \underbrace{\frac{10V_{\max}}{1-\gamma}\sqrt{\frac{C_\zeta \log(2|\mathcal{F}||\mathcal{G}|/\delta)}{n+m}}}_{\text{finite-sample error}} + \underbrace{\frac{2\zeta C_\zeta}{1-\gamma}\left[\epsilon_r + \gamma V_{\max}\left(2B_{\text{gen}} + \epsilon_{\text{ref}}\right)\right]}_{\text{synthetic-data bias}}. \tag{4.10}$$

Remark 4.3. The result separates the policy error into a finite-sample term and a synthetic-data bias term. For fixed coverage coefficient and function classes, the finite-sample term scales as $(n + m)^{-1/2}$. The synthetic-data bias depends directly on the generation error B_{gen}. A smaller B_{gen} means that the generated trajectories are closer to the desired distribution, which reduces the bias introduced by synthetic data and tightens the policy-performance bound. The remaining bias also depends on the reward prediction error ϵ_r and the target-dynamics mismatch ϵ_{ref}. Therefore, adding synthetic data can improve the bound through a larger sample size, but its benefit depends on the quality of the generated data.

5 EXPERIMENTS

In this section, we conduct experiments using the benchmark dataset provided by ODRL (Lyu et al., 2024b) to rigorously evaluate the performance of CEDGE in solving off-dynamics offline RL problems. We evaluate CEDGE across MuJoCo environments under various levels of dynamics shift.

5.1 MAIN EXPERIMENTS ON LOCOMOTION TASKS

Task and Environments. We evaluate CEDGE on MuJoCo environments with diverse dynamics shifts, following the protocol established in the ODRL benchmark (Lyu et al., 2024b). In the main experiments, we focus on two primary types of dynamics shifts: gravity and friction. We also extend experiments to kinematic and morphology shifts in Appendix H.1. For gravity and friction shifts, we rescaled the corresponding environment parameter by factors of $\{0.1, 0.5, 2.0, 5.0\}$ to introduce a range of dynamics mismatches. The source domain remains the standard, unmodified MuJoCo environment and is shared across all target domains. The target domains are created by applying the specified gravity or friction rescaling factors. This setup matches the fast-adaptation setting targeted by CEDGE. Since the source domain trajectory diffusion model can be pre-trained once on the source dataset and reused across multiple target domains, only the target-specific energy guidance functions need to be learned for each shifted target environment. We use the dataset in ODRL (Lyu et al., 2024b). More details are available in Appendix F.2.

Table 1: Performance of our methods and baselines on MuJoCo tasks (HalfCheetah, Ant, Walker2d, Hopper) under medium-level offline datasets with dynamics shifts in gravity and friction at levels $\{0.1, 0.5, 2.0, 5.0\}$. Source domains remain unchanged; target domains are shifted. We report normalized target-domain scores (mean $\pm$ std over three seeds). Best and second-best mean scores are marked in each row.

Env	Level	BOSA	IQL	TD3-BC	DARA	MOBODY	OTDF	Diffusion-based Methods				
								DmC	xTED	Diffuser	CEDGE Planner	CEDGE Policy
HalfCheetah Gravity	0.1	9.31 ± 1.94	9.62 ± 4.27	6.90 ± 0.34	12.90 ± 1.01	14.18 ± 1.06	8.03 ± 0.37	12.58 ± 3.38	11.64 ± 2.08	11.19 ± 1.18	10.60 ± 1.84	**20.04 ± 3.52**
	0.5	43.96 ± 5.68	44.23 ± 2.93	6.38 ± 3.91	46.11 ± 1.93	47.18 ± 1.23	38.60 ± 3.51	43.62 ± 0.14	2.70 ± 1.17	48.45 ± 0.68	**49.84 ± 0.17**	2.54 ± 0.59
	2.0	27.86 ± 0.94	31.34 ± 1.68	29.29 ± 3.62	31.85 ± 1.31	41.60 ± 7.35	28.55 ± 1.61	27.24 ± 1.04	**41.99 ± 2.02**	27.38 ± 0.05	27.19 ± 0.06	41.66 ± 5.84
	5.0	17.95 ± 11.97	44.00 ± 23.13	73.75 ± 14.11	27.67 ± 17.01	**83.05 ± 1.21**	72.69 ± 0.71	42.33 ± 25.21	45.42 ± 16.70	20.33 ± 3.22	4.04 ± 0.07	75.34 ± 1.63
HalfCheetah Friction	0.1	12.53 ± 3.61	26.39 ± 11.35	8.95 ± 0.71	23.69 ± 16.46	57.53 ± 2.49	8.90 ± 0.56	8.95 ± 0.36	54.08 ± 2.29	10.70 ± 0.48	10.55 ± 0.05	**64.31 ± 0.39**
	0.5	68.93 ± 0.35	69.80 ± 0.64	49.43 ± 9.91	64.89 ± 3.04	69.54 ± 0.48	67.03 ± 0.74	68.47 ± 0.25	66.56 ± 1.79	71.25 ± 0.39	**72.13 ± 0.27**	52.06 ± 6.41
	2.0	46.53 ± 0.37	46.04 ± 2.04	43.51 ± 0.74	46.25 ± 2.36	**50.02 ± 3.26**	45.60 ± 0.53	45.10 ± 0.30	40.33 ± 2.86	46.91 ± 0.62	47.93 ± 0.14	49.15 ± 4.03
	5.0	44.07 ± 9.07	44.96 ± 6.78	35.83 ± 6.65	40.06 ± 7.87	59.20 ± 4.91	36.15 ± 2.43	28.22 ± 2.80	62.37 ± 8.16	32.11 ± 1.80	32.79 ± 0.91	**68.56 ± 1.94**
Ant Gravity	0.1	25.58 ± 2.21	12.53 ± 1.11	13.23 ± 2.61	11.03 ± 1.24	37.09 ± 2.12	15.54 ± 1.25	16.23 ± 1.35	30.55 ± 2.21	11.92 ± 1.57	11.12 ± 0.11	**37.99 ± 1.13**
	0.5	19.03 ± 4.41	10.09 ± 2.00	12.91 ± 2.85	9.04 ± 1.35	**37.44 ± 2.79**	12.59 ± 1.58	26.36 ± 1.89	27.93 ± 2.08	12.44 ± 2.01	7.13 ± 0.24	35.16 ± 4.51
	2.0	41.77 ± 1.52	37.17 ± 0.96	34.04 ± 4.12	36.64 ± 0.82	**45.83 ± 1.71**	37.31 ± 1.87	42.77 ± 3.62	37.76 ± 2.75	35.84 ± 1.32	34.11 ± 0.96	44.42 ± 1.26
	5.0	31.94 ± 0.69	31.59 ± 0.35	6.37 ± 0.45	31.01 ± 0.39	65.45 ± 3.23	34.56 ± 0.09	38.38 ± 2.69	45.02 ± 9.84	10.46 ± 1.08	27.42 ± 0.12	**71.46 ± 6.56**
Ant Friction	0.1	58.95 ± 0.71	55.56 ± 0.46	49.20 ± 2.55	55.12 ± 0.24	58.79 ± 0.11	53.00 ± 0.63	50.97 ± 1.02	21.45 ± 12.38	57.07 ± 3.41	48.41 ± 0.31	**61.85 ± 3.94**
	0.5	59.72 ± 3.57	59.28 ± 0.80	25.21 ± 7.17	58.92 ± 0.80	**62.41 ± 4.10**	56.73 ± 2.97	47.63 ± 3.15	51.84 ± 7.98	58.98 ± 4.19	42.36 ± 0.56	54.06 ± 1.93
	2.0	20.18 ± 3.79	19.84 ± 3.20	22.69 ± 8.10	17.54 ± 2.47	47.41 ± 4.40	29.75 ± 6.63	56.90 ± 5.73	10.35 ± 3.77	27.41 ± 2.31	10.66 ± 0.49	**57.49 ± 4.38**
	5.0	9.07 ± 0.88	7.75 ± 0.25	10.06 ± 4.16	7.80 ± 0.12	31.17 ± 5.57	13.10 ± 3.50	12.29 ± 2.13	38.94 ± 1.85	15.67 ± 5.24	12.48 ± 0.99	**60.31 ± 9.32**
Walker2d Gravity	0.1	18.75 ± 12.02	16.04 ± 7.60	36.48 ± 0.95	20.12 ± 5.74	65.85 ± 5.08	65.30 ± 5.94	64.03 ± 6.60	55.15 ± 4.08	19.78 ± 0.73	34.08 ± 0.78	**69.48 ± 1.57**
	0.5	40.09 ± 20.37	42.05 ± 10.52	27.43 ± 3.92	29.72 ± 16.02	43.57 ± 2.32	37.72 ± 3.43	34.25 ± 5.06	23.80 ± 6.93	26.22 ± 1.95	**47.84 ± 1.52**	44.36 ± 1.07
	2.0	8.91 ± 2.28	25.69 ± 10.70	11.88 ± 9.38	32.20 ± 1.05	44.32 ± 4.58	37.49 ± 1.77	38.70 ± 2.04	34.43 ± 5.42	8.46 ± 0.29	6.97 ± 0.04	**44.36 ± 2.35**
	5.0	5.25 ± 0.50	5.42 ± 0.29	5.12 ± 0.18	5.44 ± 0.08	46.05 ± 20.73	4.85 ± 0.06	5.12 ± 0.13	5.23 ± 0.05	4.31 ± 0.04	5.36 ± 0.16	**54.01 ± 10.25**
Walker2d Friction	0.1	7.88 ± 1.88	5.72 ± 0.23	29.60 ± 24.90	5.65 ± 0.06	28.23 ± 9.13	29.67 ± 12.21	17.94 ± 8.27	16.13 ± 4.11	9.32 ± 0.22	8.71 ± 0.14	**61.66 ± 4.70**
	0.5	63.94 ± 20.40	66.26 ± 3.03	45.01 ± 18.98	68.81 ± 1.12	76.96 ± 1.99	73.79 ± 4.40	**85.52 ± 0.70**	55.30 ± 15.67	64.43 ± 0.98	70.36 ± 0.76	70.16 ± 2.25
	2.0	39.06 ± 17.36	65.40 ± 7.13	67.89 ± 1.66	72.91 ± 0.37	**73.74 ± 0.49**	64.28 ± 1.12	65.49 ± 5.69	24.28 ± 18.48	48.38 ± 3.14	52.66 ± 1.20	73.05 ± 0.02
	5.0	10.07 ± 4.91	5.39 ± 0.03	5.76 ± 0.84	5.36 ± 0.28	27.38 ± 3.87	12.40 ± 6.16	15.58 ± 3.47	8.00 ± 2.42	4.50 ± 0.20	6.19 ± 0.41	**36.44 ± 8.23**
Hopper Gravity	0.1	27.82 ± 13.41	13.10 ± 0.98	15.59 ± 6.09	23.40 ± 11.62	**36.25 ± 1.50**	31.01 ± 5.02	35.54 ± 0.80	25.23 ± 6.67	25.59 ± 0.34	29.37 ± 0.41	36.02 ± 0.33
	0.5	28.54 ± 12.77	16.24 ± 7.89	23.00 ± 14.87	12.86 ± 0.18	33.57 ± 6.71	41.34 ± 6.51	16.31 ± 2.47	33.09 ± 16.36	34.92 ± 1.27	63.28 ± 0.39	**68.25 ± 3.03**
	2.0	11.84 ± 2.37	16.10 ± 1.64	18.62 ± 6.88	14.65 ± 2.47	23.79 ± 2.09	21.27 ± 2.30	18.74 ± 1.75	20.86 ± 1.96	14.72 ± 0.15	**31.38 ± 0.19**	26.94 ± 0.59
	5.0	7.36 ± 0.13	8.12 ± 0.16	**9.08 ± 1.15**	7.90 ± 1.27	8.06 ± 0.03	8.57 ± 0.42	8.76 ± 0.01	6.17 ± 0.98	9.00 ± 0.04	8.55 ± 0.11	8.80 ± 0.02
Hopper Friction	0.1	25.55 ± 2.69	24.16 ± 4.50	18.64 ± 3.37	26.13 ± 4.24	51.19 ± 2.56	24.99 ± 2.53	24.55 ± 1.36	33.87 ± 6.87	39.32 ± 0.89	36.65 ± 1.56	**52.94 ± 7.71**
	0.5	25.22 ± 4.48	23.56 ± 1.68	19.60 ± 15.45	26.94 ± 2.86	41.34 ± 0.49	27.57 ± 2.63	27.50 ± 2.19	**49.80 ± 13.88**	39.82 ± 1.46	36.43 ± 0.52	41.97 ± 6.51
	2.0	10.32 ± 0.06	10.15 ± 0.06	9.89 ± 0.20	10.15 ± 0.03	11.00 ± 0.14	10.44 ± 0.33	10.27 ± 0.06	10.29 ± 0.84	11.09 ± 0.02	**14.06 ± 0.15**	11.08 ± 0.02
	5.0	7.90 ± 0.06	7.93 ± 0.01	7.80 ± 1.04	7.86 ± 0.05	8.07 ± 0.04	8.06 ± 0.08	8.06 ± 0.09	8.20 ± 0.05	7.94 ± 0.01	**8.27 ± 0.00**	8.24 ± 0.04
Total		875.88	901.52	779.14	890.62	1427.26	1056.87	1044.40	998.75	865.91	908.92	**1504.16**

Baselines. We compare our method against both standard offline RL baselines and specialized off-dynamics offline RL baselines. For standard offline RL, we include IQL (Kostrikov et al., 2021) and TD3-BC (Fujimoto & Gu, 2021). For off-dynamics offline RL, we incorporate DARA (Liu et al., 2022), BOSA (Liu et al., 2024a), OTDF (Lyu et al., 2025) and MOBODY (Guo et al., 2026). We also include diffusion-based baselines including Diffuser (Janner et al., 2022), DmC (Van et al., 2025) and xTED (Niu et al., 2024). Comprehensive implementation details and hyperparameter configurations for all compared baselines are provided in Appendix F.4.

Implementation Details. We implement CEDGE as an energy-guided trajectory diffusion model operating on trajectories with normalized observations. The source domain trajectory diffusion model is parameterized by a temporal 1D U-Net denoiser. Energy guidance is implemented through

three distinct learned energy modules. The weighting coefficients for the return and domain guidance are selected from the set $\{0.1, 0.5, 1.0, 2.0\}$, while the policy guidance weight is fixed at 0.1. Additional architecture and hyperparameter specifics are detailed in Appendix G.

Main Results. We report the main results on MuJoCo tasks featuring gravity and friction shifts in Table 1. We evaluate two distinct applications of the proposed framework: CEDGE-Planner, which employs energy-guided diffusion directly for inference-time planning; and CEDGE-Policy, which utilizes the generated trajectories as synthetic data for downstream policy optimization.

Table 1 demonstrates that CEDGE achieves strong overall performance across both MuJoCo gravity and friction shifts. Among all tested methods, CEDGE-Policy obtains the highest aggregate normalized score, improving the total score from 1427.26 (achieved by the strongest baseline, MOBODY) to 1504.16. This improvement suggests that trajectory-level energy-guided generation provides significantly more valuable synthetic data than the transition-level model rollouts utilized by MOBODY under off-dynamics shifts. Furthermore, compared to standard offline RL baselines, CEDGE-Policy substantially outperforms IQL, improving the aggregate score from 901.52 to 1504.16. Since the downstream optimization of CEDGE-Policy is based on IQL, this performance gap confirms that the substantial gain is not simply attributable to the underlying offline RL backbone. CEDGE-Policy also outperforms the DmC and xTED, showing the effectiveness of our energy guidance on generating synthetic data for downstream policy learning. We also compare CEDGE with the diffusion baselines to show the effectiveness of the energy guidance in diffusion-based trajectory generation. The vanilla Diffuser baseline trains a trajectory diffusion model on the combined source and target dataset without employing the proposed energy guidance. CEDGE-Planner improves the aggregate score over Diffuser from 865.91 to 908.92, demonstrating that energy guidance significantly enhances diffusion planning under dynamics shifts. **Overall, these findings highlight that diffusion modeling alone is insufficient in off-dynamics settings, and our proposed energy guidance is critical for generating trajectories that are useful for both planning and policy learning.**

5.2 Illustrative Example

We conduct a toy Windy Maze experiment to provide an example for using trajectory-level generation under dynamics shift. Specifically, we compare MOBODY, which generates transitions step by step through a learned transition model, with our CEDGE, which generates the trajectory using the diffusion model. The environment is a 5×5 Windy Maze, where the leftward movement on state in the wind zone increases from 0.10 in the source domain to 0.30 in the target domain. Both methods use 2000 source and 10 target training trajectories of horizon 32 collected with the controller policy. Starting from the same initial state, MOBODY recursively predicts the next state using its learned transition model. In contrast, CEDGE directly generates the complete future state-action trajectory from the initial state using energy guidance. For this experiment, we evaluate CEDGE as a trajectory generator without planning or policy learning. Table 2 reports the average distance to the mean target trajectory from five target trajectories. CEDGE produces trajectories closer to the target reference than MOBODY, motivating trajectory-level generation under dynamics shift. Furthermore, MOBODY shows larger variation across its generated trajectories, while the trajectories generated by CEDGE are more concentrated around the target reference, as illustrated in Figure 1 and Figure 5. Details are provided in Appendix F.1.

Table 2: Average distance for MOBODY and CEDGE. Results are averaged over 32 samples.

Method	Average distance ↓
MOBODY	0.269 ± 0.052
CEDGE	$\mathbf{0.236 \pm 0.043}$

5.3 Ablation Study

In this section, we present additional experiments to offer deeper insights of our proposed CEDGE.

Contribution of energy guidance in planning. We study the contribution of each energy term to CEDGE-Planner. All variants use the same source trajectory diffusion model, planning procedure, and return energy for candidate selection, and differ only in the guidance used during reverse diffusion. We compare the full planner with **No Guidance**, **Domain Only**, **Return Only**, and **Policy Only**. As shown in Figure 3, the CEDGE-Planner achieves the best performance across HalfCheetah and Walker2d shifts. Adding a single energy term can improve performance in some settings,

but it does not provide consistent gains across tasks and shift levels. This suggests that no individual energy term is sufficient by itself. Detailed results are provided in Table 12 in Appendix H.2.

Are guided trajectories useful for policy learning? We evaluate whether energy-guided trajectories provide useful data for downstream policy learning. We use IQL for all variants and compare training on the source dataset, the target dataset, source-target combination, target data augmented with trajectories generated by Diffuser, and target data augmented with trajectories generated by CEDGE. The two generative methods use the same synthetic data budget. As shown in Table 14 in Appendix H.4, CEDGE-Policy outperforms Diffuser in all 16 settings, target data in 15 of 16 settings, and combined data in 14 of 16 settings. These results show that CEDGE provides more useful data for policy learning than unguided diffusion generation or directly using the offline datasets.

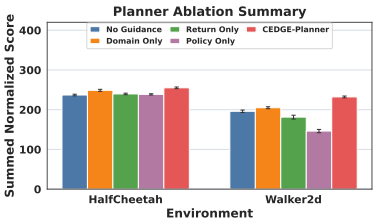

Figure 3: Planner-only ablation of energy guidance on HalfCheetah and Walker2d.

Is limited target data enough for diffusion training? We study whether the limited target dataset alone is sufficient for training a trajectory diffusion model. We compare CEDGE-Policy with an unguided diffusion model trained only on the target dataset, while keeping the downstream IQL training the same. As shown in Table 15 in Appendix H.5, CEDGE-Policy performs better in all tested gravity-shift settings. This shows that using the source domain diffusion model and adapting it with target energy guidance is more effective than directly training a diffusion model only on limited target data.

Is the gain merely due to filtering? We study whether the improvement of CEDGE-Policy can be explained by filtering alone. We apply the same filtering procedure to the generated trajectories and the original source trajectories, and train the same downstream IQL policy with the retained data combined target dataset. As shown in Figure 4, filtered synthetic trajectories generally outperform filtered source trajectories across different filtering ratios. This shows that filtering alone does not explain the improvement, and that the generated trajectories provide additional benefit. Detailed results are provided in Table 16 in Appendix H.6.

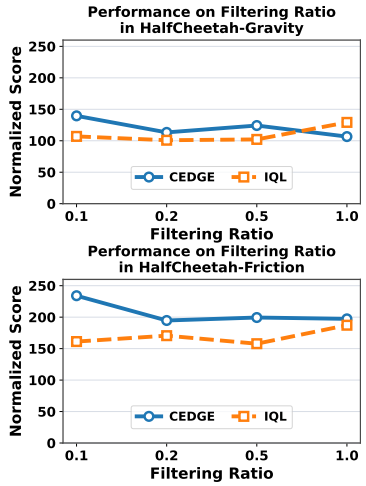

Figure 4: Filtering ratio performance on HalfCheetah.

Runtime analysis. We study the computational cost of CEDGE. As discussed in the experimental setup, the source diffusion model is trained once on the source dataset, while only the target-specific energy guidance functions need to be learned for each shifted target environment. As shown in Table 13 in Appendix H.3, this reuse substantially reduces the adaptation cost of both CEDGE-Planner and CEDGE-Policy, supporting efficient adaptation to new target environments.

6 Conclusion and Limitations

In this paper, we proposed CEDGE, an energy-guided diffusion generation framework for offline off-dynamics reinforcement learning. CEDGE learns a trajectory-level diffusion from abundant source-domain data and generates energy-guided trajectories. The guidance is derived from the distributional mismatch between the source trajectory distribution and the desired target-domain trajectory distribution, leading to three decomposed energy terms including domain energy, return energy and policy energy. CEDGE supports two downstream uses of the generated energy-guided trajectories. First, the guided diffusion model can be used directly as a planner. Second, the generated trajectories can serve as synthetic offline data for policy optimization. Experiments on the ODRL benchmark under gravity and friction dynamics shifts show improved performance compared with baselines. Our work has several limitations. Its performance depends on learned guidance modules, which may be inaccurate with very limited target data. Experiments are limited to simulated MuJoCo tasks, and the method adds computation for energy-module training and diffusion sampling.

AI USE STATEMENT

In this work, we used generative AI tools for coding, assisting with mathematical proofs, and polishing the writing. We have not used generative AI tools for retrieval and discovery, research ideation, drafting sections of the paper, or generating synthetic datasets, and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. AI-assisted code was verified and tested for correctness, mathematical derivations and proofs were checked for correctness, and AI-polished writing was reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide the source code and implementation in the supplementary materials. The main paper describes the proposed method, training procedure, experimental setup, evaluation protocol and all datasets and environments used in our experiments. Additional implementation details, hyperparameters, and experimental configurations are provided in the appendix. For the theoretical results, we state all required assumptions and provide complete proofs in the appendix.

REFERENCES

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International conference on machine learning*, pp. 1042–1051. PMLR, 2019.

Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint arXiv:2006.13916*, 2020.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Yihong Guo, Yixuan Wang, Yuanyuan Shi, Pan Xu, and Anqi Liu. Off-dynamics reinforcement learning via domain adaptation and reward augmented imitation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 136326–136360, 2024.

Yihong Guo, Yu Yang, Pan Xu, and Anqi Liu. MOBODY: Model-based off-dynamics offline reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7c0YS3cuno.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Matthew Thomas Jackson, Michael Tryfan Matthews, Cong Lu, Benjamin Ellis, Shimon Whiteson, and Jakob Foerster. Policy-guided diffusion. *arXiv preprint arXiv:2404.06356*, 2024.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Jinxin Liu, Hongyin Zhang, and Donglin Wang. Dara: Dynamics-aware reward augmentation in offline reinforcement learning. *arXiv preprint arXiv:2203.06662*, 2022.

Jinxin Liu, Ziqi Zhang, Zhenyu Wei, Zifeng Zhuang, Yachen Kang, Sibo Gai, and Donglin Wang. Beyond ood state actions: Supported cross-domain offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13945–13953, 2024a.

Xu-Hui Liu, Tian-Shuo Liu, Shengyi Jiang, Ruifeng Chen, Zhilong Zhang, Xinwei Chen, and Yang Yu. Energy-guided diffusion sampling for offline-to-online reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 31541–31565, 2024b.

Zhishuai Liu and Pan Xu. Distributionally robust off-dynamics reinforcement learning: Provable efficiency with linear function approximation. In *International Conference on Artificial Intelligence and Statistics*, pp. 2719–2727. PMLR, 2024.

Zhishuai Liu, Weixin Wang, and Pan Xu. Upper and lower bounds for distributionally robust off-dynamics reinforcement learning. *arXiv preprint arXiv:2409.20521*, 2024c.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pp. 22825–22855. PMLR, 2023a.

Cong Lu, Philip Ball, Yee Whye Teh, and Jack Parker-Holder. Synthetic experience replay. *Advances in Neural Information Processing Systems*, 36:46323–46344, 2023b.

Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? In *The Thirteenth International Conference on Learning Representations*, 2025.

Jiafei Lyu, Chenjia Bai, Jingwen Yang, Zongqing Lu, and Xiu Li. Cross-domain policy adaptation by capturing representation mismatch. *arXiv preprint arXiv:2405.15369*, 2024a.

Jiafei Lyu, Kang Xu, Jiacheng Xu, Jing-Wen Yang, Zongzhang Zhang, Chenjia Bai, Zongqing Lu, Xiu Li, et al. Odrl: A benchmark for off-dynamics reinforcement learning. *Advances in Neural Information Processing Systems*, 37:59859–59911, 2024b.

Jiafei Lyu, Mengbei Yan, Zhongjian Qiao, Runze Liu, Xiaoteng Ma, Deheng Ye, Jing-Wen Yang, Zongqing Lu, and Xiu Li. Cross-domain offline policy adaptation with optimal transport and dataset constraint. In *The Thirteenth International Conference on Learning Representations*, 2025.

Wen Zheng Terence Ng, Jianda Chen, and Tianwei Zhang. Off-dynamics conditional diffusion planners. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7461–7468. IEEE, 2024.

Haoyi Niu, Qimao Chen, Tenglong Liu, Jianxiong Li, Guyue Zhou, Yi Zhang, Jianming Hu, and Xianyuan Zhan. xted: Cross-domain adaptation via diffusion-based trajectory editing. *arXiv preprint arXiv:2409.08687*, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL `https://openreview.net/forum?id=St1giarCHLP`.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021b.

Cheng Tang, Zhishuai Liu, and Pan Xu. Robust offline reinforcement learning with linearly structured f-divergence regularization. *arXiv preprint arXiv:2411.18612*, 2024.

Linh Le Pham Van, Minh Hoang Nguyen, Duc Kieu, Hung Le, Hung The Tran, and Sunil Gupta. Dmc: Nearest neighbor guidance diffusion model for offline cross-domain reinforcement learning. *arXiv preprint arXiv:2507.20499*, 2025.

Ruhan Wang, Yu Yang, Zhishuai Liu, Dongruo Zhou, and Pan Xu. Return augmented decision transformer for off-dynamics reinforcement learning. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=QDVOr5J9Xp.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Xiaoyu Wen, Chenjia Bai, Kang Xu, Xudong Yu, Yang Zhang, Xuelong Li, and Zhen Wang. Contrastive representation for data filtering in cross-domain offline reinforcement learning. *arXiv preprint arXiv:2405.06192*, 2024.

Zhangjie Xia, Yu Yang, and Pan Xu. Localized dynamics-aware domain adaption for off-dynamics offline reinforcement learning. *arXiv preprint arXiv:2602.21072*, 2026.

Kang Xu, Chenjia Bai, Xiaoteng Ma, Dong Wang, Bin Zhao, Zhen Wang, Xuelong Li, and Wei Li. Cross-domain policy adaptation via value-guided data filtering. *Advances in Neural Information Processing Systems*, 36:73395–73421, 2023.

A Proofs of the Main Theoretical Results

In this section, we prove Proposition 4.1 and Theorem 4.2, respectively. The theoretical results do not directly analyze trajectory filtering, planner candidate selection, or the IQL implementation used in our empirical algorithm.

A.1 Proof of Proposition 4.1

Setup and errors. Let q_u and $p_u^\star$ be the marginals obtained by applying the same forward diffusion $\mathrm{d}X_u = b_u(X_u)\,\mathrm{d}u + g(u)\,\mathrm{d}W_u$ to q_{src} and $p^\star$ in (4.2) and (4.1), respectively, where $u \in [0,1]$ and $g(u) > 0$. Thus, $q_0 = q_{\mathrm{src}}$ and $p_0^\star = p^\star$.

Define the clean energy $\mathcal{E}^\star(\tau) = \log q_{\mathrm{src}}(\tau) - \log p^\star(\tau)$ and the noisy-time energy

$$\mathcal{H}_u^\star(z) = -\log \mathbb{E}\left[e^{-\mathcal{E}^\star(X_0)} \mid X_u = z\right].$$

The expectation is under the source forward process. Let $s_{\theta,u}$ be the learned source score and $\widehat{v}_u$ the guidance gradient used by the sampler. The applied score is

$$\widetilde{s}_u = s_{\theta,u} - \widehat{v}_u. \tag{A.1}$$

Let $\widehat{P}_0$ be the output distribution of the continuous reverse process initialized from $\widehat{p}_1$ with score $\widetilde{s}_u$. Define

$$\epsilon_T = D_{\mathrm{KL}}(p_1^\star \| \widehat{p}_1), \tag{A.2}$$

$$\epsilon_s^2 = \int_0^1 g(u)^2\, \mathbb{E}_{X_u \sim p_u^\star}\left[\|s_{\theta,u}(X_u) - \nabla \log q_u(X_u)\|_2^2\right] \mathrm{d}u, \tag{A.3}$$

$$\epsilon_g^2 = \int_0^1 g(u)^2\, \mathbb{E}_{X_u \sim p_u^\star}\left[\|\widehat{v}_u(X_u) - \nabla \mathcal{H}_u^\star(X_u)\|_2^2\right] \mathrm{d}u. \tag{A.4}$$

Here ϵ_T measures the mismatch between the sampler initialization and the noised desired distribution, ϵ_s measures the difference between the learned and exact source scores, and ϵ_g measures the difference between the applied and exact noisy-time guidance.

For the finite-step output distribution P_{syn}, let ϵ_{num} bound its discrepancy from the continuous-process output:

$$D_{\mathrm{TV}}(P_{\mathrm{syn}}, \widehat{P}_0) \le \epsilon_{\mathrm{num}}. \tag{A.5}$$

Assumption A.1 (Conditions for the generation bound)**.** The distributions q_{src} and $p^\star$ are normalized and have positive densities on a common support. The forward and reverse SDEs have solutions unique in law, with densities satisfying the conditions for score-based time reversal. The conditions for Girsanov's change of measure from the applied to the exact reverse process hold when both start from $p_1^\star$. The errors in (A.2)–(A.5) are finite.

Proof. **Step 1: Exact guidance.** Using the same Bayes argument as in the proof of Lu et al. (2023a, Theorem 3.1), the common forward diffusion gives, for $u > 0$,

$$\nabla \log p_u^\star(z) = \nabla \log q_u(z) - \nabla \mathcal{H}_u^\star(z). \tag{A.6}$$

Step 2: Continuous-sampler error. Using (A.1) and (A.6),

$$\nabla \log p_u^\star - \widetilde{s}_u = (\nabla \log q_u - s_{\theta,u}) + (\widehat{v}_u - \nabla \mathcal{H}_u^\star).$$

By Lemma B.1 and $\|a+b\|_2^2 \le 2\|a\|_2^2 + 2\|b\|_2^2$,

$$\begin{aligned} D_{\mathrm{KL}}(p^\star \| \widehat{P}_0) &\le \epsilon_T + \frac{1}{2}\int_0^1 g(u)^2\, \mathbb{E}_{p_u^\star}\left[\|\nabla \log p_u^\star - \widetilde{s}_u\|_2^2\right] \mathrm{d}u \\ &\le \epsilon_T + \int_0^1 g(u)^2\, \mathbb{E}_{p_u^\star}\left[\|\nabla \log q_u - s_{\theta,u}\|_2^2 + \|\widehat{v}_u - \nabla \mathcal{H}_u^\star\|_2^2\right] \mathrm{d}u \\ &= \epsilon_T + \epsilon_s^2 + \epsilon_g^2. \end{aligned} \tag{A.7}$$

The last equality follows from (A.3) and (A.4).

Step 3: Finite-step error. Combining (A.5) and (A.7) with the triangle inequality and Pinsker's inequality gives

$$\begin{aligned} D_{\mathrm{TV}}(P_{\mathrm{syn}}, p^\star) &\le D_{\mathrm{TV}}(P_{\mathrm{syn}}, \widehat{P}_0) + D_{\mathrm{TV}}(\widehat{P}_0, p^\star) \\ &\le \epsilon_{\mathrm{num}} + \sqrt{\frac{1}{2} D_{\mathrm{KL}}(p^\star \| \widehat{P}_0)} \\ &\le \epsilon_{\mathrm{num}} + \sqrt{\frac{\epsilon_T + \epsilon_s^2 + \epsilon_g^2}{2}} \\ &= B_{\mathrm{gen}}. \end{aligned}$$

□

A.2 Proof of Theorem 4.2

Setup and errors. Selecting one transition uniformly from each trajectory gives transition distributions ξ_{syn} and ξ_{des} under P_{syn} and $p^\star$, with state-action marginals μ_{syn} and μ_{des}. Write $\xi_{\mathrm{syn}} = \mu_{\mathrm{syn}} p_{\mathrm{syn}}$, where p_{syn} is the conditional next-state distribution and μp denotes $\mu(\mathrm{d}s, \mathrm{d}a) p(\mathrm{d}s' \mid s, a)$. Define

$$\epsilon_r = \mathbb{E}_{\xi_{\mathrm{syn}}} \big| \widehat{r}_\eta(s, a, s') - r(s, a) \big|, \tag{A.8}$$

$$\epsilon_{\mathrm{ref}} = D_{\mathrm{TV}}(\xi_{\mathrm{des}}, \mu_{\mathrm{des}} p_{\mathrm{trg}}). \tag{A.9}$$

Here ϵ_r measures reward-prediction error on synthetic transitions, and ϵ_{ref} measures the mismatch between transitions from $p^\star$ and the target dynamics at the same state-action marginal. The latter can remain nonzero because return weighting can change conditional next-state distributions.

Combine n real target transitions and m synthetic transitions, with $N = n + m > 0$ and $\zeta = m/N$. Their state-action distribution is

$$\mu_\zeta = (1 - \zeta)\mu_{\mathrm{trg}} + \zeta \mu_{\mathrm{syn}},$$

where μ_{trg} is the state-action distribution of the real target data. The synthetic fraction conditional on (s, a) is $w_\zeta(s, a) = \frac{\mathrm{d}(\zeta \mu_{\mathrm{syn}})}{\mathrm{d}\mu_\zeta}(s, a)$, with $0 \le w_\zeta \le 1$.

Assume $0 \le r, \widehat{r}_\eta \le R_{\max}$ and $\gamma \in (0, 1)$, and set $V_{\max} = R_{\max}/(1 - \gamma)$. For $Q : \mathcal{S} \times \mathcal{A} \to [0, V_{\max}]$, let $V_Q(s) = \max_a Q(s, a)$ and define

$$\begin{aligned} (\mathcal{T}_{\mathrm{trg}} Q)(s, a) &= r(s, a) + \gamma \mathbb{E}_{p_{\mathrm{trg}}(\cdot \mid s, a)} V_Q(s'), \\ (\mathcal{T}_{\mathrm{syn}} Q)(s, a) &= \mathbb{E}_{p_{\mathrm{syn}}(\cdot \mid s, a)} [\widehat{r}_\eta(s, a, s') + \gamma V_Q(s')], \\ (\mathcal{T}_\zeta Q)(s, a) &= (1 - w_\zeta(s, a))(\mathcal{T}_{\mathrm{trg}} Q)(s, a) \\ &\quad + w_\zeta(s, a)(\mathcal{T}_{\mathrm{syn}} Q)(s, a). \end{aligned}$$

Learner and assumptions. For observation i, define

$$Y_i(Q) = \begin{cases} r(s_i, a_i) + \gamma V_Q(s_i'), & i \le n, \\ \widehat{r}_\eta(s_i, a_i, s_i') + \gamma V_Q(s_i'), & i > n, \end{cases}$$

$$\widehat{L}(f; Q) = \frac{1}{N} \sum_{i=1}^{N} \left(f(s_i, a_i) - Y_i(Q) \right)^2.$$

For finite classes $\mathcal{F} \subseteq \mathcal{G} \subseteq [0, V_{\max}]^{\mathcal{S} \times \mathcal{A}}$, we use the learner of Chen & Jiang (2019, Eq. (6)):

$$\begin{aligned} \widehat{\mathcal{E}}(Q) &= \widehat{L}(Q; Q) - \min_{f \in \mathcal{G}} \widehat{L}(f; Q), \\ \widehat{Q} &\in \arg\min_{Q \in \mathcal{F}} \widehat{\mathcal{E}}(Q), \qquad \widehat{\pi}(s) \in \arg\max_a \widehat{Q}(s, a). \end{aligned} \tag{A.10}$$

Let d^π_{trg} be the normalized discounted state-action visitation distribution:

$$d^\pi_{\mathrm{trg}}(A) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \Pr_{\rho_0, \pi, p_{\mathrm{trg}}} \big((s_t, a_t) \in A \big).$$

Assumption A.2 (Conditions for policy learning)**.** Regular conditional distributions and measurable maximizing actions exist, and a measurable tie-breaking rule is fixed. On μ_ζ-null pairs, $\mathcal{T}_\zeta$ is extended as a Bellman operator with rewards in $[0, R_{\max}]$.

Sampling. The generator, reward predictor, and finite function classes are fixed independently of the real RL sample. Conditional on them, the n real transitions have distribution $\mu_{\mathrm{trg}} p_{\mathrm{trg}}$, the m synthetic transitions have distribution ξ_{syn}, and all N observations are independent.

Realizability and completeness. The fixed point $Q^\star_\zeta = \mathcal{T}_\zeta Q^\star_\zeta$ belongs to $\mathcal{F}$, and $\mathcal{T}_\zeta Q \in \mathcal{G}$ for every $Q \in \mathcal{F}$. The learner solves (A.10) exactly.

Coverage. A fixed finite C_ζ satisfies $d^\pi_{\mathrm{trg}}(A) \le C_\zeta \mu_\zeta(A)$ for every measurable A, for an optimal target policy $\pi^\star$ and every policy the learner can return.

The probability bound is conditional on the fixed learned components. It does not directly cover dependent transitions from the same trajectory or the experimental protocol.

Proof. **Step 1: Finite-sample error.** By Lemma B.2, with probability at least $1-\delta$,

$$\|\widehat{Q} - \mathcal{T}_\zeta \widehat{Q}\|_{L^2(\mu_\zeta)} \le 5 V_{\max} \sqrt{\frac{\log(2|\mathcal{F}||\mathcal{G}|/\delta)}{N}}. \tag{A.11}$$

The rest of the proof holds on this event.

Step 2: Synthetic-data bias. Transition extraction is a Markov kernel, and marginalization is another Markov kernel, so neither operation can increase total variation distance. Therefore, under Assumption A.1 and Proposition 4.1,

$$D_{\mathrm{TV}}(\mu_{\mathrm{syn}}, \mu_{\mathrm{des}}) \le D_{\mathrm{TV}}(\xi_{\mathrm{syn}}, \xi_{\mathrm{des}}) \le B_{\mathrm{gen}}.$$

Because $\xi_{\mathrm{syn}} = \mu_{\mathrm{syn}} p_{\mathrm{syn}}$ and $\mu_{\mathrm{syn}} p_{\mathrm{trg}}$ share the same state-action marginal,

$$\mathbb{E}_{\mu_{\mathrm{syn}}} D_{\mathrm{TV}}(p_{\mathrm{syn}}, p_{\mathrm{trg}}) = D_{\mathrm{TV}}(\xi_{\mathrm{syn}}, \mu_{\mathrm{syn}} p_{\mathrm{trg}}).$$

The triangle inequality, (A.9), and TV contraction under the common kernel p_{trg} then give

$$\begin{aligned}
\mathbb{E}_{\mu_{\mathrm{syn}}} D_{\mathrm{TV}}(p_{\mathrm{syn}}, p_{\mathrm{trg}}) &\le D_{\mathrm{TV}}(\xi_{\mathrm{syn}}, \xi_{\mathrm{des}}) + \epsilon_{\mathrm{ref}} \\
&\quad + D_{\mathrm{TV}}(\mu_{\mathrm{des}} p_{\mathrm{trg}}, \mu_{\mathrm{syn}} p_{\mathrm{trg}}) \\
&\le B_{\mathrm{gen}} + \epsilon_{\mathrm{ref}} + D_{\mathrm{TV}}(\mu_{\mathrm{des}}, \mu_{\mathrm{syn}}) \\
&\le 2B_{\mathrm{gen}} + \epsilon_{\mathrm{ref}}.
\end{aligned} \tag{A.12}$$

For $0 \le V_Q \le V_{\max}$ and next-state distributions p, q, $|\mathbb{E}_p V_Q - \mathbb{E}_q V_Q| \le V_{\max} D_{\mathrm{TV}}(p, q)$. Thus,

$$\begin{aligned}
|((\mathcal{T}_{\mathrm{syn}} - \mathcal{T}_{\mathrm{trg}})Q)(s,a)| &\le \mathbb{E}_{p_{\mathrm{syn}}(\cdot|s,a)} |\widehat{r}_\eta(s,a,s') - r(s,a)| \\
&\quad + \gamma V_{\max} D_{\mathrm{TV}}(p_{\mathrm{syn}}(\cdot \mid s,a), p_{\mathrm{trg}}(\cdot \mid s,a)).
\end{aligned}$$

Using $w_\zeta \, \mathrm{d}\mu_\zeta = \zeta \, \mathrm{d}\mu_{\mathrm{syn}}$, (A.8), and (A.12),

$$\begin{aligned}
\mathbb{E}_{\mu_\zeta} |(\mathcal{T}_\zeta - \mathcal{T}_{\mathrm{trg}})Q| &= \zeta \mathbb{E}_{\mu_{\mathrm{syn}}} |(\mathcal{T}_{\mathrm{syn}} - \mathcal{T}_{\mathrm{trg}})Q| \\
&\le \zeta \left[\epsilon_r + \gamma V_{\max} \mathbb{E}_{\mu_{\mathrm{syn}}} D_{\mathrm{TV}}(p_{\mathrm{syn}}, p_{\mathrm{trg}}) \right] \\
&\le \zeta \left[\epsilon_r + \gamma V_{\max} (2B_{\mathrm{gen}} + \epsilon_{\mathrm{ref}}) \right].
\end{aligned} \tag{A.13}$$

Step 3: Error under target-policy distributions. For $\pi \in \{\pi^\star, \widehat{\pi}\}$ and any bounded measurable f, coverage and Cauchy–Schwarz give

$$\begin{aligned}
\mathbb{E}_{d^\pi_{\mathrm{trg}}} |f| &\le \sqrt{C_\zeta} \, \|f\|_{L^2(\mu_\zeta)}, \\
\mathbb{E}_{d^\pi_{\mathrm{trg}}} |f| &\le C_\zeta \mathbb{E}_{\mu_\zeta} |f|.
\end{aligned}$$

Splitting the target Bellman residual and using (A.11) and (A.13),

$$\begin{aligned}
&\mathbb{E}_{d^\pi_{\mathrm{trg}}} |\mathcal{T}_{\mathrm{trg}} \widehat{Q} - \widehat{Q}| \\
&\quad \le \sqrt{C_\zeta} \|\mathcal{T}_\zeta \widehat{Q} - \widehat{Q}\|_{L^2(\mu_\zeta)} + C_\zeta \mathbb{E}_{\mu_\zeta} |(\mathcal{T}_{\mathrm{trg}} - \mathcal{T}_\zeta)\widehat{Q}| \\
&\quad \le 5 V_{\max} \sqrt{\frac{C_\zeta \log(2|\mathcal{F}||\mathcal{G}|/\delta)}{N}} + \zeta C_\zeta \left[\epsilon_r + \gamma V_{\max} (2B_{\mathrm{gen}} + \epsilon_{\mathrm{ref}}) \right].
\end{aligned} \tag{A.14}$$

Step 4: Policy-performance bound. For any stationary policy π, telescoping gives

$$J_{\mathrm{trg}}(\pi) - \mathbb{E}_{\rho_0} V_{\widehat{Q}}(s_0) = \frac{1}{1-\gamma} \mathbb{E}_{(s,a)\sim d^{\pi}_{\mathrm{trg}}} \left[(\mathcal{T}_{\mathrm{trg}} \widehat{Q})(s,a) - V_{\widehat{Q}}(s) \right].$$

Since $\widehat{\pi}$ is greedy, $\widehat{Q}(s, \widehat{\pi}(s)) = V_{\widehat{Q}}(s)$ and $\widehat{Q}(s,a) \le V_{\widehat{Q}}(s)$ for every action. Subtracting the identities for $\pi^\star$ and $\widehat{\pi}$ gives

$$\begin{aligned} J_{\mathrm{trg}}(\pi^\star) - J_{\mathrm{trg}}(\widehat{\pi}) &\le \frac{1}{1-\gamma} \left[\mathbb{E}_{d^{\pi^\star}_{\mathrm{trg}}} [\mathcal{T}_{\mathrm{trg}} \widehat{Q} - \widehat{Q}] - \mathbb{E}_{d^{\widehat{\pi}}_{\mathrm{trg}}} [\mathcal{T}_{\mathrm{trg}} \widehat{Q} - \widehat{Q}] \right] \\ &\le \frac{1}{1-\gamma} \left[\mathbb{E}_{d^{\pi^\star}_{\mathrm{trg}}} |\mathcal{T}_{\mathrm{trg}} \widehat{Q} - \widehat{Q}| + \mathbb{E}_{d^{\widehat{\pi}}_{\mathrm{trg}}} |\mathcal{T}_{\mathrm{trg}} \widehat{Q} - \widehat{Q}| \right]. \end{aligned}$$

Applying (A.14) to both policies gives

$$J_{\mathrm{trg}}(\pi^\star) - J_{\mathrm{trg}}(\widehat{\pi}) \le \frac{10 V_{\max}}{1-\gamma} \sqrt{\frac{C_\zeta \log(2|\mathcal{F}||\mathcal{G}|/\delta)}{n+m}} + \frac{2\zeta C_\zeta}{1-\gamma} \left[\epsilon_r + \gamma V_{\max} (2B_{\mathrm{gen}} + \epsilon_{\mathrm{ref}}) \right].$$

□

B Useful Lemmas

B.1 Reverse-diffusion relative entropy

Lemma B.1 (Reverse-diffusion relative entropy)**.** Under Assumption A.1,

$$D_{\mathrm{KL}}(p^\star \| \widehat{P}_0) \le D_{\mathrm{KL}}(p^\star_1 \| \widehat{p}_1) + \frac{1}{2} \int_0^1 g(u)^2 \mathbb{E}_{p^\star_u} \| \nabla \log p^\star_u - \widetilde{s}_u \|_2^2 \, \mathrm{d}u.$$

This follows from the Girsanov and data-processing argument in Song et al. (2021b, Theorem 1), with target distribution $p^\star$, initialization $\widehat{p}_1$, and applied score $\widetilde{s}_u$.

B.2 Finite-sample Bellman residual for mixed data

Lemma B.2 (Finite-sample Bellman residual)**.** Under the sampling, realizability, and completeness conditions of Assumption A.2, for every $\delta \in (0,1)$, with probability at least $1-\delta$,

$$\| \widehat{Q} - \mathcal{T}_\zeta \widehat{Q} \|_{L^2(\mu_\zeta)} \le 5 V_{\max} \sqrt{\frac{\log(2|\mathcal{F}||\mathcal{G}|/\delta)}{N}}.$$

Proof. We adapt the squared-loss analysis in Chen & Jiang (2019, Lemma 16 and the proof of Theorem 17) to fixed counts of independent real and synthetic observations.

Fix $Q \in \mathcal{F}$ and $f \in \mathcal{G}$. Define

$$Z_i = (f(s_i,a_i) - Y_i(Q))^2 - ((\mathcal{T}_\zeta Q)(s_i,a_i) - Y_i(Q))^2.$$

Using the conditional means of $Y_i(Q)$ and the mixture weights,

$$\begin{aligned} &\frac{1}{N} \sum_{i=1}^N \mathbb{E} \left[(f - \mathcal{T}_\zeta Q)(s_i,a_i) \big((\mathcal{T}_\zeta Q)(s_i,a_i) - Y_i(Q) \big) \right] \\ &\quad = (1-\zeta) \mathbb{E}_{\mu_{\mathrm{trg}}} \left[(f - \mathcal{T}_\zeta Q)(\mathcal{T}_\zeta Q - \mathcal{T}_{\mathrm{trg}} Q) \right] \\ &\qquad + \zeta \mathbb{E}_{\mu_{\mathrm{syn}}} \left[(f - \mathcal{T}_\zeta Q)(\mathcal{T}_\zeta Q - \mathcal{T}_{\mathrm{syn}} Q) \right] \\ &\quad = \mathbb{E}_{\mu_\zeta} \left[(f - \mathcal{T}_\zeta Q) \big\{ \mathcal{T}_\zeta Q - (1-w_\zeta) \mathcal{T}_{\mathrm{trg}} Q - w_\zeta \mathcal{T}_{\mathrm{syn}} Q \big\} \right] = 0. \end{aligned}$$

Thus, the cross term cancels after averaging the two groups, and

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}Z_i = \|f-\mathcal{T}_\zeta Q\|^2_{L^2(\mu_\zeta)}.$$

Since $f, \mathcal{T}_\zeta Q, Y_i(Q) \in [0, V_{\max}]$,

$$|Z_i - \mathbb{E}Z_i| \le 2V_{\max}^2,$$
$$\sum_{i=1}^{N}\mathrm{Var}(Z_i) \le 4NV_{\max}^2\|f-\mathcal{T}_\zeta Q\|^2_{L^2(\mu_\zeta)}.$$

Set $c_\delta = \frac{16V_{\max}^2}{3N}\log(2|\mathcal{F}||\mathcal{G}|/\delta)$. Bernstein's inequality and a union bound give, with probability at least $1-\delta$, simultaneously for all $Q \in \mathcal{F}$ and $f \in \mathcal{G}$,

$$\begin{aligned}
&\widehat{L}(f;Q) - \widehat{L}(\mathcal{T}_\zeta Q; Q) \\
&\quad\ge \|f-\mathcal{T}_\zeta Q\|^2_{L^2(\mu_\zeta)} - \sqrt{\frac{8V_{\max}^2\log(2|\mathcal{F}||\mathcal{G}|/\delta)}{N}}\|f-\mathcal{T}_\zeta Q\|_{L^2(\mu_\zeta)} \\
&\qquad - \frac{4V_{\max}^2\log(2|\mathcal{F}||\mathcal{G}|/\delta)}{3N} \\
&\quad\ge \frac{1}{2}\|f-\mathcal{T}_\zeta Q\|^2_{L^2(\mu_\zeta)} - c_\delta. \qquad \text{(B.1)}
\end{aligned}$$

Realizability gives $\mathcal{T}_\zeta Q^\star_\zeta = Q^\star_\zeta$. Applying (B.1) with $Q = Q^\star_\zeta$ and minimizing over $f \in \mathcal{G}$ yields $\widehat{\mathcal{E}}(Q^\star_\zeta) \le c_\delta$. Completeness gives $\mathcal{T}_\zeta\widehat{Q} \in \mathcal{G}$. Applying (B.1) with $f = Q = \widehat{Q}$ and using the optimality of $\widehat{Q}$,

$$\begin{aligned}
\frac{1}{2}\|\widehat{Q}-\mathcal{T}_\zeta\widehat{Q}\|^2_{L^2(\mu_\zeta)} - c_\delta &\le \widehat{L}(\widehat{Q};\widehat{Q}) - \widehat{L}(\mathcal{T}_\zeta\widehat{Q};\widehat{Q}) \\
&\le \widehat{\mathcal{E}}(\widehat{Q}) \le \widehat{\mathcal{E}}(Q^\star_\zeta) \le c_\delta.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\|\widehat{Q}-\mathcal{T}_\zeta\widehat{Q}\|_{L^2(\mu_\zeta)} &\le 2\sqrt{c_\delta} \\
&= \frac{8}{\sqrt{3}}V_{\max}\sqrt{\frac{\log(2|\mathcal{F}||\mathcal{G}|/\delta)}{N}} \\
&\le 5V_{\max}\sqrt{\frac{\log(2|\mathcal{F}||\mathcal{G}|/\delta)}{N}}.
\end{aligned}$$

□

C Additional Related Works

Diffusion models in RL. Diffusion models (Ho et al., 2020; Karras et al., 2022) have recently emerged as a powerful paradigm for decision-making and offline reinforcement learning. A significant line of work utilizes diffusion models for trajectory-level planning. For example, Diffuser (Janner et al., 2022) formulates planning as trajectory generation, showing that diffusion models can effectively capture complex, long-horizon trajectory structures and support flexible planning over entire trajectories. More recently, Veteran (Lu et al., 2025) provides systematic analyses of diffusion planning, identifying which architectural and sampling choices are most appropriate for achieving high performance on offline RL tasks. Another line of work uses diffusion models as expressive policy classes for offline RL. Diffusion-QL (Wang et al., 2022) and IDQL (Hansen-Estruch et al., 2023) uses diffusion-based action modeling combined with value-guided policy improvement, demonstrating that diffusion policies can represent complex and multimodal behavior distributions. Beyond direct planning or policy representation, recent studies (Lu et al., 2023b; Jackson et al., 2024) have also explored using diffusion models as data generators for downstream policy learning.

In particular, PGD (Jackson et al., 2024) uses diffusion models to synthesize policy guided trajectories for offline RL, highlighting the potential of generated trajectories as synthetic data, rather than merely as action proposals during test time. Compared to these works, our method leverages diffusion not just as a planner or a policy class, but as a source-domain trajectory prior that can be adapted to novel target dynamics through energy guidance and subsequently coupled with offline policy optimization.

Energy-guided diffusion in RL. Energy-guided diffusion provides a flexible mechanism to steer a learned generative model toward a desired distribution without necessitating the complete retraining of the generator. In the context of offline RL, CEP (Lu et al., 2023a) investigated energy-guided diffusion sampling from a theoretical standpoint and proposed a contrastive energy prediction method to estimate the necessary intermediate guidance for exact energy-guided sampling. EDIS (Liu et al., 2024b) learns a diffusion model from offline data and designs specific energy functions to generate samples that are better aligned with the distribution induced during online fine-tuning. EDIS successfully demonstrated that energy-guided diffusion can mitigate the mismatch between fixed offline data and the data distribution resulting from online learning. However, EDIS is focused on the offline-to-online setting, where additional online interaction is available and the policy is explicitly improved through fine-tuning. In sharp contrast, our work addresses the fully offline, off-dynamics RL problem, where no additional target-domain interaction is permissible, making the central difficulty the transition-dynamics mismatch between the source and target domains.

Off-dynamics RL. Off-dynamics reinforcement learning (Eysenbach et al., 2020; Liu et al., 2022; 2024c; Tang et al., 2024) addresses the challenge of transferring policies across domains with mismatched transition dynamics. A common approach is reward augmentation based on domain classifiers. For instance, DARC (Eysenbach et al., 2020) derives reward augmentation from the source-target dynamics discrepancy, and DARA (Liu et al., 2022) extends this idea to the offline setting. Another prominent line of work focuses on selecting or filtering source data that aligns more closely with the target domain. BOSA (Liu et al., 2024a) proposes off-dynamics offline RL under dataset support constraints. Data filtering methods utilize contrastive representations (Wen et al., 2024), optimal transport (Lyu et al., 2025), or localized dynamics-aware filtering (Xia et al., 2026) to select data from the source dataset. While these methods effectively mitigate harmful transfer from mismatched source data, their major limitation is that they remain largely restricted to the original collected offline dataset. More recently, a model-based approach, MOBODY (Guo et al., 2026), attempts to surpass the dataset support constraint by learning target-aware dynamics and generating synthetic transitions for policy learning. Our work differentiates itself by addressing off-dynamics offline RL at the trajectory level. Instead of correcting individual transitions or learning a comprehensive target dynamics model for rollouts, we adapt a source-domain diffusion trajectory model to the target domain using energy-guided sampling. We then utilize the resulting synthetic target trajectories for downstream offline policy optimization.

D Derivation of the Domain Classifiers

Following DARC (Eysenbach et al., 2020) and DARA (Liu et al., 2022), we use two domain classifiers to estimate the source-target dynamics difference. The first classifier takes (s_t, a_t, s_{t+1}) as input, while the second takes only (s_t, a_t). The second classifier is needed because the source and target datasets may contain different state-action distributions. Combining the two classifiers removes this difference and isolates the transition-dynamics ratio.

Clean-data derivation. Let $d \in \{\mathrm{src}, \mathrm{trg}\}$ denote the domain. For a clean transition (s_t, a_t, s_{t+1}),

$$p(s_t, a_t, s_{t+1} \mid d) = p(s_t, a_t \mid d) p_d(s_{t+1} \mid s_t, a_t). \tag{D.1}$$

By Bayes' rule,

$$p(d \mid s_t, a_t, s_{t+1}) = \frac{p_d(s_{t+1} \mid s_t, a_t) p(s_t, a_t \mid d) p(d)}{p(s_t, a_t, s_{t+1})}, \tag{D.2}$$

$$p(d \mid s_t, a_t) = \frac{p(s_t, a_t \mid d) p(d)}{p(s_t, a_t)}. \tag{D.3}$$

Taking the source-to-target ratio in both equations gives

$$\frac{p(\mathrm{src} \mid s_t, a_t, s_{t+1})}{p(\mathrm{trg} \mid s_t, a_t, s_{t+1})} = \frac{p_{\mathrm{src}}(s_{t+1} \mid s_t, a_t)}{p_{\mathrm{trg}}(s_{t+1} \mid s_t, a_t)} \frac{p(s_t, a_t \mid \mathrm{src})}{p(s_t, a_t \mid \mathrm{trg})} \frac{p(\mathrm{src})}{p(\mathrm{trg})}, \tag{D.4}$$

$$\frac{p(\mathrm{src} \mid s_t, a_t)}{p(\mathrm{trg} \mid s_t, a_t)} = \frac{p(s_t, a_t \mid \mathrm{src})}{p(s_t, a_t \mid \mathrm{trg})} \frac{p(\mathrm{src})}{p(\mathrm{trg})}. \tag{D.5}$$

Dividing (D.4) by (D.5) cancels the state-action term and the domain prior:

$$\frac{p_{\mathrm{src}}(s_{t+1} \mid s_t, a_t)}{p_{\mathrm{trg}}(s_{t+1} \mid s_t, a_t)} = \frac{p(\mathrm{src} \mid s_t, a_t, s_{t+1}) / p(\mathrm{trg} \mid s_t, a_t, s_{t+1})}{p(\mathrm{src} \mid s_t, a_t) / p(\mathrm{trg} \mid s_t, a_t)}. \tag{D.6}$$

Taking logarithms gives

$$\log \frac{p_{\mathrm{src}}(s_{t+1} \mid s_t, a_t)}{p_{\mathrm{trg}}(s_{t+1} \mid s_t, a_t)} = \log \frac{p(\mathrm{src} \mid s_t, a_t, s_{t+1})}{p(\mathrm{trg} \mid s_t, a_t, s_{t+1})} - \log \frac{p(\mathrm{src} \mid s_t, a_t)}{p(\mathrm{trg} \mid s_t, a_t)}. \tag{D.7}$$

Therefore, the classifier using (s_t, a_t, s_{t+1}) alone is not enough, because it also reflects how frequently (s_t, a_t) appears in each domain. The second classifier removes this effect. This is the two-classifier construction used in DARC (Eysenbach et al., 2020) and DARA (Liu et al., 2022).

Using (D.7), the clean domain energy becomes

$$\begin{aligned} \mathcal{E}_1(\tau_0) &= \sum_{t=0}^{H-1} \log \frac{p_{\mathrm{src}}(s_{t+1} \mid s_t, a_t)}{p_{\mathrm{trg}}(s_{t+1} \mid s_t, a_t)} \\ &= \sum_{t=0}^{H-1} \left[\log \frac{p(\mathrm{src} \mid s_t, a_t, s_{t+1})}{p(\mathrm{trg} \mid s_t, a_t, s_{t+1})} - \log \frac{p(\mathrm{src} \mid s_t, a_t)}{p(\mathrm{trg} \mid s_t, a_t)} \right]. \end{aligned} \tag{D.8}$$

Noisy-time implementation. During reverse diffusion, the classifiers receive noisy variables $(s_{t,k}, a_{t,k}, s_{t+1,k})$ instead of clean transitions. For a fixed diffusion step k, we similarly have

$$\begin{aligned} &p(s_{t,k}, a_{t,k}, s_{t+1,k} \mid d, k) \\ &\quad = p(s_{t,k}, a_{t,k} \mid d, k) p(s_{t+1,k} \mid s_{t,k}, a_{t,k}, d, k). \end{aligned} \tag{D.9}$$

Applying the same Bayes-rule calculation gives

$$\log \frac{p(s_{t+1,k} \mid s_{t,k}, a_{t,k}, \mathrm{src}, k)}{p(s_{t+1,k} \mid s_{t,k}, a_{t,k}, \mathrm{trg}, k)} = \log \frac{p(\mathrm{src} \mid s_{t,k}, a_{t,k}, s_{t+1,k}, k)}{p(\mathrm{trg} \mid s_{t,k}, a_{t,k}, s_{t+1,k}, k)} - \log \frac{p(\mathrm{src} \mid s_{t,k}, a_{t,k}, k)}{p(\mathrm{trg} \mid s_{t,k}, a_{t,k}, k)}. \tag{D.10}$$

We estimate these probabilities using the timestep-conditioned classifiers $p_{\theta_{\mathrm{SAS}}}$ and $p_{\theta_{\mathrm{SA}}}$ trained with (4.4) and (4.5). The domain energy used during diffusion sampling is

$$\begin{aligned} \mathcal{E}_1(\tau_k, k) = \sum_{t=0}^{H-1} \Big[&\log p_{\theta_{\mathrm{SAS}}}(\mathrm{src} \mid s_{t,k}, a_{t,k}, s_{t+1,k}, k) \\ &- \log p_{\theta_{\mathrm{SAS}}}(\mathrm{trg} \mid s_{t,k}, a_{t,k}, s_{t+1,k}, k) \\ &- \log p_{\theta_{\mathrm{SA}}}(\mathrm{src} \mid s_{t,k}, a_{t,k}, k) + \log p_{\theta_{\mathrm{SA}}}(\mathrm{trg} \mid s_{t,k}, a_{t,k}, k) \Big]. \end{aligned} \tag{D.11}$$

Its gradient, $\nabla_{\tau_k} \mathcal{E}_1(\tau_k, k)$, is used to guide the reverse diffusion process.

E Algorithm Details

This section provides the technical details of the source-domain trajectory diffusion model, the energy guidance functions, and the energy-guided reverse diffusion generation. We first train a trajectory diffusion model on source-domain trajectories. For a new target environment, we keep the diffusion model fixed and train the energy guidance functions using the available source and target offline datasets. During reverse diffusion sampling, the energy functions are evaluated on the current noisy trajectory sample, and their gradients are added to the diffusion score to guide the generation process. We summarize the process in Algorithm 1, Algorithm 2 and Algorithm 3.

Algorithm 1 Training the Source-Domain Trajectory Diffusion Model

Require: Source dataset $\mathcal{D}_{\mathrm{src}}$, denoiser D_θ, diffusion schedule $\{\beta_k\}_{k=1}^T$.

for epoch $= 1, \ldots, E_{\mathrm{diff}}$ **do**
  Sample a source trajectory batch $\tau \sim \mathcal{D}_{\mathrm{src}}$
  Sample diffusion step $k \sim \mathrm{Uniform}(\{1, \ldots, T\})$ and noise $\epsilon \sim \mathcal{N}(0, I)$
  Construct the noisy trajectory τ_k according to the forward diffusion process
  Update the denoiser D_θ using the diffusion loss in (3.1)
end for

Algorithm 2 Training Energy Guidance Functions for a Target Environment

Require: Source dataset $\mathcal{D}_{\mathrm{src}}$, target dataset $\mathcal{D}_{\mathrm{trg}}$, fixed denoiser D_θ with diffusion schedule $\{\bar{\alpha}_k\}_{k=1}^T$, domain classifiers $p_{\theta_{\mathrm{SAS}}}, p_{\theta_{\mathrm{SA}}}$, return predictor J_ψ, Gaussian policy model π_ω.

for epoch $= 1, \ldots, E_{\mathrm{energy}}$ **do**
  Sample source and target transition batches from $\mathcal{D}_{\mathrm{src}}$ and $\mathcal{D}_{\mathrm{trg}}$
  Construct noisy transitions $(s_{t,k}, a_{t,k}, s_{t+1,k})$ and extract their state-action components $(s_{t,k}, a_{t,k})$
  Update $p_{\theta_{\mathrm{SAS}}}$ and $p_{\theta_{\mathrm{SA}}}$ using the noisy inputs and k in (4.4) and (4.5)
  Sample trajectory batches τ_0 from $\mathcal{D}_{\mathrm{src}} \cup \mathcal{D}_{\mathrm{trg}}$
  Construct noisy trajectory τ_k.
  Update J_ψ on (τ_k, k) using clean-trajectory returns as labels in (4.6)
  Sample source state-action pairs from $\mathcal{D}_{\mathrm{src}}$
  Construct noisy pairs $(s_{t,k}, a_{t,k})$
  Update $\pi_\omega(a_{t,k} \mid s_{t,k}, k)$ using (4.7)
end for

Algorithm 3 Energy-Guided Reverse Diffusion Sampling

Require: Fixed denoiser D_θ, guidance functions $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$, guidance weights $\lambda_1, \lambda_2, \lambda_3$, guidance scales $\{\rho_k\}_{k=1}^T$, diffusion schedule $\{\beta_k\}_{k=1}^T$, number of samples N.

Let $\alpha_k = 1 - \beta_k$ and $\bar{\alpha}_k = \prod_{i=1}^k \alpha_i$
Sample initial noisy trajectories $\{\tau_T^{(n)}\}_{n=1}^N \sim \mathcal{N}(0, I)$
for $k = T, \ldots, 1$ **do**
  Predict the clean trajectory estimate: $\hat{\tau}_0^{(n)} = D_\theta(\tau_k^{(n)}, k)$, where $n = 1, \ldots, N$.
  Convert denoiser into the diffusion score: $s_\theta(\tau_k^{(n)}, k) = -\frac{\tau_k^{(n)} - \sqrt{\bar{\alpha}_k}\hat{\tau}_0^{(n)}}{1-\bar{\alpha}_k}$.
  Compute the weighted guidance energy:

$$\mathcal{E}(\tau_k^{(n)}, k) = \lambda_1 \mathcal{E}_1(\tau_k^{(n)}, k) + \lambda_2 \mathcal{E}_2(\tau_k^{(n)}, k) + \lambda_3 \mathcal{E}_3(\tau_k^{(n)}, k).$$

  Compute the energy guidance score: $h_k^{(n)} = -\nabla_{\tau_k^{(n)}} \mathcal{E}(\tau_k^{(n)}, k)$.
  Add the energy guidance score to the diffusion score: $\tilde{s}_\theta(\tau_k^{(n)}, k) = s_\theta(\tau_k^{(n)}, k) + \rho_k h_k^{(n)}$.
  Compute the guided reverse mean: $\tilde{\mu}_k^{(n)} = \frac{1}{\sqrt{\alpha_k}} \left(\tau_k^{(n)} + \beta_k \tilde{s}_\theta(\tau_k^{(n)}, k) \right)$.
  Sample $\tau_{k-1}^{(n)} \sim \mathcal{N}(\tilde{\mu}_k^{(n)}, \sigma_k^2 I)$
end for

F Experimental Details

F.1 Windy Maze Details

Environment. We consider a continuous 5×5 navigation environment in which the agent moves from $(0, 0)$ to $(4, 4)$ while avoiding an obstacle in $[1.5, 3] \times [1.5, 3]$. A wind zone at $x \in [2, 3]$ introduces a leftward movement of 0.10 in the source environment and 0.30 in the target environment. The reward is -1 per step and 10 at the goal. A trajectory has a fixed length of 32.

Offline data. We collect 2000 source and 10 target trajectories using a noisy waypoint controller. For the comparison shown in Figure 1, the controller follows $(0, 0) \to (1, 3.5) \to (3.5, 3.5) \to (4, 4)$.

At each step, the action is $\tilde{a}_t = d_t + \epsilon_t, \epsilon_t \sim \mathcal{N}(0, 0.08^2 I_2)$, where d_t is the unit direction toward the current waypoint. Actions with norm larger than one are normalized to unit norm. The controller switches to the next waypoint when it is within 0.35 of the current waypoint. Five target trajectories following this route are used to construct the reference trajectory for evaluation.

Trajectory generation. Both methods start from $(0, 0)$. MOBODY generates a trajectory recursively by predicting the next state from its previous predicted state and an action provided by the waypoint controller. In contrast, CEDGE generates the states and actions of the entire trajectory jointly, conditioned on the initial state. We evaluate 32 generated trajectories from each method using models trained with seed 100.

Evaluation. We average the five target reference trajectories at each time step to obtain a mean target trajectory. For each generated trajectory, we compute the mean Euclidean distance between its generated states and the corresponding states of this reference trajectory, excluding the shared initial state. Table 2 reports the mean and sample standard deviation of this distance across the 32 generated trajectories. Figure 5 further visualizes a single generated trajectory, all 32 generated trajectories, and their mean trajectory for each method. MOBODY shows greater variation across its generated trajectories, whereas the trajectories generated by CEDGE are more tightly concentrated around the target reference, consistent with the smaller standard deviation.

F.2 Environments and datasets

We evaluate on the ODRL benchmark (Lyu et al., 2024b). Following the ODRL protocol, the source domain is the original MuJoCo environment, while each target domain is constructed by modifying the simulator dynamics parameters. The source and target domains share the same state space, action space, reward function, and initial-state distribution, and differ only in the transition dynamics. For MuJoCo locomotion, we consider four environments: HalfCheetah, Ant, Walker2d, and Hopper.

We evaluate two representative types of dynamics shifts from ODRL: gravity shifts and friction shifts. For gravity shifts, we modify only the magnitude of the gravity vector while keeping its direction unchanged. For friction shifts, we modify all MuJoCo friction components associated with the robot-environment contact, including static, dynamic, and rolling friction. In both cases, the target environment is obtained by multiplying the corresponding source-domain simulator parameter by a factor in $\{0.1, 0.5, 2.0, 5.0\}$. These factors cover both mild and severe dynamics mismatch and follow the task construction used in ODRL.

For the offline datasets, we use the original D4RL medium-level datasets (Fu et al., 2020) collected from the unmodified MuJoCo environments as the source datasets. Each source dataset contains 1M transitions. For each shifted target environment, we use the limited target-domain dataset provided by ODRL, which contains $5{,}000$ transitions collected from the corresponding shifted environment by a medium-performance SAC policy. All methods are trained only on the provided offline source and target datasets and are evaluated in the shifted target environments.

The normalized score is computed as

$$\mathrm{NS} = \frac{J - J_{\mathrm{random}}}{J_{\mathrm{expert}} - J_{\mathrm{random}}} \times 100, \tag{F.1}$$

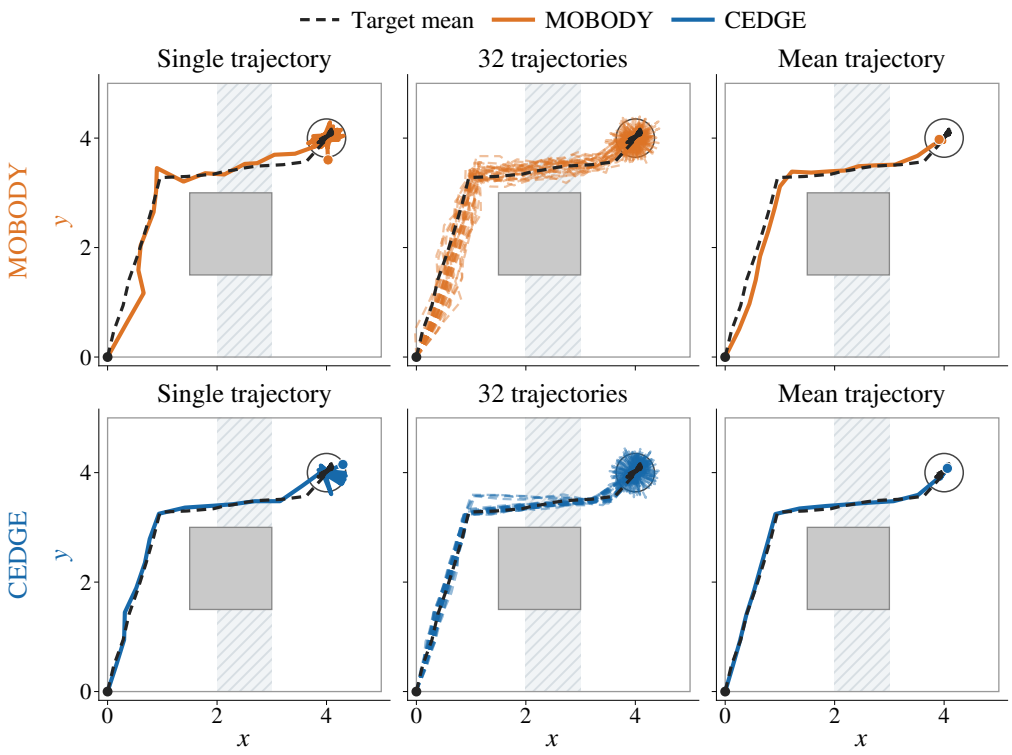

Figure 5: Windy Maze trajectory comparison between MOBODY and CEDGE. The left column shows one generated trajectory, the middle column shows all 32 generated trajectories, and the right column shows their mean trajectory. MOBODY exhibits larger variation across generated trajectories, while CEDGE produces trajectories that are more concentrated around the target reference trajectory.

where J is the evaluated return in the target domain, and J_{random} and J_{expert} are the random-policy and expert-policy returns in the same target environment.

F.3 ODRL TASK CONFIGURATION

ODRL constructs target-domain locomotion environments by editing the MuJoCo XML files.

Gravity Shift. Following the ODRL benchmark (Lyu et al., 2024b), gravity shifts are generated by editing the `gravity` attribute in the MuJoCo `option` element. Only the magnitude of the gravity vector is modified, while its direction is kept unchanged. For example, to construct `halfcheetah-gravity-0.5`, the gravity in the target domain is set to 0.5 times the gravity in the source domain:

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

Friction Shift. Friction shifts are generated by modifying the `friction` attribute in the MuJoCo `geom` elements. ODRL adjusts the frictional components to $\{0.1, 0.5, 2.0, 5.0\}$ times the corresponding source-domain values. The following example from ODRL illustrates a $5.0\times$ friction shift for Hopper:

```
# torso
<geom friction="4.5" fromto="0 0 1.45 0 0 1.05"
      name="torso_geom" size="0.05" type="capsule"/>

# thigh
<geom friction="4.5" fromto="0 0 1.05 0 0 0.6"
      name="thigh_geom" size="0.05" type="capsule"/>

# leg
<geom friction="4.5" fromto="0 0 0.6 0 0 0.1"
      name="leg_geom" size="0.04" type="capsule"/>

# foot
<geom friction="10.0" fromto="-0.13 0 0.1 0.26 0 0.1"
      name="foot_geom" size="0.06" type="capsule"/>
```

F.4 BASELINES

We compare our method with standard offline RL baselines and off-dynamics offline RL baselines. For standard offline RL baselines, we include IQL (Kostrikov et al., 2021) and TD3-BC (Fujimoto & Gu, 2021). IQL learns an upper-expectile value function and extracts the policy via advantage-weighted behavioral cloning, while TD3-BC adds a behavior cloning regularizer to the policy update. For off-dynamics offline RL baselines, we include DARA (Liu et al., 2022), BOSA (Liu et al., 2024a), and MOBODY (Guo et al., 2026). DARA uses domain classifiers to estimate the source-target dynamics discrepancy and performs dynamics-aware reward augmentation. BOSA addresses out-of-distribution state-action pairs and dynamics mismatch through support-constrained policy and value optimization. MOBODY is a model-based off-dynamics offline RL method that learns target-aware dynamics and generates transition-level rollouts for policy improvement. We also include a Diffuser baseline (Janner et al., 2022), which trains a trajectory diffusion model on the combined source and target offline datasets without the proposed energy guidance.

IQL. Implicit Q-Learning (IQL) (Kostrikov et al., 2021) is an offline RL algorithm that learns a value function, a Q-function, and then extracts a policy by weighted behavior cloning. Let $\tau_{\exp}$ be the expectile parameter and define $\ell_{\tau_{\exp}}(u) = |\tau_{\exp} - \mathbf{1}\{u < 0\}|u^2$. IQL trains the value function and Q-function with

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\ell_{\tau_{\exp}}\left(Q_\theta(s,a) - V_\psi(s)\right)\right], \tag{F.2}$$

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma V_{\bar{\psi}}(s') - Q_\theta(s,a)\right)^2\right], \tag{F.3}$$

where $\mathcal{D}$ is the training dataset. The policy is trained by advantage-weighted behavior cloning:

$$\mathcal{L}_\pi(\phi) = -\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp\left(\beta\left(Q_\theta(s,a) - V_\psi(s)\right)\right)\log \pi_\phi(a \mid s)\right]. \tag{F.4}$$

In the main comparison, IQL is trained on $\mathcal{D}_{\mathrm{src}} \cup \mathcal{D}_{\mathrm{trg}}$.

TD3-BC. TD3-BC (Fujimoto & Gu, 2021) is an offline variant of TD3 with an additional behavior cloning term in the actor loss. The critic target is

$$y = r + \gamma \min_{i=1,2} Q_{\bar{\theta}_i}\left(s', \pi_{\bar{\phi}}(s') + \epsilon\right), \tag{F.5}$$

where ϵ is the target-policy smoothing noise. The actor is trained with

$$\mathcal{L}_\pi(\phi) = -\lambda \mathbb{E}_{s\sim\mathcal{D}}\left[Q_{\theta_1}(s, \pi_\phi(s))\right] + \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\|\pi_\phi(s) - a\|_2^2\right], \tag{F.6}$$

where λ balances Q maximization and behavior cloning. In the main comparison, TD3-BC is trained on $\mathcal{D}_{\mathrm{src}} \cup \mathcal{D}_{\mathrm{trg}}$.

DARA. DARA (Liu et al., 2022) uses domain classifiers to estimate how source-like or target-like a transition is under dynamics shift. Following the classifier-ratio form used in off-dynamics RL (Eysenbach et al., 2020), DARA trains one classifier on transitions, $p_{\theta_{\mathrm{SAS}}}(d \mid s, a, s')$, and one classifier on state-action pairs, $p_{\theta_{\mathrm{SA}}}(d \mid s, a)$, where $d \in \{\mathrm{src}, \mathrm{trg}\}$. The source-over-target dynamics ratio is estimated as

$$\widehat{\Delta}_{\mathrm{dyn}}(s, a, s') = \log \frac{p_{\theta_{\mathrm{SAS}}}(\mathrm{src} \mid s, a, s')}{p_{\theta_{\mathrm{SAS}}}(\mathrm{trg} \mid s, a, s')} - \log \frac{p_{\theta_{\mathrm{SA}}}(\mathrm{src} \mid s, a)}{p_{\theta_{\mathrm{SA}}}(\mathrm{trg} \mid s, a)}. \tag{F.7}$$

DARA then modifies source-domain rewards by

$$\tilde{r}(s,a,s') = r(s,a,s') - \eta\,\widehat{\Delta}_{\mathrm{dyn}}(s,a,s'), \tag{F.8}$$

where η is the reward correction coefficient. This lowers the reward of source transitions that are estimated to be more source-like than target-like. The policy is trained on the reward-augmented source data together with the limited target data.

BOSA. BOSA (Liu et al., 2024a) addresses the off-dynamics offline RL problem from two aspects. First, it handles out-of-distribution (OOD) state-action pairs through supported policy optimization. Second, it addresses the OOD dynamics issue through supported value optimization with data filtering. Specifically, the policy is updated by maximizing

$$\mathcal{L}_{\mathrm{actor}} = \mathbb{E}_{s\sim\mathcal{D}_{\mathrm{src}}\cup\mathcal{D}_{\mathrm{trg}},\,a\sim\pi_\phi(\cdot|s)}\left[Q(s,a)\right], \quad \text{s.t. } \mathbb{E}_{s\sim\mathcal{D}_{\mathrm{src}}\cup\mathcal{D}_{\mathrm{trg}}}\left[\hat{\pi}_{\mathrm{offline}}(\pi_\phi(s)\mid s)\right] > \epsilon, \tag{F.9}$$

where ϵ is a threshold and $\hat{\pi}_{\mathrm{offline}}$ is the learned behavior policy on the combined offline dataset. The value critic is updated with

$$\begin{aligned}\mathcal{L}_{\mathrm{critic}} = {} & \mathbb{E}_{(s,a)\sim\mathcal{D}_{\mathrm{src}}}\left[Q(s,a)\right] \\ & + \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\mathrm{src}}\cup\mathcal{D}_{\mathrm{trg}},\,a'\sim\pi_\phi(\cdot|s')}\left[I(\hat{p}_{\mathrm{trg}}(s'\mid s,a)>\epsilon')\left(Q_{\theta_i}(s,a)-y\right)^2\right],\end{aligned} \tag{F.10}$$

where $I(\cdot)$ is the indicator function, $\hat{p}_{\mathrm{trg}}(s'\mid s,a)$ denotes the estimated target-domain dynamics model, and ϵ' is a threshold.

MOBODY. MOBODY (Guo et al., 2026) is a model-based off-dynamics offline RL method that learns target-domain dynamics and uses model rollouts for policy optimization. It is motivated by the limitation that reward-regularization and data-filtering methods mainly rely on low-shift source transitions and may fail to explore high-return target-domain regions under large dynamics mismatch. To learn target dynamics from abundant source data and limited target data, MOBODY uses separate source and target action encoders, ψ_{src} and ψ_{trg}, together with a shared state encoder ϕ_E and a shared transition model ϕ_T:

$$\hat{s}'_{\mathrm{src}} = \phi_T\left(\phi_E(s) + \psi_{\mathrm{src}}(\phi_E(s),a)\right), \tag{F.11}$$

$$\hat{s}'_{\mathrm{trg}} = \phi_T\left(\phi_E(s) + \psi_{\mathrm{trg}}(\phi_E(s),a)\right). \tag{F.12}$$

After learning the target dynamics, MOBODY generates synthetic rollout data and trains the policy on an enhanced dataset composed of DARA-regularized source data, limited target data, and generated rollout data. It further uses a target-Q-weighted behavior cloning loss to regularize the policy toward actions with high target-domain Q-values.

Diffuser. Diffuser (Janner et al., 2022) is a trajectory-level diffusion planning method for offline RL. It models state-action trajectories directly and generates plans through iterative denoising rather than autoregressive one-step rollout. In our experiments, the Diffuser baseline trains a trajectory denoiser on the combined source and target trajectory dataset $\mathcal{D}_{\mathrm{src}} \cup \mathcal{D}_{\mathrm{trg}}$ with the data-prediction diffusion objective

$$\mathcal{L}_{\mathrm{diff}}(\theta) = \mathbb{E}_{\tau_0\sim\mathcal{D}_{\mathrm{src}}\cup\mathcal{D}_{\mathrm{trg}},\,k,\epsilon}\left[\|D_\theta(\tau_k,k)-\tau_0\|_2^2\right], \tag{F.13}$$

where

$$\tau_k = \sqrt{\bar{\alpha}_k}\tau_0 + \sqrt{1-\bar{\alpha}_k}\epsilon, \quad \epsilon\sim\mathcal{N}(0,I). \tag{F.14}$$

At evaluation time, Diffuser samples candidate trajectories by reverse denoising. The first state of each trajectory is constrained to the current state, and the first action of the selected trajectory is executed in a receding-horizon manner.

OTDF. Optimal Transport Data Filtering (OTDF) (Lyu et al., 2025) uses optimal transport to compare source and target transitions. The resulting scores are used to select and weight source transitions when training the Q-function. The policy update also includes a regularization term that encourages actions supported by the target dataset. OTDF therefore changes how the collected source data are used for policy learning, rather than generating new transitions.

DmC. DmC (Van et al., 2025) estimates the difference between source and target data using k-nearest-neighbor distances between transition tuples (s, a, s'). It trains a diffusion model on source transitions, using the resulting target-specific scores as conditioning inputs. Classifier-free guidance then generates additional transitions with scores indicating greater similarity to the target data. The generated transitions are added to the source data for policy learning. DmC also uses these scores to select and weight source transitions and regularizes the policy toward actions supported by the target dataset.

xTED. xTED (Niu et al., 2024) trains a trajectory diffusion model on the target dataset and uses it to edit source trajectories. Its denoiser separately encodes state, action, and reward sequences and uses attention to model their dependencies. During editing, Gaussian noise is added to a source trajectory, which is then denoised using the target-trained model. The noise level controls how much of the original trajectory is changed. The edited trajectories are combined with the target dataset for downstream policy learning.

F.5 Implementation Details.

We implement CEDGE as an energy-guided trajectory diffusion model over normalized state-action sequences. The source domain trajectory diffusion model is parameterized by a temporal 1D U-Net denoiser and is trained only on source trajectories. Energy guidance is introduced through three learned energy modules, corresponding to domain energy, return energy, and policy energy. We use a discrete diffusion model with a cosine noise schedule, DDPM sampling, and 20 reverse sampling steps. The denoiser and energy modules are trained with a batch size of 64. For energy guided generation, the return and domain guidance weights are chosen from $\{0.1, 0.5, 1.0, 2.0\}$, while the policy guidance weight is set to be 0.1. Additional architecture and hyperparameter details are provided in Appendix G.

G Hyperparameters and Architectures

This section summarizes the architectures and hyperparameters used in CEDGE. All trajectory-generation components operate on state-action trajectory tensors. In the implementation, a trajectory segment is stored as a length-H state-action sequence

$$\tau = (s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}, s_H).$$

Trajectory segments are extracted within individual D4RL episodes split by terminal or timeout flags and do not cross episode boundaries. Observations are normalized using a Gaussian normalizer, while actions are kept in their original scale. During planning, generated actions are clipped to $[-1, 1]$ before environment execution.

Diffusion model. The source-domain trajectory diffusion model is parameterized by a temporal 1D U-Net denoiser. The denoiser takes a noisy trajectory segment τ_k and diffusion timestep k as input and predicts the clean trajectory τ_0. The model architecture is summarized in Table 3. Task-specific horizons and channel multipliers are reported in Table 4, and the diffusion training hyperparameters are listed in Table 5. The source-domain diffusion model is trained once and reused across target dynamics shifts.

Energy modules. For each target environment, we keep the source-domain trajectory diffusion model fixed and train target-specific energy modules using the available source and target offline datasets. The sampling energy is

$$\mathcal{E}(\tau) = \lambda_1 \mathcal{E}_1(\tau) + \lambda_2 \mathcal{E}_2(\tau) + \lambda_3 \mathcal{E}_3(\tau), \tag{G.1}$$

where $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$ denote the domain energy, return energy, and source behavior-policy energy, respectively. The architectures of these energy modules are shown in Table 6, and their training and guidance hyperparameters are reported in Table 7. During reverse diffusion sampling, the energy functions are evaluated on the current noisy trajectory sample in the normalized trajectory space.

CEDGE-Planner. CEDGE-Planner samples candidate trajectories conditioned on the current state and executes the first action of the trajectory with the lowest return energy. The planning hyperparameters are summarized in Table 8.

Table 3: Architecture of the trajectory diffusion denoiser.

Component	Setting
Denoiser	Temporal 1D U-Net
Input / output shape	$H \times (d_s + d_a)$
Prediction target	τ_0
Base channels	32
Timestep embedding dimension	32
Timestep embedding	Positional embedding
Convolution kernel size	5

Table 4: Task-specific diffusion horizon and channel multipliers.

Environment	Horizon H	Channel multipliers
HalfCheetah	4	$[1, 4, 2]$
Ant	4	$[1, 4, 2]$
Walker2d	32	$[1, 2, 2, 2]$
Hopper	32	$[1, 2, 2, 2]$

CEDGE-Policy. For CEDGE-Policy, we use the energy-guided sampler to generate synthetic trajectories for downstream offline policy optimization. The synthetic-data construction protocol is summarized in Table 9. The final policy is trained with IQL on $\mathcal{D}_{\mathrm{syn}} \cup \mathcal{D}_{\mathrm{trg}}$, with architecture and hyperparameters listed in Table 10.

H Additional Results

H.1 Results on kinematic and morphology shift

In this section, we present additional results on various types of dynamics shifts, including Kinematic shift (kin) and Morphology shift (morph), on MuJoCo. We present the results in Table 11. We observe that our method performs best or second-best in 24 of 32 tasks and receives the overall best performance.

H.2 Additional results on energy guidance

As shown in Table 12, the full CEDGE-Planner achieves the best performance across HalfCheetah and Walker2d settings in this ablation. The full planner benefits from the synergy of these energy terms which leads to more stable planning performance under dynamics shift.

H.3 Runtime analysis

We analyzed the computational cost of CEDGE. The primary efficiency advantage of CEDGE comes from the ability to reuse the source-domain trajectory diffusion model across different target shifts. As discussed in the experimental setup, the source diffusion model is trained once on the source dataset, while only the target-specific energy guidance functions need to be learned for each shifted target environment. Table 13 reports the runtime for two representative MuJoCo tasks. The **Full** column measures the cost incurred when all required components are trained for a new target shift. The **Reuse** column measures the additional cost required after the reusable diffusion model is already available. For baselines like IQL, Diffuser, and MOBODY, the runtime remains constant across tasks because they do not leverage component reuse and must retrain models for every new task. In contrast, CEDGE can reuse the source diffusion model and only retrain the target-specific energy module. Consequently, CEDGE-Planner reduces the adaptation cost from approximately 7.6 hours to 3.3 hours on Walker2d, and from approximately 5.6 hours to 1.9 hours on HalfCheetah. CEDGE-Policy also benefits from this reuse capability. These results strongly support the fast-adaptation design of CEDGE.

Table 5: Training hyperparameters of the source-domain trajectory diffusion model.

Hyperparameter	Value
Noise schedule	Cosine
Diffusion steps	20
Optimizer	AdamW
Learning rate	2×10^{-4}
Weight decay	10^{-5}
AdamW betas	$(0.9, 0.999)$
Batch size	64
Training steps	5×10^5
EMA decay	0.9999
Learning-rate schedule	Cosine annealing

Table 6: Architectures of the energy modules.

Module	Input	Architecture
Domain energy $\mathcal{E}_1$	(s, a, s'), (s, a)	Two-layer MLP, hidden size 256
Return energy $\mathcal{E}_2$	Trajectory τ	Half U-Net, kernel size 3
Source behavior-policy energy $\mathcal{E}_3$	(s, a)	Gaussian behavior policy, hidden size 256
Reward model $\hat{r}_\eta$	(s, a, s')	Two-layer MLP, hidden size 256

H.4 Effectiveness of guided synthetic data in policy optimization

We then evaluate whether energy-guided trajectories provide useful synthetic data for policy optimization. We fix IQL as the offline RL policy for all variants and only change the data used to train IQL. **Source** uses only the source domain D4RL dataset, **Target** uses only the limited target dataset, and **Source+Target** uses their combination. **Diffuser** uses synthetic trajectories generated by the vanilla Diffuser and combined with target dataset. As shown in Table 14, CEDGE outperforms Diffuser in all 16 settings. This shows that vanilla diffusion generation alone is not enough under dynamics shift and energy guidance is important for generating useful synthetic trajectories. CEDGE also outperforms Target in 15 out of 16 settings, showing that the generated trajectories provide useful information beyond the scarce target dataset. Compared with Source+Target, CEDGE performs better in 14 out of 16 settings, indicating that energy-guided synthetic data is usually more effective than directly combining the source and target datasets. Overall, these results show that the policy improvement comes from the quality of the energy-guided synthetic trajectories.

H.5 Diffusion Training with Limited Target Data

We compare CEDGE-Policy with a target-only diffusion baseline. The target-only baseline trains an unguided trajectory diffusion model using only the limited target dataset. Its generated trajectories are combined with the real target dataset for downstream IQL training. CEDGE-Policy uses the same downstream IQL training, but generates trajectories by reusing the source-domain diffusion model with target-specific energy guidance.

As shown in Table 15, CEDGE-Policy achieves higher mean scores in all eight gravity-shift settings on HalfCheetah and Walker2d. These results show that, with limited target data, reusing the source-domain diffusion model and adapting its generation through energy guidance is more effective than training the trajectory diffusion model directly on the target dataset.

H.6 Additional results on filtering ratio

Table 16 provides the detailed results of the filtering-ratio control experiment. We apply the same filtering criterion to both energy-guided synthetic trajectories and the original source trajectories, retain the top $\{0.1, 0.2, 0.5, 1.0\}$ fraction, and train the same downstream IQL policy with the limited target dataset. The purpose is to isolate the effect of trajectory generation from the effect of filtering.

Table 7: Training and guidance hyperparameters of the energy modules.

Hyperparameter	Value
Energy optimizer	Adam
Energy learning rate	2×10^{-4}
Energy weight decay	10^{-4}
Energy training steps	2×10^{5}
Return-energy batch size	64
Domain-energy batch size	64 source + 64 target
Domain source-target ratio	1:1
Source behavior-policy batch size	64 source transitions
Reward annotator optimizer	Adam
Reward annotator learning rate	3×10^{-4}
Reward annotator training steps	2×10^{5}
Reward annotator batch size	64
Domain guidance weight λ_1	$\{0.1, 0.5, 1.0, 2.0\}$
Return guidance weight λ_2	$\{0.1, 0.5, 1.0, 2.0\}$
Source behavior-policy guidance weight λ_3	0.1

Table 8: Planning hyperparameters for CEDGE-Planner.

Hyperparameter	Value
Sampler	DDPM
Reverse steps	20
Candidates	64
Temperature	0.5
Conditioning	Current state
Selection score	Return energy
Action	First action
Action clipping	$[-1, 1]$
Replanning	Every step

If the gain of CEDGE-Policy were caused only by filtering, filtered source data would perform comparably to filtered synthetic data at the same retention ratio. In contrast, filtered synthetic data generally yields stronger target-domain performance, showing that CEDGE-Policy benefits from generating target-compatible trajectories rather than merely filtering the original source data.

H.7 Sensitivity to energy guidance weights

We study the sensitivity of CEDGE-Planner to the energy guidance weights on HalfCheetah-Gravity-0.1. The weights λ_1, λ_2, and λ_3 control the strengths of domain energy, return energy, and policy energy, respectively. Starting from $(\lambda_1, \lambda_2, \lambda_3) = (0.1, 0.1, 0.1)$, we vary one weight over $\{0, 0.1, 0.5, 1.0, 2.0\}$ while keeping the other two fixed at 0.1. All settings use the same trained models, so the study examines the effect of changing guidance strength during sampling.

As shown in Table 17, the return-energy weight λ_2 produces relatively small performance changes over the tested range, whereas larger domain-energy and policy-energy weights can substantially reduce performance. For example, increasing λ_1 from 0.1 to 1.0 reduces the mean score from 11.29 to -14.01. Increasing λ_3 from 0.1 to 0.5 gives a higher mean score of 12.12, but further increasing it to 2.0 reduces the mean score to -9.10. These results show that the effect of guidance strength differs across energy terms and that excessive guidance can harm planning performance in this setting.

Table 9: Synthetic-data construction for CEDGE-Policy.

Hyperparameter	Value
Generation initialization	Gaussian noise
Sampler	Energy-guided reverse diffusion
Generation budget	10^6 synthetic transitions
Filtering order	Domain energy, then return energy
Filtering ratios	$\{(0.1, 0.5)\}$
Reward annotation	Learned reward model $\hat{r}_\eta(s, a, s')$
Downstream dataset	$\mathcal{D}_{\text{syn}} \cup \mathcal{D}_{\text{trg}}$
Target data	Original 5,000 target transitions

Table 10: Architecture and hyperparameters of downstream IQL.

Hyperparameter	Value
Actor architecture	Two-layer MLP, hidden size 256
Critic architecture	Two-layer MLP, hidden size 256
Policy distribution	Tanh-squashed Gaussian
Log standard deviation bounds	$[-20, 2]$
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Discount factor	0.99
Target update rate	0.005
Expectile parameter	0.7
Advantage temperature	3.0
Batch size	256 per replay buffer
Training steps	5×10^5
Evaluation frequency	Every 10,000 steps
Evaluation episodes	10
Random seeds	$\{0, 1, 2\}$

Table 11: Performance comparison on MuJoCo environments with kinematic and morphology shifts. Values are normalized scores. Best and second-best are marked per row based on the mean score. **M** and **H** denote medium and hard levels.

Env	Type	Level	BOSA	IQL	TD3-BC	DARA	MOBODY	Diffuser	Planner	Policy
HalfCheetah	morph-thigh	M	22.83 ± 0.03	20.49 ± 0.50	19.49 ± 0.50	10.90 ± 0.43	27.18 ± 6.80	12.22 ± 0.12	12.31 ± 0.59	**28.91 ± 0.41**
		H	20.77 ± 0.66	21.69 ± 0.58	22.19 ± 1.08	10.35 ± 2.10	28.51 ± 9.20	28.31 ± 0.16	**29.37 ± 5.12**	26.19 ± 2.53
	morph-torso	M	1.67 ± 0.87	1.87 ± 0.80	5.86 ± 0.21	2.91 ± 0.08	23.92 ± 12.24	17.49 ± 1.63	15.34 ± 0.92	**26.54 ± 0.98**
		H	17.09 ± 15.71	27.81 ± 3.14	2.73 ± 1.25	29.41 ± 7.88	**40.45 ± 1.26**	36.18 ± 4.94	33.08 ± 1.10	39.98 ± 1.32
	kin-footjnt	M	36.79 ± 0.92	34.71 ± 0.72	30.19 ± 3.73	33.48 ± 0.34	31.88 ± 3.70	22.53 ± 3.36	29.85 ± 1.34	**42.14 ± 3.45**
		H	14.70 ± 0.92	**31.68 ± 2.35**	14.05 ± 2.96	31.19 ± 4.08	18.51 ± 7.30	14.69 ± 2.96	27.08 ± 5.41	12.97 ± 5.42
	kin-thighjnt	M	14.92 ± 0.01	41.27 ± 3.16	41.77 ± 2.66	15.47 ± 0.62	59.17 ± 0.85	61.98 ± 8.87	61.41 ± 4.18	**62.86 ± 6.07**
		H	31.72 ± 0.17	31.60 ± 9.36	31.10 ± 9.86	31.46 ± 2.31	56.72 ± 0.08	34.49 ± 2.82	48.27 ± 3.42	**58.43 ± 3.39**
Ant	morph-halflegs	M	49.94 ± 5.98	73.65 ± 2.70	46.60 ± 6.24	70.66 ± 3.36	**79.25 ± 0.61**	71.46 ± 4.22	74.29 ± 1.45	75.08 ± 4.13
		H	58.40 ± 3.41	57.51 ± 1.25	45.07 ± 2.82	58.46 ± 4.45	63.76 ± 3.27	59.95 ± 2.51	53.12 ± 2.58	**66.39 ± 4.59**
	morph-alllegs	M	72.02 ± 3.57	61.12 ± 9.73	47.18 ± 6.89	64.83 ± 4.49	**75.24 ± 7.85**	58.01 ± 6.63	45.29 ± 0.47	59.49 ± 2.43
		H	18.50 ± 4.33	10.44 ± 0.51	14.53 ± 3.74	4.47 ± 6.18	24.13 ± 0.10	14.50 ± 0.16	11.94 ± 0.92	**27.05 ± 5.19**
	kin-anklejnt	M	72.06 ± 4.63	**77.60 ± 3.35**	44.72 ± 15.96	75.43 ± 2.03	74.92 ± 6.46	75.51 ± 1.51	76.43 ± 0.41	75.19 ± 4.62
		H	63.78 ± 7.97	62.95 ± 7.88	66.22 ± 26.98	61.06 ± 4.92	**76.97 ± 8.36**	52.49 ± 5.70	53.29 ± 1.99	73.51 ± 5.23
	kin-hipjnt	M	38.52 ± 5.88	**60.97 ± 1.72**	26.85 ± 4.26	55.73 ± 1.93	54.75 ± 4.58	42.24 ± 3.47	57.21 ± 7.26	59.07 ± 1.13
		H	50.57 ± 4.89	59.31 ± 2.92	33.85 ± 5.59	58.47 ± 3.42	**59.61 ± 3.11**	51.08 ± 4.29	44.01 ± 3.58	54.91 ± 4.45
Walker	morph-torso	M	8.26 ± 4.83	12.35 ± 1.45	18.93 ± 9.36	15.79 ± 1.33	38.67 ± 2.05	47.97 ± 7.03	**53.28 ± 7.27**	39.54 ± 1.50
		H	1.61 ± 0.12	2.30 ± 0.58	1.54 ± 0.44	3.32 ± 1.13	**11.96 ± 5.41**	1.04 ± 0.23	4.55 ± 0.95	5.63 ± 2.51
	morph-leg	M	46.70 ± 8.39	41.12 ± 13.58	22.24 ± 9.95	39.71 ± 13.67	**57.57 ± 2.00**	23.50 ± 3.49	27.09 ± 0.58	42.53 ± 1.51
		H	14.37 ± 3.34	16.15 ± 3.70	49.07 ± 2.38	13.13 ± 1.24	49.12 ± 0.52	49.17 ± 2.72	38.08 ± 1.84	**51.24 ± 7.68**
	kin-footjnt	M	17.99 ± 1.15	56.62 ± 12.10	43.31 ± 20.48	55.81 ± 1.36	**67.56 ± 3.05**	52.46 ± 3.00	61.43 ± 5.48	64.06 ± 0.55
		H	25.76 ± 15.99	6.52 ± 1.61	26.34 ± 13.24	9.63 ± 0.91	**57.93 ± 0.37**	33.93 ± 2.92	19.94 ± 1.96	45.41 ± 0.67
	kin-thighjnt	M	47.63 ± 27.26	61.28 ± 14.24	35.64 ± 11.74	56.28 ± 13.79	**69.48 ± 4.22**	41.55 ± 3.08	52.84 ± 0.79	64.49 ± 11.07
		H	48.66 ± 14.73	51.66 ± 2.05	43.88 ± 11.54	63.76 ± 2.06	**78.14 ± 2.50**	54.35 ± 4.75	72.19 ± 4.61	77.89 ± 3.73
Hopper	morph-foot	M	12.67 ± 0.00	32.99 ± 0.16	12.69 ± 0.43	40.61 ± 1.64	13.05 ± 0.48	40.84 ± 2.47	41.65 ± 1.15	**53.29 ± 1.68**
		H	10.13 ± 0.62	11.78 ± 0.09	14.15 ± 4.30	13.32 ± 1.48	65.02 ± 11.98	36.62 ± 2.68	49.67 ± 1.36	**66.82 ± 5.75**
	morph-torso	M	15.88 ± 1.18	13.38 ± 0.05	13.94 ± 0.75	13.29 ± 0.19	20.23 ± 1.29	22.05 ± 2.67	22.97 ± 2.04	**28.42 ± 2.72**
		H	11.73 ± 0.33	7.77 ± 3.73	11.54 ± 0.81	4.15 ± 0.05	12.34 ± 0.20	12.35 ± 1.27	9.41 ± 1.56	**12.75 ± 0.19**
	kin-legjnt	M	36.51 ± 1.51	42.28 ± 0.08	11.76 ± 4.60	44.67 ± 0.58	54.89 ± 0.26	57.56 ± 12.84	62.01 ± 2.00	**66.78 ± 11.34**
		H	36.13 ± 1.70	45.02 ± 4.08	18.87 ± 1.46	**65.44 ± 4.10**	56.88 ± 3.68	34.97 ± 7.97	32.19 ± 1.41	42.28 ± 2.28
	kin-footjnt	M	14.92 ± 0.01	15.58 ± 0.11	17.09 ± 0.04	15.47 ± 0.62	33.94 ± 14.81	27.56 ± 1.56	55.19 ± 4.14	**62.39 ± 6.01**
		H	31.72 ± 0.17	32.41 ± 0.16	32.21 ± 0.00	32.99 ± 0.78	33.35 ± 0.89	31.29 ± 2.61	35.93 ± 4.41	**39.13 ± 1.32**
Total			964.95	1123.88	865.60	1101.65	1515.10	1220.34	1310.71	**1551.36**

Table 12: Planner-only ablation of energy guidance on HalfCheetah and Walker2d. All variants use the same source trajectory diffusion model, planning protocol, and candidate-selection score. We report normalized target-domain scores (mean ± std over three seeds).

Env	Level	No Guidance	Domain Only	Return Only	Policy Only	CEDGE-Planner
HalfCheetah Gravity	0.1	6.83 ± 1.64	8.90 ± 1.91	8.20 ± 0.94	7.12 ± 0.61	$\mathbf{10.60 \pm 1.84}$
	0.5	48.26 ± 0.10	47.35 ± 1.51	48.92 ± 0.95	48.63 ± 1.15	$\mathbf{49.84 \pm 0.17}$
	2.0	26.75 ± 0.01	27.04 ± 0.05	26.98 ± 0.03	27.05 ± 0.03	$\mathbf{27.19 \pm 0.06}$
	5.0	-0.17 ± 0.04	3.41 ± 0.04	1.24 ± 0.06	0.48 ± 0.006	$\mathbf{4.04 \pm 0.07}$
HalfCheetah Friction	0.1	10.03 ± 0.10	10.32 ± 0.04	10.50 ± 0.08	10.33 ± 0.07	$\mathbf{10.55 \pm 0.05}$
	0.5	71.07 ± 0.19	71.74 ± 0.03	71.96 ± 0.23	71.16 ± 0.68	$\mathbf{72.13 \pm 0.27}$
	2.0	46.98 ± 0.96	47.52 ± 0.24	47.74 ± 0.15	47.07 ± 0.34	$\mathbf{47.93 \pm 0.14}$
	5.0	27.20 ± 0.74	32.49 ± 0.64	24.13 ± 1.38	26.70 ± 0.37	$\mathbf{32.79 \pm 0.91}$
Walker2d Gravity	0.1	16.84 ± 0.11	22.27 ± 1.49	21.99 ± 1.69	16.20 ± 2.02	$\mathbf{34.08 \pm 0.78}$
	0.5	35.58 ± 1.40	38.76 ± 1.76	27.26 ± 3.03	24.15 ± 1.43	$\mathbf{47.84 \pm 1.52}$
	2.0	6.80 ± 0.09	6.87 ± 0.09	6.51 ± 0.14	5.90 ± 0.06	$\mathbf{6.97 \pm 0.04}$
	5.0	3.84 ± 0.08	4.12 ± 0.04	4.36 ± 0.18	3.49 ± 0.33	$\mathbf{5.36 \pm 0.16}$
Walker2d Friction	0.1	8.26 ± 0.26	8.70 ± 0.34	7.96 ± 0.04	8.05 ± 0.31	$\mathbf{8.71 \pm 0.14}$
	0.5	68.76 ± 2.04	67.65 ± 0.72	66.74 ± 3.08	57.67 ± 2.16	$\mathbf{70.36 \pm 0.76}$
	2.0	51.46 ± 1.83	51.69 ± 0.18	41.87 ± 1.16	25.99 ± 2.50	$\mathbf{52.66 \pm 1.20}$
	5.0	4.44 ± 0.03	4.90 ± 0.08	4.61 ± 0.06	4.74 ± 0.14	$\mathbf{6.19 \pm 0.41}$

Table 13: Runtime analysis for Walker2d-Gravity-2.0 and HalfCheetah-Gravity-2.0. Experiments are conducted on one NVIDIA RTX A5000 GPU with 24GB memory. **Full** reports the whole cost of CEDGE. **Reuse** reports the extra cost when diffusion models are already available.

Method	Walker2d-Gravity-2.0		HalfCheetah-Gravity-2.0	
	Full	Reuse	Full	Reuse
IQL	$\sim$ 3.0h	$\sim$ 3.0h	$\sim$ 3.0h	$\sim$ 3.0h
MOBODY	$\sim$ 7.8h	$\sim$ 7.8h	$\sim$ 6.1h	$\sim$ 6.1h
Diffuser	$\sim$ 7.6h	$\sim$ 7.6h	$\sim$ 5.5h	$\sim$ 5.5h
CEDGE-Planner	$\sim$ 7.6h	$\sim$ **3.3**h	$\sim$ 5.6h	$\sim$ **1.9**h
CEDGE-Policy	$\sim$ 9.1h	$\sim$ 4.8h	$\sim$ 7.4h	$\sim$ 3.7h

Table 14: Effectiveness of guided synthetic data on HalfCheetah and Walker2d. All variants train IQL with the indicated data source; Diffuser and CEDGE use the same synthetic-data budget and are combined with the target dataset. Scores are mean ± std over three seeds.

Shift Type	Level	Source	Target	Source+Target	Diffuser	CEDGE-Policy
HalfCheetah Gravity	0.1	8.87 ± 1.24	12.07 ± 0.99	9.62 ± 4.27	7.77 ± 8.70	$\mathbf{20.04 \pm 3.52}$
	0.5	$\mathbf{45.01 \pm 3.54}$	1.52 ± 0.17	44.23 ± 2.93	-0.19 ± 0.29	2.54 ± 0.59
	2.0	31.43 ± 0.42	27.77 ± 6.53	31.34 ± 1.68	13.41 ± 7.26	$\mathbf{41.66 \pm 5.84}$
	5.0	2.80 ± 0.77	73.81 ± 0.12	44.00 ± 23.13	55.10 ± 15.36	$\mathbf{75.34 \pm 1.63}$
HalfCheetah Friction	0.1	10.59 ± 0.27	63.76 ± 0.21	26.39 ± 11.35	62.47 ± 0.44	$\mathbf{64.31 \pm 0.39}$
	0.5	$\mathbf{76.15 \pm 0.26}$	44.86 ± 8.99	69.80 ± 0.64	4.78 ± 0.22	52.06 ± 6.41
	2.0	$\mathbf{52.02 \pm 0.24}$	35.95 ± 2.36	46.04 ± 2.04	5.25 ± 0.70	49.15 ± 4.03
	5.0	16.52 ± 3.55	64.11 ± 5.14	44.96 ± 6.78	5.74 ± 5.39	$\mathbf{68.56 \pm 1.94}$
Walker2d Gravity	0.1	16.59 ± 1.25	67.29 ± 1.19	16.04 ± 7.60	44.49 ± 3.35	$\mathbf{69.48 \pm 1.57}$
	0.5	16.18 ± 4.29	36.30 ± 4.19	42.05 ± 10.52	12.82 ± 5.33	$\mathbf{44.36 \pm 1.07}$
	2.0	8.68 ± 0.68	41.17 ± 3.46	25.69 ± 10.70	19.18 ± 4.46	$\mathbf{44.36 \pm 2.35}$
	5.0	2.22 ± 1.64	$\mathbf{55.78 \pm 14.38}$	5.42 ± 0.29	5.27 ± 0.18	54.01 ± 10.25
Walker2d Friction	0.1	5.39 ± 0.05	52.97 ± 9.10	5.72 ± 0.23	10.27 ± 1.90	$\mathbf{61.66 \pm 4.70}$
	0.5	66.78 ± 0.59	62.21 ± 6.37	66.26 ± 3.03	25.25 ± 8.28	$\mathbf{70.16 \pm 2.25}$
	2.0	45.66 ± 5.00	72.96 ± 0.61	65.40 ± 7.13	17.71 ± 2.88	$\mathbf{73.05 \pm 0.02}$
	5.0	4.82 ± 0.06	22.77 ± 3.07	5.39 ± 0.03	6.56 ± 0.64	$\mathbf{36.44 \pm 8.23}$

Table 15: Comparison with the target-only unguided diffusion baseline under gravity shifts. Scores are normalized target-domain scores (mean ± std over three seeds).

Shift Type	Level	Target-only	CEDGE-Policy
HalfCheetah Gravity	0.1	15.04 ± 3.68	$\mathbf{20.04 \pm 3.52}$
	0.5	1.02 ± 0.47	$\mathbf{2.54 \pm 0.59}$
	2.0	25.24 ± 7.62	$\mathbf{41.66 \pm 5.84}$
	5.0	73.55 ± 0.18	$\mathbf{75.34 \pm 1.63}$
Walker2d Gravity	0.1	45.31 ± 0.87	$\mathbf{69.48 \pm 1.57}$
	0.5	5.56 ± 0.16	$\mathbf{44.36 \pm 1.07}$
	2.0	17.42 ± 1.43	$\mathbf{44.36 \pm 2.35}$
	5.0	7.04 ± 1.85	$\mathbf{54.01 \pm 10.25}$

Table 16: Filtering-only control. We apply the same filtering criterion to both the synthetic dataset and the original source dataset, retain the top $\{0.1, 0.2, 0.5, 1.0\}$ fraction, and train the same downstream policy. If the improvement came solely from filtering, then filtered source data should perform comparably to filtered synthetic data at matched retention ratios.

Task	Shift	Synthetic performance				Source performance			
		0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
HalfCheetah Gravity	0.1	20.04 ± 3.52	12.03 ± 2.87	11.76 ± 4.93	7.58 ± 2.18	3.25 ± 2.78	0.87 ± 1.44	4.46 ± 2.69	9.62 ± 4.27
	0.5	2.54 ± 0.59	2.62 ± 0.17	2.31 ± 0.58	2.10 ± 0.90	12.87 ± 1.36	19.60 ± 3.84	14.87 ± 4.32	44.23 ± 2.93
	2.0	41.66 ± 5.84	25.04 ± 7.90	39.19 ± 6.05	25.24 ± 7.88	26.21 ± 2.71	28.23 ± 2.56	30.66 ± 0.83	31.34 ± 1.68
	5.0	75.34 ± 1.63	73.61 ± 0.25	70.94 ± 1.54	71.72 ± 2.13	64.61 ± 8.96	52.14 ± 22.09	52.17 ± 20.60	44.00 ± 23.13
HalfCheetah Friction	0.1	64.31 ± 0.39	63.49 ± 0.15	64.19 ± 0.31	64.22 ± 0.19	25.44 ± 7.60	26.36 ± 8.52	20.99 ± 3.32	26.39 ± 11.35
	0.5	52.06 ± 6.41	26.20 ± 6.53	30.56 ± 7.01	34.23 ± 5.68	59.02 ± 4.62	61.37 ± 2.63	66.14 ± 1.89	69.80 ± 0.64
	2.0	49.15 ± 4.03	38.97 ± 5.97	38.03 ± 4.81	31.69 ± 7.71	26.41 ± 6.57	31.46 ± 5.13	24.99 ± 7.83	46.04 ± 2.04
	5.0	68.56 ± 1.94	66.04 ± 2.17	66.63 ± 1.08	67.16 ± 1.77	50.35 ± 7.28	51.39 ± 10.36	45.68 ± 2.94	44.96 ± 6.78

Table 17: Sensitivity of CEDGE-Planner to the energy guidance weights on HalfCheetah-Gravity-0.1. The three entries at 0.1 share the same baseline configuration and result.

Varied weight	0	0.1	0.5	1.0	2.0
λ_1 (Domain)	9.61 ± 2.17	11.29 ± 2.55	1.99 ± 1.39	-14.01 ± 2.05	-10.53 ± 2.02
λ_2 (Return)	8.48 ± 0.38	11.29 ± 2.55	9.14 ± 0.89	8.64 ± 0.81	8.22 ± 1.21
λ_3 (Policy)	10.38 ± 0.96	11.29 ± 2.55	12.12 ± 1.33	3.34 ± 1.70	-9.10 ± 0.78