DIFFUSION-AUGMENTED MARKOV DECISION PROCESSES FOR MAXIMUM ENTROPY REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models provide an expressive framework for sampling from complex, unnormalized distributions. In this work, we extend Maximum Entropy Reinforcement Learning (ME-RL) to diffusion-based policies by introducing *Diffusion-Augmented Markov Decision Processes* (DA-MDPs). DA-MDPs interpret each reverse-diffusion transition as an individual reinforcement-learning decision, while only the final denoised action is executed in the environment. Our DA-MDPs follow from a principled derivation based on the variational-inference formulation of ME-RL. By augmenting policy and target trajectories with intermediate diffusion variables, we obtain a tractable reverse-KL upper bound via the data-processing inequality. This bound decomposes across denoising transitions, yielding diffusion-augmented variants of soft rewards, value functions, and local policy objectives. This provides a general framework for adapting ME-RL algorithms to diffusion policies while differentiating through only one diffusion transition at a time. We instantiate the framework with PPO, REPPO, and a maximum-entropy extension of WPO. Experiments demonstrate improved continuous-control performance, benefits from additional diffusion steps, and memory-efficient training. On the StackCube and PushT manipulation tasks, DA-MDP methods learn alternative successful strategies from the same initial state and achieve higher success rates and generally higher success-weighted mode entropy than the Gaussian ME-RL baseline. We also demonstrate successful training when using action chunking.

1 INTRODUCTION

Diffusion models have emerged as powerful generative models for complex distributions (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021a). Diffusion samplers extend this idea to distributions known only through an unnormalized density: they learn a reverse process that transports samples from a simple prior toward high-density regions of the target (Zhang & Chen, 2022; Berner et al., 2022; Vargas et al., 2023; 2024; Richter & Berner, 2024; Sanokowski et al., 2024; 2025a). Maximum-entropy reinforcement learning (ME-RL) naturally fits this setting because it can be viewed as sampling trajectories from an unnormalized distribution that favors high cumulative reward (Todorov, 2008; Ziebart, 2010; Kappen et al., 2012; Levine, 2018). Diffusion policies are therefore a natural choice for ME-RL: they provide a flexible representation of complex, multimodal action distributions (Ren et al., 2025; Celik et al., 2025; Dong et al., 2025; Ma et al., 2025; McAllister et al., 2025). The difficulty, however, in using diffusion models in ME-RL is the computation of the likelihood of the action executed in the environment. For a diffusion policy, obtaining this likelihood requires marginalizing all intermediate diffusion variables and is generally intractable. Existing full-path objectives avoid this marginalization, but require backpropagation through the complete diffusion chain (Celik et al., 2025; Le et al., 2026).

We introduce **Diffusion-Augmented Markov Decision Processes (DA-MDPs)** to address these challenges by treating each reverse-diffusion transition as an individual RL decision. This formulation replaces the intractable marginal action likelihood with tractable transition likelihoods and requires backpropagation through only one diffusion step at a time. At each denoising step, the policy samples a partially denoised action conditioned on the current noisy action and the environment state. Only the final denoised action is executed in the environment.

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.

Each denoising transition receives a reward based on the log ratio of the forward and reverse transition densities. This reward encourages consistency with the prescribed noising process while promoting entropy maximization at each reverse step. To derive it, we augment the policy and target trajectories with intermediate diffusion variables and apply the data-processing inequality. The resulting upper bound on the reverse KL decomposes into local critic and actor objectives.

Our main contributions are as follows:

- We derive **DA-MDPs** from a reverse KL between augmented trajectory distributions, obtaining a principled local reward for each diffusion transition.
- We extend TRPO's monotonic-improvement bound (Schulman et al., 2015) to DA-MDPs, giving sufficient conditions for improved augmented return, including when forward kernels are learned.
- We instantiate **DA: PPO**, **DA: REPPO**, and **DA: WPO** from PPO (Schulman et al., 2017), REPPO (Voelcker et al., 2026), and our maximum-entropy extension of WPO (Pfau et al., 2025).
- Across twelve continuous-control tasks, the DA variants improve on their corresponding base algorithms. On StackCube, DA policies learn both stacking orders, including when executing action chunks. On PushT, DA: REPPO maintains successful clockwise and counter-clockwise strategies near the direction-switching boundary, where stochastic Gaussian REPPO rarely succeeds. Further experiments demonstrate benefits from additional diffusion steps and faster training than a full-chain baseline under tight memory budgets.

2 Setup and Background

We first review the two ingredients required for our construction: the probabilistic-inference formulation of ME-RL and discrete-time diffusion policies. A summary of the notation used throughout the paper is provided in Table 1 in App. B.

2.1 Maximum-Entropy Reinforcement Learning as Variational Inference

Reinforcement learning can be interpreted as an inference problem in which the objective is to sample trajectories with large cumulative reward. Consider a finite-horizon MDP with initial-state distribution $p(s_0)$, transition dynamics $p(s_{t+1} \mid s_t, a_t)$, and reward function $R_{\text{env}}(s_t, a_t)$. Here, $s_t = (x_t, t) \in \mathcal{S}$ denotes the time-dependent state, $a_t \in \mathcal{A} \subset \mathbb{R}^N$ denotes the action, where $\mathcal{A}$ is compact, and T is the final environment-time index. We further assume that the reward is uniformly bounded above: $R_{\text{env}}(s, a) \leq R_{\max} < \infty$.

The unnormalized target over action trajectories $\widetilde{\pi}(a_{0:T})$ is obtained by marginalizing the unnormalized joint state–action target $\widetilde{\pi}(a_{0:T}, s_{0:T+1})$:

$$\widetilde{\pi}(a_{0:T}) := \int_{\mathcal{S}^{T+2}} \widetilde{\pi}(a_{0:T}, s_{0:T+1})\, ds_{0:T+1}, \quad \pi(a_{0:T}) := \frac{\widetilde{\pi}(a_{0:T})}{\mathcal{Z}},$$

$$\widetilde{\pi}(a_{0:T}, s_{0:T+1}) := p(s_0) \prod_{t=0}^{T} p(s_{t+1} \mid s_t, a_t) \exp(\alpha R_{\text{env}}(s_t, a_t))\,.$$

Here, $\alpha := 1/\mathcal{T} > 0$ is the inverse of the entropy temperature $\mathcal{T} > 0$. The temperature $\mathcal{T}$ controls the trade-off between reward maximization and entropy: larger values place greater weight on entropy, whereas smaller values concentrate the target distribution on high-reward trajectories. A tilde denotes an unnormalized density, whereas the corresponding symbol without a tilde denotes the normalized distribution. The partition function is $\mathcal{Z} := \int_{\mathcal{A}^{T+1}} \widetilde{\pi}(a_{0:T})\, da_{0:T}$, and the compact-action and bounded-reward assumptions imply $0 < \mathcal{Z} < \infty$ for every fixed finite horizon; the same $\mathcal{Z}$ normalizes the joint target. Because $\prod_{t=0}^{T} \exp(\alpha R_{\text{env}}(s_t, a_t)) = \exp(\alpha \sum_{t=0}^{T} R_{\text{env}}(s_t, a_t))$, trajectories with larger cumulative reward receive exponentially more probability mass.

A policy $q_\theta(a_t \mid s_t)$ analogously induces the action-trajectory marginal and the corresponding joint variational distribution

$$q_\theta(a_{0:T}) := \int_{\mathcal{S}^{T+2}} q_\theta(a_{0:T}, s_{0:T+1})\, ds_{0:T+1}, \quad q_\theta(a_{0:T}, s_{0:T+1}) := p(s_0) \prod_{t=0}^{T} p(s_{t+1} \mid s_t, a_t) q_\theta(a_t \mid s_t).$$

In principle, variational inference would minimize the reverse KL divergence between $q_\theta(a_{0:T})$ and $\pi(a_{0:T})$. Evaluating this divergence is generally intractable, however, because both marginal densities require integration over all state trajectories. Applying the data-processing inequality (DPI) to this marginalization gives

$$D_{\mathrm{KL}}(q_\theta(a_{0:T}) \,\|\, \pi(a_{0:T})) \leq D_{\mathrm{KL}}(q_\theta(a_{0:T}, s_{0:T+1}) \,\|\, \pi(a_{0:T}, s_{0:T+1}))\,, \tag{1}$$

and we can therefore minimize the tractable joint reverse KL on the right-hand side as an upper bound on the desired action-marginal objective. Its zero-divergence optimality condition is $q_{\theta^\star}(a_{0:T}, s_{0:T+1}) = \pi(a_{0:T}, s_{0:T+1})$ and implies $q_{\theta^\star}(a_{0:T}) = \pi(a_{0:T})$. See App. F.2.1 for more details on the DPI.

Expanding the temperature-scaled finite-horizon joint KL yields the negative maximum-entropy return, up to a policy-independent constant. With stationary dynamics and policies, we introduce $\mathcal{T}_t = \gamma^t \mathcal{T}$ and $R_t = \gamma^t R_{\mathrm{env}}$, with $\gamma \in [0, 1)$. Taking $T \to \infty$ yields the discounted ME-RL objective below. An equivalent joint trajectory-KL formulation uses geometric stopping (App. F.1.3).

$$\mathcal{J}_{\mathrm{ME}}(\theta) := \mathbb{E}_{q_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \left(R_{\mathrm{env}}(s_t, a_t) - \mathcal{T} \log q_\theta(a_t \mid s_t)\right)\right]. \tag{2}$$

Here, the expectation is over trajectories initialized with $s_0 \sim p(\cdot)$ and generated by the policy and environment dynamics. In expectation subscripts, we write $a \sim q_\theta$ for $a \sim q_\theta(\cdot \mid s)$ and $s' \sim p$ for $s' \sim p(\cdot \mid s, a)$. Maximizing $\mathcal{J}_{\mathrm{ME}}(\theta)$ maximizes the expected environment reward together with the entropy of the policy. Applying the policy-gradient theorem to Eq. 2, followed by a reverse application of the log-derivative trick, gives the stationary local ME-RL surrogate. Let θ_{roll} denote the frozen parameters of the policy used to collect the rollout. For a policy q, let $\rho_t^q(s)$ denote the probability density of the continuous state s_t evaluated at s, starting from $s_0 \sim p(s_0)$. The normalized discounted state-visitation density is $d^{q_{\theta_{\mathrm{roll}}}}(s) := (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \rho_t^{q_{\theta_{\mathrm{roll}}}}(s)$. The resulting surrogate objective is

$$\mathcal{L}_{\mathrm{ME}}(\theta; \theta_{\mathrm{roll}}) = \frac{\mathcal{T}}{1-\gamma} \mathbb{E}_{s' \sim d^{q_{\theta_{\mathrm{roll}}}}}\left[D_{\mathrm{KL}}\left(q_\theta(\cdot \mid s') \,\middle\|\, \frac{\exp(\alpha Q^{q_{\theta_{\mathrm{roll}}}}(s', \cdot))}{Z(s')}\right)\right]. \tag{3}$$

The stationary soft value functions satisfy

$$Q^{q_\theta}(s, a) = R_{\mathrm{env}}(s, a) + \gamma\, \mathbb{E}_{s' \sim p}[V^{q_\theta}(s')], \qquad V^{q_\theta}(s) = \mathbb{E}_{a \sim q_\theta}[Q^{q_\theta}(s, a) - \mathcal{T} \log q_\theta(a \mid s)].$$

Both $Q^{q_{\theta_{\mathrm{roll}}}}$ and $d^{q_{\theta_{\mathrm{roll}}}}$ are frozen during the actor update. More precisely, minimizing the surrogate matches maximizing the return at the rollout parameters, i.e. $\nabla_\theta \mathcal{L}_{\mathrm{ME}}(\theta; \theta_{\mathrm{roll}})|_{\theta=\theta_{\mathrm{roll}}} = -\nabla_\theta \mathcal{J}_{\mathrm{ME}}(\theta)|_{\theta=\theta_{\mathrm{roll}}}$. Although the gradient identity is exact only at the rollout parameters, it provides the local ascent direction underlying the trust-region update. The theoretical improvement bound of Schulman et al. (2015) shows that a positive surrogate gain implies improvement of the true objective whenever the distribution shift induced by the policy update remains sufficiently small.

2.2 Discrete-Time Diffusion Policies

A diffusion sampler targets sampling from $\pi_0(x^0) = \widetilde{\pi}_0(x^0)/Z$, given its unnormalized density. The forward process progressively noises target samples toward a simple reference distribution q_{prior}, while the learned reverse process denoises reference samples toward the target (Vargas et al., 2023; Berner et al., 2022). Their path distributions are

$$\pi_\theta(x^{0:K}) = \pi_0(x^0) \prod_{k=1}^{K} \pi_\theta(x^k \mid x^{k-1}), \qquad q_\theta(x^{0:K}) = q_{\mathrm{prior}}(x^K) \prod_{k=1}^{K} q_\theta(x^{k-1} \mid x^k). \tag{4}$$

Evaluating the reverse clean-sample density $q_\theta(x^0)$ generally requires intractable marginalization over the intermediate diffusion variables. Applying the DPI again to the projection onto x^0 gives

$$D_{\mathrm{KL}}\big(q_\theta(x^0) \,\big\|\, \pi_0(x^0)\big) \leq D_{\mathrm{KL}}\big(q_\theta(x^{0:K}) \,\big\|\, \pi_\theta(x^{0:K})\big)\,. \tag{5}$$

The path-space KL on the right can be estimated from reverse samples using the reference, transition, and unnormalized target densities, up to the parameter-independent constant $\log Z$.

For bounded action spaces, as commonly encountered in continuous-control RL, we consider a discretized variance-preserving SDE (VP-SDE) with Gaussian kernels and $a = \tanh(x^0)$, or a Beta diffusion process on $[0, 1]^N$ with $a = 2x^0 - 1$ (Avdeyev et al., 2023). In all DA-MDP experiments, we jointly learn the reverse policy and the diffusion coefficients governing the forward and reverse schedules. Below, a^0 denotes the executed action, including the terminal change of variables. We compare these diffusion processes in App. E.3.2.

3 Diffusion-Augmented Markov Decision Processes

We first derive the tractable diffusion-path objective from ME-RL, then construct the augmented MDP and its local critic and actor objectives.

3.1 Deriving the DA-MDP Objective from ME-RL

We begin with a proof sketch of the DA-MDP objective, which is detailed in App. F.2. Because the executed-action marginal of a diffusion policy is intractable, we augment both trajectory distributions from Sec. 2.1 with all intermediate diffusion variables. The normalized augmented policy and target distributions are

$$\begin{aligned} q_\theta(a_{0:T}^{0:K}, s_{0:T+1}) &= p(s_0) \prod_{t=0}^{T} \left[p(s_{t+1} \mid s_t, a_t^0) q_{\text{prior}}(a_t^K) \prod_{k=1}^{K} q_\theta(a_t^{k-1} \mid a_t^k, s_t) \right], \\ \pi_\theta(a_{0:T}^{0:K}, s_{0:T+1}) &= \frac{p(s_0)}{\mathcal{Z}} \prod_{t=0}^{T} \left[p(s_{t+1} \mid s_t, a_t^0) e^{\alpha R_{\text{env}}(s_t, a_t^0)} \prod_{k=1}^{K} \pi_\theta(a_t^k \mid a_t^{k-1}, s_t) \right]. \end{aligned} \tag{6}$$

Marginalizing $a_{0:T}^{1:K}$ recovers the state–executed-action distributions used in Eq. 1, so the data-processing inequality gives

$$D_{\text{KL}}(q_\theta(a_{0:T}, s_{0:T+1}) \,\|\, \pi(a_{0:T}, s_{0:T+1})) \le D_{\text{KL}}\big(q_\theta(a_{0:T}^{0:K}, s_{0:T+1}) \,\big\|\, \pi_\theta(a_{0:T}^{0:K}, s_{0:T+1})\big). \tag{7}$$

This then yields a tractable reverse KL between the augmented policy and target distributions, which upper-bounds their reverse KL over executed-action sequences (Eqs. 1 and 7). Its zero-divergence optimality condition is $q_{\theta^\star}(a_{0:T}^{0:K}, s_{0:T+1}) = \pi_{\theta^\star}(a_{0:T}^{0:K}, s_{0:T+1})$. As with any DPI bound, decreasing the upper bound does not imply that the marginal objective decreases monotonically after every parameter update. See App. F.2.1 for more details.

Expanding the augmented joint KL cancels the shared environment dynamics, while the fixed prior contributes only a parameter-independent constant. Applying the reward and temperature scaling from Sec. 2.1 and taking $T \to \infty$ yields the stationary discounted DA-MDP return

$$\mathcal{J}_{\text{DA}}(\theta) := \mathbb{E}_{q_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \left(R_{\text{env}}(s_t, a_t^0) - \mathcal{T} \sum_{k=1}^{K} \log \frac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi_\theta(a_t^k \mid a_t^{k-1}, s_t)} \right) \right]. \tag{8}$$

Here, the expectation is over augmented trajectories generated as in Eq. 6, including the initial state, prior samples, all denoising steps, and environment transitions. Equivalently, the temperature-scaled geometrically stopped augmented KL equals $-\mathcal{J}_{\text{DA}}(\theta)$ up to parameter-independent constants (App. F.2). The expression in parentheses identifies the local objective for each denoising decision: the environment reward appears only at $k = 1$, while the reverse–forward log ratio appears at every diffusion step.

3.2 Augmented MDP and Local Objectives

Applying the policy-gradient theorem to $\mathcal{J}_{\text{DA}}$ in Eq. 8 yields the local critic and actor objectives and the associated DA-MDP construction presented in this section (see App. F.2). We treat each denoising transition as a decision in an augmented MDP. For environment time $t = 0, 1, \ldots$ and denoising phase $k = K, \ldots, 1$, define

$$\tilde{t}(t, k) = tK + (K - k), \qquad \tilde{s}_{\tilde{t}(t,k)} = (s_t, a_t^k, k), \qquad \tilde{a}_{\tilde{t}(t,k)} = a_t^{k-1}.$$

The augmented state contains the environment state, noisy action, and diffusion index; the action is the next denoised sample. Writing a generic successor as $\tilde{s}' = (s', a', k')$, the transition kernel is

$$p_{\mathrm{DA}}\big(\tilde{s}' \mid (s_t, a_t^k, k), a_t^{k-1}\big) = \begin{cases} \delta\big(\tilde{s}' = (s_t, a_t^{k-1}, k-1)\big), & k > 1, \\ p(s' \mid s_t, a_t^0)\, q_{\mathrm{prior}}(a')\, \delta(k' = K), & k = 1. \end{cases}$$

Here, $\delta(u = v)$ denotes the Dirac measure concentrated at v. For $k > 1$, the successor is deterministic and the environment state remains fixed. At $k = 1$, the environment executes a_t^0, transitions to s_{t+1}, and a fresh prior sample starts the next denoising chain. The environment reward is $\widetilde{R}_{\mathrm{DA}} = \mathbf{1}_{\{k=1\}} R_{\mathrm{env}}(s_t, a_t^0)$. We set $\bar{\gamma}(k) = 1$ for $k > 1$ and $\bar{\gamma}(1) = \gamma$, so discounting occurs only when the environment advances. Figure 4 in App. A illustrates the construction.

For the frozen rollout parameters θ_{roll}, the soft Bellman equations are

$$\begin{aligned} Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s}_{\tilde{t}}, a_t^{k-1}) &= \widetilde{R}_{\mathrm{DA}}(\tilde{s}_{\tilde{t}}, a_t^{k-1}) + \bar{\gamma}(k) \mathbb{E}_{\tilde{s}_{\tilde{t}+1}} \left[V_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s}_{\tilde{t}+1}) \right], \\ V_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s}_{\tilde{t}}) &= \mathbb{E}_{a_t^{k-1} \sim q_{\theta_{\mathrm{roll}}}(\cdot \mid \tilde{s}_{\tilde{t}})} \left[Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s}_{\tilde{t}}, a_t^{k-1}) - \mathcal{T} \log \frac{q_{\theta_{\mathrm{roll}}}(a_t^{k-1} \mid a_t^k, s_t)}{\pi_{\theta_{\mathrm{roll}}}(a_t^k \mid a_t^{k-1}, s_t)} \right]. \end{aligned} \tag{9}$$

where $\tilde{s}_{\tilde{t}+1} \sim p_{\mathrm{DA}}(\cdot \mid \tilde{s}_{\tilde{t}}, a_t^{k-1})$. The reverse–forward log ratio replaces the usual entropy term.

Let $\rho_{t,k}^q(s, a^k)$ denote the joint density of the environment state and noisy action under policy q at environment time t and denoising phase k. For $\tilde{s} = (s, a^k, k)$, the normalized discounted occupancy is $d_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(s, a^k, k) := \frac{1-\gamma}{K} \sum_{t=0}^{\infty} \gamma^t \rho_{t,k}^{q_{\theta_{\mathrm{roll}}}}(s, a^k)$, which weights all K denoising phases equally. Following the local-surrogate construction in Eq. 3, the actor loss is

$$\mathcal{L}_{\mathrm{DA}}(\theta; \theta_{\mathrm{roll}}) = \frac{K\mathcal{T}}{1-\gamma} \mathbb{E}_{\tilde{s} \sim d_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}} \left[D_{\mathrm{KL}}^{\mathrm{u}} \Big(q_\theta(\cdot \mid \tilde{s}) \,\Big\|\, \pi_\theta(a^k \mid \cdot, s) \exp\Big(\alpha Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s}, \cdot) \Big) \Big) \right]. \tag{10}$$

Here, $D_{\mathrm{KL}}^{\mathrm{u}}$ denotes a KL-like variational free energy with an unnormalized right-hand argument (see Eq. 76). At the rollout parameters, minimizing this loss is locally equivalent to maximizing the DA-MDP return: $\nabla_\theta \mathcal{L}_{\mathrm{DA}}(\theta; \theta_{\mathrm{roll}})|_{\theta=\theta_{\mathrm{roll}}} = -\nabla_\theta \mathcal{J}_{\mathrm{DA}}(\theta)|_{\theta=\theta_{\mathrm{roll}}}$. App. E.3.10 verifies this identity numerically on an exactly enumerable finite MDP with learned forward diffusion kernels.

Proposition 3.1 (DA-MDP return improvement)**.** *Under the assumptions in App. F.3.2, let* $\Delta\mathcal{J}_{\mathrm{DA}} := \mathcal{J}_{\mathrm{DA}}(\theta) - \mathcal{J}_{\mathrm{DA}}(\theta_{\mathrm{roll}})$ *and define the surrogate gain* $\Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} := \mathcal{L}_{\mathrm{DA}}(\theta_{\mathrm{roll}}; \theta_{\mathrm{roll}}) - \mathcal{L}_{\mathrm{DA}}(\theta; \theta_{\mathrm{roll}})$. *For* $\tilde{s} = (s, a^k, k)$, *define* $D_q := \sup_{\tilde{s}} D_{\mathrm{KL}}(q_\theta(\cdot \mid \tilde{s}) \| q_{\theta_{\mathrm{roll}}}(\cdot \mid \tilde{s}))$, $\epsilon_A := \sup_{\tilde{s}, a^{k-1}} |A_{\mathrm{roll}}(\tilde{s}, a^{k-1})|$, *and* $\epsilon_G := \sup_{\tilde{s}} \left| \mathbb{E}_{a^{k-1} \sim q_\theta(\cdot \mid \tilde{s})} \left[\log \frac{\pi_\theta(a^k \mid a^{k-1}, s)}{\pi_{\theta_{\mathrm{roll}}}(a^k \mid a^{k-1}, s)} \right] \right|$, *where* A_{roll} *is the soft rollout advantage (Eq. 151). With* $B_{K,\gamma} := \frac{K}{1-\gamma} \left(K - 1 + \frac{2\gamma K}{1-\gamma} \right)$, *the return change satisfies*

$$\Delta\mathcal{J}_{\mathrm{DA}} \geq \Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} - \left(\epsilon_A B_{K,\gamma} + \frac{K\mathcal{T}}{1-\gamma} \right) D_q - \mathcal{T} B_{K,\gamma} \epsilon_G \sqrt{D_q / 2}. \tag{11}$$

Thus, the return improves whenever the surrogate gain exceeds the displayed penalties. Fixed forward kernels give $\epsilon_G = 0$. App. F.3.2 provides the proof.

3.3 Diffusion-Augmented MDP Algorithms: Examples

Our DA-MDP algorithms limit changes to the reverse policy using update constraints adapted from their Gaussian-policy counterparts. These constraints aim to keep the penalties in Proposition 3.1 small relative to the surrogate gain. We instantiate the framework with PPO, REPPO, and WPO by applying their actor and critic updates to stored local denoising transitions. Each DA-MDP diffusion variant is explained in detail in App. D and shortly summarized below.

DA: PPO applies the log-derivative trick to $\mathcal{L}_{\mathrm{DA}}$ in Eq. 10, using importance weights between the current and rollout reverse kernels for stored local denoising transitions. It estimates their advantages using a value critic and generalized advantage estimation on the DA-MDP rewards. Learning the forward process requires an additional forward log-density gradient (Sanokowski et al., 2025b). PPO's clipped surrogate controls reverse-policy updates. Setting $\mathcal{T} = 0$ recovers a DPPO-style local PPO objective on the denoising-augmented MDP (Ren et al., 2025). See App. D.6 for details.

DA: REPPO applies the reparameterization trick to $\mathcal{L}_{\mathrm{DA}}$ in Eq. 10, using an action-value critic on the augmented MDP and KL-based trust-region constraints between the rollout and current reverse kernels. App. D.3 provides the complete objective and implementation details.

DA: WPO. To instantiate the framework with WPO, we derive its maximum-entropy extension ME-WPO in App. F.4. The complete Algorithm is presented in App. C.3. Projecting the Wasserstein gradient flow of the soft policy-improvement objective onto the policy parameters makes the update depend on both the action-value gradient and the policy's entropy score. When extending this to DA-MDPs we obtain DA: WPO by applying the same score-matching construction to the local diffusion target in Eq. 10. This yields

$$\mathcal{L}_{\mathrm{DA:WPO}}(\theta) = \frac{\mathcal{T}}{2}\mathbb{E}_{\substack{\tilde{s}_{\tilde{t}}\sim d_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}} \\ a_t^{k-1}\sim q_{\theta^*}(\cdot|\tilde{s}_{\tilde{t}})}}\left[\left\|\nabla_{a_t^{k-1}}\left(\log\frac{q_\theta(a_t^{k-1}\mid a_t^k,s_t)}{\pi_\theta(a_t^k\mid a_t^{k-1},s_t)} - \alpha Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s}_{\tilde{t}},a_t^{k-1})\right)\right\|^2\right]. \quad (12)$$

The augmented state is sampled from the frozen discounted visitation distribution $d_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}$, and a_t^{k-1} is then drawn from the current reverse kernel with gradients stopped. DA: WPO otherwise uses the same augmented rollout, critic, and KL-based trust-region constraints on the reverse kernels as DA: REPPO and replaces only its policy loss by Eq. 12. For variance-preserving Gaussian reverse kernels, we approximate full parameter-space inverse-Fisher preconditioning using adjusted per-coordinate blocks in the local control and diffusion-coefficient coordinates. App. D.4 presents the DA: WPO algorithm, and App. F.5 derives its objective $\mathcal{L}_{\mathrm{DA:WPO}}$.

4 Related Work

Diffusion and flow policies provide expressive alternatives to Gaussian actors and have been trained with supervised, policy-gradient, and variational objectives (Ren et al., 2025; Celik et al., 2025; Dong et al., 2025; Ma et al., 2025; McAllister et al., 2025). DA-MDPs build on the local policy-gradient view of scalable discrete diffusion samplers (Sanokowski et al., 2025a) and extend it to continuous-control RL. DIME (Celik et al., 2025) is conceptually closest to our approach. It starts from the per-state ME-RL objective, expressed as the action-marginal divergence $\mathcal{D}_{\mathrm{marg}}(s) := D_{\mathrm{KL}}\big(q_\theta(a^0\mid s)\,\big\|\,\exp(\alpha Q(s,a^0))/Z(s)\big)$. Applying the DPI yields the tractable full-path upper bound $\mathcal{D}_{\mathrm{path}}(s) := D_{\mathrm{KL}}\big(q_\theta(a^{0:K}\mid s)\,\big\|\,\pi_\theta(a^{1:K}\mid a^0,s)\exp(\alpha Q(s,a^0))/Z(s)\big) \geq \mathcal{D}_{\mathrm{marg}}(s)$, where $\pi_\theta(a^{1:K}\mid a^0,s)$ is the conditional forward path. TruDi (Le et al., 2026) instantiates this full-path objective within REPPO and is therefore the closest algorithmic counterpart to DA: REPPO. DIME and TruDi backpropagate through the complete reverse and forward chains, which requires $\mathcal{O}(K)$ active memory. DA-MDPs instead optimize one reverse transition conditioned on $\tilde{s}_{\tilde{t}} = (s_t, a_t^k, k)$. Each sampled transition has $\mathcal{O}(1)$ cost, and a minibatch of B local transitions uses $\mathcal{O}(B)$ memory. A dataset of N environment transitions contains NK local transitions; processing it once requires approximately NK/B sequential minibatch updates. See App. E.1 for a more detailed discussion about related work.

5 Experiments

We evaluate the proposed diffusion-augmented algorithms on standard continuous-control benchmarks and tasks designed to assess multimodality and behavioral diversity. Our experiments address five questions: whether the DA-MDP variants improve their corresponding base algorithms, whether they are able to represent multimodal action distributions, whether this diversity extends to contact-rich manipulation, how performance depends on the number of diffusion steps, and how the local one-step objective trades memory for runtime compared with a full-chain method.

Multimodality and Behavioral Diversity. To assess whether the DA-MDP algorithms can represent multimodal action distributions, we introduce a simple control task with a bimodal reward landscape (see App. E.3.7 for task details). The reward has the same two symmetric optima in each state and therefore induces an inherently multimodal optimal policy. The transition dynamics discretize the state space into a small set of orientations, which allows us to visualize the complete learned action distribution at every state.

We evaluate REPPO, TruDi, DPPO, DA: REPPO, DA: WPO, and DA: PPO on this benchmark. Figure 11 in App. E.3.7 shows the learned action distributions and resulting agent behavior. The

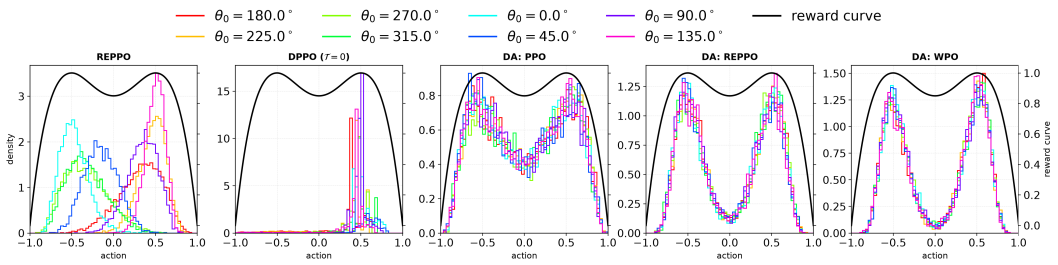
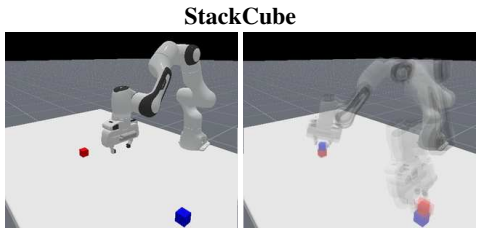


Figure 1: Learned action distributions on the Multimodal Agent benchmark. REPPO and DPPO remain unimodal; DA: PPO, DA: REPPO, and DA: WPO assign probability mass to both reward modes. Figure 11 in the appendix shows the full comparison, including TruDi, and the resulting trajectories.

REPPO policy collapses to one mode in every state and consequently drives the agent toward the lower-left corner. DPPO also remains unimodal and produces cyclic trajectories. In contrast, TruDi and all three DA-MDP methods assign probability mass to both reward modes, resulting in substantially more diverse behavior (Fig. 11b). Figure 1 compares the three DA-MDP methods with REPPO and DPPO. We also quantify the tightness of the DPI bound on this benchmark. Writing D_{aug} and D_{act} for the augmented-trajectory and executed-action-sequence KLs, respectively, the DPI gap is $D_{\text{aug}} - D_{\text{act}} = \Delta_{\text{state}} + \Delta_{\text{diffusion}}$ (Eq. 70). Lower KL values indicate closer distributional matching, while smaller gaps indicate tighter bounds. DA: REPPO achieves a lower action-marginal KL (3.742 versus 927.011 nats) and a smaller state gap (0.069 versus 2.079 nats) than REPPO, despite an additional diffusion gap of 284.603 nats. The full DA: REPPO decomposition is $288.415 \approx 3.742 + 0.069 + 284.603$ nats (details in App. E.3.9).

Contact-Rich Multimodality on StackCube. We train on the symmetric StackCube task of Le et al. (2026), where either stacking order is valid (App. E.3.3). We evaluate final checkpoints after extended training, with training success approaching a plateau for most runs; App. E.3.5 reports per-seed learning curves and budgets. For each of three training seeds, we generate repeated rollouts from a shared set of configurations drawn IID from the training distribution. We measure strict success p and success-weighted stacking-order entropy pH, where H quantifies the balance between the two successful orders. Statistics are computed within each configuration before averaging over configurations and seeds. DA-MDP methods achieve higher success rates and generally greater success-weighted stacking-order entropy than stochastic Gaussian REPPO (results table in Fig. 2b). Under stochastic execution, the unchunked DA: REPPO, DA: PPO, and DA: WPO policies achieve $pH = 0.149$–0.170 bits, more than twice Gaussian REPPO's 0.072 bits. Their advantage therefore extends beyond task completion to realizing alternative successful strategies from the same initial configuration. Figure 8 shows successful multimodality across varied cube placements. It also highlights widely separated configurations where DA: REPPO realizes both stacking orders while Gaussian REPPO rarely succeeds. The DA-MDP framework also supports **Action Chunking**, allowing several actions to be generated jointly and executed per policy call and is a widely used approach in diffusion-based robot control (Chi et al., 2025; Ren et al., 2025). With four-action chunks, DA: REPPO achieves the highest success-weighted entropy under every evaluated sampling scheme (table in Fig. 2b). Under ODE sampling, it reaches 88.9% success and $pH = 0.188$ bits, nearly twice the unchunked variant's 0.096 bits. This successful diversity accompanies a substantial computational benefit: generating four actions in one diffusion pass reduces amortized ODE inference from 67.07 to 17.01 ms per executed action on an A100, a $3.94\times$ speedup (App. E.3.3). Thus, the chunked policy combines high task success and diverse stacking strategies with nearly fourfold faster action generation.

Orientation-Conditioned Multimodality on PushT. On PushT (Le et al., 2026), we examine whether diffusion policies can maintain successful control while transitioning between distinct solution strategies. After approximately 250M interactions, DA: REPPO (ODE with sampled prior) achieves $98.3 \pm 0.4\%$ success across three training seeds, compared with $84.5 \pm 1.6\%$ for stochastic Gaussian REPPO on IID initial configurations (mean $\pm$ SD; evaluation protocols and training budgets in Apps. E.3.4 and E.3.5). Figure 2c reveals the difference near the transition between clockwise and counter-clockwise solutions. Holding the goal and robot poses fixed, we vary the block's initial orientation. DA: REPPO changes its preferred direction between 190° and 192°, sampling



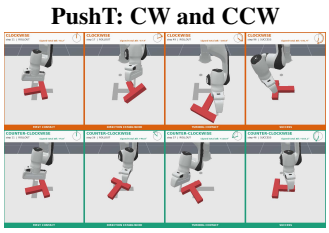
(a)

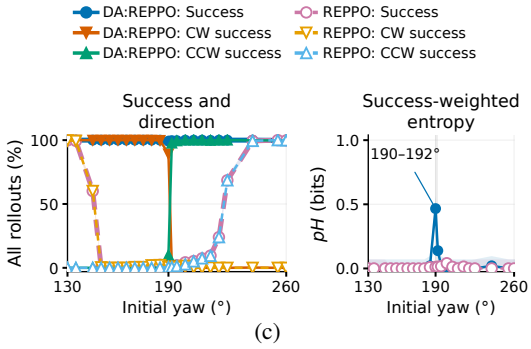
Method	Success (%) ↑ Means + prior	ODE + prior	Stoch.	Success-weighted pH (bits) ↑ Means + prior	ODE + prior	Stoch.
DA:REPPO C=1	**96.5**	**96.4**	**95.0**	.107	.096	.149
	±0.2	**±0.5**	**±0.8**	±.008	±.008	±.011
DA:REPPO C=4	92.6	88.9	88.2	**.126**	**.188**	**.211**
	±2.3	±4.4	±3.0	**±.007**	**±.015**	**±.009**
DA:PPO	93.9	92.6	89.2	.093	.069	.166
	±2.0	±3.4	±2.3	±.013	±.007	±.015
DA:WPO	96.0	96.1	94.0	.104	.104	.170
	±1.6	±1.5	±1.5	±.011	±.016	±.015
Gaussian REPPO	—	—	82.7	—	—	.072
			±2.1			±.003

Mean ± SD across three training seeds; budgets differ.

↑: higher is better. Bold: best mean per column.

(b)





(c)

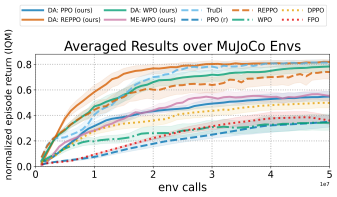
Figure 2: Multimodal manipulation. (a) Illustrative motions. (b) IID StackCube success and success-weighted mode entropy pH: mean ± SD across three training seeds. (c) PushT success, successful CW/CCW fractions of all attempts, and pH: one seed, 95% sampling intervals.

successful trajectories in both directions from the same initial configuration while maintaining over 99% success at both angles. Gaussian REPPO solves the outer orientations but struggles to connect the two strategies, achieving only 1.2%–1.4% success in this switching region. At 190°, success-weighted mode entropy is 0.467 bits for DA: REPPO versus 0.012 bits for Gaussian REPPO. These results demonstrate that the diffusion policy supports alternative successful strategies through the switching transition, consistent with the benefit of representing multimodal action distributions.

Diffusion-Step Ablation. We ablate the number of diffusion steps, $K \in \{2, 4, 8\}$, for DA: REPPO, DA: PPO, and DA: WPO on G1JoystickFlatTerrain. Figure 3b reports deterministic returns at method-specific checkpoints. Increasing K from 2 to 8 raises the mean return from 21.11 to 28.32 for DA: REPPO, from −1.89 to 31.74 for DA: PPO, and from −1.60 to 18.04 for DA: WPO. The evaluation learning curves are presented in Fig. 10. Larger diffusion depths thus improve the reported return for all three methods. See App. E.3.6. for details.

Memory–Runtime Trade-off with TruDi. To test how the local one-step objective trades active memory against training time, we benchmark DA: REPPO against the full-chain method TruDi on CheetahRun using 16 diffusion steps and vary the minibatch size of each method (Fig. 3c). Point labels show (samples per minibatch, optimizer updates), with median runtime measured per 131,072 environment transitions. Every TruDi sample requires differentiation through the complete reverse and forward diffusion chains; under a tight memory budget, TruDi must therefore process fewer samples per minibatch and perform more optimizer updates. DA: REPPO instead processes local one-step transitions, allowing substantially larger minibatches at the same memory budget. Across all measured active-memory settings up to approximately 1.9 GiB, DA: REPPO is faster, requiring 6.3–7.0 s per training block compared with 6.7–22.3 s for TruDi. TruDi becomes faster only when additional memory allows more full-chain samples to be processed in each minibatch. Thus, DA: REPPO is particularly advantageous when larger networks or more diffusion steps make memory the limiting resource. Profiling details are given in App. E.4.

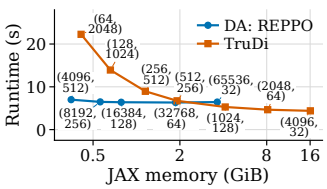
Continuous-control Experiments. On MuJoCo Playground (Zakka et al., 2025), we compare DA: PPO, DA: REPPO, and DA: WPO with their respective non-diffusion counterparts and with DPPO (Ren et al., 2025), TruDi (Le et al., 2026), and FPO (McAllister et al., 2025). Figure 3a reports the aggregate IQM (Agarwal et al., 2021); the task-level learning curves are in Fig. 5 in



(a)

G1 deterministic return ↑			
K	DA: REPPO 50.07M	DA: PPO 200.02M	DA: WPO 69.07M
2	21.11 ± 0.81	-1.89 ± 0.05	-1.60 ± 0.02
4	24.98 ± 0.68	20.30 ± 2.07	15.89 ± 0.55
8	$\mathbf{28.32 \pm 0.42}$	$\mathbf{31.74 \pm 0.78}$	$\mathbf{18.04 \pm 0.40}$

(b)



(c)

Figure 3: (a) Equal-task-weighted IQM of normalized returns across twelve tasks with 95% stratified-bootstrap confidence intervals. (b) G1 returns at the reported checkpoints (mean ± SD over seeds). Headers give method-specific interaction counts; bold marks the best depth per method. (c) Memory–runtime trade-off between DA: REPPO and TruDi on CheetahRun at $K = 16$.

App. E.3.1. Unless stated otherwise, diffusion policies use $K = 8$ steps. App. E.3.1 describes the benchmark setup and per-task results. We first evaluate the maximum-entropy extension of WPO. WPO and ME-WPO share the same REPPO implementation and differ only in their policy loss and entropy treatment: ME-WPO automatically tunes the temperature, whereas WPO sets $\mathcal{T} = 0$. The aggregate curves in Fig. 3a and task-level curves in Fig. 5 (App. E.3.1) show that ME-WPO is more stable and performs better than WPO, although REPPO remains stronger overall. At matched environment interactions, each DA variant also improves on its non-diffusion counterparts, with the largest gains on HopperHop, HopperStand, FingerSpin, and the sparse swing-up tasks. In particular, DA: PPO learns faster than PPO and reaches a higher final return. Comparing DA: PPO with DPPO isolates the maximum-entropy correction because DPPO uses the same augmented MDP with $\mathcal{T} = 0$. A fixed positive temperature improves CartpoleSwingupSparse and FishSwim and slightly increases aggregate IQM, but cannot adapt across tasks like the automatically tuned temperatures of DA: REPPO, DA: WPO, and TruDi, which span several orders of magnitude. Finally, DA: REPPO learns faster initially than TruDi and reaches a comparable final IQM, while FPO is weaker than the other diffusion baselines. See App. E.3.1 for protocol and baseline details.

Diffusion-Sampler Ablation. We compare VP-SDE and bounded Beta diffusion for DA: REPPO (Fig. 7 in App. E.3.2). Our diffusion policies use VP-SDE in all other experiments. For VP-SDE, we additionally compare latent-entropy and transformed-action entropy formulations.

6 Conclusion, Limitations and Future Work

DA-MDPs provide a general framework for adapting maximum-entropy RL algorithms to diffusion policies. Derived from the variational formulation of ME-RL, the construction expresses denoising as a sequence of local decisions with diffusion-augmented soft rewards, value functions, and policy objectives. This allows each update to differentiate through only one diffusion transition. The accompanying return-improvement bound establishes conditions under which improvements in the local surrogate translate into higher augmented return, including when the forward kernels are learned.

In our continuous-control experiments, the resulting variants of PPO, REPPO, and maximum-entropy WPO improve on their non-diffusion counterparts. On StackCube, DA: REPPO achieves higher success and success-weighted mode entropy than Gaussian REPPO, and realizes both stacking orders at some widely separated placements where the baseline fails. DA policies also retain successful diversity under action chunking. On PushT, DA: REPPO realizes both motion directions near the transition, where Gaussian REPPO has near-zero success and little successful diversity. Our ablations show that additional diffusion steps can improve performance, while the runtime study demonstrates the practical benefit of local training when memory is constrained.

Adapting a Gaussian ME-RL algorithm to a DA-MDP can also require method-specific design choices. In DA: PPO, learning the forward kernels requires an explicit gradient correction to the PPO surrogate. In DA: REPPO, auxiliary critic losses must account for both environment transitions and intermediate denoising transitions. Entropy tuning remains task dependent: greater entropy can encourage exploration and diversity but may favor lower-reward actions. Learned state-dependent priors and extensions to discrete or structured action spaces offer directions for future research. App. E.4.3 discusses additional limitations and future work.

AI USE STATEMENT

We used large language models (LLMs) to assist with drafting and revising the manuscript, developing research code, and creating scientific figures. The authors proved the original finite-horizon DA-MDP formulation by hand. LLMs assisted in proofing the math for the extension of the DA-MDP derivation to infinite-horizon reinforcement learning and the adaptation of the TRPO monotonic improvement bound to diffusion-augmented MDPs. LLMs assisted in implementing the PyTorch code, which was used for the StackCube and PushT experiments and the diffusion-sampler ablation. The JAX implementation was written by the authors without LLM assistance and was used for all remaining experiments. LLMs subsequently helped identify and fix some bugs in the JAX codebase. The authors checked AI-generated and AI-assisted text, code, and mathematical arguments for correctness. The authors take responsibility for the final manuscript, code, figures, mathematical claims, and experimental results.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998. doi: 10.1162/089976698300017746.

Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In *International Conference on Machine Learning*, pp. 1276–1301. PMLR, 2023.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. PMLR, 2017.

Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

Damiano Brigo, Bernard Hanzon, and François Le Gland. A differential geometric approach to nonlinear filtering: The projection filter. *IEEE Transactions on Automatic Control*, 43(2):247–252, 1998. doi: 10.1109/9.661075.

Damiano Brigo, Bernard Hanzon, and François Le Gland. Approximate nonlinear filtering by projection on exponential manifolds of densities. *Bernoulli*, 5(3):495–534, 1999. doi: 10.2307/3318714.

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. DIME: Diffusion-based maximum entropy reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 6958–6977. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/celik25a.html.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xQBRrtQM8u.

Xiaoyi Dong, Jian Cheng, and Xi Sheryl Zhang. Maximum entropy reinforcement learning with diffusion policy. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 13963–13983. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/dong25b.html.

Mingzhou Fan, Ruida Zhou, Chao Tian, and Xiaoning Qian. Path-guided particle-based sampling. In *International Conference on Machine Learning*, pp. 12916–12934. PMLR, 2024.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024.

M. Gil, F. Alajaji, and T. Linder. Rényi divergence measures for commonly used univariate continuous distributions. *Information Sciences*, 249:124–131, November 2013. doi: 10.1016/j.ins.2013.06.018. URL https://doi.org/10.1016/j.ins.2013.06.018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, and Juan Carrasquilla. Variational neural annealing. *Nat. Mach. Intell.*, 3(11):952–961, 2021. doi: 10.1038/s42256-021-00401-3. URL https://doi.org/10.1038/s42256-021-00401-3.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/49f85a9ed090b20c8bed85a5923c669f-Abstract.html.

Huy Le, Onur Celik, Denis Blessing, Tai Hoang, Claas A Voelcker, Axel Brunnbauer, Felix Richter, Michael Volpp, and Gerhard Neumann. Trust-region diffusion policies for massively parallel on-policy rl. *arXiv preprint arXiv:2606.15260*, 2026.

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

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

Haitong Ma, Ofir Nabati, Aviv Rosenberg, Bo Dai, Oran Lang, Idan Szpektor, Craig Boutilier, Na Li, Shie Mannor, Lior Shani, et al. Reinforcement learning with discrete diffusion policies for combinatorial action spaces. *arXiv preprint arXiv:2509.22963*, 2025.

Bálint Máté and François Fleuret. Learning interpolations between boltzmann densities. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=TH6YrEcbth.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 2025.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Trans. Graph.*, 38(5):145:1–145:19, 2019. doi: 10.1145/3341156. URL https://doi.org/10.1145/3341156.

Kirill Neklyudov, Jannes Nys, Luca Thiede, Juan Carrasquilla, Qiang Liu, Max Welling, and Alireza Makhzani. Wasserstein quantum monte carlo: a novel approach for solving the quantum many-body schrödinger equation. *Advances in Neural Information Processing Systems*, 36:63461–63482, 2023.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Frank Noé and Hao Wu. Boltzmann generators - sampling equilibrium states of many-body systems with deep learning. *CoRR*, abs/1812.01729, 2018. URL http://arxiv.org/abs/1812.01729.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KVf2SFL1pi.

David Pfau, Ian Davies, Diana L Borsa, João Guilherme Madeira Araújo, Brendan Daniel Tracey, and Hado van Hasselt. Wasserstein policy optimization. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=oAKe7MG9GM.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Allen Z. Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mEpqHvbD2h.

Enric Ribera Borrell, Lorenz Richter, and Christof Schuette. Reinforcement learning with random time horizons. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 5101–5123. PMLR, 2025.

Lorenz Richter and Julius Berner. Improved sampling via learned diffusions. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fmPpbbPjQb.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 43346–43367. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/sanokowski24a.html.

Sebastian Sanokowski, Wilhelm Berghammer, Martin Ennemoser, Haoyu Peter Wang, Sepp Hochreiter, and Sebastian Lehner. Scalable discrete diffusion samplers: Combinatorial optimization and statistical physics. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=peNgxpbdxB.

Sebastian Sanokowski, Lukas Gruber, Christoph Bartmann, Sepp Hochreiter, and Sebastian Lehner. Rethinking losses for diffusion bridge samplers. *arXiv preprint arXiv:2506.10982*, 2025b.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2256–2265. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/sohl-dickstein15.html.

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, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-Kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh N, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: GPU parallelized robot simulation and rendering for generalizable embodied AI. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025. URL https://openreview.net/forum?id=GgTxudXaU8.

Shantanu Thakoor, Mark Rowland, Diana Borsa, Will Dabney, Remi Munos, and Andre Barreto. Generalised policy improvement with geometric policy composition. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 21272–21307. PMLR, 2022.

Yifeng Tian, Nishant Panda, and Yen Ting Lin. Liouville flow importance sampler. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 48186–48210, 2024.

Emanuel Todorov. General duality between optimal control and estimation. In *2008 47th IEEE conference on decision and control*, pp. 4286–4292. IEEE, 2008.

Francisco Vargas, Will Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. *arXiv preprint arXiv:2302.13834*, 2023.

Francisco Vargas, Shreyas Padhy, Denis Blessing, and N Nüsken. Transport meets variational inference: Controlled monte carlo diffusions. In *The Twelfth International Conference on Learning Representations*, 2024.

Claas A. Voelcker, Axel Brunnbauer, Marcel Hussing, Michal Nauman, Pieter Abbeel, Eric Eaton, Radu Grosu, Amir-massoud Farahmand, and Igor Gilitschenski. Relative entropy pathwise policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4vmm8mlHkS.

Dian Wu, Lei Wang, and Pan Zhang. Solving statistical mechanics using variational autoregressive networks. *Phys. Rev. Lett.*, 122:080602, Feb 2019. doi: 10.1103/PhysRevLett.122.080602. URL https://link.aps.org/doi/10.1103/PhysRevLett.122.080602.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo playground, 2025. URL https://arxiv.org/abs/2502.08844.

Qinsheng Zhang and Yongxin Chen. Path integral sampler: A stochastic control approach for sampling. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=_uCb2ynRu7Y.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *arXiv preprint arXiv:2505.22094*, 2025.

Brian D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, December 2010. URL https://www.cs.cmu.edu/~bziebart/publications/thesis-bziebart.pdf.

APPENDIX CONTENTS

A Visual Overview of DA-MDPs **16**

B Notation **17**

C Details of the Relevant Reinforcement-Learning Algorithms **17**

C.1 Proximal Policy Optimization (PPO) 17

C.2 Relative Entropy Pathwise Policy Optimization (REPPO) 18

C.3 Maximum-Entropy Wasserstein Policy Optimization (ME-WPO) 19

D DA-MDP Algorithms **20**

D.1 From the DA-MDP to the Implementation 20

D.2 Critic Targets 21

D.3 DA: REPPO 22

D.4 DA: WPO 23

D.5 Trust-Region and Temperature Dual Updates 24

D.6 DA: PPO 25

E Extended Paper Sections **26**

E.1 Related Work 26

E.2 Diffusion Models in Reinforcement Learning 26

E.3 Additional Experimental Details 26

E.4 Runtime Comparison 43

F Derivations **45**

F.1 Standard ME-RL Trajectory KL 45

F.2 Data-Processing Inequality and DA-MDP Trajectory KL 49

F.3 Monotonic Improvement for the Local DA-MDP Loss 54

F.4 Maximum-Entropy Wasserstein Policy Optimization 62

F.5 Derivation of Diffusion-Augmented Wasserstein Policy Optimization 65

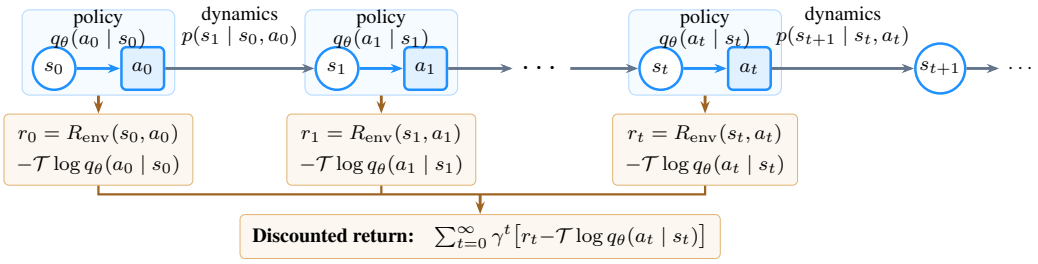
G Common Hyperparameters **69**

G.1 MuJoCo 69

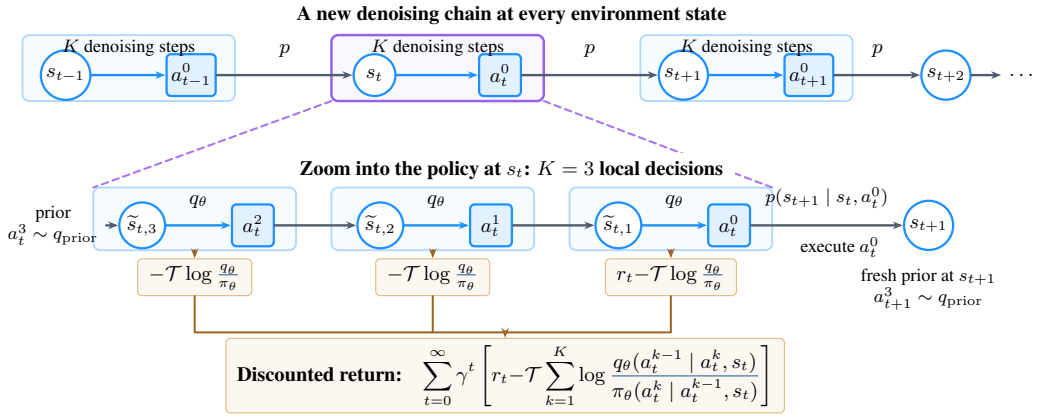
G.2 PushT Hyperparameters 70

A VISUAL OVERVIEW OF DA-MDPS

Figure 4 shows how a diffusion policy turns one environment-level decision into K local denoising decisions. The same state–action diagram describes both processes: circles denote states, rounded squares denote actions, and blue arrows denote policy samples.



(a) An ordinary maximum-entropy MDP: one policy sample per environment step.



(b) A DA-MDP across environment steps, with a zoom into $K = 3$ local decisions at s_t.

Figure 4: From environment trajectories to local denoising decisions. (a) Each action produces an environment reward and an entropy contribution before the environment advances. (b) A fresh denoising chain generates an action at each environment state. Dashed purple guides expand the highlighted policy call at s_t into local DA-MDP decisions. Gray arrows between denoising decisions carry the sampled action into the next augmented state; only the final action a_t^0 changes the environment state. Gold boxes show the local contributions and their infinite-horizon discounted returns.

States and actions. For readability, write $\tilde{s}_{t,k} := \tilde{s}_{\tilde{t}(t,k)} = (s_t, a_t^k, k)$, with $\tilde{t}(t,k) = tK + (K - k)$. The reverse kernel $q_\theta(a_t^{k-1} \mid a_t^k, s_t)$ samples the local action. For $k > 1$, the successor is deterministically $(s_t, a_t^{k-1}, k-1)$. At $k = 1$, the environment executes a_t^0, samples $s_{t+1} \sim p(\cdot \mid s_t, a_t^0)$, and starts a new denoising chain with $a_{t+1}^K \sim q_{\text{prior}}$, as formalized in Sec. 3.2.

Rewards and discounting. Here, $r_t = R_{\text{env}}(s_t, a_t)$, with $a_t = a_t^0$ in panel (b). The ordinary policy's entropy contribution is replaced by a sum of reverse–forward log-ratio contributions. In the local boxes, q_θ and π_θ denote the reverse and forward conditional densities evaluated at the displayed denoising transition; their full arguments appear in the return below. The environment reward appears only at $k = 1$. Denoising leaves the environment clock unchanged: the discount is one for $k > 1$ and γ at $k = 1$. Thus, all K contributions at environment step t share the factor γ^t. Taking expectations of the displayed returns recovers the objectives in Eqs. 2 and 8.

B NOTATION

Table 1: Notation used throughout the paper.

Symbol	Meaning
$t \in \{0, 1, \ldots\}$	Environment-time index in the stationary discounted setting; T is used only for the finite trajectory prefixes that introduce the DPI.
$k \in \{K, \ldots, 1\}$	Reverse-diffusion index; a_t^K is prior noise and a_t^0 is executed in the environment.
$\tilde{t},\ \tilde{s}_{\tilde{t}},\ \tilde{a}_{\tilde{t}}$	Augmented index, state, and action: $\tilde{t} = tK + (K-k)$, $\tilde{s}_{\tilde{t}} = (s_t, a_t^k, k)$, and $\tilde{a}_{\tilde{t}} = a_t^{k-1}$.
$q_\theta,\ \pi_\theta$	Learned reverse policy/kernel and forward diffusion kernel, respectively; $\widetilde{\pi}$ denotes an unnormalized reward target and π its normalized version.
θ	Current actor parameters with respect to which gradients are taken.
θ_{roll}	Frozen behavior-policy snapshot that generated the stored rollout states and actions.
$\theta^* = \text{sg}(\theta)$	Stop-gradient copy of the current parameters: numerically equal to θ but treated as constant. In general, $\theta^* \neq \theta_{\text{roll}}$ after the first minibatch update.
$Q^{q_{\theta_{\text{roll}}}},\ V^{q_{\theta_{\text{roll}}}}$	Discounted critic functions evaluated for the frozen rollout policy; γ is fixed and therefore not repeated as an index.
$\mathcal{T},\ \alpha = 1/\mathcal{T}$	Entropy temperature and inverse temperature.
$\gamma,\ H$	Infinite-horizon discount factor and the corresponding geometric terminal index, with $\Pr(H = h) = (1-\gamma)\gamma^h$.
$d^{q_\theta}(s)$	Normalized discounted state-visitation density for $s_0 \sim p(s_0)$.
$\bar{\gamma}(k),\ d_{\text{DA}}^{q_\theta}$	Exact phase-dependent DA-MDP discount and normalized discounted augmented-state visitation; $\bar{\gamma}(k) = 1$ for $k > 1$ and $\bar{\gamma}(1) = \gamma$.
$\mathcal{J},\ \mathcal{J}_{\text{DA}}$	Discounted maximum-entropy returns for the original and diffusion-augmented MDPs; both are maximized.
$q_\theta^{(k)}(a^k \mid s)$	Noisy-action marginal at diffusion phase k, generated from the shared prior using the reverse kernels.
$\mathcal{L}_{\text{DA}}(\theta; \theta_{\text{roll}})$	Local denoising loss with frozen rollout critic and visitation; its decrease predicts the return gain bounded in App. F.3.
$D_{\text{KL}}(p(X) \Vert r(X))$	Ordinary KL divergence; its scope is exactly the random-variable tuple X shown in both arguments.
$D_{\text{KL}}^{\text{u}}(q \Vert \widetilde{p})$	Unnormalized-target KL (variational free energy), $\int q \log(q/\widetilde{p})$; used only for the learned-forward local surrogate.
$X = a^{(0)}$ vs. $X = a^{(0:K)}$	Action marginal versus full diffusion path.
$X = a_{0:T}$ vs. $X = (a_{0:T}, s_{0:T+1})$	Action-sequence marginal versus joint environment state–action trajectory.
$X = (a_{0:T}^{0:K}, s_{0:T+1})$	Joint environment–diffusion trajectory; conditioning such as $a_t \mid s_t$ denotes a per-state KL.

C DETAILS OF THE RELEVANT REINFORCEMENT-LEARNING ALGORITHMS

C.1 PROXIMAL POLICY OPTIMIZATION (PPO)

Proximal Policy Optimization (PPO) can be derived by applying the log-derivative trick to the Maximum Entropy surrogate objective (see Eq.3) combined with importance sampling. Importance sampling is required because PPO does not learn a state–action value function Q_θ directly; thus, returns or rewards $R_{\text{env}}(s_t, a_t)$ can only be evaluated at state–action pairs observed under the behavior policy. The resulting surrogate objective can be written as

$$\mathcal{L}_{\text{ME}}(\theta, s_t) = \mathbb{E}_{a_t \sim q_{\theta_{\text{roll}}}(a_t|s_t)}\Big[\frac{q_\theta(a_t|s_t)}{q_{\theta_{\text{roll}}}(a_t|s_t)}\Big(\mathcal{T}\log q_\theta(a_t|s_t) - Q^{q_{\theta_{\text{roll}}}}(s_t, a_t)\Big)\Big] + C$$

where $q_{\theta_{\text{roll}}}$ denotes the behavior policy used to collect trajectories. As the gradient is computed with respect to θ this equation effectively implements the log-derivative trick as

$$\begin{aligned}\nabla_\theta \frac{q_\theta(a_t|s_t)}{q_{\theta_{\text{roll}}}(a_t|s_t)} &= \nabla_\theta \exp\big(\log q_\theta(a_t|s_t) - \log q_{\theta_{\text{roll}}}(a_t|s_t)\big) \\ &= \exp\big(\log q_\theta(a_t|s_t) - \log q_{\theta_{\text{roll}}}(a_t|s_t)\big)\nabla_\theta \log q_\theta(a_t|s_t) \\ &= \frac{q_\theta(a_t|s_t)}{q_{\theta_{\text{roll}}}(a_t|s_t)}\nabla_\theta \log q_\theta(a_t|s_t)\end{aligned} \tag{13}$$

The PPO objective, however, additionally uses a clipping strategy, where the objective is then given by:

$$\mathcal{L}_{\text{PPO}}(\theta, s_t) = -\mathbb{E}_{a_t\sim q_{\theta_{\text{roll}}}(a_t|s_t)}\Big[\min\Big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\Big)\Big],$$

where $r_t(\theta) = \frac{q_\theta(a_t|s_t)}{q_{\theta_{\text{roll}}}(a_t|s_t)}$ are importance weights, $\hat{A}_t$ is an estimate of the advantage function, and ϵ is the clipping parameter. The advantage function A_t quantifies how much better taking action a_t in state s_t is compared to the expected value of the policy from that state and is defined as $A_t = Q^{\theta_{\text{roll}}}(s_t, a_t) - V^{\theta_{\text{roll}}}(s_t) - \mathcal{T}\log q_\theta(a_t|s_t)$ where the subtraction of $V^{\theta_{\text{roll}}}(s_t)$ reduces the variance of the gradient estimates while leaving the average gradient unchanged. The estimate $\hat{A}_t$ is treated as constant when differentiating the clipped surrogate.

PPO evaluates the surrogate on frozen rollout states using importance-weighted actions. At the rollout parameters, its population gradient matches the local actor objective. Clipping discourages large policy updates and provides a first-order alternative to KL-based trust-region control.

C.2 Relative Entropy Pathwise Policy Optimization (REPPO)

Relative Entropy Pathwise Policy Optimization (REPPO) combines the local actor objective in Eq. 3 with an entropy target and a KL trust region. Holding the rollout critic fixed, its constrained policy-improvement problem is

$$\begin{aligned}\min_\theta \quad & -\mathbb{E}_{s\sim d^{q_{\theta_{\text{roll}}}},\, a\sim q_\theta}[Q^{q_{\theta_{\text{roll}}}}(s,a)] \\ \text{subject to} \quad & \mathbb{E}_{s\sim d^{q_{\theta_{\text{roll}}}}}[D_{\text{KL}}(q_{\theta_{\text{roll}}}(\cdot\mid s)\|q_\theta(\cdot\mid s))] \le \delta_{\text{KL}}, \\ & \mathbb{E}_{s\sim d^{q_{\theta_{\text{roll}}}}}[\mathcal{H}[q_\theta(\cdot\mid s)]] \ge \mathcal{H}_{\text{tar}}.\end{aligned} \tag{14}$$

Lagrangian formulation. The temperature $\mathcal{T} \ge 0$ is the entropy-constraint multiplier, while $\lambda_{\text{KL}} \ge 0$ controls the KL constraint. Their Lagrangian is

$$\begin{aligned}\mathcal{L}(\theta, \lambda_{\text{KL}}, \mathcal{T}) = \mathbb{E}_{s\sim d^{q_{\theta_{\text{roll}}}},\, a\sim q_\theta}[\mathcal{T}\log q_\theta(a\mid s) - Q^{q_{\theta_{\text{roll}}}}(s,a)] + \mathcal{T}\mathcal{H}_{\text{tar}} \\ + \lambda_{\text{KL}}\left(\mathbb{E}_{s\sim d^{q_{\theta_{\text{roll}}}}}[D_{\text{KL}}(q_{\theta_{\text{roll}}}(\cdot\mid s)\|q_\theta(\cdot\mid s))] - \delta_{\text{KL}}\right).\end{aligned} \tag{15}$$

For fixed $\mathcal{T}$, the first expectation equals the temperature-scaled reverse-KL actor objective up to a θ-independent normalizer. Entropy therefore enters once, through $\mathcal{T}\log q_\theta$.

Policy update. Holding the critic, rollout states, and multipliers fixed, the policy update is

$$\theta_{i+1} = \theta_i - \eta_\theta \nabla_\theta \mathcal{L}(\theta_i, \lambda^i_{\text{KL}}, \mathcal{T}^i). \tag{16}$$

We parameterize $\lambda_{\text{KL}} = \exp(\alpha_{\text{KL}})$ and $\mathcal{T} = \exp(\alpha_{\mathcal{T}})$ to keep both multipliers positive.

Dual updates. With policy statistics held fixed, gradient ascent in the log-multipliers gives

$$\alpha^{i+1}_{\text{KL}} = \alpha^i_{\text{KL}} + \eta_{\text{KL}}\lambda^i_{\text{KL}}\left(\mathbb{E}_{s\sim d^{q_{\theta_{\text{roll}}}}}[D_{\text{KL}}(q_{\theta_{\text{roll}}}(\cdot\mid s)\|q_{\theta_i}(\cdot\mid s))] - \delta_{\text{KL}}\right), \tag{17}$$

$$\alpha^{i+1}_{\mathcal{T}} = \alpha^i_{\mathcal{T}} + \eta_{\mathcal{T}}\mathcal{T}^i\left(\mathcal{H}_{\text{tar}} - \mathbb{E}_{s\sim d^{q_{\theta_{\text{roll}}}}}[\mathcal{H}[q_{\theta_i}(\cdot\mid s)]]\right). \tag{18}$$

Thus the KL penalty increases when its constraint is exceeded, and the temperature increases when entropy falls below its target.

Q-target estimation via TD(λ). REPPO directly learns an action-value function $Q_\phi(s,a)$ and does not employ a separate state-value function. Consequently, action-value targets are computed using multi-step temporal-difference updates with eligibility traces, i.e. TD(λ).

Reparameterized policy gradient. REPPO uses a learned Q-function together with the *reparameterization trick* to differentiate through freshly sampled actions. Actions are generated as

$$a_t = \mu_\theta(s_t) + \xi_t \, \sigma_\theta(s_t) := f_\theta(\xi_t; s_t), \quad \xi_t \sim \mathcal{N}(0, I), \tag{19}$$

allowing gradients to flow directly through sampled actions. The resulting policy-gradient estimator takes the form

$$\nabla_\theta \mathcal{L}_{\mathrm{REPPO}}(\theta, s_t) = \mathbb{E}_{\xi_t}\Big[\mathcal{T}\, \nabla_\theta \log q_\theta(f_\theta(\xi_t; s_t) \mid s_t) - \nabla_\theta Q^{q_{\theta_{\mathrm{roll}}}}(s_t, f_\theta(\xi_t; s_t))\Big]. \tag{20}$$

KL-aware clipped surrogate objective. To enforce the KL trust region in practice, REPPO employs a *sample-wise clipped loss* that selects between the local reverse-KL actor objective and a pure trust-region KL penalty. For each minibatch state $s_t^{(i)}$, define the behavior-to-current Monte Carlo estimate explicitly as

$$\widehat{D_{\mathrm{KL}}}^{(i)} := \widehat{D_{\mathrm{KL}}}\Big(q_{\theta_{\mathrm{roll}}}(a_t \mid s_t^{(i)}) \,\Big\|\, q_\theta(a_t \mid s_t^{(i)})\Big).$$

The final REPPO surrogate loss can then be written equivalently as

$$\mathcal{L}^{\mathrm{clip}}_{\mathrm{REPPO}}(\theta) = \mathbb{E}_{s_t^{(i)} \sim q_{\theta_{\mathrm{roll}}}}\left[\begin{cases} \mathcal{L}^{(i)}_{\mathrm{ME}}(\theta, s_t^{(i)}), & \text{if } \widehat{D_{\mathrm{KL}}}^{(i)} \le \delta_{\mathrm{KL}}, \\ \lambda_{\mathrm{KL}}\, \widehat{D_{\mathrm{KL}}}^{(i)}, & \text{if } \widehat{D_{\mathrm{KL}}}^{(i)} > \delta_{\mathrm{KL}}. \end{cases}\right], \tag{21}$$

where δ_{KL} is a predefined trust-region threshold, λ_{KL} is a KL penalty coefficient, and $\mathcal{L}^{(i)}_{\mathrm{ME}}(\theta, s_t^{(i)})$ denotes the reverse-KL surrogate loss evaluated at sample i.

The adaptive rule follows the reverse-KL update while the empirical divergence remains within the prescribed trust region and switches to a KL penalty once the boundary is exceeded. It should therefore be interpreted as a practical mechanism for keeping the sampled update within the regime motivated by the theoretical improvement bound.

Auxiliary tasks. REPPO optionally employs auxiliary objectives to stabilize representation learning in sparse-reward regimes and when using small update batches. In particular, it adopts the latent self-prediction auxiliary task introduced in the REPPO paper, which has been shown to significantly improve critic stability and sample efficiency in low-signal settings (see also Jaderberg et al., 2016). Concretely, the critic is augmented with an auxiliary prediction head that encourages the learned latent representation of state–action pairs to be predictive of future representations under the on-policy data distribution. The auxiliary loss is optimized jointly with the critic objective using on-policy rollouts. We refer to the REPPO paper for the precise formulation and architectural details.

Cross-entropy regression for critic targets. REPPO further stabilizes critic learning by replacing mean-squared-error regression with a cross-entropy-based regression loss (HL-Gauss; Farebrother et al., 2024), originally inspired by distributional reinforcement learning methods (Bellemare et al., 2017). All critic targets, including those used for policy optimization, are regressed using this loss. This formulation has been shown to yield improved numerical stability and learning dynamics, particularly in deterministic and sparse-reward environments. We refer the reader to the REPPO paper for implementation details and ablation results.

C.3 Maximum-Entropy Wasserstein Policy Optimization (ME-WPO)

ME-WPO uses the same Gaussian policy, Q-critic, TD(λ) targets, trust-region constraint, temperature update, and auxiliary losses as REPPO. It changes only the actor direction: instead of differentiating the reverse-KL objective through a reparameterized action, ME-WPO follows the Wasserstein gradient flow of the same soft policy-improvement objective. For a fresh action $a_t \sim q_\theta(\cdot \mid s_t)$ that is held fixed during differentiation, this direction is obtained from the score-matching loss

$$\widehat{\mathcal{L}}_{\text{ME-WPO}}(\theta) = \frac{\mathcal{T}}{2} \mathbb{E}_{\substack{s_t \sim q_{\theta_{\mathrm{roll}}} \\ a_t \sim q_{\theta^*}(\cdot \mid s_t)}}\left[\left\| \nabla_{a_t}\left(\log q_\theta(a_t \mid s_t) - \frac{1}{\mathcal{T}} Q^{q_{\theta_{\mathrm{roll}}}}(s_t, a_t)\right)\right\|^2\right]. \tag{22}$$

App. F.4 derives Eq. 22 from the Wasserstein gradient flow of the soft policy-improvement objective and its projection onto the parametric policy family. Let $\widehat{\mathcal{L}}^{(i)}_{\text{ME-WPO}}(\theta)$ denote the statewise contribution to Eq. 22, including the expectation over fresh actions at $s_t^{(i)}$. Using the behavior-to-current KL estimate $\widehat{D_{\text{KL}}}^{(i)}$ defined above, the complete conditional preconditioned actor gradient is

$$\widetilde{\nabla}_\theta \mathcal{L}^{\text{clip}}_{\text{ME-WPO}}(\theta) = \mathbb{E}_{s_t^{(i)} \sim q_{\theta_{\text{roll}}}} \left[\begin{cases} \mathcal{F}_{\theta\theta}^{-1} \nabla_\theta \widehat{\mathcal{L}}^{(i)}_{\text{ME-WPO}}(\theta), & \text{if } \widehat{D_{\text{KL}}}^{(i)} \le \delta_{\text{KL}}, \\ \lambda_{\text{KL}} \nabla_\theta \widehat{D_{\text{KL}}}^{(i)}, & \text{if } \widehat{D_{\text{KL}}}^{(i)} > \delta_{\text{KL}}. \end{cases} \right] \tag{23}$$

Here, $\mathcal{F}_{\theta\theta}$ is the policy Fisher matrix defined in Eq. 191, and the branch condition and λ_{KL} are held fixed during the actor update, exactly as in REPPO. Thus, states inside the trust region follow the inverse-Fisher-preconditioned score gradient, whereas states outside it contribute only the ordinary KL-penalty gradient. Within the score-matching branch, the policy score $\nabla_{a_t} \log q_\theta(a_t \mid s_t)$ is matched to the soft target score $\nabla_{a_t} Q^{q_{\theta_{\text{roll}}}}(s_t, a_t)/\mathcal{T}$ without evaluating the target normalizer. In mean–standard-deviation coordinates, the Gaussian preconditioner of Pfau et al. (2025) implements the inverse-Fisher factor through the element-wise transformations

$$\widetilde{\nabla}_{\mu_\theta} \widehat{\mathcal{L}}_{\text{ME-WPO}} = \sigma_\theta^2 \odot \nabla_{\mu_\theta} \widehat{\mathcal{L}}_{\text{ME-WPO}}, \qquad \widetilde{\nabla}_{\sigma_\theta} \widehat{\mathcal{L}}_{\text{ME-WPO}} = \frac{1}{2} \sigma_\theta^2 \odot \nabla_{\sigma_\theta} \widehat{\mathcal{L}}_{\text{ME-WPO}}. \tag{24}$$

Apart from this score-based actor gradient and inverse-Fisher scaling, ME-WPO follows the REPPO implementation described above. Standard WPO is recovered by removing the entropy contribution, whereas ME-WPO retains REPPO's automatically tuned positive temperature.

D DA-MDP Algorithms

D.1 From the DA-MDP to the Implementation

The implementation follows the construction in Sec. 3.2 directly. It replaces each environment transition by K local denoising transitions and stores them in an ordinary on-policy rollout buffer. The buffer is flattened over environment time and diffusion time, with leading dimensions $[TK, N_{\text{env}}]$; the explicit chain dimension is restored only when a loss must be summed over all K diffusion steps. Consequently, actor and critic minibatches contain local transitions and never require backpropagation through a complete diffusion chain.

Rollout collection. Let θ_{roll} denote the frozen behavior parameters used during collection. At environment state s_t, the implementation samples $a_t^K \sim q_{\text{prior}}$, applies the reverse kernels successively for $k = K, \dots, 1$, and stores each transition $(s_t, a_t^k, k, a_t^{k-1})$. It additionally stores the reverse and forward log densities, critic predictions, termination flags, and

$$r_{\tilde{t}} = \mathbf{1}_{\{k=1\}} R_{\text{env}}(s_t, a_t^0), \qquad \ell_{\tilde{t}} = \log \frac{q_{\theta_{\text{roll}}}(a_t^{k-1} \mid a_t^k, s_t)}{\pi_{\theta_{\text{roll}}}(a_t^k \mid a_t^{k-1}, s_t)}. \tag{25}$$

For $k > 1$, the successor is formed in memory without calling the environment. For Gaussian policies, the buffer stores the terminal latent x_t^0, while the environment executes $a_t^0 = \tanh(x_t^0)$; its reward and episode boundary are attached to the final row of the chain. A new prior sample initializes the next chain. For the transformed-action entropy formulation, the terminal densities in the DA-MDP objective are evaluated in executed-action coordinates:

$$q_\theta(a_t^0 \mid a_t^1, s_t) = q_\theta^X(x_t^0 \mid a_t^1, s_t) e^{-j_{\tanh}(x_t^0)}, \qquad \pi_\theta(a_t^1 \mid a_t^0, s_t) = \pi_\theta^X(a_t^1 \mid x_t^0, s_t),$$

where q_θ^X, π_θ^X are latent-coordinate densities and $j_{\tanh}(x^0) = \sum_i \log(1 - \tanh^2(x_i^0))$. The transformed-action formulation therefore stores the latent terminal log ratio minus $j_{\tanh}(x_t^0)$. The latent-entropy formulation instead stores the latent terminal log ratio without this correction. The selected convention is used consistently in the actor loss, critic soft rewards, and temperature statistic (App. E.3.2).

Environment-time discounting. Two conventions are possible. The exact, phase-dependent convention defines $\bar{\gamma}(k) = 1$ for every internal denoising transition $k > 1$ and $\bar{\gamma}(1) = \gamma$ for the transition that executes the action and advances environment time. It assigns every reward and log-ratio term associated with environment time t the intended weight γ^t, and therefore exactly preserves the discounted DA-MDP objective. An alternative constant convention sets $\bar{\gamma}(k) = \gamma^{1/K}$ for every phase. It gives one factor of γ between consecutive environment rewards, but it additionally phase-weights the intermediate log-ratio terms and therefore corresponds to a different augmented objective. We found that the exact phase-dependent convention, $\bar{\gamma}(k) = 1$ for $k > 1$ and $\bar{\gamma}(1) = \gamma$, performs better; we use it throughout the paper and in all reported experiments. The trace parameter λ is tuned separately; empirically, we find that it should be chosen slightly larger than for Gaussian policies, for example $\lambda = 0.98$.

D.2 Critic Targets

DA: REPPO and DA: WPO learn an action-value function, whereas DA: PPO learns a state-value function. Both critics are trained from trajectories recorded under the rollout policy and targets computed over the flattened augmented trajectory. We describe the two cases separately below. The log ratios $\ell_{\tilde{t}}$ follow the entropy convention selected for the actor: the transformed-action formulation includes the terminal soft-reward contribution $+\mathcal{T} j_{\tanh}(x_t^0)$, whereas the latent-entropy formulation omits it (App. E.3.2).

D.2.1 Q-Function Training

For DA: REPPO and DA: WPO, the critic estimates $Q_\omega(\tilde{s}_{\tilde{t}}, \tilde{a}_{\tilde{t}})$ for the sampled local reverse-diffusion action. The soft Q-function excludes the entropy cost of its own action. Consequently, its Bellman target contains the reverse–forward log-ratio of the *successor* action. Let $\bar{\omega}$ denote the critic parameters used during rollout collection, let $z_{\tilde{t}}$ indicate a true environment termination, and define the stored bootstrap value

$$B_{\tilde{t}}^Q := Q_{\bar{\omega}}(\tilde{s}_{\tilde{t}+1}, \tilde{a}_{\tilde{t}+1}), \qquad \bar{r}_{\tilde{t}}^Q := r_{\tilde{t}} - \gamma_{\tilde{t}}(1 - z_{\tilde{t}})\mathcal{T}\ell_{\tilde{t}+1}. \tag{26}$$

Within a diffusion chain, $(\tilde{s}_{\tilde{t}+1}, \tilde{a}_{\tilde{t}+1})$ is simply the next stored local transition. At the last diffusion step, the implementation samples the first local transition of the next chain and evaluates both $B_{\tilde{t}}^Q$ and $\ell_{\tilde{t}+1}$ using that same sample. Thus, for $\lambda = 0$, the target reduces exactly to

$$y_{\tilde{t}}^{Q,(1)} = r_{\tilde{t}} + \gamma_{\tilde{t}}(1 - z_{\tilde{t}}) \left[B_{\tilde{t}}^Q - \mathcal{T}\ell_{\tilde{t}+1} \right]. \tag{27}$$

For the multi-step target used in the experiments, let $b_{\tilde{t}}$ indicate any episode boundary, including either a true termination or a time-limit truncation. The buffer computes TD-λ backwards as

$$\begin{aligned} \delta_{\tilde{t}}^Q &= \bar{r}_{\tilde{t}}^Q + \gamma_{\tilde{t}}(1 - z_{\tilde{t}})B_{\tilde{t}}^Q - Q_{\bar{\omega}}(\tilde{s}_{\tilde{t}}, \tilde{a}_{\tilde{t}}), \\ A_{\tilde{t}}^{Q,\lambda} &= \delta_{\tilde{t}}^Q + \gamma_{\tilde{t}}\lambda_{\tilde{t}}(1 - b_{\tilde{t}})A_{\tilde{t}+1}^{Q,\lambda}, \qquad y_{\tilde{t}}^{Q,\lambda} = Q_{\bar{\omega}}(\tilde{s}_{\tilde{t}}, \tilde{a}_{\tilde{t}}) + A_{\tilde{t}}^{Q,\lambda}. \end{aligned} \tag{28}$$

A true termination sets $z_{\tilde{t}} = b_{\tilde{t}} = 1$, eliminating both the bootstrap and continuation of the trace. A time-limit truncation sets $z_{\tilde{t}} = 0$ and $b_{\tilde{t}} = 1$: it ends the trace but still bootstraps from the terminal observation recorded before the environment is reset. This prevents returns from leaking across episode boundaries without incorrectly treating a time limit as a terminal state.

The Q-network is regressed onto $y_{\tilde{t}}^{Q,\lambda}$ while the actor is frozen. In the reported experiments, an HL-Gauss head encodes each scalar target as a Gaussian-smoothed categorical distribution over a fixed support, and the critic minimizes the corresponding cross-entropy loss (Farebrother et al., 2024). The Q update precedes the actor update in every minibatch, so the DA: REPPO and DA: WPO objectives use the most recent critic while keeping its parameters fixed during actor differentiation.

D.2.2 Value-Function Training

DA: PPO instead estimates $V_\omega(\tilde{s}_{\tilde{t}})$. Since a value function averages over the current local action, the current reverse–forward log-ratio belongs to its immediate soft reward,

$$\bar{r}_{\tilde{t}}^V := r_{\tilde{t}} - \mathcal{T}\ell_{\tilde{t}}. \tag{29}$$

During rollout collection, the implementation records both $V_{\bar{\omega}}(\tilde{s}_{\tilde{t}})$ and the successor bootstrap $B^V_{\tilde{t}} := V_{\bar{\omega}}(\tilde{s}_{\tilde{t}+1})$. It then computes generalized advantage estimates and value targets using the same termination and boundary masks as above:

$$\begin{aligned} \delta^V_{\tilde{t}} &= \bar{r}^V_{\tilde{t}} + \gamma_{\tilde{t}}(1 - z_{\tilde{t}})B^V_{\tilde{t}} - V_{\bar{\omega}}(\tilde{s}_{\tilde{t}}), \\ A^{V,\lambda}_{\tilde{t}} &= \delta^V_{\tilde{t}} + \gamma_{\tilde{t}}\lambda_{\tilde{t}}(1 - b_{\tilde{t}})A^{V,\lambda}_{\tilde{t}+1}, \qquad y^{V,\lambda}_{\tilde{t}} = V_{\bar{\omega}}(\tilde{s}_{\tilde{t}}) + A^{V,\lambda}_{\tilde{t}}. \end{aligned} \tag{30}$$

We discount only when the environment advances: $\gamma_{\tilde{t}} = 1$ during internal denoising steps ($k > 1$), and $\gamma_{\tilde{t}} = \gamma$ when the final step ($k = 1$) executes the action (App. D.1). Thus, denoising introduces no additional discount. The trace parameter $\lambda_{\tilde{t}}$ controls how strongly later temporal-difference errors contribute to the current advantage. We train the value head on $y^{V,\lambda}_{\tilde{t}}$ using HL-Gauss cross-entropy and use $A^{V,\lambda}_{\tilde{t}}$ to construct the DA: PPO actor advantage. As in PPO, we update the actor using the frozen rollout critic, then fit the value function with the actor frozen.

D.3 DA: REPPO

DA: REPPO applies REPPO to the local transitions stored in the augmented rollout buffer. The construction does not require a particular reverse or forward diffusion kernel. For any reparameterizable reverse kernel, write a fresh local action as

$$\hat{a}^{k-1}_\theta := g_\theta(\epsilon; \tilde{s}), \qquad \epsilon \sim \rho(\epsilon), \qquad \tilde{s} = (s_t, a^k_t, k), \tag{31}$$

where the base distribution ρ is independent of θ. The sample-wise maximum-entropy policy loss is

$$f_\theta(\tilde{s}, a^{k-1}) := \mathcal{T} \log \frac{q_\theta(a^{k-1} \mid a^k_t, s_t)}{\pi_\theta(a^k_t \mid a^{k-1}, s_t)} - Q^{q_{\theta_{\text{roll}}}}_{\text{DA}}(\tilde{s}, a^{k-1}). \tag{32}$$

The reverse-kernel term encourages entropy in each denoising decision, the forward-kernel term enforces consistency with the prescribed noising process, and the critic favors local actions with high predicted return. The unconstrained actor objective is therefore

$$\mathcal{L}^{\text{uncon}}_{\text{DA:REPPO}}(\theta) = \mathbb{E}_{\substack{\tilde{s}\sim d^{q_{\theta_{\text{roll}}}}_{\text{DA}} \\ \epsilon\sim\rho}} \left[f_\theta(\tilde{s}, g_\theta(\epsilon; \tilde{s})) \right]. \tag{33}$$

A minibatch average of Eq. 33 estimates the unnormalized local reverse-KL objective in Eq. 10; no state-dependent partition function is evaluated. The augmented-state distribution, rollout parameters θ_{roll}, and critic parameters are held fixed, while gradients pass through the reparameterized action, the reverse and forward log densities, and the critic's action input. Consequently, learned forward-kernel parameters receive the correction implied by their log density.

DA: REPPO measures the change in each reverse kernel using the rollout-to-current KL,

$$d_\theta(\tilde{s}) := D_{\text{KL}}(q_{\theta_{\text{roll}}}(\cdot \mid \tilde{s}) \,\|\, q_\theta(\cdot \mid \tilde{s})), \qquad \widehat{D}^{\text{chain}}_\theta(\tilde{s}) := K d_\theta(\tilde{s}). \tag{34}$$

Each local KL can be computed analytically or estimated by sampling, depending on the kernel family. For uniformly sampled diffusion steps, the minibatch average of $K d_\theta$ estimates the expected sum of KLs over all K steps. Actor clipping uses the scaled KL of each sample separately. With bound δ_{KL} and multiplier $\lambda_{\text{KL}} \geq 0$, the actor loss is

$$\mathcal{L}^{\text{REPPO}}_{\text{actor}}(\theta) = \mathbb{E}_{\substack{\tilde{s}\sim d^{q_{\theta_{\text{roll}}}}_{\text{DA}} \\ \epsilon\sim\rho}} \left[\begin{cases} f_\theta(\tilde{s}, g_\theta(\epsilon; \tilde{s})), & \widehat{D}^{\text{chain}}_\theta \leq \delta_{\text{KL}}, \\ \lambda_{\text{KL}} \widehat{D}^{\text{chain}}_\theta, & \widehat{D}^{\text{chain}}_\theta > \delta_{\text{KL}}. \end{cases} \right]. \tag{35}$$

The branch indicator and λ_{KL} are stopped during the policy update. An in-bound sample therefore follows the maximum-entropy policy gradient, whereas an out-of-bound sample contributes only the trust-region gradient. The KL multiplier and temperature are updated separately by the dual objectives in Sec. D.5.

Gaussian trust region. For the Gaussian reverse kernels of the VP-SDE (App. E.3.2), the rollout-to-current KL has a closed form. At each diffusion step, let $\mu_{\theta_{\text{roll}}}, \sigma^2_{\theta_{\text{roll}}}$ denote the rollout mean and diagonal variance, and $\mu_\theta, \sigma^2_\theta$ their current values. Then

$$d_\theta(\tilde{s}) = \frac{1}{2} \sum_{i=1}^{N} \left[\log \frac{\sigma^2_{\theta,i}}{\sigma^2_{\theta_{\text{roll}},i}} + \frac{\sigma^2_{\theta_{\text{roll}},i} + (\mu_{\theta_{\text{roll}},i} - \mu_{\theta,i})^2}{\sigma^2_{\theta,i}} - 1 \right]. \tag{36}$$

Both DA: REPPO and DA: WPO evaluate the rollout-to-current KL at sampled diffusion steps and scale it by K (Eq. 34). The KL multiplier is updated using the minibatch average, while actor clipping is applied separately to each sample.

D.4 DA: WPO

DA: WPO uses the same augmented rollout, Q critic, KL trust region, and dual variables as DA: REPPO. It replaces only the in-bound maximum-entropy policy direction by a Wasserstein score-matching direction. In the following equations, $\tilde{s} = (s_t, a_t^k, k) \sim d_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}$ and $a^{k-1} \sim q_{\theta^*}(\cdot \mid a_t^k, s_t)$, with the sampled action held fixed during differentiation. For any differentiable reverse and forward kernels, the score-matching loss is

$$\mathcal{L}_{\mathrm{DA:WPO}}(\theta) := \frac{\mathcal{T}}{2}\mathbb{E}_{\tilde{s},a^{k-1}}\left[\left\|\nabla_{a^{k-1}}\left(\log\frac{q_\theta(a^{k-1}\mid\tilde{s})}{\pi_\theta(a_t^k\mid a^{k-1},s_t)} - \alpha Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s},a^{k-1})\right)\right\|_2^2\right]. \tag{37}$$

For Gaussian policies, the implementation takes score derivatives in unsquashed coordinates x^{k-1}. At $k = 1$, the differentiated terminal log ratio includes $-j_{\tanh}(x^0)$ under transformed-action entropy and omits it under latent entropy. The critic uses the same entropy convention, and its terminal action dependence is differentiated with respect to x^0. The implemented score norm thus uses latent-coordinate geometry.

This loss directly compares the reverse-kernel score with the forward-kernel score plus the temperature-scaled critic gradient. It requires only that the two kernel densities and their action derivatives can be evaluated locally; it does not depend on a Gaussian or SDE parameterization. Because the critic is nonlinear in a^{k-1}, DA: WPO generally samples the local action even when the kernel scores themselves are analytic. With the sampled action and critic held fixed, the parameter gradient is

$$\nabla_\theta\mathcal{L}_{\mathrm{DA:WPO}}(\theta) = \mathbb{E}_{\tilde{s},a^{k-1}}\left[\left(\nabla_\theta\nabla_{a^{k-1}}\log\frac{q_\theta(a^{k-1}\mid\tilde{s})}{\pi_\theta(a_t^k\mid a^{k-1},s_t)}\right)\nabla_{a^{k-1}}\left(\mathcal{T}\log\frac{q_\theta(a^{k-1}\mid\tilde{s})}{\pi_\theta(a_t^k\mid a^{k-1},s_t)} - Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s},a^{k-1})\right)\right]. \tag{38}$$

The implementation obtains Eq. 38 from the explicit first-order surrogate

$$\widehat{f}_\theta^{\mathrm{DA:WPO}} := \mathrm{sg}\left[\nabla_{a^{k-1}}\left(\mathcal{T}\log\frac{q_\theta(a^{k-1}\mid\tilde{s})}{\pi_\theta(a_t^k\mid a^{k-1},s_t)} - Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s},a^{k-1})\right)\right]^\top\nabla_{a^{k-1}}\log\frac{q_\theta(a^{k-1}\mid\tilde{s})}{\pi_\theta(a_t^k\mid a^{k-1},s_t)}. \tag{39}$$

$$\widehat{\mathcal{L}}_{\mathrm{DA:WPO}}(\theta) = \mathbb{E}_{\tilde{s},a^{k-1}}\left[\widehat{f}_\theta^{\mathrm{DA:WPO}}\right]. \tag{40}$$

The stop-gradient operator avoids derivatives through the critic gradient while retaining derivatives through the reverse and forward kernel scores.

Combining the score loss with the clipped reverse-kernel trust region gives the final DA: WPO actor loss

$$\mathcal{L}_{\mathrm{actor}}^{\mathrm{DA:WPO}}(\theta) = \mathbb{E}\left[\begin{cases}\widehat{f}_\theta^{\mathrm{DA:WPO}}, & \widehat{D}_\theta^{\mathrm{chain}} \le \delta_{\mathrm{KL}},\\ \lambda_{\mathrm{KL}}\widehat{D}_\theta^{\mathrm{chain}}, & \widehat{D}_\theta^{\mathrm{chain}} > \delta_{\mathrm{KL}}.\end{cases}\right]. \tag{41}$$

The branch condition and λ_{KL} are stopped. Hence, an in-bound sample follows the (optionally preconditioned) score-matching gradient, whereas an out-of-bound sample contributes only the KL-penalty gradient. When a separate forward-kernel trust region is enabled, its gate and KL penalty are applied independently.

Inverse-Fisher approximation. Computing and inverting the Fisher matrix over all network parameters is intractable for our policies. Following Pfau et al. (2025), we instead use the inverse Fisher of the Gaussian distribution parameters as a local preconditioner. For the VP reverse kernel in Eq. 66, this requires accounting for the coupling between the control $u_{\Psi,i}$ and diffusion coefficient $\delta_{k,i}$: both affect the mean, and $\delta_{k,i}$ also determines the variance. Expressing the Gaussian Fisher in these coordinates gives an exact 2×2 block for each action coordinate and diffusion step. Writing $v_{k,i} = \nu^2\delta_{k,i}$ and $h_{k,i} = \frac{1}{2}a_{t,i}^k + \nu^2 u_{\Psi,i}(s_t, a_t^k, k)$, this block is

$$F_{k,i}^{\mathrm{VP}} = \begin{pmatrix} v_{k,i} & h_{k,i} \\ h_{k,i} & h_{k,i}^2/v_{k,i} + 1/(2\delta_{k,i}^2)\end{pmatrix}. \tag{42}$$

We stabilize each block by diagonal normalization, correlation clipping, and relative damping before inversion, then rescale its inverse to cap the largest eigenvalue. The resulting matrix $P_{k,i}$ preconditions the score-loss gradient with its entries held fixed:

$$\widetilde{g}_{k,i} = \mathrm{sg}(P_{k,i})g_{k,i}, \qquad g_{k,i} := \nabla_{(u_{\Psi,i},\delta_{k,i})}\widehat{f}_\theta^{\mathrm{DA:WPO}}. \tag{43}$$

D.5 Trust-Region and Temperature Dual Updates

DA: REPPO and DA: WPO optimize the KL multiplier in log space. Let $\lambda_{\mathrm{KL}} = \exp(\xi_{\mathrm{KL}})$ and let $\widehat{D}_{\mathrm{KL}}$ be the minibatch mean of the estimated KL sum over diffusion steps in Eq. 34. Although each local Gaussian KL is available in closed form, its sum over diffusion steps can be estimated by Monte Carlo sampling. If κ indices k_j are drawn independently from probabilities $\rho_k > 0$, then

$$\widehat{D}_\theta^{\mathrm{chain}} := \frac{1}{\kappa}\sum_{j=1}^{\kappa}\frac{d_{\theta,k_j}}{\rho_{k_j}}, \qquad \mathbb{E}_{k_{1:\kappa}}\left[\widehat{D}_\theta^{\mathrm{chain}}\right] = \sum_{k=1}^{K} d_{\theta,k}. \tag{44}$$

For uniform sampling, $\rho_k = 1/K$, so every sampled local KL receives weight K. The KL dual loss is

$$\mathcal{L}_{\mathrm{KL}}^{\mathrm{dual}}(\xi_{\mathrm{KL}}) := \lambda_{\mathrm{KL}}\left(\delta_{\mathrm{KL}} - \mathrm{sg}(\widehat{D}_{\mathrm{KL}})\right). \tag{45}$$

Minimizing this loss gives the update direction

$$\xi_{\mathrm{KL}} \leftarrow \xi_{\mathrm{KL}} + \eta_{\mathrm{KL}}\lambda_{\mathrm{KL}}\left(\widehat{D}_{\mathrm{KL}} - \delta_{\mathrm{KL}}\right), \tag{46}$$

so the penalty increases precisely when the estimated KL sum exceeds its bound. During this dual update, the KL estimate is stopped; during the actor update, the multiplier is stopped.

For Gaussian policies, temperature tuning uses the same entropy convention as the actor and critic. The latent-entropy formulation targets a path-space lower bound on terminal latent entropy; the transformed-action formulation adds the expected terminal $\tanh$ log-Jacobian to this bound (App. E.3.2). Here $x^{0:K}$ denotes the unsquashed path, and all displayed kernels are latent-coordinate densities. At a fixed environment state s, the latent-space path-entropy lower bound is

$$\mathcal{H}_{\mathrm{lower}}[q_\theta](s) = \mathcal{H}[q_{\mathrm{prior}}] + \mathbb{E}_{x^{0:K}\sim q_\theta(\cdot|s)}\left[\sum_{k=1}^{K}\log\frac{\pi_\theta(x^k \mid x^{k-1}, s)}{q_\theta(x^{k-1} \mid x^k, s)}\right]. \tag{47}$$

Here, $q_\theta(x^{0:K} \mid s) = q_{\mathrm{prior}}(x^K)\prod_{k=1}^{K} q_\theta(x^{k-1} \mid x^k, s)$ is the complete current-policy reverse path at fixed s. The state-averaged bound is $\mathcal{H}_{\mathrm{lower}}[q_\theta] := \mathbb{E}_{s\sim d^{q_{\theta_{\mathrm{roll}}}}}[\mathcal{H}_{\mathrm{lower}}[q_\theta](s)]$. An unbiased Monte Carlo estimator draws $s_b \sim d^{q_{\theta_{\mathrm{roll}}}}$, then $x_b^{0:K} \sim q_\theta(\cdot \mid s_b)$, and independently κ indices $k_{b,j} \sim \rho$ for each path b:

$$\begin{aligned}\widehat{\mathcal{H}}_{\mathrm{lower}} &:= \mathcal{H}[q_{\mathrm{prior}}] + \frac{1}{B\kappa}\sum_{b=1}^{B}\sum_{j=1}^{\kappa}\frac{1}{\rho_{k_{b,j}}}\log\frac{\pi_\theta(x_b^{k_{b,j}} \mid x_b^{k_{b,j}-1}, s_b)}{q_\theta(x_b^{k_{b,j}-1} \mid x_b^{k_{b,j}}, s_b)},\\ \mathbb{E}\left[\widehat{\mathcal{H}}_{\mathrm{lower}}\right] &= \mathcal{H}_{\mathrm{lower}}[q_\theta].\end{aligned} \tag{48}$$

The expectation in the second line averages over the sampled states, complete reverse paths, and diffusion indices. In implementation, states and noisy actions x^k are taken from the rollout buffer, while local actions are sampled freshly from $q_\theta(\cdot \mid x^k, s)$. Thus, the state average is empirical, and reusing rollout noisy actions approximates current-policy path sampling when $\theta \neq \theta_{\mathrm{roll}}$. Only the sampled local log ratios need to be evaluated. Under uniform sampling, the inverse-probability factor is again K; for nonuniform sampling, $1/\rho_k$ preserves unbiasedness with respect to diffusion-index sampling. Let $\mathcal{H}_{\mathrm{tar}}$ be the target, and parameterize $\mathcal{T} = \exp(\xi_{\mathcal{T}})$. In the dual update below, $\widehat{\mathcal{H}}_{\mathrm{lower}}$ denotes the entropy estimate for the selected formulation: Eq. 48 for latent entropy, and that estimate plus the sampled terminal log-Jacobian for transformed-action entropy (App. E.3.2). The temperature dual loss and its update direction are

$$\mathcal{L}_{\mathcal{T}}^{\mathrm{dual}}(\xi_{\mathcal{T}}) := \mathcal{T}\left(\mathrm{sg}(\widehat{\mathcal{H}}_{\mathrm{lower}}) - \mathcal{H}_{\mathrm{tar}}\right), \tag{49}$$

$$\xi_{\mathcal{T}} \leftarrow \xi_{\mathcal{T}} + \eta_{\mathcal{T}}\mathcal{T}\left(\mathcal{H}_{\mathrm{tar}} - \widehat{\mathcal{H}}_{\mathrm{lower}}\right). \tag{50}$$

Thus, the temperature increases when the estimated entropy is below target and decreases when it is above target. The exponential parameterizations keep both dual variables positive. Equations 46 and 50 display the gradient-descent directions; the implementation applies them with separate Adam parameter groups and learning rates.

D.6 DA: PPO

DA: PPO applies PPO to stored local denoising transitions. We describe the clipped-forward variant used in the diffusion-step ablation (Mode 2 in the implementation). It combines the current soft advantage with a learned-forward-kernel correction restricted to PPO's active set. For the augmented transition indexed by (t, k), Eq. 30 gives the rollout-policy advantage as

$$A^{\mathrm{roll}}_{t,k} := y^{V,\lambda}_{t,k} - V_{\bar{\omega}}(\tilde{s}_{t,k}) = A^{V,\lambda}_{t,k}. \tag{51}$$

Thus, $A^{\mathrm{roll}}_{t,k}$ is the GAE estimate obtained from the rollout soft rewards $\bar{r}^{\theta_{\mathrm{roll}}}_{t,k} = r_{t,k} - \mathcal{T}\ell^{\theta_{\mathrm{roll}}}_{t,k}$ and the phase-dependent discounts in Sec. D.1.

The soft reward depends on the policy and must therefore be updated during the actor optimization. For the stored rollout action a^{k-1}_t, define the current path log-ratio and soft reward

$$\ell^{\theta}_{t,k} := \log q_\theta(a^{k-1}_t \mid a^k_t, s_t) - \log \pi_\theta(a^k_t \mid a^{k-1}_t, s_t), \qquad \bar{r}^{\theta}_{t,k} := r_{t,k} - \mathcal{T}\ell^{\theta}_{t,k}, \tag{52}$$

Replacing the current rollout reward in the GAE estimate gives

$$A^{\theta}_{t,k} = A^{\mathrm{roll}}_{t,k} - \bar{r}^{\theta_{\mathrm{roll}}}_{t,k} + \bar{r}^{\theta}_{t,k}. \tag{53}$$

The implementation inserts this current soft reward into the stopped PPO advantage:

$$A^{\mathrm{cur}}_{t,k} := A^{\mathrm{roll}}_{t,k} - \bar{r}^{\theta_{\mathrm{roll}}}_{t,k} + r_{t,k} - \mathcal{T}\ell^{\theta}_{t,k}, \qquad \widehat{A}_{t,k} := \mathrm{sg}\left(\frac{A^{\mathrm{cur}}_{t,k} - \mu_A}{\sigma_A}\right). \tag{54}$$

Here, μ_A and σ_A are the minibatch advantage mean and scale, with $\sigma_A = 1$ when variance normalization is disabled. The current transition's soft reward is recomputed under the current reverse and forward diffusion kernels and then stopped inside the PPO surrogate, whereas successor soft rewards and critic targets stored at rollout time remain fixed.

For the stored action a^{k-1}_t, the current-to-behavior likelihood ratio and its clipped version are

$$\rho_{t,k}(\theta) := \frac{q_\theta(a^{k-1}_t \mid a^k_t, s_t)}{q_{\theta_{\mathrm{roll}}}(a^{k-1}_t \mid a^k_t, s_t)}, \qquad \bar{\rho}_{t,k} := \mathrm{clip}(\rho_{t,k}, 1-\epsilon, 1+\epsilon). \tag{55}$$

The standard PPO contribution is

$$\mathcal{L}_{\mathrm{clip}}(\theta) = -\mathbb{E}\Big[\min\Big(\rho_{t,k}\widehat{A}_{t,k}, \bar{\rho}_{t,k}\widehat{A}_{t,k}\Big)\Big]. \tag{56}$$

Stopping the advantage leaves the direct derivative through a learned forward kernel to be supplied separately. We restrict this correction to the same active set as the reverse-policy PPO update. The implementation uses the mask

$$m_{t,k} := \mathbf{1}\Big\{\rho_{t,k}\widehat{A}_{t,k} < \bar{\rho}_{t,k}\widehat{A}_{t,k} \quad \text{or} \quad 1-\epsilon \le \rho_{t,k} \le 1+\epsilon\Big\}. \tag{57}$$

The forward correction is evaluated on stored rollout actions. Up to terms with stopped gradients, it is

$$\mathcal{L}^{\mathrm{clip}}_{\mathrm{fwd}}(\theta) = -\frac{\mathcal{T}}{\sigma_A}\,\mathbb{E}_{\mathrm{roll}}\big[\mathrm{sg}(m_{t,k}\rho_{t,k}) \log \pi_\theta(a^k_t \mid a^{k-1}_t, s_t)\big]. \tag{58}$$

Here, the expectation uses stored local transitions, and the actions, mask, importance ratio, temperature, and advantage scale are held fixed during differentiation. The ratio corrects for sampling actions from the rollout policy; stopping it avoids an additional reverse-policy score term. Samples outside the active set contribute no direct forward-kernel gradient. The full actor loss is

$$\mathcal{L}^{\mathrm{PPO}}_{\mathrm{actor}} = \mathcal{L}_{\mathrm{clip}} + \mathcal{L}^{\mathrm{clip}}_{\mathrm{fwd}}. \tag{59}$$

An alternative is the unmasked fresh-sample correction (Mode 1). At each stored augmented state, sample $\hat{a}^{k-1}_t \sim q_\theta(\cdot \mid a^k_t, s_t)$ without reparameterization and replace $\mathcal{L}^{\mathrm{clip}}_{\mathrm{fwd}}$ by

$$\mathcal{L}^{\mathrm{fresh}}_{\mathrm{fwd}}(\theta) = -\frac{\mathcal{T}}{\sigma_A}\,\mathbb{E}\big[\log \pi_\theta\big(a^k_t \mid \mathrm{sg}(\hat{a}^{k-1}_t), s_t\big)\big]. \tag{60}$$

This alternative needs neither an importance ratio nor a PPO mask. At the rollout parameters, $\rho_{t,k} = m_{t,k} = 1$, so both variants recover the same population actor gradient, up to fixed advantage scaling. They differ away from the rollout policy: Mode 2 gates the forward correction with PPO clipping, whereas Mode 1 retains it on fresh samples. If the forward kernel is fixed, neither correction contributes a parameter gradient. Empirical clipping and KL control do not guarantee monotonic improvement with finite samples. Setting $\mathcal{T} = 0$ removes the path correction and yields a DPPO-style local PPO objective (Ren et al., 2025).

E Extended Paper Sections

E.1 Related Work

E.1.1 Sampling from Unnormalized Target Distributions

Sampling from unnormalized target distributions is a vibrant and active research area. Classical methods, such as Markov Chain Monte Carlo (MCMC) (Metropolis et al., 1953), have long been foundational, but recent advances increasingly leverage neural networks to approximate target distributions (Noé & Wu, 2018; Wu et al., 2019). These approaches find broad applications, including molecule configuration prediction (Noé & Wu, 2018), statistical physics (Wu et al., 2019), Monte Carlo integration (Müller et al., 2019), and combinatorial optimization (Hibat-Allah et al., 2021). Diffusion samplers offer an expressive solution by transporting noise samples toward high-density regions of the target distribution π via a learned reverse diffusion process. These methods have been successfully applied in both continuous (Zhang & Chen, 2022; Berner et al., 2022; Vargas et al., 2023; 2024; Richter & Berner, 2024) and discrete domains (Sanokowski et al., 2024; 2025a), demonstrating robust performance on challenging sampling problems. Adjoint Matching (Domingo-Enrich et al., 2025) also employs a score-matching-based loss similar to our DA: WPO loss, though it is derived from the adjoint method and, in contrast to our method, does not allow for learnable diffusion coefficients. Physics-informed neural network (PINN) (Raissi et al., 2019) losses, derived from the continuity equation of continuous normalizing flows, have been explored in (Tian et al., 2024; Máté & Fleuret, 2023; Fan et al., 2024). Reinforcement learning (RL) naturally aligns with the unnormalized-sampling paradigm. A substantial body of work has established that RL can be formulated as probabilistic inference (Todorov, 2008; Ziebart, 2010; Kappen et al., 2012; Levine, 2018; Abdolmaleki et al., 2018).

E.2 Diffusion Models in Reinforcement Learning

Diffusion models have recently gained traction in RL for representing complex action distributions. Dong et al. (2025) optimize a forward-KL objective, whereas Ma et al. (2025) consider both forward- and reverse-KL updates toward a policy-mirror-descent target. Their target-weighted diffusion training differs from our local maximum-entropy trajectory construction. DIME (Celik et al., 2025) integrates diffusion models with Soft Actor-Critic (SAC), but unlike our approach, it does not evaluate a Q-function at each diffusion step. Instead, its objective is based on applying the Data Processing Inequality directly to the surrogate loss in Eq. 3. This method requires backpropagation through the entire diffusion chain for every environment action, which becomes memory-intensive for large numbers of diffusion steps K. The work most closely related to ours is Sanokowski et al. (2025a), who use RL to train diffusion models scalably for unnormalized sampling. We extend their local policy-gradient construction to RL itself: the diffusion model becomes the policy acting in an environment, and its local transition rewards are derived from the ME-RL trajectory objective. At $\mathcal{T} = 0$, DA: PPO yields a DPPO-style local PPO objective (Ren et al., 2025); the original DPPO algorithm additionally specifies denoising discounting and fine-tuning choices. Our formulation adds the maximum-entropy path correction and supports learned forward kernels. Recent works have also explored flow models in RL (Liu et al., 2025; Park et al., 2025; Zhang et al., 2025; McAllister et al., 2025), further expanding the intersection of generative models and RL.

E.3 Additional Experimental Details

E.3.1 Continuous-Control Experiments

We compare the three DA-MDP methods with PPO, REPPO, WPO, ME-WPO, TruDi, DPPO, and FPO on twelve DeepMind Control tasks in MuJoCo Playground (Zakka et al., 2025). Our experiments use one configuration per method across tasks and 50M environment interactions. We aggregate eight independent training seeds per method–task pair. Diffusion policies use $K = 8$ steps; DPPO is implemented as DA: PPO with $\mathcal{T} = 0$. PPO (r) denotes the externally reported results of Voelcker et al. (2026); all other curves are from our experiments. The NaNs originated *within the environment simulations themselves*, caused by bugs in the Humanoid Run, Walk, and Stand environments. These simulation failures occurred more frequently for diffusion-based variants. After four consecutive NaN failures despite restarting with new random seeds, we excluded these environments

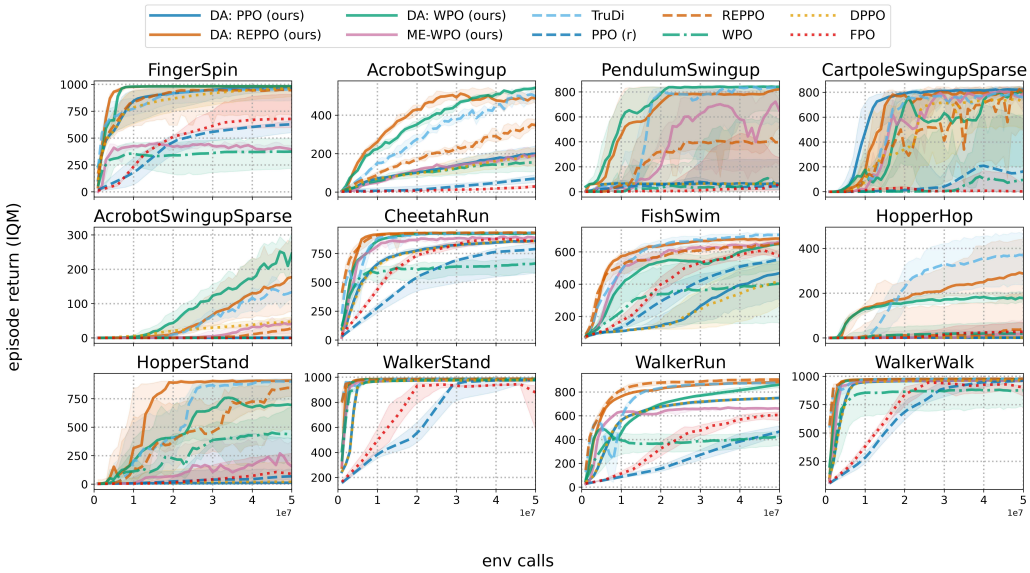


Figure 5: Continuous-control performance across twelve tasks. Curves show IQM returns; shaded regions indicate 95% bootstrap confidence intervals.

from the experiments. Common settings are given in App. G, with exact experiment configurations provided alongside the code.

The twelve-task comparison evaluates diffusion policies using ODE sampling (Eq. 67). We report the interquartile mean (IQM) (Agarwal et al., 2021), both per task (Fig. 5) and across tasks (Fig. 3a). For aggregate results, we divide returns by 1000 and compute an equally task-weighted IQM over seed-level scores. We obtain 95% confidence intervals from 10,000 bootstrap replicates, resampling seeds independently within each task.

All three DA-MDP variants improve on their corresponding base algorithms in aggregate. DA: REPPO learns fastest overall and reaches final performance comparable to TruDi. DA: WPO substantially improves on WPO and ME-WPO, while DA: PPO outperforms PPO and DPPO. The task-level results show particularly large gains for the Q-based DA methods on PendulumSwingup and AcrobotSwingupSparse, although TruDi remains stronger on HopperHop. These results show that local denoising updates can retain the performance benefits of diffusion policies across diverse control tasks.

E.3.2 Action-Transform and Entropy-Measure Ablation

Continuous-control environments require bounded actions, whereas the Gaussian reverse process used by DA: REPPO produces a terminal latent $x^0 \in \mathbb{R}^d$. The usual solution is to execute $a = \tanh(x^0)$. The Gaussian ablation compares two entropy conventions throughout training. The latent-entropy formulation omits the terminal $\tanh$ log-Jacobian from the actor objective, critic soft rewards, and temperature statistic. The transformed-action entropy formulation includes this correction in all three components. Both formulations execute the same action transform. We ablate these two choices and compare them with a diffusion process that is bounded by construction.

Latent-space and transformed-space entropy. We distinguish two entropy measures: **Tanh latent**, which constrains entropy in latent coordinates $x^0 \in \mathbb{R}^d$, and **Tanh** $+J$, which constrains entropy in transformed-action coordinates $a = \tanh(x^0)$. In the latent formulation, the environment applies the action transform and entropy is measured before squashing. In the transformed-action formulation, the transform belongs to the policy and its log-Jacobian contributes to the actor objective, critic soft rewards, and temperature statistic. The latent-action environment is defined by

$$R_X(s, x^0) = R_{\text{env}}(s, \tanh x^0), \qquad P_X(s' \mid s, x^0) = P_{\text{env}}(s' \mid s, \tanh x^0).$$

The latent density q_X and transformed density q_A are related by change of variables:

$$\log q_A(a \mid s) = \log q_X(x^0 \mid s) - j_{\tanh}(x^0),$$
$$\mathcal{H}[q_A(\cdot \mid s)] = \mathcal{H}[q_X(\cdot \mid s)] + \mathbb{E}_{q_X(\cdot|s)}[j_{\tanh}(x^0)],$$
$$j_{\tanh}(x^0) := \sum_{i=1}^{d} \log\bigl(1 - \tanh^2(x_i^0)\bigr)\,.$$

The same correction relates the corresponding path-space entropy bounds.

For a minibatch of B stored augmented states $\tilde{s}_b = (s_b, x_b^{k_b}, k_b)$ with uniformly sampled phases $k_b \in \{1, \dots, K\}$, draw fresh local actions $\hat{x}_b^{k_b-1} \sim q_\theta(\cdot \mid x_b^{k_b}, s_b)$. The sampled entropy bounds are

$$\widetilde{\mathcal{H}}_X = \mathcal{H}[q_{\text{prior}}] + \frac{K}{B}\sum_{b=1}^{B} \log \frac{\pi_\theta(x_b^{k_b} \mid \hat{x}_b^{k_b-1}, s_b)}{q_\theta(\hat{x}_b^{k_b-1} \mid x_b^{k_b}, s_b)},$$
$$\widetilde{\mathcal{H}}_A = \widetilde{\mathcal{H}}_X + \widehat{j}_{\tanh}, \qquad \widehat{j}_{\tanh} := \frac{1}{B}\sum_{b=1}^{B} j_{\tanh}(\widetilde{x}_b^0). \tag{61}$$

The factor K accounts for sampling diffusion phases. For the Jacobian estimate, each row supplies the stored terminal-phase pair (s_b, x_b^1) from the same rollout chain, and we draw $\widetilde{x}_b^0 \sim q_\theta(\cdot \mid x_b^1, s_b)$ independently of the selected phase k_b. Thus, the terminal correction is estimated for every row and has no additional factor K. Reusing rollout conditioning states gives the approximation described in App. D.5.

Let $\mathcal{J}_{\text{env}}(\theta)$ denote expected environment return, with the same state/time weighting understood for the entropy terms. The two sampled Lagrangians are

$$\mathcal{L}_X(\theta, \mathcal{T}) = \mathcal{J}_{\text{env}}(\theta) + \mathcal{T}\bigl(\widetilde{\mathcal{H}}_X - \mathcal{H}_{\text{tar}}^X\bigr),$$
$$\mathcal{L}_A(\theta, \mathcal{T}) = \mathcal{J}_{\text{env}}(\theta) + \mathcal{T}\bigl(\widetilde{\mathcal{H}}_A - \mathcal{H}_{\text{tar}}^A\bigr). \tag{62}$$

Actors maximize the corresponding Lagrangian: the latent formulation omits the Jacobian, whereas the transformed formulation includes it. At any diffusion phase k, their local actor losses, before trust-region modifications, have the form

$$\ell_k^m(\theta; \mathcal{T}) = \mathbb{E}_{x^{k-1}\sim q_\theta(\cdot|x^k,s)}\Bigg[- \bar{Q}_{\text{DA}}^m(\tilde{s}_k, x^{k-1}) + \mathcal{T}\log\frac{q_\theta(x^{k-1} \mid x^k, s)}{\pi_\theta(x^k \mid x^{k-1}, s)}$$
$$- \mathcal{T}\,\mathbf{1}_{\{m=A\}}\mathbf{1}_{\{k=1\}} j_{\tanh}(x^{k-1})\Bigg], \qquad m \in \{X, A\}. \tag{63}$$

Here $\tilde{s}_k = (s, x^k, k)$, and $\bar{Q}_{\text{DA}}^m$ is the augmented critic for formulation m, excluding the current local entropy terms and including future soft rewards (App. D.2.1). Its parameters are held fixed during actor differentiation. Every phase uses its own local decision x^{k-1}. At $k = 1$, both formulations execute the environment action $\tanh(x^0)$, but only the transformed-action formulation ($m = A$) includes the explicit Jacobian term. The critic's future soft rewards use the same entropy convention.

For either formulation, temperature tuning minimizes

$$\mathcal{L}_{\mathcal{T}}^m(\xi_{\mathcal{T}}) = \exp(\xi_{\mathcal{T}})\ \text{sg}\Bigl(\widetilde{\mathcal{H}}_m - \mathcal{H}_{\text{tar}}^m\Bigr), \qquad \mathcal{T} = \exp(\xi_{\mathcal{T}}), \quad m \in \{X, A\}. \tag{64}$$

The entropy statistic, including the Jacobian sample for $m = A$, is detached in this loss. Gradient descent therefore increases the temperature when entropy is below its target and decreases it when entropy is above target.

The latent-space return can be optimized directly under the cost and gradient integrability conditions in App. F.2. To additionally recover the finite-horizon normalized-target KL interpretation of Secs. 2.1 and 3.1, we assume, for each fixed $\mathcal{T} > 0$,

$$M_{\mathcal{T}} := \sup_s \int_{\mathbb{R}^d} \exp(R_X(s, x)/\mathcal{T})\ dx < \infty.$$

A sufficient condition is $R_X(s,x) \le C - c\|x\|^2$, uniformly in s, with $c > 0$. Normalized forward kernels preserve the finite partition function. For geometric stopping, the sufficient condition in Eq. 101 becomes $\gamma M_{\mathcal{T}} < 1$, which a constant reward shift can enforce as in Eq. 102. This additional normalization assumption concerns the KL interpretation and is not required to define the return itself.

Variance-preserving Gaussian diffusion. For the Gaussian variants, let $x^k \in \mathbb{R}^d$ denote the latent at diffusion step k and let $q_{\text{prior}} = \mathcal{N}(\mathbf{0}, \nu^2 I_d)$. The forward process is the variance-preserving SDE

$$dx = -\frac{1}{2}\boldsymbol{\beta}(t)x\,dt + \nu\boldsymbol{\beta}(t)^{1/2}dW_t, \tag{65}$$

whose stationary distribution is $\mathcal{N}(\mathbf{0}, \nu^2 I_d)$. To parameterize the reverse process, we introduce a learned control $u_\Psi(s, x^k, k) \in \mathbb{R}^d$, predicted by a neural network with parameters Ψ from the environment state s, noisy latent x^k, and diffusion step k. This control provides the learned correction to the reverse drift. We discretize each coordinate independently. Writing $\boldsymbol{\delta}_k = \operatorname{diag}(\delta_{k,1}, \ldots, \delta_{k,d})$, with $(\delta_{k,1}, \ldots, \delta_{k,d})^\top = \boldsymbol{\beta}_k \Delta_k$, gives the paired transition kernels

$$\begin{aligned} \pi_\theta(x^k \mid x^{k-1}, s) &= \mathcal{N}\big((I - \tfrac{1}{2}\boldsymbol{\delta}_{k-1})x^{k-1}, \nu^2\boldsymbol{\delta}_{k-1}\big)\,, \\ q_\theta(x^{k-1} \mid x^k, s) &= \mathcal{N}\big((I + \tfrac{1}{2}\boldsymbol{\delta}_k)x^k + \nu^2\boldsymbol{\delta}_k u_\Psi(s, x^k, k), \nu^2\boldsymbol{\delta}_k\big)\,. \end{aligned} \tag{66}$$

The complete parameter vector is $\theta = (\Psi, \boldsymbol{\beta}_K, \ldots, \boldsymbol{\beta}_0)$. At reverse step k, q_θ uses $\boldsymbol{\beta}_k$ while its paired forward kernel uses $\boldsymbol{\beta}_{k-1}$. The pair therefore shares no learnable diffusion coefficient, although the global coefficient sequence is shared across neighboring transition pairs. App. F.3 discusses the resulting forward-kernel update and trust-region term. Starting from $x^K \sim q_{\text{prior}}$, the policy applies the reverse kernels down to x^0 and executes $a = \tanh(x^0)$ in the bounded environment. The evaluation rules below distinguish stochastic transitions, ODE integration, and conditional-mean inference.

SDE, ODE, and deterministic evaluation. All three evaluation modes start from a sampled prior latent $x^K \sim q_{\text{prior}}$. With independent $\epsilon_k \sim \mathcal{N}(0, I_d)$, their updates are

$$\begin{aligned} \text{SDE}: \quad & x^{k-1} = (I + \tfrac{1}{2}\boldsymbol{\delta}_k)x^k + \nu^2\boldsymbol{\delta}_k u_\Psi(s, x^k, k) + \nu\boldsymbol{\delta}_k^{1/2}\epsilon_k, \\ \text{ODE}: \quad & x^{k-1} = (I + \tfrac{1}{2}\boldsymbol{\delta}_k)x^k + \tfrac{1}{2}\nu^2\boldsymbol{\delta}_k u_\Psi(s, x^k, k), \\ \text{Deterministic}: \quad & x^{k-1} = (I + \tfrac{1}{2}\boldsymbol{\delta}_k)x^k + \nu^2\boldsymbol{\delta}_k u_\Psi(s, x^k, k). \end{aligned} \tag{67}$$

With an exact score in continuous time, the probability-flow ODE samples from the same marginal distributions as the reverse SDE (Song et al., 2021b). Deterministic sampling instead selects the mean, which is also the mode, of each Gaussian reverse kernel (Le et al., 2026, Sec. 4.4). Following these locally most probable transitions reduces exploration and can improve evaluation return, analogous to executing the mean of a Gaussian policy. The sampled prior still permits diverse actions at a fixed state. The G1 ablation additionally fixes the prior to its mean (App. E.3.6).

Learning Gaussian diffusion coefficients. We ensure positive diffusion coefficients by setting $\beta_{k,i} = \operatorname{softplus}(b_{k,i})$ and $\delta_{k,i} = \Delta_k \beta_{k,i}$, with fixed integration increments Δ_k. Starting from the prescribed initial coefficients, we optimize $b_{k,i}$ jointly with the control network, including gradients through both reverse and forward kernels (App. F.2.7).

In the Gaussian ablation, we fix the cosine integration schedule and instead learn positive friction f_i per action coordinate, shared across diffusion steps. This adapts the effective diffusion coefficients through $\delta_{k,i} = 2\Delta_k/(\nu^2 f_i)$.

Bounded Product-Beta diffusion. To keep the entire diffusion process within the action bounds, we use Product-Beta transitions inspired by Dirichlet diffusion with Jacobi processes (Avdeyev et al., 2023). Let $u^k \in [0,1]^d$ and map the terminal sample to the environment action as $a = 2u^0 - 1$. To describe the transition kernels, recall that a Beta distribution with parameters (α, β) can equivalently be parameterized by its mean m and total concentration κ:

$$m = \frac{\alpha}{\alpha + \beta}, \qquad \kappa = \alpha + \beta, \qquad \alpha = m\kappa, \quad \beta = (1 - m)\kappa.$$

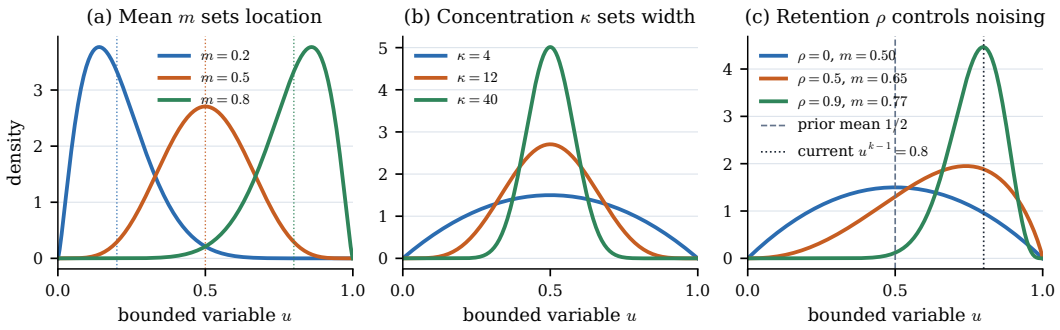


Figure 6: Product-Beta forward transitions in one dimension. (a) Varying m shifts the mean at fixed concentration. (b) Increasing κ narrows the distribution at fixed mean. (c) With $u^{k-1} = 0.8$, ρ controls the influence of the current value on the next latent. At $\rho = 0$, the transition follows the $\mathrm{Beta}(2, 2)$ reference distribution; increasing ρ moves the conditional mean toward u^{k-1} and reduces the variance.

This reparameterization introduces no additional degrees of freedom. It merely separates the two roles of (α, β): m locates the density, whereas κ controls its width, as illustrated in Fig. 6(a–b).

We use the factorized reference distribution $\mathrm{Beta}(2, 2)^{\otimes d}$, whose mean is $1/2$ and whose total concentration is 4. At diffusion step k, the forward transition is defined by

$$\pi_k(u^k \mid u^{k-1}) = \prod_{i=1}^{d} \mathrm{Beta}\big(u_i^k; m_{k,i}\kappa_k, (1 - m_{k,i})\kappa_k\big)\,,$$

$$m_{k,i} = (1 - \rho_k)\frac{1}{2} + \rho_k u_i^{k-1}, \qquad \kappa_k = \frac{4}{1 - \rho_k^2}.$$

The coefficient $\rho_k \in [0, 1)$ controls how strongly each diffusion step depends on its input. If $\rho_k = 0$, then $m_{k,i} = 1/2$ and $\kappa_k = 4$, so the transition is exactly $\mathrm{Beta}(2, 2)$ and forgets its input. As ρ_k approaches one, $m_{k,i}$ approaches u_i^{k-1} and κ_k increases, yielding an increasingly narrow transition around the input; Fig. 6(c) visualizes this effect. The particular relation $\kappa_k = 4/(1 - \rho_k^2)$ makes the first two moments of the reference distribution stationary under every forward step.

It remains to choose how quickly information is removed over the complete chain. We initialize the schedule by defining the cumulative signal $c_k^{(0)}$ and setting $\rho_k^{(0)} = c_k^{(0)}/c_{k-1}^{(0)}$:

$$c_k^{(0)} = \cos\left(\frac{\pi k}{2K}\right), \qquad \rho_k^{(0)} = \frac{c_k^{(0)}}{c_{k-1}^{(0)}}, \qquad k = 1, \ldots, K.$$

At initialization, $\prod_{j=1}^{k} \rho_j^{(0)} = c_k^{(0)}$, and therefore

$$\mathbb{E}\left[u_i^k - \frac{1}{2} \,\middle|\, u_i^0\right] = c_k^{(0)}\left(u_i^0 - \frac{1}{2}\right).$$

The initialization therefore interpolates from $c_0^{(0)} = 1$ to $c_K^{(0)} = 0$. We use $K = 8$, keep $\rho_K = 0$ so that the final forward step reaches the $\mathrm{Beta}(2, 2)$ reference distribution exactly, and learn $\rho_1, \ldots, \rho_{K-1}$ jointly with the reverse policy. The cosine construction is thus an initialization rather than a fixed schedule.

The reverse policy is likewise factorized,

$$q_\theta(u^{k-1} \mid u^k, s) = \prod_{i=1}^{d} \mathrm{Beta}\big(u_i^{k-1}; \alpha_{\theta,k,i}, \beta_{\theta,k,i}\big)\,,$$

The score network predicts corrections to the forward-kernel mean and log concentration. This construction is a discrete-time analogue of a Jacobi diffusion whose complete path remains inside the action domain. The affine map from u^0 to a contributes the constant log-Jacobian $d \log 2$; unlike the Gaussian alternatives, it requires no sample-dependent terminal correction. Both the transition entropy and the KL between old and updated reverse kernels are available in closed form.

Product-Beta trust region. For Product-Beta kernels, we likewise constrain the sum of rollout-to-current KLs over diffusion steps. At a fixed augmented state, suppress k and write $\kappa_{\theta,i} = \alpha_{\theta,i} + \beta_{\theta,i}$. By independence, the local KL is the sum of the closed-form Beta divergences (Gil et al., 2013, Table 3):

$$d_\theta(\tilde{s}) = \sum_{i=1}^{d} \Bigg[\log \frac{B(\alpha_{\theta,i}, \beta_{\theta,i})}{B(\alpha_{\theta_{\text{roll}},i}, \beta_{\theta_{\text{roll}},i})} + (\alpha_{\theta_{\text{roll}},i} - \alpha_{\theta,i})\psi(\alpha_{\theta_{\text{roll}},i}) + (\beta_{\theta_{\text{roll}},i} - \beta_{\theta,i})\psi(\beta_{\theta_{\text{roll}},i}) + (\kappa_{\theta,i} - \kappa_{\theta_{\text{roll}},i})\psi(\kappa_{\theta_{\text{roll}},i}) \Bigg], \tag{68}$$

where B is the Beta function and ψ the digamma function. We sum these local KLs over diffusion steps, or use $K d_\theta$ when sampling a phase uniformly (Eq. 34); no action sampling is needed to evaluate each local KL.

Learning $\rho_{1:K-1}$ also changes the forward kernels that define the DA objective. We therefore constrain the forward chain divergence from the rollout kernels to the updated kernels, $\sum_{k=1}^{K} D_{\text{KL}}(\pi_{\theta_{\text{roll}},k} \,\|\, \pi_{\theta,k})$, to 0.08 and optimize its Lagrange multiplier jointly with the policy. This forward trust region complements the reverse-policy chain-KL constraint above.

Product-Beta evaluation. The discrete Beta kernels support stochastic sampling and conditional-mean evaluation:

$$\begin{aligned} \text{Stochastic}: &\quad u^K \sim \text{Beta}(2,2)^{\otimes d}, \quad u_i^{k-1} \sim \text{Beta}(\alpha_{\theta,k,i}, \beta_{\theta,k,i}), \\ \text{Deterministic}: &\quad u^K \sim \text{Beta}(2,2)^{\otimes d}, \quad u_i^{k-1} = \frac{\alpha_{\theta,k,i}}{\alpha_{\theta,k,i} + \beta_{\theta,k,i}}. \end{aligned} \tag{69}$$

Both modes sample the Beta prior; deterministic sampling then replaces each reverse transition by its conditional mean. The reported Beta ablation uses the stochastic rule.

Learning Product-Beta kernel parameters. The reverse network learns both the location and noise level of each denoising transition. Let $h_{\Psi,k,i}(s,u^k)$ and $g_{\Psi,k,i}(s,u^k)$ denote its mean and log-concentration corrections, and define $\bar{m}_{k,i} = (1-\rho_k)/2 + \rho_k u_i^k$ and $\omega_k = \sqrt{1-\rho_k^2}$. Before numerical stabilization, the reverse parameters are

$$\begin{aligned} \widehat{m}_{\theta,k,i} &= \text{sigmoid}\big(\text{logit}(\bar{m}_{k,i}) + \omega_k h_{\Psi,k,i}(s,u^k)\big), \\ \widehat{\kappa}_{\theta,k,i} &= \kappa_k \exp\big(\text{clip}\big(\omega_k g_{\Psi,k,i}(s,u^k), -5, 5\big)\big), \\ \alpha_{\theta,k,i} &= \widehat{m}_{\theta,k,i}\widehat{\kappa}_{\theta,k,i}, \qquad \beta_{\theta,k,i} = (1-\widehat{m}_{\theta,k,i})\widehat{\kappa}_{\theta,k,i}. \end{aligned}$$

The sigmoid and exponential enforce a valid mean and positive concentration. Actor updates train both network outputs. To learn the forward schedule, we parameterize $\rho_k = \varepsilon + (1-2\varepsilon)\,\text{sigmoid}(r_k)$ for $k < K$, with $\varepsilon = 10^{-6}$, and fix $\rho_K = 0$ to preserve the exact prior endpoint. We initialize the unconstrained parameters r_k to reproduce the cosine schedule and optimize them jointly with the reverse network. The concentrations follow $\kappa_k = 4/(1-\rho_k^2)$, and gradients account for the coefficients' effects on both forward and reverse kernels.

Experimental setup. We compare Gaussian diffusion under latent-entropy and transformed-action entropy formulations with Product-Beta diffusion. For each family, we test three target-entropy factors in $\mathcal{H}_{\text{tar}}(c) = \mathcal{H}[q_{\text{prior}}] + cd$, where d is the action dimension. The latent-space Gaussian runs use $c \in \{-4,-5,-6\}$, the transformed-space Gaussian runs use $c \in \{-5,-6,-7\}$, and the Product-Beta runs use $c \in \{-4,-5,-6\}$. Different ranges are necessary because the two Gaussian estimators differ by the negative $\tanh$ Jacobian and because the Beta prior has a different entropy. The Gaussian prior has standard deviation 2.5. Both Gaussian variants set the environment-action clipping bound to 1.0, with no additional interior clamp. The Product-Beta runs use a $\text{Beta}(2,2)$ prior in each coordinate, initialize the forward coefficients from the cosine schedule, and learn them under a forward chain-KL bound of 0.08.

Table 2: Policy families in the action-transform ablation. The Gaussian variants differ in the entropy terms used by the actor, critic, and temperature estimator. The transformed-action formulation includes the terminal $\tanh$ log-Jacobian in all three components, whereas the latent formulation omits it.

Policy family	Executed action	Terminal log-Jacobian in actor/critic	Entropy used for temperature tuning	Target factors c
$\tanh$, latent entropy	$a = \tanh(x^0)$	Omitted	$\widehat{\mathcal{H}}_X$	$-4, -5, -6$
$\tanh$, transformed entropy	$a = \tanh(x^0)$	Included at $k = 1$	$\widehat{\mathcal{H}}_X + \widehat{\mathbb{E}}[j_{\tanh}(x^0)]$	$-5, -6, -7$
Product-Beta / Jacobi	$a = 2u^0 - 1$	Constant $d \log 2$	Bounded path-space entropy	$-4, -5, -6$

For every policy family and target factor, we train four independent seeds on PendulumSwingup, AcrobotSwingup, HopperHop, FishSwim, and AcrobotSwingupSparse. This gives nine settings per environment and 180 runs in total. All settings use DA: REPPO, eight diffusion steps, 50 million environment interactions, no action chunking, and a chain-KL bound of 0.08. For Product-Beta, the additional forward-kernel chain-KL bound is also 0.08. The score network has three hidden layers of width 256, and the critic has two hidden layers of width 512. All remaining optimization settings are shared.

We evaluate each policy every 10% of training using 128 independent evaluation episodes. Both Gaussian variants use sampled priors and ODE transitions (Eq. 67); Product-Beta uses the stochastic rule in Eq. 69. We report mean returns over four training seeds with 95% bootstrap confidence intervals.

Results. Figure 7 compares learned Product-Beta diffusion with Gaussian diffusion under latent-entropy and transformed-action entropy formulations. After 50M interactions, the learned Product-Beta policy reaches aggregate returns of 564.2 for $c = -4$ and 561.4 for $c = -6$, compared with 584.9 and 583.7 for the corresponding latent-entropy Gaussian settings. PendulumSwingup is largely saturated, while HopperHop remains variable across seeds. Over the tested target-entropy ranges, the transformed-action entropy formulation is substantially less robust than the latent-entropy formulation, particularly on FishSwim and AcrobotSwingupSparse. This comparison changes the actor, critic, and temperature update together and does not isolate the effect of temperature tuning. Overall, learned Product-Beta diffusion achieves aggregate performance close to the latent-entropy Gaussian formulation without requiring a $\tanh$ transform.

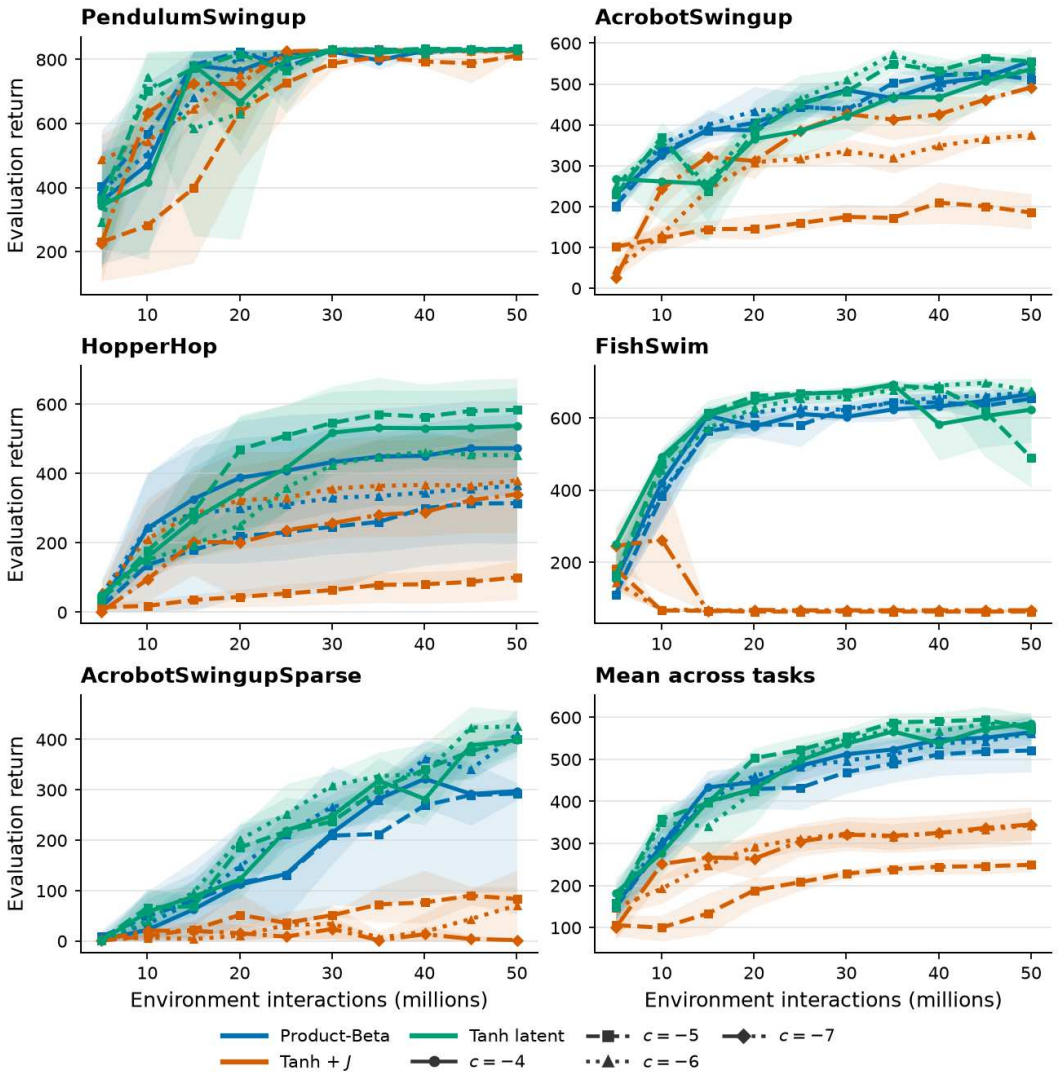


Figure 7: Action-transform and entropy-measure ablation. Curves show mean evaluation return over four training seeds, with 95% bootstrap confidence intervals. The aggregate weights the five tasks equally and bootstraps seeds within each task. Gaussian policies use ODE sampling; Product-Beta uses stochastic reverse transitions and a learned forward schedule initialized from the cosine coefficients and constrained by a forward chain-KL bound of 0.08.

E.3.3 StackCube: Task, Training, and Matched-State Evaluation

We compare final checkpoints after extended, method-specific training budgets, with success rates approaching a plateau for most runs. Gaussian REPPO reaches its success plateau substantially earlier than the DA variants, as shown in Fig. 9.

Task and randomized training states. We adapt ManiSkill3 StackCube (Tao et al., 2025) to the symmetric task of Le et al. (2026). A Panda controls seven joint increments and its gripper to stack two 4 cm cubes. Centers are sampled over $x \in [-0.2, 0.2]$, $y \in [-0.4, 0.4]$ m, rejecting overlaps, with randomized yaws and robot joint-reset noise of 0.02 rad standard deviation. Either order is valid; no target order is sampled or observed. Training resets on success or at 50 physical steps, stopping traces but bootstrapping both boundaries from the final observation.

Observations and permutation invariance. Training and evaluation share 85-dimensional states: robot state, both cube poses, tool-to-cube and directed relative/stack-goal vectors, headings, velocities, grasp flags and symmetric pair features. Permutation-invariant actor and critic encoders compute $z(s) = [r, c, \frac{1}{2}\{\phi([r, o_A]) + \phi([r, o_B])\}]$, where $r \in \mathbb{R}^{25}$ is robot state, $c \in \mathbb{R}^6$ is pair center/absolute displacement, and $o_i \in \mathbb{R}^{27}$ collects object features. The shared MLP ϕ is $52 \to 64 \to 54$ (LayerNorm, GeLU); normalization is tied across object slots. Label swapping leaves z invariant, without enforcing equally likely spatial strategies.

Reward and strict success. Let p_i denote cube centers, p_{TCP} the tool position, $g_i = p_j + (0, 0, 0.04)$ the target above the other cube, and $f(d) = 1 - \tanh(5d)$. The implemented raw dense reward is

$$r_{\mathrm{raw}} = \begin{cases} 8, & \text{strict success,} \\ 6 + (u + s)/2, & \text{cubes geometrically stacked,} \\ 4 + \max_i f(\|g_i - p_i\|), & \text{either cube grasped,} \\ 2f(\min_i \|p_{\mathrm{TCP}} - p_i\|), & \text{otherwise.} \end{cases}$$

Cases have the displayed priority. Training uses $r_{\mathrm{raw}}/8$; u is gripper opening (one if the upper cube is released), and $s = 1 - \tanh(10\|v_{\mathrm{top}}\| + \|\omega_{\mathrm{top}}\|)$. Geometric stacking requires horizontal distance ≤ 3.33 cm and height difference 4 ± 0.5 cm. Strict success also requires release, upper-cube linear speed < 1 cm/s and angular speed < 0.5 rad/s: lifting alone is insufficient.

Training configurations. Each method uses three training seeds and the final checkpoint of each run. All use 1024 environments, randomized initial episode counters, $\gamma = 0.8$, $\lambda = 0.98$, Adam, gradient clipping 0.5, and 301-bin HL-Gauss critics on $[-10, 10]$. DA actors have three residual hidden layers (LayerNorm/GeLU), critics two; $K = 8$, cosine time grid, prior standard deviation 2.5. Gaussian REPPO has a three-layer state-dependent tanh-Gaussian actor (SiLU/RMSNorm) and a 512-wide critic encoder/head. DA critic auxiliary weight is 0.05; Gaussian REPPO disables it and uses exact forward policy KL.

	Gaussian	DA:REPPO	DA:REPPO	DA:PPO	DA:WPO
Chunk size C	1	1	4	1	1
Interactions/seed (M)	50.07	100.14	250.09	300.15	250.09
Hidden width	512	512	768	512	512
Policy steps/rollout	128	128	32	128	128
Epochs/minibatches	8/64	8/64	16/16	8/64	8/64
Network LR (10^{-4})	3	3	3	3	6
KL bound / PPO clip	.08	.08	.16	.2 (clip)	.04
Entropy factor	$-.5$	-4	-2	-3	-10

DA entropy targets are $8C[\frac{1}{2}\log(2\pi e\,\sigma_{\mathrm{ref}}^2) + c]$, with $\sigma_{\mathrm{ref}} = 2.5$ for PPO and 3 otherwise; Gaussian REPPO targets -0.5 nats/action coordinate. DA:REPPO dual LRs are $2.12 \cdot 10^{-4}$, PPO temperature LR $3 \cdot 10^{-4}$, WPO dual LRs $6 \cdot 10^{-4}$. DA:REPPO/WPO use local forward-score/current-Q guidance. For $C = 4$, KL scales as $0.08\sqrt{C}$, entropy factor as $-4/\sqrt{C}$; temperature is learned. Chunk rewards, γ^m and λ^m use actual physical substeps m, with deferred resets.

Action-chunk inference timing. We benchmark the unchunked ($C = 1$, width 512) and chunked ($C = 4$, width 768) DA: REPPO architectures on a single NVIDIA A100-SXM4-40GB with $K = 8$. Using freshly initialized policies, we time single-state ODE inference in five blocks of 20 CUDA-synchronized calls after 10 warmup calls. Mean call latency is 67.07 ± 0.18 ms for $C = 1$ and 68.03 ± 0.16 ms for $C = 4$ (standard deviations across blocks). Dividing by the number of predicted actions gives 67.07 versus 17.01 ms per action, a $3.94\times$ amortized inference speedup when all four actions are executed. A control with width 768 for both policies yields a comparable $4.01\times$ speedup. These measurements concern policy computation, not physical motion duration or training time to a target success rate.

Matched-state evaluation and metrics. For every training seed and sampler, we evaluate 128 rollouts from each of the same 128 independently sampled training-reset configurations. Every rollout lasts 100 physical steps; success is checked at every substep, including within action chunks, without resetting or freezing after success. DA reverse means and probability-flow ODE resample the prior at each decision; SDE additionally samples reverse-step noise. Gaussian REPPO samples its action distribution.

For each state, count orders n_{ib}, n_{ir} at *first strict success*, with $n_i = n_{ib} + n_{ir}$. Report success $p_i = n_i/128$, entropy $H_i = -\sum_{m\in\{b,r\}} q_{im} \log_2 q_{im}$ for $q_{im} = n_{im}/n_i$, and success-weighted entropy $J_i = p_i H_i$. We first average p_i and J_i across initial states separately for each trained policy, then report the equal-weight mean and sample standard deviation across the three training seeds (Fig. 2b). These error bars describe between-seed variability, not 95% confidence intervals. In general, $\overline{J} \neq \overline{p}\,\overline{H}$. If $n_i = 0$, H_i is undefined and $J_i = 0$; the state remains in the averages of success and J. Modes are never pooled across initial states or training seeds. Full-horizon returns include failed rollouts and retain both raw and normalized units.

Spatial robustness and successful diversity. Figure 8 uses for all five methods and 512 rollouts at each of 540 valid blue-cube positions on a uniform 2.5 cm grid. Red remains at $(0, -35)$ cm; the robot pose and cube orientations are fixed, with initial end effector $(0, 0, 17.0)$ cm. Each rollout lasts 100 physical steps. These are controlled configuration slices, not IID resets, and no spatial interpolation is applied. At the original 70 cm separation, unchunked DA: REPPO succeeds in $492/512$ attempts ($pH = 0.424$ bits), whereas Gaussian REPPO succeeds in $18/512$ ($pH = 0.030$ bits). The Gaussian policy's conditional entropy is nevertheless $H = 0.852$ bits, illustrating why success must accompany a diversity measure. Distance alone does not explain the spatial pattern; the mixed-order band describes initial configurations, not a motion path.

TruDi's multimodality evaluation always executes Gaussian mean actions, while diffusion ODE sampling retains prior randomness (Le et al., 2026, p. 8). This is not a matched comparison of behavioral diversity. Fixing the diffusion prior to its mean and taking the mean at every reverse step would likewise produce a single trajectory from a fixed initial state under deterministic dynamics. Zero entropy under such execution does not establish collapse of the learned stochastic policy. We therefore evaluate REPPO using stochastic samples from its Gaussian policy. Stochastic Gaussian REPPO can still realize both orders: action noise can break symmetry and feedback sustain different trajectories. This caveat concerns conditional diversity, not return quality.

Spatial comparison across methods. The XY maps reveal configuration-dependent successful diversity. Unchunked DA: REPPO and DA: WPO exhibit extended mixed-order regions; DA: PPO solves most placements but usually favors one order. Chunked DA: REPPO also samples both orders, but its success deteriorates at large positive blue-cube y. The pH maps distinguish reliably successful diversity from balanced orders supported by only a few successful attempts.

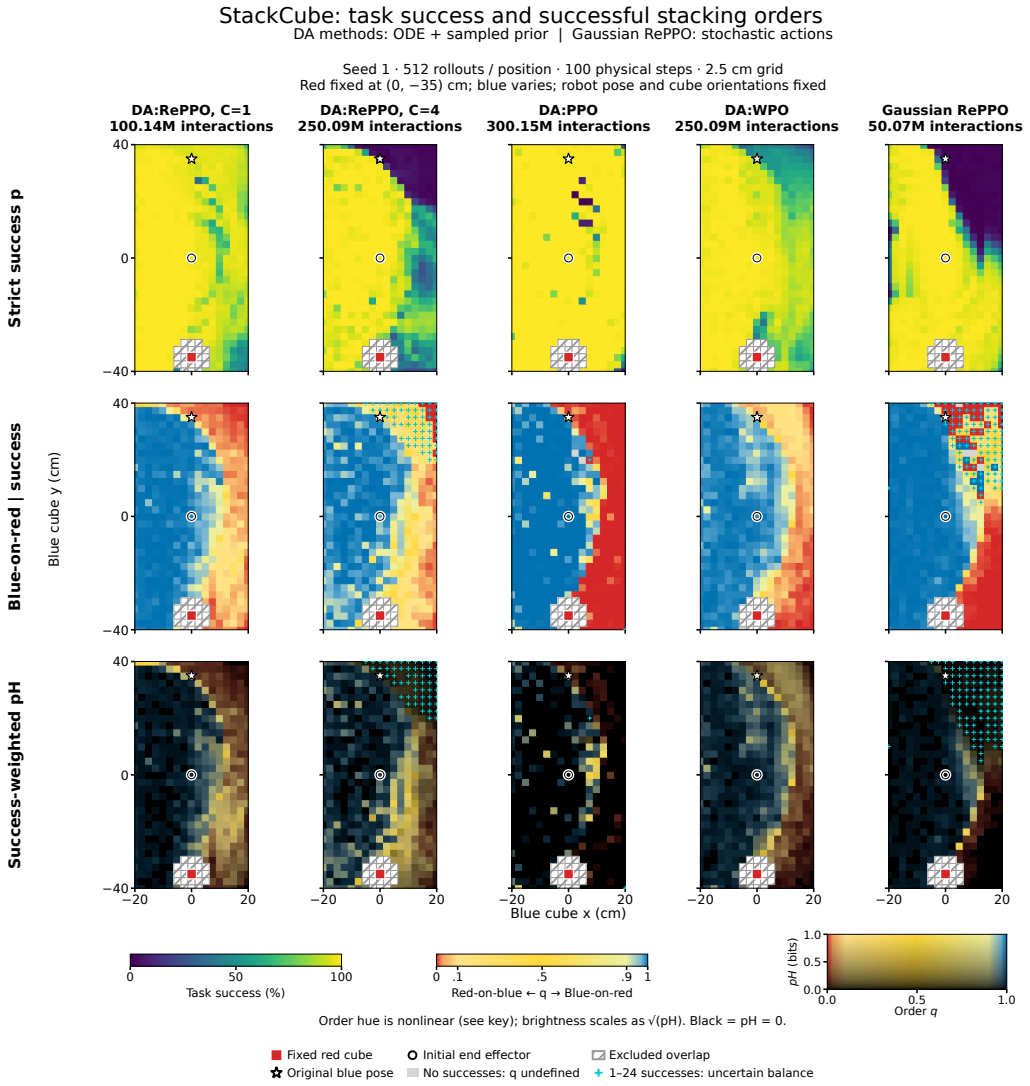


Figure 8: StackCube spatial evaluation: 540 positions, 512 rollouts per position. Rows show strict success p, blue-on-red fraction q among first strict successes, and success-weighted entropy pH. Bottom-row hue encodes q and brightness scales as $\sqrt{pH}$; black means $pH = 0$. Gray denotes no successes; cyan crosses flag fewer than 25 successes. Hatching excludes overlaps. Circles, stars, and squares mark initial end-effector, blue-cube, and red-cube positions. Budgets differ across methods.

E.3.4 PUSHT ORIENTATION-SWEEP MULTIMODALITY EXPERIMENT

Setup and training. We follow the PushT multimodality setup of Le et al. (2026). A Panda robot pushes a T-shaped block until it overlaps at least 90% of a T-shaped goal. Block and goal positions and orientations are randomized during training, and state observations include their poses and the robot state. Training episodes last at most 100 steps. The dense reward combines orientation and position alignment with an end-effector proximity bonus; success receives raw reward 3, and training uses rewards divided by 3. Success resets the episode while retaining the final observation for bootstrapping. We train three seeds per method and evaluate their final checkpoints: unchunked DA: REPPO at 250.35M interactions and Gaussian REPPO at 250.09M. Both use 1024 environments, 128 rollout steps, $\gamma = 0.9$, $\lambda = 0.98$, and eight optimization epochs. The DA policy uses eight diffusion steps (App. G.2); the Gaussian baseline uses 512-wide actor and critic networks, configured entropy target -1, and KL bound 0.1. Budgets and evaluation conditions are matched approximately; this is not an otherwise identical hyperparameter ablation.

IID training-distribution evaluation. For each of six trained policies, we generate 128 rollouts from each of 128 shared configurations drawn IID from native training resets, randomizing block and goal poses as during training. Each rollout restores the initial state and lasts 200 physical steps without resetting or freezing after success. DA: REPPO uses ODE sampling with a fresh prior per action; Gaussian REPPO samples stochastic actions. Using the first-success CW/CCW classifier defined below, we compute success p_i and successful-direction entropy H_i within each configuration. We average p_i and $p_i H_i$ over configurations separately for each policy, then report the equal-weight mean and sample SD across three training seeds. Modes are never pooled across configurations. With no successes, H_i is undefined and $p_i H_i = 0$. The evaluation comprises $98{,}304$ rollouts.

Orientation-sweep evaluation. we fix the block and goal centers at $(-0.1, -0.05)\,\mathrm{m}$, the goal yaw at zero, and the robot pose at its test value. We evaluate 512 rollouts from each of 32 initial block yaws between 30° and 330°; Fig. 2c displays 130°–260°. DA: REPPO uses ODE sampling with a fresh Gaussian prior per action; REPPO samples its Gaussian action distribution. Each rollout lasts 200 physical steps without resetting or freezing after success. We record whether task success occurs and classify direction by the sign of accumulated unwrapped yaw at the *first success*. Failed trajectories receive no mode label.

For N attempts with n_{CW} and n_{CCW} successful directional rollouts, $p = (n_{\mathrm{CW}} + n_{\mathrm{CCW}})/N$ and $q = n_{\mathrm{CW}}/(n_{\mathrm{CW}} + n_{\mathrm{CCW}})$. We plot n_{CW}/N, n_{CCW}/N, and pH, where $H = -q \log_2 q - (1 - q)\log_2(1-q)$. Thus the two directional frequencies sum to success probability, and $pH = 0$ with no successes. Probability bands are 95% Wilson intervals. For pH, we map separate 97.5% Clopper–Pearson intervals for p and q through $pH(q)$, giving a conservative 95% interval by a Bonferroni bound.

Results. At 190°, DA: REPPO records 457 CW and 51 CCW successes ($508/512$, $pH = 0.467$ bits). At 192°, the dominant direction reverses: 10 CW and 500 CCW successes ($510/512$, $pH = 0.139$ bits). The grid therefore brackets the switching region but does not resolve the exact equal-probability angle or the entropy maximum between these samples. Gaussian REPPO records only $6/512$ successes at 190° and $7/512$ at 192°, despite high success at outer angles ($511/512$ at both 130° and 260°). It has no observed successes at 155°, 160°, or 165°. Thus the failure is concentrated in a range of orientations rather than explained by uniformly poor task performance.

Earlier deterministic Gaussian REPPO control. As a separate diagnostic, the earlier 200.02M REPPO checkpoint was evaluated using mean actions, $a_t = \tanh(\mu_\theta(s_t))$, from the same fixed poses for 200 steps. It succeeded in $0/512$ executions at every tested yaw from 145° through 225°, but in $512/512$ at 135°, 240°, and 255°. These are repeated deterministic executions, not independent policy samples; we assign no policy-sampling confidence intervals. This historical control does not evaluate the final 250.09M checkpoint used in Fig. 2c.

E.3.5 Manipulation Training Curves and Evaluation Checkpoints

Figure 9 reports all three training seeds for each manipulation configuration. We evaluate the final checkpoint at the stated budget, not a checkpoint selected by held-out success or mode entropy. StackCube budgets are 50.07M interactions for Gaussian REPPO, 100.14M for unchunked DA: REPPO, 250.09M for chunked DA: REPPO and DA: WPO, and 300.15M for DA: PPO; both PushT methods receive approximately 250M. These method-specific budgets assess performance after extended training, rather than equal-budget sample efficiency.

Most success curves approach a plateau, but training ends at a prescribed budget rather than a convergence-triggered stopping rule. Comparing the final two non-overlapping 10M-interaction windows, PushT gains range from 0.18 to 0.50 percentage points for DA: REPPO and 0.12 to 0.72 for Gaussian REPPO. Some StackCube seeds continue to improve: DA: PPO seed 2 gains 2.12 points and chunked DA: REPPO seed 3 gains 1.46 points. These diagnostics concern success on randomized training episodes; they establish neither convergence of mode entropy nor uniform reliability on fixed-state slices.

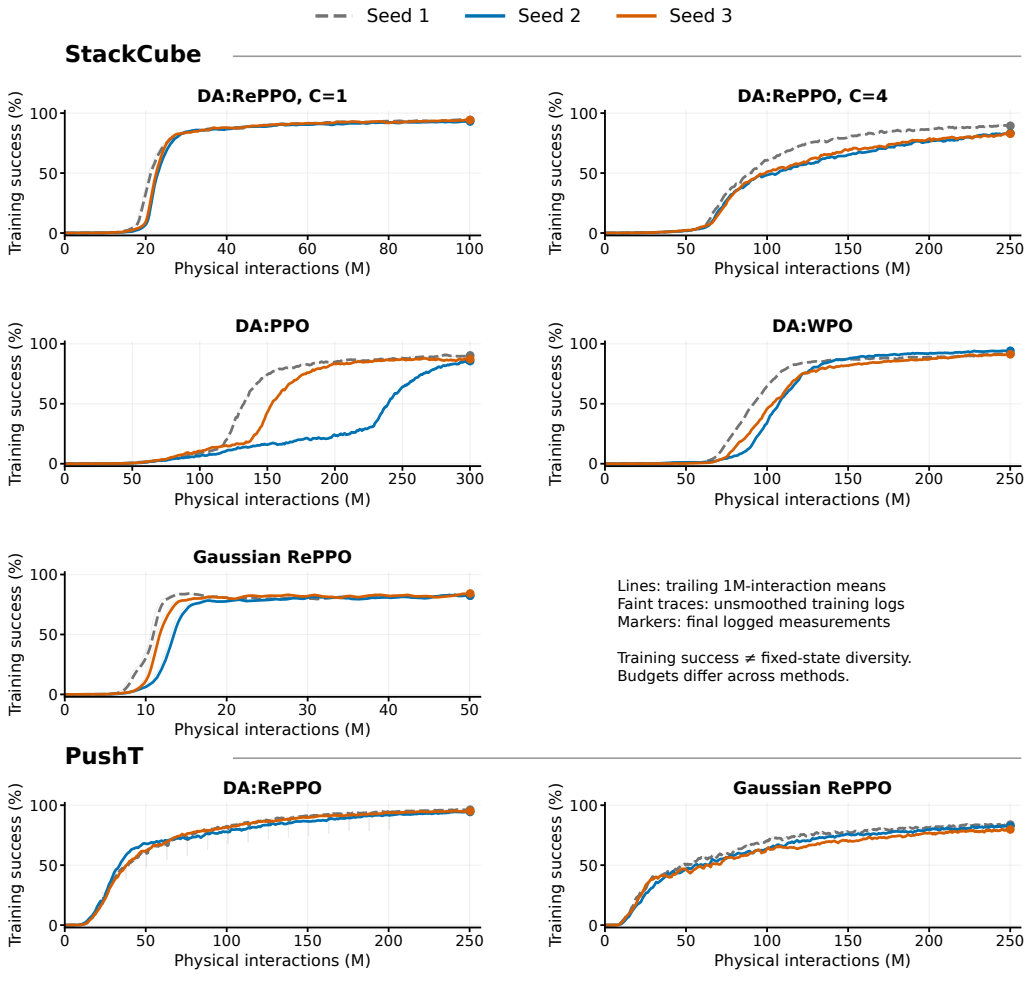


Figure 9: Training success for all manipulation runs, grouped by task: StackCube above and PushT below. Lines show trailing 1M-interaction means; faint traces show raw logs and markers identify final measurements. Original and resumed stages share a physical-interaction axis. Curves use randomized training starts and are not fixed-state multimodality evaluations. All three seeds are shown separately; panel budgets differ.

E.3.6 Diffusion-Step Ablation Details

Training protocol. We vary the diffusion depth $K \in \{2, 4, 8\}$ on G1JoystickFlatTerrain. DA: REPPO uses seven training seeds and 50.07M environment interactions per run; DA: PPO uses eight seeds and 200.02M interactions. Within each method, architectures, hyperparameters, and optimizer-update counts are fixed across depths, with all diffusion transitions included. Both use 1024 environments, $\gamma = 0.98$ and a Gaussian prior standard deviation of 2.5; domain randomization and external pushes are disabled. The environment-action clipping bound is 1.0, with no additional interior clamp. The DA: PPO ablation uses the clipped-forward PPO variant, which restricts the forward correction to the PPO active set. Budgets and optimization settings differ between methods, so these are within-method depth comparisons. DA: WPO uses six seeds and 128-step rollouts, with $\lambda = 0.98$ and the same environment count, discount, prior, and randomization settings.

Deterministic checkpoint evaluation. We initialize at the Gaussian prior mean and apply the conditional mean of each reverse Gaussian kernel. Evaluations occur approximately every 1M interactions and record the first episode from each of 128 environments, ending on failure or at 1000 steps. Returns sum environment rewards without entropy bonuses. Figure 10 shows the evaluation learning curves separately for each method, with a shared return scale and method-specific interaction ranges. For DA: REPPO and DA: PPO, the curves include the initial policy, periodic evaluations, and dedicated final-checkpoint evaluations. The DA: WPO curves use periodic evaluations within the 0–70M display window, whose last common logged checkpoint is 69.07M. All six WPO seeds contribute equally, with evaluation times aligned by linear interpolation within their observed ranges. Figure 3b summarizes the returns at these reported checkpoints. Results report the mean and sample standard deviation across all training seeds, without temporal smoothing.

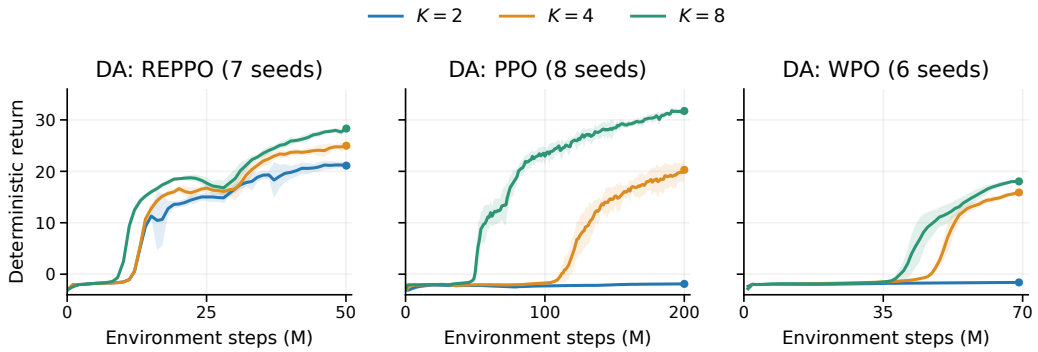


Figure 10: G1 diffusion-step ablation: deterministic evaluation learning curves for DA: REPPO, DA: PPO, and DA: WPO (left to right), using seven, eight, and six training seeds. Colors indicate K; bands show ± 1 sample standard deviation, without temporal smoothing. Panels share a return scale and use method-specific interaction ranges. Endpoint markers correspond to the table in Fig. 3b: 50.07M, 200.02M, and 69.07M interactions. WPO is displayed through 70M. Evaluation follows prior and reverse-kernel means and excludes entropy bonuses.

E.3.7 Multimodal Agent Environment Details

We provide the full specification of the environment used in Sec. 5.

State and Action Space. The state is an orientation on the circle, represented by $\mathcal{S} = \{0°, 45°, 90°, \ldots, 315°\}$, with angles identified modulo 360°. The action $a_t \in [-1, 1]$ represents a relative movement direction, mapped to angles in $[-90°, 90°]$.

Reward Function. For $a \in [-1, 1]$, the state-independent reward is

$$R_{\text{env}}(s, a) = \begin{cases} 1 - 0.15(1 - 4a^2)^2, & |a| \le \frac{1}{2}, \\ 1 - \left(\dfrac{4a^2 - 1}{3}\right)^2, & \frac{1}{2} < |a| \le 1. \end{cases}$$

It has two symmetric maxima at $a = \pm\frac{1}{2}$, corresponding to $\pm 45°$. The positive-temperature maximum-entropy optimum is bimodal; reward maximization alone also admits policies that select only one mode.

Transition Dynamics. After an action is taken, the environment maps it to the nearest mode of the reward landscape, while the reward is evaluated at the original (continuous) action:

$$s_{t+1} = \begin{cases} (s_t + 45°) \bmod 360° & \text{if } a_t > 0 \\ (s_t - 45°) \bmod 360° & \text{if } a_t \leq 0 \end{cases}$$

Discrete State Space. The modulo convention closes the dynamics on these eight states; for example, $315° + 45° \equiv 0°$ and $0° - 45° \equiv 315°$ modulo $360°$. This finite circular state space allows explicit visualization of the action distributions conditioned on each state.

Evaluating Multimodality. This benchmark illustrates why reward and state entropy are insufficient for evaluating multimodal policies. Selecting only one of two equally good actions can achieve the optimal reward, and DPPO attains high state entropy by cycling through the discrete states while remaining unimodal in its action distribution (Fig. 11). The entropy of state–action trajectories would provide a more informative measure of multimodality, although it is difficult to estimate reliably in high-dimensional control problems.

E.3.8 DPI Gap for the KL Divergence

App. F.2.1 proves that the joint KL over $(a^{0:K}_{0:T}, s_{0:T+1})$ decomposes exactly into the marginal KL over the executed actions $a_{0:T}$ and two nonnegative conditional KL terms. We restate the decomposition here to define the quantities used in the numerical evaluation.

The resulting identity is

$$\begin{aligned} &D_{\mathrm{KL}}\big(q_\theta(a^{0:K}_{0:T}, s_{0:T+1}) \big\| \pi_\theta(a^{0:K}_{0:T}, s_{0:T+1})\big) \\ &\quad = D_{\mathrm{KL}}(q_\theta(a_{0:T}) \| \pi(a_{0:T})) + \Delta_{\text{state}} + \Delta_{\text{diffusion}}. \end{aligned} \tag{70}$$

The two terms are

$$\begin{aligned} \Delta_{\text{state}} &:= \mathbb{E}_{q_\theta(a_{0:T})}[D_{\mathrm{KL}}(q_\theta(s_{0:T+1} \mid a_{0:T}) \| \pi(s_{0:T+1} \mid a_{0:T}))]\,, \\ \Delta_{\text{diffusion}} &:= \mathbb{E}_{q_\theta(a_{0:T}, s_{0:T+1})}\big[D_{\mathrm{KL}}\big(q_\theta(a^{1:K}_{0:T} \mid a_{0:T}, s_{0:T+1}) \big\| \pi_\theta(a^{1:K}_{0:T} \mid a_{0:T}, s_{0:T+1})\big)\big]\,. \end{aligned}$$

Thus, Δ_{state} measures the discrepancy between the conditional state trajectories after fixing $a_{0:T}$, while $\Delta_{\text{diffusion}}$ measures the remaining discrepancy between the conditional intermediate diffusion variables after additionally fixing $s_{0:T+1}$. Their sum is exactly the DPI gap, and both terms are nonnegative. The joint bound is tight if and only if, almost surely under the corresponding q_θ marginals,

$$q_\theta(s_{0:T+1} \mid a_{0:T}) = \pi(s_{0:T+1} \mid a_{0:T}), \qquad q_\theta(a^{1:K}_{0:T} \mid a_{0:T}, s_{0:T+1}) = \pi_\theta(a^{1:K}_{0:T} \mid a_{0:T}, s_{0:T+1}). \tag{71}$$

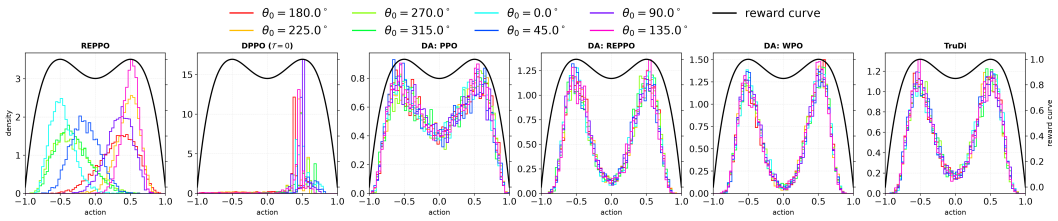
Consequently, two policies with the same executed-action marginal can have different joint KLs because their conditional state or intermediate diffusion distributions differ. If $q_\theta(a^{0:K}_{0:T}, s_{0:T+1}) = \pi_\theta(a^{0:K}_{0:T}, s_{0:T+1})$, the joint KL is zero. Marginalizing over the states and intermediate diffusion variables then yields $q_\theta(a_{0:T}) = \pi(a_{0:T})$. The finite-horizon DPI bound and the stationary discounted local-gradient identity are

$$D_{\mathrm{KL}}(q_\theta(a_{0:T}) \| \pi(a_{0:T})) \leq D_{\mathrm{KL}}\big(q_\theta(a^{0:K}_{0:T}, s_{0:T+1}) \big\| \pi_\theta(a^{0:K}_{0:T}, s_{0:T+1})\big)\,, \tag{72}$$

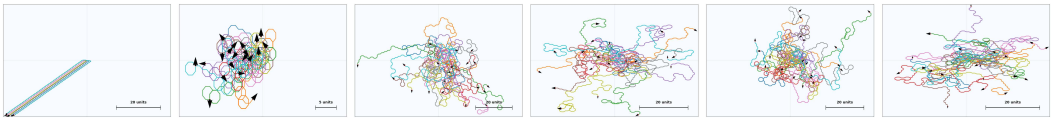
$$\nabla_\theta \mathcal{L}_{\mathrm{DA}}(\theta; \theta_{\text{roll}})|_{\theta=\theta_{\text{roll}}} = -\nabla_\theta \mathcal{J}_{\mathrm{DA}}(\theta)|_{\theta=\theta_{\text{roll}}}\,. \tag{73}$$

E.3.9 Numerical Quantification on the Multimodal Agent Benchmark

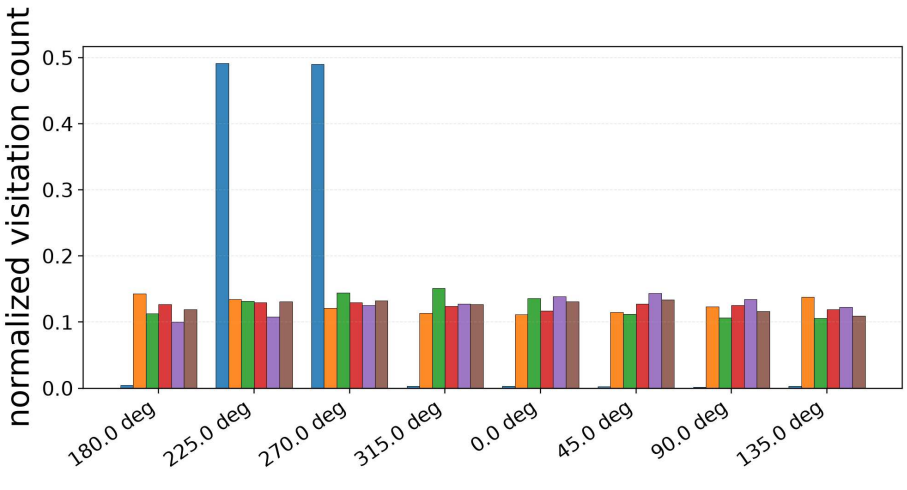
We quantify every term of Eq. 70 on the Multimodal Agent benchmark using one trained policy per method, horizon $H = 100$, $K = 8$, and $\mathcal{T} = 0.1$. We use numerical integration to compute the target partition function and the per-state executed-action density $q_\theta(a^0_t \mid s_t)$, marginalizing over all eight continuous intermediate variables. We estimate the state gap directly from the posterior



(a) Action distributions and bimodal reward landscape.



(b) Resulting agent trajectories.



(c) State visitation histogram of each method on the Multimodal Agent Benchmark.

Figure 11: Extended Results on the Multimodal Agent benchmark, which show that DA: WPO and TruDi are also able to cover the multimodal action distribution. Panel (c) complements the action and trajectory visualizations with normalized state-visitation counts; it illustrates why high state entropy alone does not imply multimodal action coverage.

over the eight initial headings using $262{,}144$ sampled action sequences. We then obtain the action-sequence KL by subtracting this gap from the numerically integrated state–action KL. The joint KL over $(a_{0:T}^{0:K}, s_{0:T+1})$ is evaluated from the tractable local log-density ratios; for REPPO, which has no intermediate diffusion variables, the analogous quantity is its state–action trajectory KL. All entries below are unscaled KLs in nats, and $\pm$ denotes Monte Carlo standard error for the fixed policy, conditional on the numerical density estimates. REPPO's Monte Carlo error is negligible at the reported precision; numerical integration error is not included.

Table 3: Numerical estimates of the DPI-gap decomposition on the Multimodal Agent benchmark. The trajectory KL in the final column is method-specific because only DA-MDP contains auxiliary diffusion variables. These stochastic checkpoints achieve mean per-step environment rewards of 0.9981 (REPPO) and 0.9572 (DA: REPPO).

Method	Marginal action-sequence KL ↓	State gap ↓	Diffusion gap ↓	Method-specific trajectory KL ↓
REPPO	927.01125	2.07944	N/A	929.091
DA: REPPO	3.74249 ± 0.00012	0.06882 ± 0.00012	284.603 ± 0.104	288.415 ± 0.104

For DA: REPPO, the numerical identity is therefore $288.415 \approx 3.742 + 0.069 + 284.603$ up to rounding. The action marginal closely matches the target even though the auxiliary diffusion conditional makes the augmented bound loose. This is the standard extended-objective behavior: tightness of the path-space variational bound alone does not determine the quality of the terminal marginal, and a tighter path bound need not produce better marginal samples. In particular, the final column of Table 3 does not compare the same divergence across the two methods.

Numerical integration gives $\log Z_H = 985.152$. DA: REPPO's action-sequence and augmented-trajectory ELBOs are therefore 981.410 and 696.737, respectively. Directly estimating this partition function from $131{,}072$ augmented-trajectory importance samples yields an effective sample size of only 1.24, illustrating the severe weight degeneracy even in this one-dimensional task.

In a general continuous-control task, evaluating the executed-action marginal requires

$$q_\theta(a_{0:T}) = \int q_\theta(a_{0:T}^{0:K}, s_{0:T+1})\, da_{0:T}^{1:K}\, ds_{0:T+1}. \tag{74}$$

This integral contains the complete continuous state trajectory and HK auxiliary diffusion variables; its dimension grows with horizon, state and action dimension, and diffusion depth, while the target normalizer is also unknown. The exact marginal action-sequence KL and DPI gap are therefore computationally intractable on our high-dimensional, long-horizon control benchmarks. The augmented-trajectory KL remains tractable precisely because its log density decomposes into local Markov transition terms.

E.3.10 Numerical Verification of the Surrogate Loss

We numerically verify that, at $\theta = \theta_{\text{roll}}$, the gradients of the local surrogate and the augmented DA-MDP objective are equal in expectation, including when the forward diffusion kernels are learned. Specifically, we test

$$\nabla_\theta \mathcal{L}_{\text{DA}}(\theta; \theta_{\text{roll}})|_{\theta=\theta_{\text{roll}}} = -\nabla_\theta \mathcal{J}_{\text{DA}}(\theta)|_{\theta=\theta_{\text{roll}}}. \tag{75}$$

Exact enumeration setup. We verify Eq. 75 on an exactly enumerable finite DA-MDP. The task has three environment decisions ($T = 2$), three reverse-diffusion steps ($K = 3$), two environment states, five latent-action grid points, action-dependent transitions, a tilted double-well reward, and temperature $\mathcal{T} = 0.7$. Both a small reverse model and one diffusion coefficient per step are learned. We use the VP-SDE forward process, discretized on the latent-action grid (App. E.3.2). We can therefore compute the rollout occupancies, soft Q-values, and expected gradients by enumerating all trajectories in 64-bit arithmetic.

At $\theta = \theta_{\text{roll}}$, we compare the gradient of the augmented-trajectory objective with (i) the local surrogate in Eq. 10 and (ii) an intentionally incorrect loss that omits the explicit forward-kernel gradient.

Table 4: Comparison of expected gradients at $\theta = \theta_{\mathrm{roll}}$. Relative error is $\|g - g_{\mathrm{traj}}\|_2 / \|g_{\mathrm{traj}}\|_2$.

Comparison	Parameter block	Cosine	Relative error
Local surrogate	Joint	1.0000	1.16×10^{-16}
Local surrogate	Diffusion coefficients	1.0000	2.21×10^{-16}
Forward gradient omitted	Diffusion coefficients	0.2065	1.054

The correct surrogate and trajectory gradients thus agree to machine precision, whereas omitting the explicit $-\log \pi_{\beta_k}$ gradient produces a relative error greater than one. Independent Monte Carlo gradient estimates converge to the values obtained by exact enumeration: with $16{,}384$ samples, the median cosine with the exact gradient is 0.999244 for the trajectory estimator and 0.999775 for the local surrogate. Finally, at $\|\theta - \theta_{\mathrm{roll}}\|_2 = 0.4$, the median relative error of the stale surrogate is 9.74%, while recomputing the occupancy and Q-function at the updated parameters and resetting $\theta_{\mathrm{roll}} = \theta$ restores agreement to 1.75×10^{-16}. Thus, the local surrogate is exact when $\theta = \theta_{\mathrm{roll}}$, while reusing a stale rollout makes it only a local approximation.

Rationale for the Unnormalized Local Target. For completeness, with $u = a_t^{k-1}$, the learned-forward local target and the corresponding unnormalized KL are

$$\begin{aligned} \widetilde{p}_{\theta;\theta_{\mathrm{roll}}}(u \mid \tilde{s}) &:= \pi_\theta(a_t^k \mid u, s_t) \exp\Big(\alpha Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s}, u)\Big), \\ D_{\mathrm{KL}}^{\mathrm{u}}(q \| \widetilde{p}) &:= \int q(u) \log \frac{q(u)}{\widetilde{p}(u)}\, du. \end{aligned} \tag{76}$$

Normalizing $\widetilde{p}_{\theta;\theta_{\mathrm{roll}}}$ would introduce a parameter-dependent local partition function whose gradient is absent from the augmented-trajectory objective. In the exact enumeration above, differentiating through this normalizer changes the diffusion-coefficient gradient by 9.18%, whereas Eq. 76 agrees with the trajectory gradient to machine precision. We therefore leave the local target unnormalized.

E.4 Runtime Comparison

E.4.1 Controlled Active-Memory Comparison

We compare DA: REPPO and TruDi under a controlled CheetahRun configuration on a single NVIDIA H100 GPU. Both methods use FP32, matched 4×512 policy networks, width-1,024 critics, $K = 16$, and four samples for the trust-region KL estimate. Each timing covers a complete training block of 131,072 environment transitions. We exclude compilation, discard one warm-up block, and report the median runtime over the next three blocks. Figure 3c plots this runtime against the maximum active memory reported by JAX.

TruDi backpropagates through complete diffusion paths, storing intermediate values from every diffusion step. DA: REPPO computes gradients for individual diffusion transitions, requiring less memory per training sample. For either method, increasing the minibatch size reduces the number of sequential optimizer updates needed to process the training data. Lower memory use does not automatically imply faster training. For a fixed number of environment interactions, runtime depends on training throughput, which is influenced by the computation per sample, minibatch size, and GPU utilization. Available memory constrains which minibatch sizes can be used.

At approximately 1.9 GiB of active memory, DA: REPPO requires 6.331 s per block and TruDi requires 6.730 s. As the memory budget increases, TruDi can process more full-chain samples per minibatch and therefore perform fewer optimizer updates: its runtime falls to 5.289 s at 4.123 GiB and 4.388 s at 16.063 GiB. DA: REPPO trains faster under the tighter memory budgets tested here, while TruDi becomes faster when additional memory permits larger minibatches.

E.4.2 Controlled NVML Process-Memory Comparison

Section 5 reports active JAX memory to expose the working-set scaling of the two algorithms. Here we additionally report the operational GPU-process footprint observed by NVML. We use the same controlled single-H100 protocol: FP32, four trust-region KL samples, 131,072 environment transitions per block, one warm-up block, and the median runtime of three complete measured blocks,

with compilation excluded. JAX's automatic GPU-memory reservation is disabled. The reported NVML memory is the maximum process allocation observed across the three measured blocks and includes JAX allocator pools, the CUDA context, compiled code, and other process-level GPU allocations.

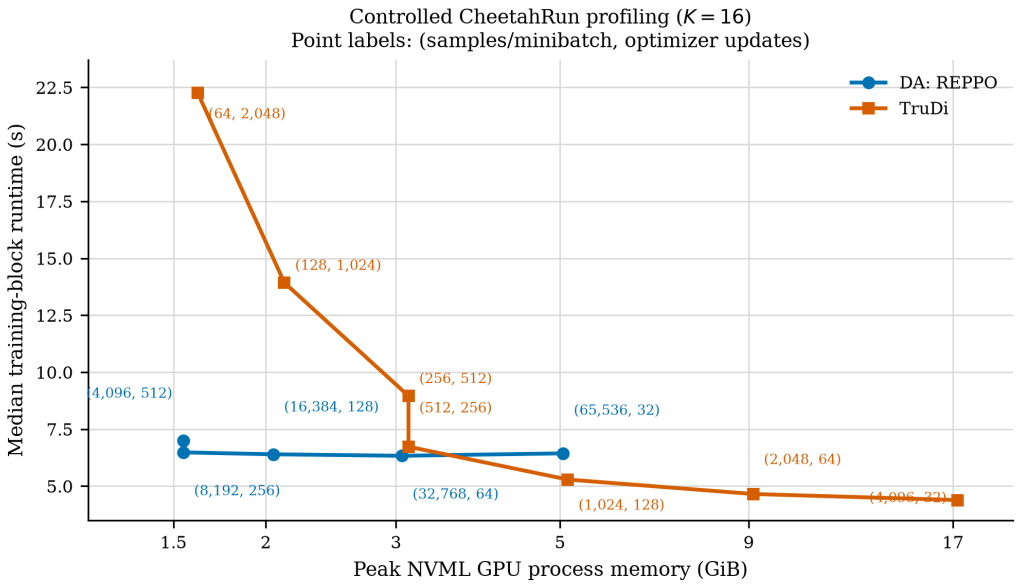


Figure 12: Controlled NVML process-memory–runtime configuration sweep for the complete $K = 16$, 4×512-policy sweep. Point labels denote (samples per minibatch, optimizer updates). Every measured configuration is shown with a filled marker, and lines connect configurations in order of increasing samples per minibatch.

Table 5: Controlled NVML process-memory–runtime comparison of DA: REPPO and TruDi. DA-MDP minibatches contain diffusion-MDP samples, whereas TruDi minibatches contain environment samples. "Updates" denotes the number of minibatch optimizer steps required to process one epoch corresponding to 131,072 environment transitions.

(a) $K = 8$**, matched** 4×256 **policies**

Method	Minibatches	Samples/minibatch	NVML peak memory	Runtime	Relative result
DA: REPPO	32	32,768	3.047 GiB	3.362 s	5.74× less memory
TruDi	16	8,192	17.502 GiB	1.646 s	2.04× faster

(b) $K = 16$**, matched** 4×512 **policies and approximately matched NVML memory**

DA samples/ minibatch	DA updates	DA NVML memory	DA runtime	TruDi samples/ minibatch	TruDi updates	TruDi NVML memory	TruDi runtime	Result
8,192	256	1.547 GiB	6.481 s	64	2,048	1.617 GiB	22.271 s	DA 3.44× faster
16,384	128	2.047 GiB	6.397 s	128	1,024	2.119 GiB	13.928 s	DA 2.18× faster
32,768	64	3.055 GiB	6.331 s	512	256	3.119 GiB	6.730 s	DA 1.06× faster
65,536	32	5.047 GiB	6.437 s	1,024	128	5.121 GiB	5.289 s	TruDi 1.22× faster

The NVML results support the same bounded conclusion as the active-memory comparison. At $K = 8$, DA: REPPO uses 5.74× less NVML peak process memory, while TruDi is 2.04× faster. At $K = 16$ and approximately matched NVML budgets near 1.6, 2.1, and 3.1 GiB, DA-MDP is respectively 3.44×, 2.18×, and 1.06× faster; near 5.1 GiB, TruDi is 1.22× faster. In the matched $K = 16$ comparison, DA: REPPO achieves shorter runtimes at lower memory budgets, whereas TruDi is faster when more memory is available.

NVML and JAX memory need not vary smoothly together because NVML includes retained allocator pools. For example, the TruDi configurations $(256, 512)$ and $(512, 256)$ both occupy 3.119

GiB according to NVML, although their active JAX peaks are 1.152 and 1.913 GiB, respectively. The smaller-minibatch configuration performs twice as many updates and is slower (8.958 versus 6.730 s). Consequently, the connected TruDi sweep contains a vertical segment at 3.119 GiB in Fig. 12, whereas Fig. 3c separates the two active working sets. Reporting both metrics distinguishes the algorithmic working set from the whole-process GPU footprint.

E.4.3 Further Directions for Future Work

Another key consideration is optimization stability, particularly when using the HL-Gauss loss (Farebrother et al., 2024) in DA: REPPO or DA: WPO with automatic temperature tuning. If the temperature overshoots, the log-ratios, which are scaled by the temperature, can grow excessively large and unbounded. This can push Q-targets outside the fixed binning range of the HL loss, resulting in numerical instability. While reducing the learning rate for the Lagrange multipliers can partially mitigate this issue, we see this as an important area for future research. We also found that the minibatch size significantly influences the learned temperature and KL regularization schedule and can make learning unstable, primarily due to noisy updates in the Lagrange multipliers. Smoothing techniques could potentially enhance robustness in this context. Furthermore, in Eq. 10, the diffusion time step is currently sampled uniformly. An alternative approach could involve sampling later diffusion steps more frequently using importance sampling, a technique commonly employed in data-based diffusion models (Song et al., 2021a), which may further improve learning efficiency.

Two architectural extensions could reduce the current computational overhead. First, separate state and noisy-action encoders would allow the state representation to be computed once and reused throughout a diffusion chain. Second, a learned state-dependent prior could initialize the chain closer to useful actions and thereby reduce the required transport. More broadly, the DA-MDP construction is not specific to continuous control: analogous augmented processes could be defined for discrete or structured action spaces, including combinatorial optimization (Karalias & Loukas, 2020) and RL fine-tuning of diffusion language models (Ouyang et al., 2022; Nie et al., 2025).

F Derivations

This appendix derives the theoretical identities used by the objectives and algorithms in the main paper. It proceeds from the standard maximum-entropy trajectory KL to its DA-MDP counterpart, establishes the monotonic-improvement result for the DA-MDP surrogate loss, and derives the ME-WPO and DA: WPO updates.

F.1 Standard ME-RL Trajectory KL

F.1.1 Finite-Horizon Identity

We first show in the finite-horizon setting that minimizing the KL divergence between the policy and target distributions over joint state–action trajectories is equivalent to maximizing the corresponding maximum-entropy RL objective. We then derive the local surrogate loss whose gradient matches the objective gradient at the rollout policy.

For a finite horizon T, recall the policy trajectory distribution and the unnormalized reward target

$$q_\theta(a_{0:T}, s_{0:T+1}) = p(s_0) \prod_{t=0}^{T} q_\theta(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t), \tag{77}$$

$$\widetilde{\pi}(a_{0:T}, s_{0:T+1}) = p(s_0) \prod_{t=0}^{T} p(s_{t+1} \mid s_t, a_t) \exp\left(\frac{R_{\text{env}}(s_t, a_t)}{\mathcal{T}}\right), \qquad \pi := \frac{\widetilde{\pi}}{\mathcal{Z}}. \tag{78}$$

The initial-state distribution and environment dynamics are identical in the two trajectories and therefore cancel inside the log ratio. Hence

$$\log \frac{q_\theta(a_{0:T}, s_{0:T+1})}{\pi(a_{0:T}, s_{0:T+1})} = \sum_{t=0}^{T} \left[\log q_\theta(a_t \mid s_t) - \frac{R_{\text{env}}(s_t, a_t)}{\mathcal{T}}\right] + \log \mathcal{Z}. \tag{79}$$

Taking the expectation under q_θ gives

$$\mathcal{T} D_{\mathrm{KL}}(q_\theta(a_{0:T}, s_{0:T+1}) \,\|\, \pi(a_{0:T}, s_{0:T+1})) = -\mathcal{J}_T(\theta) + \mathcal{T} \log \mathcal{Z}, \tag{80}$$

$$\mathcal{J}_T(\theta) := \mathbb{E}_{q_\theta}\left[\sum_{t=0}^{T} \left(R_{\mathrm{env}}(s_t, a_t) - \mathcal{T} \log q_\theta(a_t \mid s_t)\right)\right]. \tag{81}$$

Thus, minimizing the joint trajectory KL is exactly equivalent to maximizing the finite-horizon maximum-entropy return, apart from the constant $\mathcal{T} \log \mathcal{Z}$.

For completeness, define the finite-horizon soft value functions by

$$Q^{q_\theta}(s_t, a_t) = R_{\mathrm{env}}(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim p(\cdot \mid s_t, a_t)}[V^{q_\theta}(s_{t+1})], \tag{82}$$

$$V^{q_\theta}(s_t) = \mathbb{E}_{a_t \sim q_\theta(\cdot \mid s_t)}[Q^{q_\theta}(s_t, a_t) - \mathcal{T} \log q_\theta(a_t \mid s_t)], \qquad V^{q_\theta}(s_{T+1}) = 0. \tag{83}$$

Although the entropy term depends explicitly on θ, its direct derivative vanishes conditionally:

$$\mathbb{E}_{a_t \sim q_\theta(\cdot \mid s_t)}[\nabla_\theta \log q_\theta(a_t \mid s_t)] = \nabla_\theta \int q_\theta(a_t \mid s_t)\, da_t = 0. \tag{84}$$

This cancellation removes only the explicit derivative of the entropy term. The policy still changes the distribution of future states, and the causal likelihood-ratio expansion collects that dependence in Q^{q_θ}. Applying the policy-gradient theorem therefore yields

$$\begin{aligned} &\nabla_\theta[\mathcal{T} D_{\mathrm{KL}}(q_\theta(a_{0:T}, s_{0:T+1}) \,\|\, \pi(a_{0:T}, s_{0:T+1}))] \\ &\quad = \sum_{t=0}^{T} \mathbb{E}_{q_\theta}[(\mathcal{T} \log q_\theta(a_t \mid s_t) - Q^{q_\theta}(s_t, a_t))\, \nabla_\theta \log q_\theta(a_t \mid s_t)]. \end{aligned} \tag{85}$$

F.1.2 Reverse Log-Derivative Lemma and Actor Surrogate

The policy-gradient integrand in Eq. 85 is itself the gradient of a per-state reverse KL. This reverse application of the usual log-derivative trick produces the actor loss optimized in practice.

Lemma F.1 (Reverse log-derivative trick)**.** *Fix a time t and state s, freeze the rollout critic $Q^{q_{\theta_{\mathrm{roll}}}}(s, a)$, and assume that*

$$\pi_{Q,t}^{q_{\theta_{\mathrm{roll}}}}(a \mid s) := \frac{\exp(Q^{q_{\theta_{\mathrm{roll}}}}(s, a)/\mathcal{T})}{Z_{Q,t}^{q_{\theta_{\mathrm{roll}}}}(s)}, \qquad \nabla_\theta Q^{q_{\theta_{\mathrm{roll}}}}(s, a) = 0, \tag{86}$$

is normalizable. Then

$$\begin{aligned} &\nabla_\theta\left[\mathcal{T} D_{\mathrm{KL}}\left(q_\theta(\cdot \mid s) \,\middle\|\, \pi_{Q,t}^{q_{\theta_{\mathrm{roll}}}}(\cdot \mid s)\right)\right] \\ &\quad = \mathbb{E}_{a \sim q_\theta(\cdot \mid s)}[(\mathcal{T} \log q_\theta(a \mid s) - Q^{q_{\theta_{\mathrm{roll}}}}(s, a))\, \nabla_\theta \log q_\theta(a \mid s)]. \end{aligned} \tag{87}$$

Proof. Expanding the KL gives the expectation of $\mathcal{T} \log q_\theta(a \mid s) - Q^{q_{\theta_{\mathrm{roll}}}}(s, a)$ plus the parameter-independent term $\mathcal{T} \log Z_{Q,t}^{q_{\theta_{\mathrm{roll}}}}(s)$. Differentiating this expectation produces the right-hand side of Eq. 87 and the additional term $\mathcal{T}\mathbb{E}_{q_\theta}[\nabla_\theta \log q_\theta(a \mid s)]$, which vanishes by Eq. 84. □

Applying Lemma F.1 at every time step while freezing the rollout state distributions and critic gives the finite-horizon actor surrogate

$$\mathcal{L}_T^{\mathrm{ME}}(\theta; \theta_{\mathrm{roll}}) := \mathcal{T} \sum_{t=0}^{T} \mathbb{E}_{s_t \sim q_{\theta_{\mathrm{roll}}}}\left[D_{\mathrm{KL}}\left(q_\theta(\cdot \mid s_t) \,\middle\|\, \pi_{Q,t}^{q_{\theta_{\mathrm{roll}}}}(\cdot \mid s_t)\right)\right]. \tag{88}$$

At the rollout parameters, its gradient is exactly the trajectory-KL gradient, or equivalently the negative return gradient:

$$\begin{aligned} \nabla_\theta \mathcal{L}_T^{\mathrm{ME}}(\theta; \theta_{\mathrm{roll}})\big|_{\theta=\theta_{\mathrm{roll}}} &= \nabla_\theta[\mathcal{T} D_{\mathrm{KL}}(q_\theta(a_{0:T}, s_{0:T+1}) \,\|\, \pi(a_{0:T}, s_{0:T+1}))]|_{\theta=\theta_{\mathrm{roll}}} \\ &= -\nabla_\theta \mathcal{J}_T(\theta)|_{\theta=\theta_{\mathrm{roll}}}. \end{aligned} \tag{89}$$

This equality is exact only at the first actor iterate, because subsequent updates continue to use the rollout-frozen state distributions and critic. TRPO motivates optimizing the surrogate for multiple updates subject to a rollout-to-current policy KL constraint, for example

$$\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}_{s_t\sim q_{\theta_{\mathrm{roll}}}}\left[D_{\mathrm{KL}}(q_{\theta_{\mathrm{roll}}}(\cdot\mid s_t)\,\|\,q_\theta(\cdot\mid s_t))\right]\le\delta_{\mathrm{KL}}. \tag{90}$$

Following TRPO, we use a rollout-averaged KL constraint as a practical approximation to the worst-state divergence control appearing in the improvement bound (Schulman et al., 2015).

F.1.3 Infinite-Horizon Discounted Extension

In this section, we present two ways to extend the finite-horizon result above to the infinite-horizon discounted setting.

Alternative A: deterministic time weighting. For each $t\ge 0$, define a time-dependent entropy coefficient and scaled reward

$$\mathcal{T}_t:=\gamma^t\mathcal{T},\qquad\qquad R_t(s_t,a_t):=\gamma^t R_{\mathrm{env}}(s_t,a_t). \tag{91}$$

For a finite cutoff T, the resulting maximum-entropy return is

$$\begin{aligned}\mathcal{J}_T(\theta)&:=\sum_{t=0}^{T}\mathbb{E}_{q_\theta}[R_t(s_t,a_t)-\mathcal{T}_t\log q_\theta(a_t\mid s_t)]\\&=\sum_{t=0}^{T}\gamma^t\mathbb{E}_{q_\theta}[R_{\mathrm{env}}(s_t,a_t)-\mathcal{T}\log q_\theta(a_t\mid s_t)]\,.\end{aligned} \tag{92}$$

Assuming $\mathbb{E}_{q_\theta}[\sum_{t=0}^{\infty}\gamma^t|R_{\mathrm{env}}(s_t,a_t)-\mathcal{T}\log q_\theta(a_t\mid s_t)|]<\infty$, we may take $T\to\infty$ and obtain the standard discounted maximum-entropy objective (Haarnoja et al., 2018; Levine, 2018):

$$\mathcal{J}(\theta)=\mathbb{E}_{q_\theta}\left[\sum_{t=0}^{\infty}\gamma^t\left(R_{\mathrm{env}}(s_t,a_t)-\mathcal{T}\log q_\theta(a_t\mid s_t)\right)\right]. \tag{93}$$

To derive the policy gradient of this maximum-entropy RL objective, let $c_\theta(s_t,a_t):=\mathcal{T}\log q_\theta(a_t\mid s_t)-R_{\mathrm{env}}(s_t,a_t)$ and $g_t:=\nabla_\theta\log q_\theta(a_t\mid s_t)$. We first differentiate at a finite cutoff T. Applying the log-derivative trick and then the conditional score identity gives

$$\begin{aligned}-\nabla_\theta\mathcal{J}_T(\theta)&=\mathbb{E}_{q_\theta}\left[\left(\sum_{j=0}^{T}g_j\right)\left(\sum_{t=0}^{T}\gamma^t c_\theta(s_t,a_t)\right)\right]+\mathcal{T}\sum_{t=0}^{T}\gamma^t\mathbb{E}_{q_\theta}[g_t]\\&=\sum_{t=0}^{T}\gamma^t\mathbb{E}_{q_\theta}\left[c_\theta(s_t,a_t)\sum_{j=0}^{t}g_j\right].\end{aligned} \tag{94}$$

The direct entropy derivative vanishes because $\mathbb{E}_{q_\theta}[g_t]=0$, and causality gives $\mathbb{E}_{q_\theta}[c_\theta(s_t,a_t)g_j]=0$ for $j>t$. Assuming differentiation under expectation is valid and the differentiated terms admit a locally uniform summable bound, we may take $T\to\infty$ and re-index the remaining causal terms to obtain

$$\begin{aligned}-\nabla_\theta\mathcal{J}(\theta)&=\sum_{j=0}^{\infty}\gamma^j\mathbb{E}_{q_\theta}\left[g_j\,\mathbb{E}_{q_\theta}\left[\sum_{\ell=0}^{\infty}\gamma^\ell c_\theta(s_{j+\ell},a_{j+\ell})\,\middle|\,s_j,a_j\right]\right]\\&=\sum_{j=0}^{\infty}\gamma^j\mathbb{E}_{q_\theta}[(\mathcal{T}\log q_\theta(a_j\mid s_j)-Q^{q_\theta}(s_j,a_j))\,\nabla_\theta\log q_\theta(a_j\mid s_j)]\,,\end{aligned} \tag{95}$$

where Q^{q_θ} is defined by the stationary soft Bellman equations below.

Let $\rho_t^{q_\theta}(s)$ denote the probability density of the continuous state s_t evaluated at s under q_θ, with $s_0\sim p(s_0)$. The normalized discounted state-visitation density is $d^{q_\theta}(s):=(1-\gamma)\sum_{t=0}^{\infty}\gamma^t\rho_t^{q_\theta}(s)$.

Assuming normalizability, define at frozen rollout parameters θ_{roll} the soft policy-improvement target

$$\pi_Q^{q_{\theta_{\mathrm{roll}}}}(a \mid s) := \frac{\exp(Q^{q_{\theta_{\mathrm{roll}}}}(s,a)/\mathcal{T})}{Z_Q^{q_{\theta_{\mathrm{roll}}}}(s)}, \qquad Z_Q^{q_{\theta_{\mathrm{roll}}}}(s) := \int_{\mathcal{A}} \exp(Q^{q_{\theta_{\mathrm{roll}}}}(s,a)/\mathcal{T})\, da. \tag{96}$$

Applying Lemma F.1 to Eq. 95 gives the infinite-horizon surrogate loss

$$\mathcal{L}_{\mathrm{ME}}(\theta;\theta_{\mathrm{roll}}) := \frac{\mathcal{T}}{1-\gamma}\mathbb{E}_{s\sim d^{q_{\theta_{\mathrm{roll}}}}}\left[D_{\mathrm{KL}}\Big(q_\theta(a \mid s)\,\Big\|\,\pi_Q^{q_{\theta_{\mathrm{roll}}}}(a \mid s)\Big)\right]. \tag{97}$$

At the rollout parameters, $\nabla_\theta \mathcal{L}_{\mathrm{ME}}(\theta;\theta_{\mathrm{roll}})|_{\theta=\theta_{\mathrm{roll}}} = -\nabla_\theta \mathcal{J}(\theta)|_{\theta=\theta_{\mathrm{roll}}}$.

Alternative B: a single KL over geometrically stopped trajectories. This construction uses the standard connection between geometric horizons and discounted state visitation (Thakoor et al., 2022; Ribera Borrell et al., 2025). Let $H \in \{0,1,\ldots\}$ be independent of the MDP trajectory and distributed as

$$w(h) := \Pr(H=h) = (1-\gamma)\gamma^h, \qquad \Pr(H \geq t) = \gamma^t.$$

Write $\tau_h = (s_0, a_0, \ldots, s_h, a_h, s_{h+1})$. For a stationary policy $q_\theta(a \mid s)$, define the normalized policy trajectory distribution

$$q_\theta(h,\tau_h) := w(h)p(s_0)\prod_{t=0}^{h} q_\theta(a_t \mid s_t)p(s_{t+1} \mid s_t, a_t), \tag{98}$$

and the corresponding unnormalized reward target and its partition function as

$$\widetilde{\pi}(h,\tau_h) := w(h)p(s_0)\prod_{t=0}^{h} p(s_{t+1} \mid s_t, a_t)\exp\left(\frac{R_{\mathrm{env}}(s_t,a_t)}{\mathcal{T}}\right), \tag{99}$$

$$\mathcal{Z} := \sum_{h=0}^{\infty}\int \widetilde{\pi}(h,\tau_h)\, d\tau_h, \qquad \pi := \frac{\widetilde{\pi}}{\mathcal{Z}}. \tag{100}$$

Let $V_{\mathcal{A}} := \mathrm{vol}(\mathcal{A})$ denote the Lebesgue volume of the action space. The compact-action and bounded-reward assumptions give the explicit sufficient condition

$$M := V_{\mathcal{A}} e^{R_{\max}/\mathcal{T}}, \qquad \gamma M < 1, \qquad 0 < \mathcal{Z} \leq (1-\gamma)\sum_{h=0}^{\infty}\gamma^h M^{h+1} = \frac{(1-\gamma)M}{1-\gamma M} < \infty. \tag{101}$$

This condition can always be enforced without changing the optimizer. For any constant b satisfying $\gamma V_{\mathcal{A}} \exp((R_{\max}-b)/\mathcal{T}) < 1$, replace R_{env} by $R_{\mathrm{env}}^{(b)} := R_{\mathrm{env}} - b$. Then

$$\mathcal{J}^{(b)}(\theta) = \mathcal{J}(\theta) - \frac{b}{1-\gamma}. \tag{102}$$

Hence the shift makes the geometric target normalizable while leaving every policy gradient and maximizer unchanged. The initial-state, geometric-horizon, and raw environment-transition factors are identical in Eqs. 98 and 99. They therefore cancel inside the trajectory log ratio, giving

$$\log\frac{q_\theta(h,\tau_h)}{\pi(h,\tau_h)} = \sum_{t=0}^{h}\left[\log q_\theta(a_t \mid s_t) - \frac{R_{\mathrm{env}}(s_t,a_t)}{\mathcal{T}}\right] + \log\mathcal{Z}. \tag{103}$$

To make the sampling distributions explicit, let $\tau_\infty = (s_0, a_0, s_1, a_1, \ldots) \sim q_\theta$ denote an infinite trajectory generated by the policy and environment dynamics, and draw $H \sim w$ independently. For any integrable quantity $x_t = x_t(\tau_\infty)$, the stopped sum satisfies $\sum_{t=0}^{H} x_t = \sum_{t=0}^{\infty}\mathbf{1}_{\{H\geq t\}}x_t$. Assuming $\mathbb{E}_{\substack{H\sim w\\ \tau_\infty\sim q_\theta}}[\sum_{t=0}^{\infty}\mathbf{1}_{\{H\geq t\}}|x_t|] < \infty$, Fubini's theorem and the independence of H and τ_∞ give

$$\begin{aligned}\mathbb{E}_{\substack{H\sim w\\ \tau_\infty\sim q_\theta}}\left[\sum_{t=0}^{H} x_t\right] &= \sum_{t=0}^{\infty}\mathbb{E}_{\substack{H\sim w\\ \tau_\infty\sim q_\theta}}\left[\mathbf{1}_{\{H\geq t\}}x_t\right] \\ &= \sum_{t=0}^{\infty}\Pr_{H\sim w}(H\geq t)\,\mathbb{E}_{\tau_\infty\sim q_\theta}[x_t] = \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\tau_\infty\sim q_\theta}[x_t].\end{aligned} \tag{104}$$

The last equality uses the geometric survival probability $\Pr_{H\sim w}(H \geq t) = \gamma^t$; this random-horizon representation of discounted objectives is also stated by Ribera Borrell et al. (2025, Lemma 2.2). Taking the expectation of Eq. 103 under $(H, \tau_H) \sim q_\theta(H, \tau_H)$ and using Eq. 104 yields

$$\mathcal{T}\, D_{\mathrm{KL}}(q_\theta(H, \tau_H) \,\|\, \pi(H, \tau_H)) = -\mathcal{J}(\theta) + \mathcal{T} \log \mathcal{Z}, \tag{105}$$

$$\mathcal{J}(\theta) := \mathbb{E}_{\tau_\infty \sim q_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \left(R_{\mathrm{env}}(s_t, a_t) - \mathcal{T} \log q_\theta(a_t \mid s_t)\right)\right]. \tag{106}$$

Thus, minimizing the single trajectory KL in Eq. 105 is exactly equivalent, up to the parameter-independent additive constant $\mathcal{T} \log \mathcal{Z}$ and a sign, to maximizing the infinite-horizon discounted maximum-entropy RL objective.

Stationary soft value functions and policy gradient. Because the MDP, reward, and policy are stationary and the geometric stopping distribution is memoryless, the value functions no longer require a time index. They satisfy the discounted soft Bellman evaluation equations (Levine, 2018)

$$Q^{q_\theta}(s, a) = R_{\mathrm{env}}(s, a) + \gamma\, \mathbb{E}_{s' \sim p(\cdot|s,a)}[V^{q_\theta}(s')], \tag{107}$$

$$V^{q_\theta}(s) = \mathbb{E}_{a \sim q_\theta(\cdot|s)}\left[Q^{q_\theta}(s, a) - \mathcal{T} \log q_\theta(a \mid s)\right]. \tag{108}$$

The causal likelihood-ratio expansion of Eq. 105 gives

$$\begin{aligned}&\nabla_\theta[\mathcal{T} D_{\mathrm{KL}}(q_\theta(H, \tau_H) \,\|\, \pi(H, \tau_H))] \\ &\quad = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{q_\theta}[(\mathcal{T} \log q_\theta(a_t \mid s_t) - Q^{q_\theta}(s_t, a_t))\, \nabla_\theta \log q_\theta(a_t \mid s_t)].\end{aligned} \tag{109}$$

As in the finite-horizon derivation, the explicit entropy derivative vanishes by the conditional score identity; its dependence through future state visitation remains and is contained in Q^{q_θ}.

Using the normalized discounted state-visitation density d^{q_θ}, Eq. 109 is equivalently

$$\begin{aligned}&\nabla_\theta[\mathcal{T} D_{\mathrm{KL}}(q_\theta(H, \tau_H) \,\|\, \pi(H, \tau_H))] \\ &\quad = \frac{1}{1-\gamma} \mathbb{E}_{\substack{s' \sim d^{q_\theta} \\ a \sim q_\theta(\cdot|s')}}\left[(\mathcal{T} \log q_\theta(a \mid s') - Q^{q_\theta}(s', a))\, \nabla_\theta \log q_\theta(a \mid s')\right].\end{aligned} \tag{110}$$

F.2 Data-Processing Inequality and DA-MDP Trajectory KL

This section first derives the data-processing inequality from the KL chain rule for a general latent variable. We then specialize it to Gaussian-policy state–action trajectories, where marginalizing the states introduces Δ_{state}, and to diffusion policies, where marginalizing the intermediate diffusion variables additionally introduces $\Delta_{\mathrm{diffusion}}$. Finally, we instantiate these identities for the finite-prefix DA-MDP trajectory and derive its stationary discounted objective, Bellman equations, actor surrogate, and learned-forward-kernel correction.

F.2.1 General Latent-Variable DPI from the KL Chain Rule

Let X be an observed random variable and Z a latent variable, with normalized joint densities $q(X, Z)$ and $\pi(X, Z)$. Assume $q(X, Z)$ is absolutely continuous with respect to $\pi(X, Z)$ and the joint KL is finite. The chain-rule factorizations

$$q(X, Z) = q(X)q(Z \mid X), \qquad \pi(X, Z) = \pi(X)\pi(Z \mid X) \tag{111}$$

give the pointwise identity

$$\log \frac{q(X, Z)}{\pi(X, Z)} = \log \frac{q(X)}{\pi(X)} + \log \frac{q(Z \mid X)}{\pi(Z \mid X)}. \tag{112}$$

Taking the expectation under $q(X, Z)$ yields the KL chain rule

$$\begin{aligned}D_{\mathrm{KL}}(q(X, Z) \,\|\, \pi(X, Z)) &= D_{\mathrm{KL}}(q(X) \,\|\, \pi(X)) \\ &\quad + \mathbb{E}_{X \sim q(X)}[D_{\mathrm{KL}}(q(Z \mid X) \,\|\, \pi(Z \mid X))].\end{aligned} \tag{113}$$

The conditional KL in the second line is nonnegative. Therefore

$$D_{\mathrm{KL}}(q(X) \,\|\, \pi(X)) \leq D_{\mathrm{KL}}(q(X, Z) \,\|\, \pi(X, Z)), \tag{114}$$

which is the data-processing inequality for the deterministic map $(X, Z) \mapsto X$ that discards Z. Its exact gap is the expected conditional KL in Eq. 113; equality holds if and only if $q(Z \mid X) = \pi(Z \mid X)$ for $q(X)$-almost every X.

F.2.2 Gaussian-Policy Trajectories and Δ_{state}

For the Gaussian-policy algorithms, the conditional action distribution has the form

$$q_\theta(a_t \mid s_t) = \mathcal{N}\big(a_t; \mu_\theta(s_t), \operatorname{diag}(\sigma_\theta(s_t)^2)\big)\,, \tag{115}$$

and induces the joint state–action trajectory distribution in Eq. 77. To apply Eq. 113, set $X = a_{0:T}$ and $Z = s_{0:T+1}$. This gives

$$D_{\mathrm{KL}}(q_\theta(a_{0:T}, s_{0:T+1}) \,\|\, \pi(a_{0:T}, s_{0:T+1})) = D_{\mathrm{KL}}(q_\theta(a_{0:T}) \,\|\, \pi(a_{0:T})) + \Delta_{\text{state}}, \tag{116}$$

where

$$\Delta_{\text{state}} := \mathbb{E}_{a_{0:T}\sim q_\theta(a_{0:T})}[D_{\mathrm{KL}}(q_\theta(s_{0:T+1} \mid a_{0:T}) \,\|\, \pi(s_{0:T+1} \mid a_{0:T}))]\,. \tag{117}$$

Thus, Δ_{state} is exactly the information lost when the state trajectory is marginalized and only the action sequence is retained. Although the policy and target joint distributions contain the same environment transition kernel, their conditional state distributions given a complete action sequence need not agree: the policy weights a state path through the state-dependent Gaussian action likelihoods, whereas the target weights it through the exponentiated rewards. The bound in Eq. 1 is tight precisely when these two conditional state-trajectory distributions agree almost surely. The chain-rule result itself does not rely on Gaussianity; Eq. 115 identifies the specialization used by the non-diffusion algorithms in this paper.

F.2.3 Diffusion Trajectories and $\Delta_{\text{diffusion}}$

A diffusion policy augments each executed action sequence $a_{0:T} = a^0_{0:T}$ with the intermediate diffusion variables $a^{1:K}_{0:T}$. Apply Eq. 113 with $X = (a_{0:T}, s_{0:T+1})$ and $Z = a^{1:K}_{0:T}$. The resulting chain rule is

$$\begin{aligned} &D_{\mathrm{KL}}\big(q_\theta(a^{0:K}_{0:T}, s_{0:T+1}) \,\big\|\, \pi_\theta(a^{0:K}_{0:T}, s_{0:T+1})\big) \\ &\quad = D_{\mathrm{KL}}(q_\theta(a_{0:T}, s_{0:T+1}) \,\|\, \pi(a_{0:T}, s_{0:T+1})) + \Delta_{\text{diffusion}}, \end{aligned} \tag{118}$$

where

$$\Delta_{\text{diffusion}} := \mathbb{E}_{(a_{0:T}, s_{0:T+1})\sim q_\theta(a_{0:T}, s_{0:T+1})}\Big[D_{\mathrm{KL}}\Big(q_\theta(a^{1:K}_{0:T} \mid a_{0:T}, s_{0:T+1}) \,\Big\|\, \pi_\theta(a^{1:K}_{0:T} \mid a_{0:T}, s_{0:T+1})\Big)\Big]\,. \tag{119}$$

Substituting the Gaussian-policy state decomposition in Eq. 116 gives

$$\begin{aligned} &D_{\mathrm{KL}}\big(q_\theta(a^{0:K}_{0:T}, s_{0:T+1}) \,\big\|\, \pi_\theta(a^{0:K}_{0:T}, s_{0:T+1})\big) \\ &\quad = D_{\mathrm{KL}}(q_\theta(a_{0:T}) \,\|\, \pi(a_{0:T})) + \Delta_{\text{state}} + \Delta_{\text{diffusion}}. \end{aligned} \tag{120}$$

The two nonnegative terms have distinct meanings. Δ_{state} is already present for an ordinary Gaussian policy and measures the conditional state-trajectory mismatch after fixing the executed actions. $\Delta_{\text{diffusion}}$ is introduced only by the diffusion augmentation and measures the additional mismatch between the conditional intermediate diffusion paths after fixing both the executed actions and state trajectory. The augmented DPI bound is tight if and only if both conditional distributions agree almost surely:

$$q_\theta(s_{0:T+1} \mid a_{0:T}) = \pi(s_{0:T+1} \mid a_{0:T}), \qquad q_\theta(a^{1:K}_{0:T} \mid a_{0:T}, s_{0:T+1}) = \pi_\theta(a^{1:K}_{0:T} \mid a_{0:T}, s_{0:T+1}). \tag{121}$$

F.2.4 Finite-Prefix DA-MDP Construction

We now instantiate the preceding diffusion chain rule for the finite-prefix DA-MDP distributions. Retaining the complete denoising path can only reveal more discrepancy between policy and target than observing the final actions alone.

The augmented policy and target distributions are

$$q^T_{\mathrm{DA},\theta}(a^{0:K}_{0:T}, s_{0:T+1}) := p(s_0) \prod_{t=0}^{T} \left[p(s_{t+1} \mid s_t, a^0_t) q_{\text{prior}}(a^K_t) \prod_{k=1}^{K} q_\theta(a^{k-1}_t \mid a^k_t, s_t) \right], \tag{122}$$

$$\pi^T_{\mathrm{DA},\theta}(a^{0:K}_{0:T}, s_{0:T+1}) := \frac{p(s_0)}{\mathcal{Z}_T} \prod_{t=0}^{T} \left[p(s_{t+1} \mid s_t, a^0_t) e^{R_{\text{env}}(s_t, a^0_t)/\mathcal{T}} \prod_{k=1}^{K} \pi_\theta(a^k_t \mid a^{k-1}_t, s_t) \right]. \tag{123}$$

Here, $\mathcal{Z}_T$ is the partition function that normalizes the finite-horizon augmented target. Every forward kernel is normalized, so marginalizing $a_{0:T}^{1:K}$ from the target recovers the joint state–executed-action target; marginalizing the same variables from the policy recovers its corresponding joint distribution. Applying the data-processing inequality first to $a_{0:T}^{1:K}$ and then to $s_{0:T+1}$ gives

$$\begin{aligned} D_{\mathrm{KL}}(q_\theta(a_{0:T}) \,\|\, \pi(a_{0:T})) &\le D_{\mathrm{KL}}(q_\theta(a_{0:T}, s_{0:T+1}) \,\|\, \pi(a_{0:T}, s_{0:T+1})) \\ &\le D_{\mathrm{KL}}\big(q^T_{\mathrm{DA},\theta}(a_{0:T}^{0:K}, s_{0:T+1}) \,\big\|\, \pi^T_{\mathrm{DA},\theta}(a_{0:T}^{0:K}, s_{0:T+1})\big)\,. \end{aligned} \tag{124}$$

The augmented bound is zero exactly when $q^T_{\mathrm{DA},\theta^\star}(a_{0:T}^{0:K}, s_{0:T+1}) = \pi^T_{\mathrm{DA},\theta^\star}(a_{0:T}^{0:K}, s_{0:T+1})$ almost everywhere. This condition is sufficient for equality of the executed-action marginals, whereas a nonzero augmented KL may still coexist with a good executed-action marginal because the conditional state and intermediate diffusion distributions may still differ.

F.2.5 Infinite-Horizon Discounted Objective

We now prove the stationary discounted counterpart of the finite-prefix construction. In simple terms, all denoising decisions used to produce one environment action belong to the same environment time and must therefore receive the same factor γ^t. As above, all trajectory distributions include $s_0 \sim p(s_0)$.

Discounted augmented return. For compactness, define the local reverse–forward log ratio and augmented cost

$$\ell^\theta_{t,k} := \log \frac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi_\theta(a_t^k \mid a_t^{k-1}, s_t)}, \qquad c^\theta_{t,k} := \mathcal{T}\ell^\theta_{t,k} - \mathbf{1}_{\{k=1\}} R_{\mathrm{env}}(s_t, a_t^0). \tag{125}$$

The corresponding deterministic time-weighted maximum-entropy return is

$$\mathcal{J}_{\mathrm{DA}}(\theta) := \mathbb{E}_{q_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \left(R_{\mathrm{env}}(s_t, a_t^0) - \mathcal{T}\sum_{k=1}^{K} \ell^\theta_{t,k} \right)\right]. \tag{126}$$

As in Alternative A of App. F.1.3, this objective can be obtained by assigning $\mathcal{T}_t = \gamma^t\mathcal{T}$ to every reverse–forward log ratio in environment step t and assigning $R_t = \gamma^t R_{\mathrm{env}}$ to its terminal denoising phase. We assume the discounted trajectory costs and their parameter derivatives satisfy the integrability condition below. For the finite trajectory prefix $\xi_t = (s_{0:t+1}, a_{0:t}^{0:K})$, define

$$f_t(\theta, \xi_t) := q^t_{\mathrm{DA},\theta}(\xi_t) \sum_{k=1}^{K} c^\theta_{t,k}, \qquad -\mathcal{J}_{\mathrm{DA}}(\theta) = \sum_{t=0}^{\infty} \gamma^t \int f_t(\theta, \xi_t)\, d\xi_t.$$

At each parameter value under consideration, assume that f_t is continuously differentiable in an open neighborhood U, with a parameter-independent integration domain, and that

$$\sum_{t=0}^{\infty} \gamma^t \int \sup_{\vartheta \in U} \left(|f_t(\vartheta, \xi_t)| + \|\nabla_\vartheta f_t(\vartheta, \xi_t)\| \right) d\xi_t < \infty.$$

This condition justifies interchanging differentiation, integration, and the discounted sum. The derivative of f_t includes both the trajectory-density derivative and the explicit cost derivative.

A single KL over geometrically stopped DA-MDP trajectories. The same objective admits an exact single-KL representation. Let the environment-step terminal index H be independent of the trajectory, with

$$w(h) := \Pr(H = h) = (1-\gamma)\gamma^h, \qquad \Pr(H \ge t) = \gamma^t. \tag{127}$$

For $\xi_h = (s_{0:h+1}, a_{0:h}^{0:K})$, define

$$q_{\mathrm{DA},\theta}(h, \xi_h) := w(h)p(s_0) \prod_{t=0}^{h} \left[q_{\mathrm{prior}}(a_t^K) \prod_{k=1}^{K} q_\theta(a_t^{k-1} \mid a_t^k, s_t) p(s_{t+1} \mid s_t, a_t^0) \right], \tag{128}$$

$$\widetilde{\pi}_{\mathrm{DA},\theta}(h, \xi_h) := w(h)p(s_0) \prod_{t=0}^{h} \left[e^{R_{\mathrm{env}}(s_t, a_t^0)/\mathcal{T}} \prod_{k=1}^{K} \pi_\theta(a_t^k \mid a_t^{k-1}, s_t) p(s_{t+1} \mid s_t, a_t^0) \right], \tag{129}$$

$$\mathcal{Z}_{\mathrm{DA}} := \sum_{h=0}^{\infty} \int \widetilde{\pi}_{\mathrm{DA},\theta}(h, \xi_h)\, d\xi_h, \qquad \pi_{\mathrm{DA},\theta} := \frac{\widetilde{\pi}_{\mathrm{DA},\theta}}{\mathcal{Z}_{\mathrm{DA}}}. \tag{130}$$

Because the forward kernels are normalized, integrating over $a_t^{1:K}$ leaves only an integral over the compact executed-action space. Consequently, the sufficient condition $\gamma V_{\mathcal{A}} e^{R_{\max}/\mathcal{T}} < 1$ from Eq. 101 also guarantees $0 < \mathcal{Z}_{\mathrm{DA}} < \infty$. If it is not satisfied initially, the constant reward shift in Eq. 102 enforces it and changes $\mathcal{J}_{\mathrm{DA}}$ only by the parameter-independent constant $-b/(1-\gamma)$. Because each forward kernel is normalized, integrating Eq. 129 over $a_t^{1:K}$ leaves only the reward potential over a_t^0; hence the partition function is independent of the forward-kernel parameters. The initial-state, geometric-horizon, and raw environment-transition factors in Eqs. 128 and 129 cancel inside the log ratio. Consequently,

$$\log \frac{q_{\mathrm{DA},\theta}(h,\xi_h)}{\pi_{\mathrm{DA},\theta}(h,\xi_h)} = \sum_{t=0}^{h}\left[\log q_{\mathrm{prior}}(a_t^K) + \sum_{k=1}^{K} \ell_{t,k}^{\theta} - \frac{R_{\mathrm{env}}(s_t,a_t^0)}{\mathcal{T}}\right] + \log \mathcal{Z}_{\mathrm{DA}}. \tag{131}$$

Using Eq. 104 gives the exact identity

$$\mathcal{T} D_{\mathrm{KL}}(q_{\mathrm{DA},\theta}(H,\xi_H) \,\|\, \pi_{\mathrm{DA},\theta}(H,\xi_H)) = -\mathcal{J}_{\mathrm{DA}}(\theta) + C_{\mathrm{prior}} + \mathcal{T}\log \mathcal{Z}_{\mathrm{DA}}, \tag{132}$$

$$C_{\mathrm{prior}} := \frac{\mathcal{T}}{1-\gamma}\mathbb{E}_{a^K\sim q_{\mathrm{prior}}}[\log q_{\mathrm{prior}}(a^K)]. \tag{133}$$

The prior and normalizer terms are independent of the optimized reverse-policy parameters and, for normalized forward kernels, of the forward-kernel parameters. Minimizing the geometrically stopped augmented-trajectory KL is therefore equivalent, up to parameter-independent additive constants, to maximizing the infinite-horizon discounted DA-MDP objective in Eq. 126.

F.2.6 Bellman Equations, Occupancy, and Actor Surrogate

We derive the stationary soft Bellman equations, normalize the discounted augmented-state visitation distribution, and express the policy gradient as a local actor-target KL. In simple terms, these identities show how the global path objective becomes the critic and actor losses evaluated on individual denoising transitions.

Stationary augmented Bellman equations. An exact stationary recursion is obtained with the phase-dependent discount

$$\bar{\gamma}(k) := \begin{cases} 1, & k > 1, \\ \gamma, & k = 1. \end{cases} \tag{134}$$

Thus, there is no discount between denoising decisions within one chain, and the factor γ is applied exactly when a_t^0 is executed and the next environment step begins. Let $\tilde{s} = (s, a^k, k)$, $\tilde{a} = a^{k-1}$, and let p_{DA} denote the augmented transition kernel in Sec. 3.2. For a stationary reverse policy q_θ, the soft evaluation equations are

$$Q_{\mathrm{DA}}^{q_\theta}(\tilde{s}, a^{k-1}) = \mathbf{1}_{\{k=1\}} R_{\mathrm{env}}(s,a^0) + \bar{\gamma}(k)\,\mathbb{E}_{\tilde{s}'\sim p_{\mathrm{DA}}(\cdot\mid\tilde{s},a^{k-1})}[V_{\mathrm{DA}}^{q_\theta}(\tilde{s}')], \tag{135}$$

$$V_{\mathrm{DA}}^{q_\theta}(\tilde{s}) = \mathbb{E}_{a^{k-1}\sim q_\theta(\cdot\mid\tilde{s})}\left[Q_{\mathrm{DA}}^{q_\theta}(\tilde{s}, a^{k-1}) - \mathcal{T}\log\frac{q_\theta(a^{k-1}\mid a^k, s)}{\pi_\theta(a^k\mid a^{k-1}, s)}\right]. \tag{136}$$

Starting Eq. 136 at (s_0, a_0^K, K) and averaging $s_0 \sim p(s_0)$ and $a_0^K \sim q_{\mathrm{prior}}$ recovers $\mathcal{J}_{\mathrm{DA}}$ exactly. The prior-density term occurs only in the full trajectory KL and is isolated as the parameter-independent constant in Eq. 133. In particular, the stationary augmented state retains the diffusion phase k, but it needs neither an environment-time index nor a finite-horizon countdown.

Discounted occupancy and actor gradient. Let $\rho_{t,k}^{q_\theta}(s,a^k)$ denote the joint probability density of (s_t, a_t^k) at denoising phase k. The normalized discounted visitation density at $\tilde{s} = (s,a^k,k)$ is $d_{\mathrm{DA}}^{q_\theta}(s,a^k,k) := \frac{1-\gamma}{K}\sum_{t=0}^{\infty}\gamma^t \rho_{t,k}^{q_\theta}(s,a^k)$. Each $\rho_{t,k}^{q_\theta}(s,a^k)$ integrates to one over the continuous state and noisy-action variables. Consequently,

$$\begin{aligned}\sum_{k=1}^{K}\int\!\!\int d_{\mathrm{DA}}^{q_\theta}(s,a^k,k)\,da^k\,ds &= \frac{1-\gamma}{K}\sum_{t=0}^{\infty}\gamma^t\sum_{k=1}^{K}\underbrace{\int\!\!\int \rho_{t,k}^{q_\theta}(s,a^k)\,da^k\,ds}_{1} \\ &= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t = 1.\end{aligned} \tag{137}$$

Moreover, for any integrable function g of the augmented state, substituting the definition of $d_{\mathrm{DA}}^{q_\theta}$ gives

$$\sum_{t=0}^{\infty}\gamma^t\sum_{k=1}^{K}\mathbb{E}_{q_\theta}[g(\tilde{s}_{t,k})]=\frac{K}{1-\gamma}\mathbb{E}_{\tilde{s}\sim d_{\mathrm{DA}}^{q_\theta}}[g(\tilde{s})]. \tag{138}$$

Applying this identity to the local policy-gradient integrand, the causal likelihood-ratio argument for a fixed forward kernel gives

$$\begin{aligned}-\nabla_\theta\mathcal{J}_{\mathrm{DA}}(\theta)&=\sum_{t=0}^{\infty}\gamma^t\sum_{k=1}^{K}\mathbb{E}_{q_\theta}\left[\left(\mathcal{T}\ell_{t,k}^{\theta}-Q_{\mathrm{DA}}^{q_\theta}(\tilde{s}_{t,k},a_t^{k-1})\right)\nabla_\theta\log q_\theta(a_t^{k-1}\mid\tilde{s}_{t,k})\right]\\&=\frac{K}{1-\gamma}\mathbb{E}_{\substack{\tilde{s}\sim d_{\mathrm{DA}}^{q_\theta}\\a^{k-1}\sim q_\theta(\cdot\mid\tilde{s})}}\left[\left(\mathcal{T}\ell_\theta(\tilde{s},a^{k-1})-Q_{\mathrm{DA}}^{q_\theta}(\tilde{s},a^{k-1})\right)\nabla_\theta\log q_\theta(a^{k-1}\mid\tilde{s})\right],\end{aligned} \tag{139}$$

where $\tilde{s}=(s',a^k,k)$ and $\ell_\theta(\tilde{s},a^{k-1}):=\log\frac{q_\theta(a^{k-1}|a^k,s')}{\pi(a^k|a^{k-1},s')}$.

At frozen rollout parameters θ_{roll} and for a fixed forward kernel, define the stationary local target

$$\pi_{Q,\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(a^{k-1}\mid\tilde{s}):=\frac{\pi_{\theta_{\mathrm{roll}}}(a^k\mid a^{k-1},s')\exp\!\Big(Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s},a^{k-1})/\mathcal{T}\Big)}{Z_{Q,\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s})}. \tag{140}$$

The time-independent actor surrogate is then

$$\mathcal{L}_{\mathrm{DA}}^{\mathrm{fixed}}(\theta;\theta_{\mathrm{roll}}):=\frac{K\mathcal{T}}{1-\gamma}\mathbb{E}_{\tilde{s}\sim d_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}}\left[D_{\mathrm{KL}}\Big(q_\theta(a^{k-1}\mid\tilde{s})\,\Big\|\,\pi_{Q,\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(a^{k-1}\mid\tilde{s})\Big)\right]. \tag{141}$$

At $\theta=\theta_{\mathrm{roll}}$, its gradient equals Eq. 139. Thus, the local KL is first-order exact at the rollout parameters even though its occupancy and critic remain frozen during subsequent minibatch updates.

Here the forward kernel and critic are fixed, so normalizing the local target does not change the actor gradient. The next subsection considers learned forward kernels and uses the unnormalized objective to avoid introducing an additional normalizer gradient that is absent from the trajectory objective.

F.2.7 Learned Forward-Kernel Correction

We prove that learning the forward diffusion coefficients adds one explicit score term and that the unnormalized local target retains this term without introducing a spurious normalizer gradient. In simple terms, the forward kernel appears inside the reward correction, so it must receive a direct gradient even though actions are sampled from the reverse kernel.

When π_θ is learned, differentiating the local reverse–forward log ratio adds

$$-\mathcal{T}\sum_{t=0}^{\infty}\gamma^t\sum_{k=1}^{K}\mathbb{E}_{q_\theta}\left[\nabla_\theta\log\pi_\theta(a_t^k\mid a_t^{k-1},s_t)\right] \tag{142}$$

to the fixed-forward policy gradient in Eq. 139. The derivatives of the policy-induced state visitation are already summarized by $Q_{\mathrm{DA}}^{q_\theta}$; Eq. 142 is the only additional explicit derivative.

At frozen rollout parameters, the corresponding gradient-matching surrogate is

$$\mathcal{L}_{\mathrm{DA}}(\theta;\theta_{\mathrm{roll}}):=\frac{K}{1-\gamma}\mathbb{E}_{\substack{\tilde{s}\sim d_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}\\a^{k-1}\sim q_\theta(\cdot\mid\tilde{s})}}\left[\mathcal{T}\log\frac{q_\theta(a^{k-1}\mid a^k,s')}{\pi_\theta(a^k\mid a^{k-1},s')}-Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s},a^{k-1})\right]. \tag{143}$$

To see the unnormalized-KL form explicitly, define the local target over a^{k-1} by

$$\widetilde{\pi}_{Q,\theta}^{q_{\theta_{\mathrm{roll}}}}(a^{k-1}\mid\tilde{s}):=\pi_\theta(a^k\mid a^{k-1},s')\exp\left(\frac{Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s},a^{k-1})}{\mathcal{T}}\right). \tag{144}$$

This target is generally not normalized as a distribution over a^{k-1}. For a normalized reverse kernel $q_\theta(\cdot\mid\tilde{s})$, define

$$D_{\mathrm{KL}}^{\mathrm{u}}\Big(q_\theta(\cdot\mid\tilde{s})\,\Big\|\,\widetilde{\pi}_{Q,\theta}^{q_{\theta_{\mathrm{roll}}}}(\cdot\mid\tilde{s})\Big):=\int q_\theta(a^{k-1}\mid\tilde{s})\log\frac{q_\theta(a^{k-1}\mid\tilde{s})}{\widetilde{\pi}_{Q,\theta}^{q_{\theta_{\mathrm{roll}}}}(a^{k-1}\mid\tilde{s})}\,da^{k-1}. \tag{145}$$

Substituting Eq. 144 into this definition gives

$$\begin{aligned}&\mathcal{T} D_{\mathrm{KL}}^{\mathrm{u}}\Big(q_\theta(\cdot \mid \tilde{s}) \,\Big\|\, \widetilde{\pi}_{Q,\theta}^{q_{\theta_{\mathrm{roll}}}}(\cdot \mid \tilde{s})\Big) \\ &\quad = \mathbb{E}_{a^{k-1}\sim q_\theta(\cdot|\tilde{s})}\left[\mathcal{T}\log\frac{q_\theta(a^{k-1} \mid a^k, s')}{\pi_\theta(a^k \mid a^{k-1}, s')} - Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s}, a^{k-1})\right].\end{aligned} \tag{146}$$

Hence Eq. 143 can be written exactly as

$$\mathcal{L}_{\mathrm{DA}}(\theta;\theta_{\mathrm{roll}}) = \frac{K\mathcal{T}}{1-\gamma}\mathbb{E}_{\tilde{s}\sim d_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}}\left[D_{\mathrm{KL}}^{\mathrm{u}}\Big(q_\theta(\cdot \mid \tilde{s}) \,\Big\|\, \widetilde{\pi}_{Q,\theta}^{q_{\theta_{\mathrm{roll}}}}(\cdot \mid \tilde{s})\Big)\right]. \tag{147}$$

Finally, let $Z_{Q,\theta}(\tilde{s}) := \int \widetilde{\pi}_{Q,\theta}^{q_{\theta_{\mathrm{roll}}}}(a^{k-1} \mid \tilde{s})\, da^{k-1}$ and $p_{Q,\theta}(a^{k-1} \mid \tilde{s}) := \widetilde{\pi}_{Q,\theta}^{q_{\theta_{\mathrm{roll}}}}(a^{k-1} \mid \tilde{s})/Z_{Q,\theta}(\tilde{s})$. Then

$$\begin{aligned}&D_{\mathrm{KL}}\big(q_\theta(a^{k-1} \mid \tilde{s}) \,\big\|\, p_{Q,\theta}(a^{k-1} \mid \tilde{s})\big) \\ &\quad = D_{\mathrm{KL}}^{\mathrm{u}}\Big(q_\theta(a^{k-1} \mid \tilde{s}) \,\Big\|\, \widetilde{\pi}_{Q,\theta}^{q_{\theta_{\mathrm{roll}}}}(a^{k-1} \mid \tilde{s})\Big) + \log Z_{Q,\theta}(\tilde{s}).\end{aligned} \tag{148}$$

Because the learned forward kernel makes $Z_{Q,\theta}$ parameter dependent, differentiating the normalized KL would add $\mathcal{T}\nabla_\theta \log Z_{Q,\theta}(\tilde{s})$, which is absent from the augmented-trajectory gradient. The unnormalized KL retains the required explicit forward-kernel gradient without introducing this additional term.

F.3 Monotonic Improvement for the Local DA-MDP Loss

Statement to be proved. We prove Proposition 3.1 by relating the change in the augmented return of Eq. 8 directly to the change in the local denoising loss of Eq. 10. Let θ be the candidate parameters and θ_{roll} the frozen rollout parameters, and write $q_{\mathrm{roll}} := q_{\theta_{\mathrm{roll}}}$ and $\pi_{\mathrm{roll}} := \pi_{\theta_{\mathrm{roll}}}$. Define

$$\begin{aligned}\Delta\mathcal{J}_{\mathrm{DA}} &:= \mathcal{J}_{\mathrm{DA}}(\theta) - \mathcal{J}_{\mathrm{DA}}(\theta_{\mathrm{roll}}),\\ \Delta\mathcal{L}_{\mathrm{DA}} &:= \mathcal{L}_{\mathrm{DA}}(\theta;\theta_{\mathrm{roll}}) - \mathcal{L}_{\mathrm{DA}}(\theta_{\mathrm{roll}};\theta_{\mathrm{roll}}),\\ \Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} &:= -\Delta\mathcal{L}_{\mathrm{DA}}.\end{aligned} \tag{149}$$

The return and loss differences are candidate minus rollout. A positive surrogate return gain $\Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}}$ corresponds to a decrease in the local loss and predicts an increase in the true return. The second argument of the loss keeps its critic and sampling distribution fixed at the rollout parameters.

We use the exact local rollout value functions from Eq. 9, abbreviated as $Q_{\mathrm{roll}} := Q_{\mathrm{DA}}^{q_{\mathrm{roll}}}$ and $V_{\mathrm{roll}} := V_{\mathrm{DA}}^{q_{\mathrm{roll}}}$. For $\tilde{s} = (s, a^k, k)$, define the reverse–forward log ratio and the local soft advantage by

$$\ell_\theta(\tilde{s}, a^{k-1}) := \log\frac{q_\theta(a^{k-1} \mid a^k, s)}{\pi_\theta(a^k \mid a^{k-1}, s)}, \tag{150}$$

$$A_{\mathrm{roll}}(\tilde{s}, a^{k-1}) := Q_{\mathrm{roll}}(\tilde{s}, a^{k-1}) - \mathcal{T}\ell_{\mathrm{roll}}(\tilde{s}, a^{k-1}) - V_{\mathrm{roll}}(\tilde{s}). \tag{151}$$

The rollout expectation of this advantage is zero at every augmented state. The local policy-change KL and expected forward-log-density change are

$$\kappa(\tilde{s}) := D_{\mathrm{KL}}\big(q_\theta(\cdot \mid \tilde{s}) \| q_{\mathrm{roll}}(\cdot \mid \tilde{s})\big), \tag{152}$$

$$\bar{G}_\theta(\tilde{s}) := \mathbb{E}_{a^{k-1}\sim q_\theta(\cdot|\tilde{s})}\left[\log\frac{\pi_\theta(a^k \mid a^{k-1}, s)}{\pi_{\mathrm{roll}}(a^k \mid a^{k-1}, s)}\right]. \tag{153}$$

Define the constants

$$\begin{aligned}\epsilon_A &:= \sup_{\tilde{s},a^{k-1}} |A_{\mathrm{roll}}(\tilde{s}, a^{k-1})|, \qquad D_q := \sup_{\tilde{s}} \kappa(\tilde{s}) = D_{\mathrm{KL}}^{\max}(q_\theta \| q_{\mathrm{roll}}),\\ \epsilon_G &:= \sup_{\tilde{s}} |\bar{G}_\theta(\tilde{s})|, \qquad B_{K,\gamma} := \frac{K}{1-\gamma}\left(K - 1 + \frac{2\gamma K}{1-\gamma}\right).\end{aligned} \tag{154}$$

Here D_q is the largest candidate-to-rollout KL across augmented states, ϵ_A bounds the rollout advantage, and ϵ_G bounds the expected change in forward log density. Under the assumptions below,

we prove the KL-based bound

$$\boxed{\begin{aligned}\Delta\mathcal{J}_{\mathrm{DA}} \geq \Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} - \left(\epsilon_A B_{K,\gamma} + \frac{K\mathcal{T}}{1-\gamma}\right) D_q \\ - \mathcal{T} B_{K,\gamma}\epsilon_G\sqrt{D_q/2}.\end{aligned}} \tag{182}$$

Consequently, the return strictly improves when the surrogate gain $\Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}}$ exceeds the displayed penalties. For fixed forward kernels, $\bar{G}_\theta = 0$ and $\epsilon_G = 0$, leaving a penalty proportional to the reverse-policy KL alone.

F.3.1 Relating the Bound to Kernel Constraints

Direction and averaging of the reverse KL. The final bound uses the candidate-to-rollout direction because the entropy correction κ averages the reverse log ratio under q_θ. Total variation can be bounded using either KL direction, but the opposite-direction KL alone does not bound κ. The theorem uses the supremum over augmented states; replacing it by a rollout-minibatch average is a practical relaxation of this worst-state constraint.

Learned forward kernels. Learning the forward coefficients affects both the surrogate gain and its prediction error. The term $+\mathcal{T}\bar{G}_\theta$ in the surrogate measures the expected increase in forward log density on candidate reverse samples: increasing π_θ relative to π_{roll} at these samples raises this contribution, whereas decreasing it lowers it. The final bound additionally subtracts $\mathcal{T}B_{K,\gamma}\epsilon_G\sqrt{D_q/2}$ to account for evaluating this change on rollout rather than candidate augmented states. Since $\epsilon_G = \sup_{\tilde{s}}|\bar{G}_\theta(\tilde{s})|$, this penalty is nonnegative regardless of the sign of $\bar{G}_\theta$. It vanishes when the forward kernel is fixed or $D_q = 0$; in the latter case, a forward-induced return change is still captured by the surrogate.

Our algorithms encourage small reverse-policy updates through likelihood-ratio clipping in DA: PPO and reverse-kernel KL control in DA: REPPO and DA: WPO. These mechanisms indirectly target the visitation change responsible for the additional error: for a fixed ϵ_G, reducing D_q reduces $\mathcal{T}B_{K,\gamma}\epsilon_G\sqrt{D_q/2}$. The empirical rollout-to-current KL used by the latter methods (Eq. 34) is a practical proxy. Thus, the implementation encourages the small-update regime motivated by the bound without certifying its improvement condition.

A complementary direction for future work is to control ϵ_G more directly. A sufficient forward-update constraint is

$$\epsilon_G \leq \sup_{\tilde{s}} \mathbb{E}_{a^{k-1}\sim q_\theta(\cdot|\tilde{s})}\left|\log\frac{\pi_\theta(a^k \mid a^{k-1}, s)}{\pi_{\mathrm{roll}}(a^k \mid a^{k-1}, s)}\right| \leq \delta_\pi. \tag{155}$$

This bounds the additional penalty by $\mathcal{T}B_{K,\gamma}\delta_\pi\sqrt{D_q/2}$. Estimates on rollout augmented states with fresh candidate reverse samples could guide a dedicated forward trust region, although finite-sample checks do not certify the uniform constraint.

F.3.2 Proof

Assumptions and notation. We consider continuous environment states and latent actions, with initial-state and environment-transition densities. The candidate and rollout policies share the same initial-state distribution, environment dynamics, number $K \geq 1$ of diffusion steps, and fixed prior q_{prior}. The temperature $\mathcal{T} > 0$ and discount $0 \leq \gamma < 1$ are fixed. Assume common support for the reverse and forward densities, bounded environment rewards, and absolute integrability of the discounted costs under both policies and of the candidate-to-rollout reverse and forward log ratios appearing below. In addition, require

$$\sup_{\tilde{s}}|V_{\mathrm{roll}}(\tilde{s})| < \infty, \qquad \epsilon_A < \infty, \quad D_q < \infty, \quad \epsilon_G < \infty,$$

with the constants defined in Eq. 154. These assumptions justify the expectations below; bounded rollout values make the terminal telescoping term vanish. These uniform boundedness assumptions need not hold for Gaussian kernels on unbounded latent spaces; the proposition provides a conditional improvement guarantee. For this theoretical bound, we assume access to the exact value

functions of the frozen rollout policy. The normalized augmented visitation density $d_{\mathrm{DA}}^{q_\theta}$ is defined in App. F.2. Its L^1 norm integrates the continuous variables and sums over the discrete diffusion phase.

Proof roadmap. The proof follows the maximum-entropy TRPO argument: express both gains through the same advantage and entropy correction, bound the advantage error using state visitation, then bound the entropy correction using $0 \le \kappa \le D_q$. Diffusion adds intermediate noisy-action visitation and, when the forward kernels are learned, the $\bar{G}_\theta$ term. The proof first rewrites the two changes exactly, then bounds their difference.

1. **Expand the local-loss change.** Substep 1.3 identifies the advantage, entropy correction, and forward correction. Substep 1.4 averages them over rollout augmented states.
2. **Expand the return change.** Substep 2.2 telescopes the rollout values across both diffusion and environment steps. Substep 2.3 obtains the candidate augmented-state average of the same terms.
3. **Bound the advantage and forward-correction errors.** Substep 3.1 bounds the expected rollout advantage. Substeps 3.2–3.5 control the change in augmented states, and Substep 3.6 converts their product into a KL bound.
4. **Complete the improvement bound.** Substep 4.1 bounds the remaining entropy correction, and Substep 4.2 combines the bounds. Substeps 4.3–4.4 give the fixed-forward case and the sufficient condition for return improvement.

STEP 1: EXPRESS THE LOCAL-LOSS IMPROVEMENT

We show that the negative loss change is the rollout-state average of the expected advantage, minus a policy-change KL, plus the forward correction. This step uses only the loss definition and the rollout Bellman equation.

Substep 1.1: Restate the loss and evaluate its rollout value. Expanding the unnormalized KL in Eq. 10 gives precisely Eq. 143:

$$\mathcal{L}_{\mathrm{DA}}(\theta;\theta_{\mathrm{roll}}) = \frac{K}{1-\gamma}\mathbb{E}_{\substack{\tilde{s}\sim d_{\mathrm{DA}}^{q_{\mathrm{roll}}} \\ a^{k-1}\sim q_\theta(\cdot|\tilde{s})}}[\mathcal{T}\ell_\theta(\tilde{s},a^{k-1}) - Q_{\mathrm{roll}}(\tilde{s},a^{k-1})]. \tag{156}$$

At the rollout parameters, the inner expectation is $-V_{\mathrm{roll}}(\tilde{s})$, because

$$V_{\mathrm{roll}}(\tilde{s}) = \mathbb{E}_{a^{k-1}\sim q_{\mathrm{roll}}(\cdot|\tilde{s})}[Q_{\mathrm{roll}}(\tilde{s},a^{k-1}) - \mathcal{T}\ell_{\mathrm{roll}}(\tilde{s},a^{k-1})]. \tag{157}$$

Thus $\mathcal{L}_{\mathrm{DA}}(\theta_{\mathrm{roll}};\theta_{\mathrm{roll}}) = -\frac{K}{1-\gamma}\mathbb{E}_{d_{\mathrm{DA}}^{q_{\mathrm{roll}}}}[V_{\mathrm{roll}}]$.

Substep 1.2: Subtract the two losses. Both losses use the same rollout distribution and critic. We can therefore subtract their inner expectations at each augmented state. For this calculation, abbreviate the expected local improvement as

$$g_\theta(\tilde{s}) := \mathbb{E}_{a^{k-1}\sim q_\theta(\cdot|\tilde{s})}[Q_{\mathrm{roll}}(\tilde{s},a^{k-1}) - \mathcal{T}\ell_\theta(\tilde{s},a^{k-1})] - V_{\mathrm{roll}}(\tilde{s}). \tag{158}$$

The loss difference is

$$\begin{aligned}\Delta\mathcal{L}_{\mathrm{DA}} &= \mathcal{L}_{\mathrm{DA}}(\theta;\theta_{\mathrm{roll}}) - \mathcal{L}_{\mathrm{DA}}(\theta_{\mathrm{roll}};\theta_{\mathrm{roll}}) \\ &= \frac{K}{1-\gamma}\mathbb{E}_{\tilde{s}\sim d_{\mathrm{DA}}^{q_{\mathrm{roll}}}}\left[\mathbb{E}_{a^{k-1}\sim q_\theta}[\mathcal{T}\ell_\theta - Q_{\mathrm{roll}}] + V_{\mathrm{roll}}(\tilde{s})\right] \\ &= -\frac{K}{1-\gamma}\mathbb{E}_{\tilde{s}\sim d_{\mathrm{DA}}^{q_{\mathrm{roll}}}}[g_\theta(\tilde{s})].\end{aligned}$$

The inner expectation is conditional on $\tilde{s}$; its arguments are the same as in Eq. 156.

Substep 1.3: Identify the advantage and the two log-density corrections. We express g_θ through the rollout advantage and explicit changes in the reverse and forward kernels. Recall the rollout soft-advantage definition from Eq. 151, $A_{\mathrm{roll}}(\tilde{s}, a^{k-1}) := Q_{\mathrm{roll}}(\tilde{s}, a^{k-1}) - \mathcal{T}\ell_{\mathrm{roll}}(\tilde{s}, a^{k-1}) - V_{\mathrm{roll}}(\tilde{s})$. The first equality below simply substitutes this definition; the second expands the difference of log ratios:

$$\begin{aligned} &Q_{\mathrm{roll}}(\tilde{s}, a^{k-1}) - \mathcal{T}\ell_\theta(\tilde{s}, a^{k-1}) - V_{\mathrm{roll}}(\tilde{s}) \\ &\quad = A_{\mathrm{roll}}(\tilde{s}, a^{k-1}) - \mathcal{T}\big[\ell_\theta(\tilde{s}, a^{k-1}) - \ell_{\mathrm{roll}}(\tilde{s}, a^{k-1})\big] \\ &\quad = A_{\mathrm{roll}}(\tilde{s}, a^{k-1}) - \mathcal{T}\log\frac{q_\theta(a^{k-1} \mid a^k, s)}{q_{\mathrm{roll}}(a^{k-1} \mid a^k, s)} + \mathcal{T}\log\frac{\pi_\theta(a^k \mid a^{k-1}, s)}{\pi_{\mathrm{roll}}(a^k \mid a^{k-1}, s)}. \end{aligned}$$

Define $\bar{A}_\theta(\tilde{s}) := \mathbb{E}_{a^{k-1}\sim q_\theta(\cdot|\tilde{s})}[A_{\mathrm{roll}}(\tilde{s}, a^{k-1})]$. Taking the same candidate expectation of the preceding identity gives

$$g_\theta(\tilde{s}) = \bar{A}_\theta(\tilde{s}) - \mathcal{T}\kappa(\tilde{s}) + \mathcal{T}\bar{G}_\theta(\tilde{s}). \tag{159}$$

The reverse log ratio has expectation κ because the action is sampled from the candidate kernel. This determines the candidate-to-rollout direction of the KL. The forward log ratio instead gives $\bar{G}_\theta$ as defined in Eq. 153.

Substep 1.4: Average over rollout augmented states. Insert Eq. 159 into the loss difference from Substep 1.2. The predicted return gain is

$$\begin{aligned} \Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} = -\Delta\mathcal{L}_{\mathrm{DA}} &= \frac{K}{1-\gamma}\mathbb{E}_{\tilde{s}\sim d_{\mathrm{DA}}^{q_{\mathrm{roll}}}}[g_\theta(\tilde{s})] \\ &= \frac{K}{1-\gamma}\mathbb{E}_{\tilde{s}\sim d_{\mathrm{DA}}^{q_{\mathrm{roll}}}}[\bar{A}_\theta(\tilde{s}) - \mathcal{T}\kappa(\tilde{s}) + \mathcal{T}\bar{G}_\theta(\tilde{s})]. \end{aligned} \tag{160}$$

STEP 2: EXPRESS THE RETURN IMPROVEMENT AND COMPARE THE TWO GAINS

We now obtain the candidate-state average of the same local gain. We start from the return itself and insert a telescoping sum of rollout values. No distribution approximation is made in this step.

Substep 2.1: Restate the return and its initial-value baseline. At $\tilde{s} = (s, a^k, k)$, the regularized one-step reward is

$$r_\theta(\tilde{s}, a^{k-1}) := \mathbf{1}_{\{k=1\}} R_{\mathrm{env}}(s, a^0) - \mathcal{T}\ell_\theta(\tilde{s}, a^{k-1}). \tag{161}$$

Only the final denoising step receives environment reward; each step receives its reverse–forward log-ratio contribution. Writing $\tilde{s}_{t,k} = (s_t, a_t^k, k)$, the return of Eq. 126 is

$$\mathcal{J}_{\mathrm{DA}}(\theta) = \mathbb{E}_\theta\left[\sum_{t=0}^{\infty}\gamma^t\sum_{k=1}^{K} r_\theta(\tilde{s}_{t,k}, a_t^{k-1})\right]. \tag{162}$$

Here and throughout this proof, $\mathbb{E}_\theta$ averages over augmented trajectories generated by $s_0 \sim p(s_0)$ and, at each environment step $t \geq 0$, $a_t^K \sim q_{\mathrm{prior}}$, $a_t^{k-1} \sim q_\theta(\cdot \mid a_t^k, s_t)$ for $k = K, \ldots, 1$, and $s_{t+1} \sim p(\cdot \mid s_t, a_t^0)$. By definition, $V_{\mathrm{roll}}(\tilde{s}_{0,K})$ is the expected discounted sum of $r_{\theta_{\mathrm{roll}}}$ under the rollout policy, conditional on the initial augmented state $\tilde{s}_{0,K}$. Averaging over this state therefore gives $\mathcal{J}_{\mathrm{DA}}(\theta_{\mathrm{roll}}) = \mathbb{E}_{s_0\sim p(s_0),\, a_0^K\sim q_{\mathrm{prior}}}[V_{\mathrm{roll}}(\tilde{s}_{0,K})]$. Since both policies share the initial-state distribution and prior, this expectation is unchanged under the candidate policy. Hence,

$$\begin{aligned} \Delta\mathcal{J}_{\mathrm{DA}} &= \mathcal{J}_{\mathrm{DA}}(\theta) - \mathcal{J}_{\mathrm{DA}}(\theta_{\mathrm{roll}}) \\ &= \mathbb{E}_\theta\left[\sum_{t=0}^{\infty}\gamma^t\sum_{k=1}^{K} r_\theta(\tilde{s}_{t,k}, a_t^{k-1}) - V_{\mathrm{roll}}(\tilde{s}_{0,K})\right]. \end{aligned} \tag{163}$$

Substep 2.2: Telescope the rollout values across both time scales. Let $\tilde{s}_{t,k}^+$ be the successor augmented state. For $k > 1$ it is $\tilde{s}_{t,k-1}$; for $k = 1$ it is the next environment state with fresh prior

noise, $\tilde{s}_{t+1,K}$. Recall that $\bar{\gamma}(k) = 1$ for $k > 1$ and $\bar{\gamma}(1) = \gamma$. To display the cancellation, write $V_{t,k} := V_{\mathrm{roll}}(\tilde{s}_{t,k})$. Within one chain,

$$\sum_{k=1}^{K}[\bar{\gamma}(k)V_{\mathrm{roll}}(\tilde{s}^{+}_{t,k}) - V_{t,k}]$$
$$= \sum_{k=2}^{K}(V_{t,k-1} - V_{t,k}) + \gamma V_{t+1,K} - V_{t,1} = \gamma V_{t+1,K} - V_{t,K}.$$

Every intermediate noisy-action value cancels. Summing the remaining terms over environment steps gives, for finite N,

$$\sum_{t=0}^{N}\gamma^t(\gamma V_{t+1,K} - V_{t,K}) = \sum_{t=1}^{N+1}\gamma^t V_{t,K} - \sum_{t=0}^{N}\gamma^t V_{t,K}$$
$$= -V_{0,K} + \gamma^{N+1}V_{N+1,K}.$$

Bounded rollout values and $\gamma < 1$ make the last term vanish. We can therefore replace the initial-value subtraction in Eq. 163 by these value differences:

$$\Delta\mathcal{J}_{\mathrm{DA}} = \mathbb{E}_\theta\left[\sum_{t=0}^{\infty}\gamma^t\sum_{k=1}^{K}\big(r_\theta(\tilde{s}_{t,k}, a_t^{k-1}) + \bar{\gamma}(k)V_{\mathrm{roll}}(\tilde{s}^{+}_{t,k}) - V_{\mathrm{roll}}(\tilde{s}_{t,k})\big)\right]. \tag{164}$$

Substep 2.3: Identify the local gain and average over candidate states. Condition on the current augmented state and action, and sample $\tilde{s}^{+} \sim p_{\mathrm{DA}}(\cdot \mid \tilde{s}, a^{k-1})$. The rollout Bellman equation uses the same augmented transition kernel as the candidate, so

$$\mathbb{E}_{\tilde{s}^{+}|\tilde{s},a^{k-1}}[r_\theta(\tilde{s}, a^{k-1}) + \bar{\gamma}(k)V_{\mathrm{roll}}(\tilde{s}^{+})] = Q_{\mathrm{roll}}(\tilde{s}, a^{k-1}) - \mathcal{T}\ell_\theta(\tilde{s}, a^{k-1}). \tag{165}$$

Averaging next over $a^{k-1} \sim q_\theta(\cdot \mid \tilde{s})$ and subtracting $V_{\mathrm{roll}}(\tilde{s})$ gives exactly $g_\theta(\tilde{s})$. Hence

$$\boxed{\Delta\mathcal{J}_{\mathrm{DA}} = \sum_{t=0}^{\infty}\gamma^t\sum_{k=1}^{K}\mathbb{E}_\theta[g_\theta(\tilde{s}_{t,k})] = \frac{K}{1-\gamma}\mathbb{E}_{\tilde{s}\sim d_{\mathrm{DA}}^{q_\theta}}[g_\theta(\tilde{s})].} \tag{166}$$

The last equality is the discounted-occupancy identity in Eq. 138.

Substep 2.4: Restate both gains and isolate their difference. Equations 160 and 166, with Eq. 159, have the same integrand:

$$\Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} = \frac{K}{1-\gamma}\mathbb{E}_{\tilde{s}\sim d_{\mathrm{DA}}^{q_{\mathrm{roll}}}}[\bar{A}_\theta(\tilde{s}) - \mathcal{T}\kappa(\tilde{s}) + \mathcal{T}\bar{G}_\theta(\tilde{s})],$$
$$\Delta\mathcal{J}_{\mathrm{DA}} = \frac{K}{1-\gamma}\mathbb{E}_{\tilde{s}\sim d_{\mathrm{DA}}^{q_\theta}}[\bar{A}_\theta(\tilde{s}) - \mathcal{T}\kappa(\tilde{s}) + \mathcal{T}\bar{G}_\theta(\tilde{s})].$$

Their only difference is the augmented-state weighting. The surrogate uses rollout environment states and noisy actions; the true return uses those visited by the candidate. Subtract the two identities and keep the three contributions separate:

$$\Delta\mathcal{J}_{\mathrm{DA}} = \Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} + \mathcal{E}_A + \mathcal{E}_\kappa + \mathcal{E}_G, \tag{167}$$

where

$$\mathcal{E}_A := \frac{K}{1-\gamma}\left(\mathbb{E}_{d_{\mathrm{DA}}^{q_\theta}}[\bar{A}_\theta] - \mathbb{E}_{d_{\mathrm{DA}}^{q_{\mathrm{roll}}}}[\bar{A}_\theta]\right),$$
$$\mathcal{E}_\kappa := \frac{K\mathcal{T}}{1-\gamma}\left(\mathbb{E}_{d_{\mathrm{DA}}^{q_{\mathrm{roll}}}}[\kappa] - \mathbb{E}_{d_{\mathrm{DA}}^{q_\theta}}[\kappa]\right), \tag{168}$$
$$\mathcal{E}_G := \frac{K\mathcal{T}}{1-\gamma}\left(\mathbb{E}_{d_{\mathrm{DA}}^{q_\theta}}[\bar{G}_\theta] - \mathbb{E}_{d_{\mathrm{DA}}^{q_{\mathrm{roll}}}}[\bar{G}_\theta]\right).$$

The sign in $\mathcal{E}_\kappa$ is reversed because both gains contain $-\mathcal{T}\kappa$. All three terms arise from changing the state weighting; the local loss already includes both log-density corrections. We bound the advantage and forward errors in Step 3, then use the nonnegativity of κ in Step 4.

STEP 3: BOUND THE ADVANTAGE AND FORWARD-CORRECTION ERRORS

We first bound the expected rollout advantage, then the change in visited augmented states. Multiplying the two bounds gives the advantage error, just as in the maximum-entropy TRPO proof. The visitation bound also controls the forward-correction error.

Substep 3.1: Bound the expected rollout advantage. Define the largest reverse-kernel change in total variation by

$$\delta_q := \sup_{\tilde{s}} D_{\mathrm{TV}}\big(q_\theta(\cdot \mid \tilde{s}), q_{\mathrm{roll}}(\cdot \mid \tilde{s})\big) = \frac{1}{2} \sup_{\tilde{s}} \int |q_\theta(a^{k-1} \mid \tilde{s}) - q_{\mathrm{roll}}(a^{k-1} \mid \tilde{s})| \, da^{k-1}. \quad (169)$$

The rollout Bellman equation implies $\mathbb{E}_{q_{\mathrm{roll}}}[A_{\mathrm{roll}} \mid \tilde{s}] = 0$. Subtract this zero from the candidate expectation to introduce the difference between the two kernels:

$$\begin{aligned} |\bar{A}_\theta(\tilde{s})| &= \left| \int (q_\theta - q_{\mathrm{roll}})(a^{k-1} \mid \tilde{s}) A_{\mathrm{roll}}(\tilde{s}, a^{k-1}) \, da^{k-1} \right| \\ &\le \epsilon_A \int |q_\theta(a^{k-1} \mid \tilde{s}) - q_{\mathrm{roll}}(a^{k-1} \mid \tilde{s})| \, da^{k-1} \le 2\epsilon_A \delta_q. \end{aligned} \quad (170)$$

It remains to control the distributions used to average this quantity. An augmented state contains both an environment state and a noisy action, so both must be included.

Substep 3.2: Bound the noisy-action marginals at each diffusion step. For $i \in \{\theta, \mathrm{roll}\}$, let $q_i^{(k)}(a^k \mid s)$ be the marginal density of the noisy action at phase k when a chain is generated at environment state s. These marginals satisfy

$$q_i^{(K)} = q_{\mathrm{prior}}, \qquad q_i^{(k-1)}(a^{k-1} \mid s) = \int q_i^{(k)}(a^k \mid s) q_i(a^{k-1} \mid a^k, s) \, da^k. \quad (171)$$

Fix an environment state s. The L^1 distance between the two noisy-action densities is their integrated absolute difference:

$$\|q_\theta^{(k)}(\cdot \mid s) - q_{\mathrm{roll}}^{(k)}(\cdot \mid s)\|_1 := \int \left| q_\theta^{(k)}(a^k \mid s) - q_{\mathrm{roll}}^{(k)}(a^k \mid s) \right| \, da^k.$$

The integral is over all values of a^k, while s remains fixed. This distance equals twice the total variation distance and lies in $[0, 2]$. We suppress the common conditioning on s in the norms below. To compare the marginals after one denoising step, add and subtract the rollout marginal propagated through the candidate kernel. The triangle inequality gives

$$\begin{aligned} \|q_\theta^{(k-1)} - q_{\mathrm{roll}}^{(k-1)}\|_1 &\le \int \int |q_\theta^{(k)}(a^k \mid s) - q_{\mathrm{roll}}^{(k)}(a^k \mid s)| q_\theta(a^{k-1} \mid a^k, s) \, da^{k-1} da^k \\ &\quad + \int q_{\mathrm{roll}}^{(k)}(a^k \mid s) \int |q_\theta(a^{k-1} \mid a^k, s) - q_{\mathrm{roll}}(a^{k-1} \mid a^k, s)| \, da^{k-1} da^k \\ &\le \|q_\theta^{(k)} - q_{\mathrm{roll}}^{(k)}\|_1 + 2\delta_q. \end{aligned}$$

In the first term, integrating the candidate kernel over a^{k-1} gives one, leaving the previous marginal distance. In the second term, the inner integral is at most $2\delta_q$ by Eq. 169; averaging it against the normalized rollout marginal preserves this bound. Thus, each denoising step adds at most $2\delta_q$ to the distance.

We now apply induction as the diffusion index decreases from K to 0. The base case is $k = K$: both marginals equal q_{prior}, so their distance is zero. For the induction step, suppose the distance at phase k is at most $2(K-k)\delta_q$. The preceding one-step inequality then gives

$$\|q_\theta^{(k-1)} - q_{\mathrm{roll}}^{(k-1)}\|_1 \le 2(K-k)\delta_q + 2\delta_q = 2\big(K - (k-1)\big)\delta_q.$$

Hence, after $K - k$ denoising steps the distance is at most $2(K-k)\delta_q$. Since δ_q bounds the kernel change at every augmented state, this holds uniformly in s:

$$\sup_s \|q_\theta^{(k)}(\cdot \mid s) - q_{\mathrm{roll}}^{(k)}(\cdot \mid s)\|_1 \le 2(K-k)\delta_q, \qquad 0 \le k \le K. \quad (172)$$

In particular, the executed-action marginal at $k = 0$ differs by at most $2K\delta_q$ in L^1.

Substep 3.3: Express the discounted environment-state distribution. We next track how the changed executed actions affect future environment states. The environment transition under policy q_i is

$$P_i(s' \mid s) := \int q_i^{(0)}(a^0 \mid s) p(s' \mid s, a^0)\, da^0. \tag{173}$$

Let p_0 be the common initial-state density and ρ_t^i the density of s_t under q_i, so $\rho_0^i = p_0$ and $\rho_{t+1}^i = \rho_t^i P_i$. Here $(fP_i)(s') := \int f(s) P_i(s' \mid s)\, ds$ denotes propagation through one environment transition. Write $d_i(s) := (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \rho_t^i(s)$ for the normalized discounted state-visitation density d^{q_i}. Separating its initial term and shifting the remaining index gives

$$\begin{aligned} d_i &= (1-\gamma)p_0 + (1-\gamma)\sum_{t=1}^{\infty} \gamma^t \rho_t^i \\ &= (1-\gamma)p_0 + \gamma \left[(1-\gamma)\sum_{u=0}^{\infty} \gamma^u \rho_u^i \right] P_i = (1-\gamma)p_0 + \gamma d_i P_i. \end{aligned} \tag{174}$$

The bracketed quantity is the defining series for d_i itself. Subtracting the candidate and rollout identities cancels the initial-state term. Adding and subtracting $\gamma d_{\mathrm{roll}} P_\theta$ then yields

$$\begin{aligned} d_\theta - d_{\mathrm{roll}} &= \gamma(d_\theta P_\theta - d_{\mathrm{roll}} P_{\mathrm{roll}}) \\ &= \gamma(d_\theta - d_{\mathrm{roll}})P_\theta + \gamma d_{\mathrm{roll}}(P_\theta - P_{\mathrm{roll}}). \end{aligned} \tag{175}$$

The first term propagates an existing state-distribution difference; the second introduces a difference through the changed actions.

Substep 3.4: Bound the environment-state difference. Two normalization properties bound the terms in Eq. 175. First, propagating any signed density f through a Markov kernel cannot increase its L^1 norm:

$$\|fP_\theta\|_1 \le \int\int |f(s)| P_\theta(s' \mid s)\, ds\, ds' = \int |f(s)|\, ds = \|f\|_1.$$

Second, integrating against the normalized environment transition cannot increase the executed-action difference. Equation 172 therefore implies

$$\begin{aligned} \int |P_\theta(s' \mid s) - P_{\mathrm{roll}}(s' \mid s)|\, ds' &\le \int\int |q_\theta^{(0)}(a^0 \mid s) - q_{\mathrm{roll}}^{(0)}(a^0 \mid s)| p(s' \mid s, a^0)\, da^0 ds' \\ &= \|q_\theta^{(0)}(\cdot \mid s) - q_{\mathrm{roll}}^{(0)}(\cdot \mid s)\|_1 \le 2K\delta_q. \end{aligned}$$

Averaging over the normalized d_{roll} preserves this bound. Taking norms in Eq. 175 thus gives

$$\|d_\theta - d_{\mathrm{roll}}\|_1 \le \gamma \|d_\theta - d_{\mathrm{roll}}\|_1 + 2\gamma K \delta_q.$$

Moving the first term to the left and dividing by $1-\gamma$ yields

$$\boxed{\|d_\theta - d_{\mathrm{roll}}\|_1 \le \frac{2\gamma K}{1-\gamma}\delta_q.} \tag{176}$$

This is the discounted state-visitation argument used in TRPO (Schulman et al., 2015, Theorem 1), with the executed-action change bounded through the K denoising kernels.

Substep 3.5: Combine environment-state and noisy-action differences. The augmented-state visitation density factorizes as

$$d_{\mathrm{DA}}^{q_i}(s, a^k, k) = \frac{1}{K} d_i(s) q_i^{(k)}(a^k \mid s), \qquad k = 1, \ldots, K, \tag{177}$$

This follows directly by conditioning each term in the defining sum for $d_{\mathrm{DA}}^{q_i}$ on s_t: the conditional noisy-action marginal is $q_i^{(k)}$, independent of environment time. The factor $1/K$ assigns equal weight to the phases.

For each phase, split the joint-density difference as $(d_\theta - d_{\text{roll}})q_\theta^{(k)} + d_{\text{roll}}(q_\theta^{(k)} - q_{\text{roll}}^{(k)})$. The triangle inequality and normalization of $q_\theta^{(k)}$ give

$$\begin{aligned}\|d_{\text{DA}}^{q_\theta} - d_{\text{DA}}^{q_{\text{roll}}}\|_1 &\le \|d_\theta - d_{\text{roll}}\|_1 + \frac{1}{K}\sum_{k=1}^{K}\int d_{\text{roll}}(s)\|q_\theta^{(k)}(\cdot \mid s) - q_{\text{roll}}^{(k)}(\cdot \mid s)\|_1\, ds \\ &\le \frac{2\gamma K}{1-\gamma}\delta_q + \frac{2}{K}\sum_{k=1}^{K}(K-k)\delta_q \\ &= \left(\frac{2\gamma K}{1-\gamma} + K - 1\right)\delta_q.\end{aligned} \tag{178}$$

The last equality uses $\sum_{k=1}^{K}(K-k) = K(K-1)/2$. The first contribution comes from future environment states; the second comes from the noisy actions inside the current chain. For $K = 1$, the latter vanishes because the only noisy input is the shared prior.

Substep 3.6: Combine the bounds and express the errors in KL. We now multiply the augmented-state distance by the magnitude of the function being averaged. The visitation estimate from Eq. 178 and the definition of $B_{K,\gamma}$ give

$$\frac{K}{1-\gamma}\|d_{\text{DA}}^{q_\theta} - d_{\text{DA}}^{q_{\text{roll}}}\|_1 \le B_{K,\gamma}\delta_q.$$

The expected advantage is also proportional to δ_q by Eq. 170. Their product is therefore proportional to δ_q^2. Using Pinsker's inequality with natural logarithms, $D_{\text{TV}}(p,q)^2 \le \frac{1}{2}D_{\text{KL}}(p\|q)$, gives

$$\delta_q^2 \le \frac{D_q}{2}, \qquad \delta_q \le \sqrt{D_q/2}. \tag{179}$$

The KL direction here is candidate to rollout, as in the definition of κ. Applying the inequality $|\int fh| \le \|f\|_1 \sup|h|$ to the errors in Eq. 168 yields

$$\begin{aligned}|\mathcal{E}_A| &\le \frac{K}{1-\gamma}\|d_{\text{DA}}^{q_\theta} - d_{\text{DA}}^{q_{\text{roll}}}\|_1 \sup_{\tilde{s}}|\bar{A}_\theta(\tilde{s})| \le 2\epsilon_A B_{K,\gamma}\delta_q^2 \le \epsilon_A B_{K,\gamma} D_q, \\ |\mathcal{E}_G| &\le \frac{K\mathcal{T}}{1-\gamma}\|d_{\text{DA}}^{q_\theta} - d_{\text{DA}}^{q_{\text{roll}}}\|_1 \epsilon_G \le \mathcal{T} B_{K,\gamma}\epsilon_G\delta_q \le \mathcal{T} B_{K,\gamma}\epsilon_G\sqrt{D_q/2}.\end{aligned} \tag{180}$$

The first line is the usual TRPO mechanism: a small policy change limits both the expected advantage and the change in visited states. Diffusion changes the visitation coefficient through the intermediate noisy actions. The second line accounts for learned forward kernels and vanishes when those kernels stay fixed.

STEP 4: COMPLETE THE KL-BASED RETURN-IMPROVEMENT BOUND

We bound the remaining entropy-correction error using the nonnegativity of the local KL, then combine it with Step 3. This is the same final step as in the maximum-entropy TRPO proof.

Substep 4.1: Bound the entropy-correction error. The contribution from $-\mathcal{T}\kappa$ is $\mathcal{E}_\kappa$ in Eq. 168. Since $0 \le \kappa(\tilde{s}) \le D_q$ at every augmented state and both visitation distributions are normalized, both averages of κ lie in $[0, D_q]$. Their difference is therefore at least $-D_q$, so

$$\mathcal{E}_\kappa = \frac{K\mathcal{T}}{1-\gamma}\left(\mathbb{E}_{d_{\text{DA}}^{q_{\text{roll}}}}[\kappa] - \mathbb{E}_{d_{\text{DA}}^{q_\theta}}[\kappa]\right) \ge -\frac{K\mathcal{T}}{1-\gamma}D_q. \tag{181}$$

This bounds a difference of entropy corrections already present in both gains. It does not introduce a new term into the actor loss.

Substep 4.2: Combine the advantage, entropy, and forward bounds. Insert Eqs. 180 and 181 into the exact decomposition from Substep 2.4:

$$\begin{aligned}\Delta\mathcal{J}_{\mathrm{DA}} &= \Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} + \mathcal{E}_A + \mathcal{E}_\kappa + \mathcal{E}_G \\ &\geq \Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} - |\mathcal{E}_A| - \frac{K\mathcal{T}}{1-\gamma} D_q - |\mathcal{E}_G| \\ &\geq \Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} - \left(\epsilon_A B_{K,\gamma} + \frac{K\mathcal{T}}{1-\gamma}\right) D_q \\ &\quad - \mathcal{T} B_{K,\gamma}\epsilon_G \sqrt{D_q/2}.\end{aligned} \tag{182}$$

This proves the statement at the beginning of the section. The first penalty combines the advantage and entropy-correction errors and is proportional to the reverse-kernel KL. The second controls the interaction between reverse-policy change and forward-log-density change. No supremum of g_θ remains in the result.

Substep 4.3: Recover the fixed-forward maximum-entropy bound. When $\pi_\theta = \pi_{\mathrm{roll}}$, we have $\bar{G}_\theta = 0$ and $\epsilon_G = 0$. Substituting the definition of $B_{K,\gamma}$ makes the KL coefficient explicit:

$$\boxed{\begin{aligned}\Delta\mathcal{J}_{\mathrm{DA}} \geq \Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} \\ - \left(\frac{2\gamma K^2 \epsilon_A}{(1-\gamma)^2} + \frac{K(K-1)\epsilon_A + K\mathcal{T}}{1-\gamma}\right) D_{\mathrm{KL}}^{\max}(q_\theta \| q_{\mathrm{roll}}).\end{aligned}} \tag{183}$$

For $K = 1$, this becomes

$$\Delta\mathcal{J}_{\mathrm{DA}} \geq \Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} - \left(\frac{2\gamma\epsilon_A}{(1-\gamma)^2} + \frac{\mathcal{T}}{1-\gamma}\right) D_{\mathrm{KL}}^{\max}(q_\theta \| q_{\mathrm{roll}}), \tag{184}$$

which is the single-step maximum-entropy bound with the sharper Pinsker constant. For $K > 1$, the term proportional to $K(K-1)$ accounts for changing noisy-action distributions within each chain; the K^2 term accounts for their accumulated effect on future environment states.

Substep 4.4: Sufficient Condition for Return Improvement. With learned forward kernels, a sufficient condition for strict return improvement is

$$\Delta\mathcal{J}_{\mathrm{DA}}^{\mathrm{surrogate}} > \left(\epsilon_A B_{K,\gamma} + \frac{K\mathcal{T}}{1-\gamma}\right) D_q + \mathcal{T} B_{K,\gamma}\epsilon_G \sqrt{D_q/2} \quad \Longrightarrow \quad \Delta\mathcal{J}_{\mathrm{DA}} > 0. \tag{185}$$

For fixed forward kernels, only the KL penalty remains. In words, the surrogate return gain must exceed the bound on its prediction error. A small KL alone does not imply improvement, and failure of this sufficient condition does not imply that an update is harmful. Both the surrogate and the true return evaluate the same local advantage and log-density corrections; the penalties bound the effect of using rollout rather than candidate augmented states.

F.4 Maximum-Entropy Wasserstein Policy Optimization

This section derives ME-WPO as an alternative update for the per-state maximum-entropy surrogate objective. We first compute its corresponding 2-Wasserstein gradient flow in policy-distribution space, then express this flow as a score-matching loss and project it onto the parametric policy family using an inverse-Fisher preconditioner.

Functional derivative. Fix a state s and the rollout-frozen critic $Q^{q_{\theta_{\mathrm{roll}}}}$. Expanding the per-state actor-target KL gives, up to a constant independent of q,

$$\mathcal{L}_{\mathrm{ME}}(q, s) = \int q(a \mid s) \left[\log q(a \mid s) - \alpha Q^{q_{\theta_{\mathrm{roll}}}}(s, a)\right] da + \mathrm{const}. \tag{186}$$

Its functional derivative is

$$\frac{\delta\mathcal{L}_{\mathrm{ME}}(q, s)}{\delta q}(a) = \log q(a \mid s) - \alpha Q^{q_{\theta_{\mathrm{roll}}}}(s, a) + \mathrm{const}. \tag{187}$$

The constant contains both the target log-partition function and the $+1$ produced by differentiating $q \log q$; it disappears under the action gradient.

Wasserstein gradient flow. To keep the projection calculation compact, define the spatial residual and its gradient by

$$\phi_t(a,s) := \log q_t(a \mid s) - \alpha Q^{q_{\theta_{\text{roll}}}}(s,a), \qquad v_t(a,s) := \nabla_a \phi_t(a,s). \tag{188}$$

The 2-Wasserstein gradient flow that minimizes $\mathcal{L}_{\text{ME}}$ is (Benamou & Brenier, 2000)

$$\partial_t q_t(a \mid s) = \nabla_a \cdot (q_t(a \mid s) v_t(a,s)) \tag{189}$$

Equivalently, the continuity-equation velocity is $-v_t(a,s)$: the value term transports actions toward larger $Q^{q_{\theta_{\text{roll}}}}$, while the entropy term spreads probability mass. We write $v_\theta(a,s) := v_t(a,s)|_{q_t=q_\theta}$ below and substitute its full expression only at the end of the calculation.

Projection to a parametric policy. Define the instantaneous density change at the current policy by $\dot{q}_\theta(a \mid s) := \partial_t q_t(a \mid s)|_{q_t=q_\theta}$. The Wasserstein flow prescribes the desired infinitesimal density $q_\theta + \dot{q}_\theta \, \mathrm{d}t$. In contrast, a small parameter update produces $q_{\theta+\Delta\theta} = q_\theta + (\nabla_\theta q_\theta)^\top \Delta\theta + o(\|\Delta\theta\|)$. Because the parametric policy family generally cannot represent the Wasserstein density change exactly, we choose $\Delta\theta$ such that its induced density change best approximates $\dot{q}_\theta \, \mathrm{d}t$. The remaining approximation error is $\dot{q}_\theta \, \mathrm{d}t - (\nabla_\theta q_\theta)^\top \Delta\theta$. Measuring this infinitesimal residual by the local KL geometry gives the projection used by Neklyudov et al. (2023); Pfau et al. (2025). More broadly, this is the classical information-geometric construction of projecting an infinite-dimensional density evolution onto the tangent space of a finite-dimensional statistical manifold; the local KL induces the Fisher metric (Amari, 1998), as in projection-filter methods (Brigo et al., 1998; 1999). Concretely, the projection minimizes

$$\begin{aligned} &\mathbb{E}_{s\sim d^{q_{\theta_{\text{roll}}}}} \left[D_{\text{KL}}\Big(q_\theta(\cdot \mid s) \,\Big\|\, q_\theta(\cdot \mid s) + \dot{q}_\theta(\cdot \mid s)\,\mathrm{d}t - \big(\nabla_\theta q_\theta(\cdot \mid s)\big)^\top \Delta\theta \Big) \right] \\ &\quad \approx \frac{1}{2} \begin{bmatrix} \mathrm{d}t \\ -\Delta\theta \end{bmatrix}^\top \begin{bmatrix} \mathcal{F}_{tt} & \mathcal{F}_{t\theta}^\top \\ \mathcal{F}_{t\theta} & \mathcal{F}_{\theta\theta} \end{bmatrix} \begin{bmatrix} \mathrm{d}t \\ -\Delta\theta \end{bmatrix}. \end{aligned} \tag{190}$$

This form follows from the second-order expansion of the KL divergence, whose curvature is the Fisher information matrix. Its three blocks are

$$\begin{aligned} \mathcal{F}_{tt} &:= \mathbb{E}_{\substack{s\sim d^{q_{\theta_{\text{roll}}}} \\ a\sim q_\theta(\cdot|s)}} \left[\left(\frac{\dot{q}_\theta(a \mid s)}{q_\theta(a \mid s)} \right)^2 \right], \\ \mathcal{F}_{t\theta} &:= \mathbb{E}_{\substack{s\sim d^{q_{\theta_{\text{roll}}}} \\ a\sim q_\theta(\cdot|s)}} \left[\nabla_\theta \log q_\theta(a \mid s) \frac{\dot{q}_\theta(a \mid s)}{q_\theta(a \mid s)} \right], \\ \mathcal{F}_{\theta\theta} &:= \mathbb{E}_{\substack{s\sim d^{q_{\theta_{\text{roll}}}} \\ a\sim q_\theta(\cdot|s)}} \left[\nabla_\theta \log q_\theta(a \mid s) \nabla_\theta \log q_\theta(a \mid s)^\top \right]. \end{aligned} \tag{191}$$

Here, $\mathcal{F}_{tt}$ measures the size of the unrestricted density flow, $\mathcal{F}_{t\theta}$ measures how that flow aligns with changes realizable by the policy parameters, and $\mathcal{F}_{\theta\theta}$ is the usual policy Fisher matrix. Minimizing Eq. 190 with respect to $\Delta\theta$ gives

$$\Delta\theta^\star = \mathcal{F}_{\theta\theta}^{-1} \mathcal{F}_{t\theta} \, \mathrm{d}t. \tag{192}$$

Evaluation of the mixed Fisher block. Substituting the density flow from Eq. 189 gives

$$\mathcal{F}_{t\theta} = \mathbb{E}_{s\sim d^{q_{\theta_{\text{roll}}}}} \left[\int \nabla_\theta \log q_\theta(a \mid s) \nabla_a \cdot \Big(q_\theta(a \mid s) v_\theta(a,s) \Big) da \right]. \tag{193}$$

For a fixed state, expanding the divergence and using $q_\theta(a \mid s) \nabla_\theta \log q_\theta(a \mid s) = \nabla_\theta q_\theta(a \mid s)$ gives

$$\mathcal{F}_{t\theta}(s) = \int \nabla_\theta q_\theta(a \mid s) \nabla_a \cdot v_\theta(a,s)\, da + \int \nabla_\theta \log q_\theta(a \mid s) v_\theta(a,s)^\top \nabla_a q_\theta(a \mid s)\, da. \tag{194}$$

Integration by parts in the first term yields

$$\begin{aligned} \int \nabla_\theta q_\theta(a \mid s) \nabla_a \cdot v_\theta(a,s)\, da &= \int_{\partial\Omega} \nabla_\theta q_\theta(a \mid s) v_\theta(a,s)^\top n \, dS \\ &\quad - \int \nabla_a (\nabla_\theta q_\theta(a \mid s))\, v_\theta(a,s)\, da. \end{aligned} \tag{195}$$

Here, $\Omega = \mathbb{R}^N$, and the boundary integral vanishes when q_θ and its parameter derivatives decay sufficiently fast. Expanding

$$\nabla_a(\nabla_\theta q_\theta(a \mid s)) = \nabla_\theta \log q_\theta(a \mid s) \nabla_a q_\theta(a \mid s)^\top + q_\theta(a \mid s) \nabla_\theta \nabla_a \log q_\theta(a \mid s) \tag{196}$$

shows that the first term on the right cancels the second integral in Eq. 194. Consequently,

$$\mathcal{F}_{t\theta} = -\mathbb{E}_{\substack{s\sim d^{q_{\theta_{\text{roll}}}} \\ a\sim q_\theta(\cdot|s)}} \left[\nabla_\theta \nabla_a \log q_\theta(a \mid s) v_\theta(a, s) \right]. \tag{197}$$

Substituting $v_\theta(a, s) = \nabla_a[\log q_\theta(a \mid s) - \alpha Q^{q_{\theta_{\text{roll}}}}(s, a)]$ recovers the mixed-derivative identity used by WPO. In the corresponding first-order surrogate, ∇_θ acts only on the policy score $\nabla_a \log q_\theta(a \mid s)$; the sampled action, critic, and score residual multiplying this mixed derivative are held fixed.

Multiplying the reverse-KL objective by $\mathcal{T}$ gives the maximum-entropy RL objective used in this paper. Its detached-sample Fisher-divergence surrogate is

$$\widehat{\mathcal{L}}_{\text{ME-WPO}}(\theta) := \frac{\mathcal{T}}{2} \mathbb{E}_{\substack{s\sim d^{q_{\theta_{\text{roll}}}} \\ a\sim q_{\theta^*}(\cdot|s)}} \left[\left\| \nabla_a \left(\log q_\theta(a \mid s) - \frac{1}{\mathcal{T}} Q^{q_{\theta_{\text{roll}}}}(s, a) \right) \right\|^2 \right]. \tag{198}$$

Indeed, holding both the sampled action and the critic fixed gives

$$\nabla_\theta \widehat{\mathcal{L}}_{\text{ME-WPO}}(\theta) = \mathbb{E}_{\substack{s\sim d^{q_{\theta_{\text{roll}}}} \\ a\sim q_{\theta^*}(\cdot|s)}} \left[\nabla_\theta \nabla_a \log q_\theta(a \mid s)\, \nabla_a (\mathcal{T} \log q_\theta(a \mid s) - Q^{q_{\theta_{\text{roll}}}}(s, a)) \right]. \tag{199}$$

The exact continuous-time projected parameter flow is therefore

$$\dot{\theta} = -\mathcal{F}_{\theta\theta}^{-1} \nabla_\theta \widehat{\mathcal{L}}_{\text{ME-WPO}}(\theta). \tag{200}$$

The inverse Fisher may be computed exactly for small policy classes or approximated by a diagonal, block-diagonal, or structured preconditioner. Omitting the preconditioner recovers ordinary gradient descent on the same Fisher-divergence objective.

Diagonal-Gaussian specialization. For the Gaussian policy used by ME-WPO, write

$$q_\theta(a \mid s) = \mathcal{N}\big(a\,;\, \mu_\theta(s), \operatorname{diag}\big(\sigma_\theta(s)^2\big)\big)\,, \tag{201}$$

Its action score and its scores in the local distribution coordinates (μ, σ) are

$$\nabla_a \log q_\theta(a \mid s) = -\operatorname{diag}\big(\sigma_\theta(s)^{-2}\big)\,(a - \mu_\theta(s))\,, \tag{202}$$

$$\nabla_\mu \log q_\theta(a \mid s) = \operatorname{diag}\big(\sigma_\theta(s)^{-2}\big)\,(a - \mu_\theta(s))\,,$$

$$\nabla_\sigma \log q_\theta(a \mid s) = -\sigma_\theta(s)^{-1} + (a - \mu_\theta(s))^{\odot 2} \odot \sigma_\theta(s)^{\odot(-3)}, \tag{203}$$

where powers and products involving vectors are element-wise. Consequently, the score residual appearing in Eq. 198 becomes

$$\nabla_a \left(\log q_\theta(a \mid s) - \frac{1}{\mathcal{T}} Q^{q_{\theta_{\text{roll}}}}(s, a) \right) = -\operatorname{diag}\big(\sigma_\theta(s)^{-2}\big)\,(a - \mu_\theta(s)) - \frac{1}{\mathcal{T}} \nabla_a Q^{q_{\theta_{\text{roll}}}}(s, a). \tag{204}$$

The negative of this residual is the continuity-equation velocity for the unscaled reverse KL. Thus, the policy-score contribution spreads samples away from high-density regions, whereas the critic contribution moves them toward increasing value.

At a fixed state, the exact Fisher matrix in the output coordinates (μ, σ) is block diagonal:

$$\mathcal{F}_{(\mu,\sigma)} = \begin{pmatrix} \operatorname{diag}(\sigma_\theta(s)^{-2}) & 0 \\ 0 & 2\operatorname{diag}(\sigma_\theta(s)^{-2}) \end{pmatrix}, \qquad \mathcal{F}_{(\mu,\sigma)}^{-1} = \begin{pmatrix} \operatorname{diag}(\sigma_\theta(s)^2) & 0 \\ 0 & \frac{1}{2}\operatorname{diag}(\sigma_\theta(s)^2) \end{pmatrix}. \tag{205}$$

The corresponding preconditioned output-coordinate gradients are therefore

$$\widetilde{\nabla}_\mu \widehat{\mathcal{L}}_{\text{ME-WPO}} = \sigma_\theta(s)^2 \odot \nabla_\mu \widehat{\mathcal{L}}_{\text{ME-WPO}}, \qquad \widetilde{\nabla}_\sigma \widehat{\mathcal{L}}_{\text{ME-WPO}} = \frac{1}{2} \sigma_\theta(s)^2 \odot \nabla_\sigma \widehat{\mathcal{L}}_{\text{ME-WPO}}. \tag{206}$$

For a neural policy, these exact local Gaussian blocks provide the tractable natural-gradient approximation of Pfau et al. (2025) to the full parameter Fisher.

F.5 Derivation of Diffusion-Augmented Wasserstein Policy Optimization

We now derive the DA: WPO objective for a DA-MDP transition, first for a learnable reverse diffusion kernel and then for a learnable forward diffusion kernel. Fix an augmented state $\tilde{s}_{\tilde{t}} = (s_t, a_t^k, k)$ and the rollout-frozen critic $Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}$. To avoid carrying the full DA-MDP notation through every intermediate step, define

$$\begin{aligned} x &:= a_t^{k-1}, & q_\theta(x) &:= q_\theta(a_t^{k-1} \mid a_t^k, s_t), \\ \pi_\theta(x) &:= \pi_\theta(a_t^k \mid a_t^{k-1}, s_t), & Q(x) &:= Q_{\mathrm{DA}}^{q_{\theta_{\mathrm{roll}}}}(\tilde{s}_{\tilde{t}}, a_t^{k-1}). \end{aligned} \tag{207}$$

We derive the updates by transporting the reverse density and the local forward-target shape at fixed critic and temperature, then projecting these density-space flows onto the kernel parameters. Assuming $0 < Z_\theta < \infty$, we define

$$\widetilde{p}_\theta(x) := \pi_\theta(x) \exp(\alpha Q(x)), \qquad Z_\theta := \int \widetilde{p}_\theta(x)\, dx, \qquad \bar{p}_\theta(x) := \frac{\widetilde{p}_\theta(x)}{Z_\theta}. \tag{208}$$

With $\alpha = 1/\mathcal{T}$, the normalized functional is

$$\begin{aligned} \mathcal{E}_{\tilde{s}_{\tilde{t}}}(q_\theta, \bar{p}_\theta) &:= \mathcal{T} D_{\mathrm{KL}}(q_\theta \,\|\, \bar{p}_\theta) = \mathcal{T} \int q_\theta(x) \log \frac{q_\theta(x)}{\bar{p}_\theta(x)}\, dx \\ &= \int q_\theta(x) \left[\mathcal{T} \log \frac{q_\theta(x)}{\pi_\theta(x)} - Q(x) \right] dx + \mathcal{T} \log Z_\theta. \end{aligned} \tag{209}$$

To distinguish normalization from the choice of transport geometry, for a positive density ρ with $Z[\rho] := \int \rho(x)\, dx \in (0, \infty)$ define

$$\begin{aligned} \mathcal{F}^{\mathrm{u}}(q, \rho) &:= \mathcal{T} \int q(x) \log \frac{q(x)}{\rho(x)}\, dx, \\ \mathcal{G}(q, \rho) &:= \mathcal{T} D_{\mathrm{KL}}(q \,\|\, \rho / Z[\rho]) = \mathcal{F}^{\mathrm{u}}(q, \rho) + \mathcal{T} \log Z[\rho]. \end{aligned} \tag{210}$$

For $\rho = \widetilde{p}_\theta$, $\mathcal{F}^{\mathrm{u}}$ is the local unnormalized functional in Eq. 10, and $\mathcal{G}$ equals $\mathcal{E}_{\tilde{s}_{\tilde{t}}}$ above. Normalization leaves the target's spatial score and its mixed spatial–parameter derivative unchanged:

$$\nabla_x \log \bar{p}_\theta = \nabla_x \log \widetilde{p}_\theta, \qquad \nabla_\theta \nabla_x \log \bar{p}_\theta = \nabla_\theta \nabla_x \log \widetilde{p}_\theta. \tag{211}$$

Because Z_θ is independent of x, it disappears from the spatial derivatives above.

F.5.1 Learnable Reverse Diffusion Kernel

We first vary q while holding $\bar{p}$ fixed. For $q_\epsilon = q + \epsilon\eta$ with $\int \eta(x)dx = 0$,

$$\frac{d}{d\epsilon} \mathcal{E}_{\tilde{s}_{\tilde{t}}}(q_\epsilon, \bar{p}) \bigg|_{\epsilon=0} = \mathcal{T} \int \left(\log \frac{q(x)}{\bar{p}(x)} + 1 \right) \eta(x) dx. \tag{212}$$

Thus, the functional derivative of Eq. 209 is

$$\frac{\delta \mathcal{E}_{\tilde{s}_{\tilde{t}}}}{\delta q}(x) = \mathcal{T} \log \frac{q(x)}{\bar{p}(x)} + \mathrm{const}. \tag{213}$$

Hence its minimizing 2-Wasserstein flow, with λ denoting flow time, is

$$\partial_\lambda q_\lambda(x) = \mathcal{T} \nabla_x \cdot \left[q_\lambda(x) \nabla_x \log \frac{q_\lambda(x)}{\bar{p}_{\theta^*}(x)} \right]. \tag{214}$$

For compactness, define the reverse-kernel score residual

$$r_\theta^q(x) := \nabla_x \log \frac{q_\theta(x)}{\bar{p}_{\theta^*}(x)}. \tag{215}$$

The residual is available without evaluating the normalizer because

$$\begin{aligned} \nabla_x \log \frac{q_\theta(x)}{\bar{p}_{\theta^*}(x)} &= \nabla_x [\log q_\theta(x) - \log \pi_{\theta^*}(x) - \alpha Q(x) + \log Z_{\theta^*}] \\ &= \nabla_x [\log q_\theta(x) - \log \pi_{\theta^*}(x) - \alpha Q(x)]. \end{aligned} \tag{216}$$

Applying the projection in Eqs. 190–192, and using the integration-by-parts identity in Eq. 197, gives

$$\begin{aligned}\mathcal{F}^{q}_{\lambda\theta} &:= \int \nabla_\theta \log q_\theta(x)\; \partial_\lambda q_\lambda(x)|_{q_\lambda=q_\theta}\, dx \\ &= -\mathcal{T}\mathbb{E}_{x\sim q_{\theta^*}}\left[\nabla_\theta \nabla_x \log q_\theta(x)\, r^q_\theta(x)\right] = -\nabla_\theta \mathcal{L}_q(\theta).\end{aligned} \tag{217}$$

Restoring the DA-MDP variables, the projected reverse-kernel flow is therefore represented by

$$\mathcal{L}_q(\theta) := \frac{\mathcal{T}}{2}\mathbb{E}_{a_t^{k-1}\sim q_{\theta^*}(\cdot|a_t^k,s_t)}\left[\left\|\nabla_{a_t^{k-1}}\left(\log\frac{q_\theta(a_t^{k-1}\mid a_t^k,s_t)}{\pi_{\theta^*}(a_t^k\mid a_t^{k-1},s_t)} - \alpha Q^{q_{\theta_{\text{roll}}}}_{\text{DA}}(\tilde{s}_{\tilde{t}},a_t^{k-1})\right)\right\|^2\right]. \tag{218}$$

Here ∇_θ acts only on $\nabla_x \log q_\theta(x)$; the forward target, sampled x, and critic are detached.

F.5.2 Learnable Forward Diffusion Kernel

Normalization preserves the Wasserstein flow. **Eq. 211** establishes that normalization leaves the target's spatial score and its mixed spatial–parameter derivative unchanged. For the forward-target flow, the corresponding invariance follows from the functional-derivative calculation below.

We hold q, the critic, and the temperature fixed and consider mass-preserving spatial transport of the forward target. Assume sufficient regularity, finite second moments, and vanishing boundary flux for the Wasserstein flows and integrations by parts below. Comparing the two functionals on the same density variable ρ and with the same transport metric gives

$$\begin{aligned}&\frac{\delta\mathcal{F}^{\mathrm{u}}}{\delta\rho}(x) = -\mathcal{T}\frac{q(x)}{\rho(x)}, \quad \frac{\delta\mathcal{G}}{\delta\rho}(x) = -\mathcal{T}\frac{q(x)}{\rho(x)} + \frac{\mathcal{T}}{Z[\rho]}, \\ &\nabla_x\frac{\delta\mathcal{F}^{\mathrm{u}}}{\delta\rho} = \nabla_x\frac{\delta\mathcal{G}}{\delta\rho}.\end{aligned} \tag{219}$$

The added first-variation term is spatially constant. Consequently, both functionals induce the same flow $\partial_\lambda\rho = \nabla_x\cdot(\rho\nabla_x(\delta\mathcal{F}^{\mathrm{u}}/\delta\rho))$, which preserves $\int\rho\,dx$ under the stated boundary condition. Thus, for the following calculation, we use the normalized target $\bar{p}_\theta = \widetilde{p}_\theta/Z_\theta$ and its Wasserstein metric. The normalizer affects ordinary parameter differentiation, as discussed in App. F.2.7, but does not affect the mass-preserving spatial flows considered here. The resulting projected transport update is therefore distinct from the ordinary parameter gradient of the local objective.

Derivation. We now vary the normalized target through π_θ. For $\bar{p}_\epsilon = \bar{p} + \epsilon\eta$ with $\int\eta(x)\,dx = 0$, Eq. 209 gives

$$\frac{d}{d\epsilon}\mathcal{E}_{\tilde{s}_{\tilde{t}}}(q_{\theta^*},\bar{p}_\epsilon)\Big|_{\epsilon=0} = -\mathcal{T}\int q_{\theta^*}(x)\frac{\eta(x)}{\bar{p}(x)}dx. \tag{220}$$

Therefore,

$$\frac{\delta\mathcal{E}_{\tilde{s}_{\tilde{t}}}}{\delta\bar{p}}(x) = -\mathcal{T}\frac{q_{\theta^*}(x)}{\bar{p}(x)}. \tag{221}$$

The corresponding spatial transport of the target shape is

$$\partial_\lambda\bar{p}_\lambda(x) = -\mathcal{T}\nabla_x\cdot\left[\bar{p}_\lambda(x)\nabla_x\frac{q_{\theta^*}(x)}{\bar{p}_\lambda(x)}\right]. \tag{222}$$

Equivalently, defining

$$v^\pi_\lambda(x) := \mathcal{T}\nabla_x\frac{q_{\theta^*}(x)}{\bar{p}_\lambda(x)}, \qquad \partial_\lambda\bar{p}_\lambda + \nabla_x\cdot(\bar{p}_\lambda v^\pi_\lambda) = 0, \tag{223}$$

makes explicit that the forward-target shape is transported by the velocity v^π_λ. Because the critic is frozen,

$$\nabla_\theta\nabla_x\log\bar{p}_\theta(x) = \nabla_\theta\nabla_x[\log\pi_\theta(x) + \alpha Q(x) - \log Z_\theta] = \nabla_\theta\nabla_x\log\pi_\theta(x). \tag{224}$$

The middle equality displays the cancellation directly: although $\nabla_\theta \log Z_\theta$ need not vanish, its derivative with respect to x is zero. Define the forward-kernel residual by

$$r^\pi_\theta(x) := \nabla_x \log \frac{q_{\theta^*}(x)}{\bar{p}_\theta(x)}. \tag{225}$$

We can now project Eq. 222 onto the forward-kernel parameter block. Substitution of the flow, one integration by parts, and the identity $\bar{p}\,\nabla_x(q/\bar{p}) = q\,\nabla_x \log(q/\bar{p})$ give the full mixed-block calculation

$$\begin{aligned}
\mathcal{F}^\pi_{\lambda\theta} &:= \int \nabla_\theta \log \bar{p}_\theta(x)\; \partial_\lambda \bar{p}_\lambda(x)|_{\bar{p}_\lambda = \bar{p}_{\theta^*}}\, dx \\
&= -\mathcal{T} \int \nabla_\theta \log \bar{p}_\theta(x) \nabla_x \cdot \left[\bar{p}_{\theta^*}(x) \nabla_x \frac{q_{\theta^*}(x)}{\bar{p}_{\theta^*}(x)} \right] dx \\
&= \mathcal{T} \int \bar{p}_{\theta^*}(x) \nabla_\theta \nabla_x \log \bar{p}_\theta(x) \nabla_x \frac{q_{\theta^*}(x)}{\bar{p}_{\theta^*}(x)} dx \\
&= \mathcal{T} \int q_{\theta^*}(x) \nabla_\theta \nabla_x \log \pi_\theta(x) \nabla_x \log \frac{q_{\theta^*}(x)}{\bar{p}_{\theta^*}(x)} dx \\
&= \mathcal{T} \mathbb{E}_{x \sim q_{\theta^*}} \left[\nabla_\theta \nabla_x \log \pi_\theta(x)\, r^\pi_\theta(x) \right] = -\nabla_\theta \mathcal{L}_\pi(\theta).
\end{aligned} \tag{226}$$

The last equality follows because q_{θ^*} is detached and $\nabla_\theta r^\pi_\theta = -\nabla_\theta \nabla_x \log \pi_\theta$. Restoring the DA-MDP variables, the forward-kernel loss is

$$\mathcal{L}_\pi(\theta) := \frac{\mathcal{T}}{2} \mathbb{E}_{a_t^{k-1} \sim q_{\theta^*}(\cdot \mid a_t^k, s_t)} \left[\left\| \nabla_{a_t^{k-1}} \left(\log \frac{q_{\theta^*}(a_t^{k-1} \mid a_t^k, s_t)}{\pi_\theta(a_t^k \mid a_t^{k-1}, s_t)} - \alpha Q^{q_{\theta_{\text{roll}}}}_{\text{DA}}(\tilde{s}_{\tilde{t}}, a_t^{k-1}) \right) \right\|^2 \right]. \tag{227}$$

Here ∇_θ acts only on $\nabla_x \log \pi_\theta(x)$; the reverse-kernel sample and score and the critic gradient are held fixed.

Why the parameter blocks are disjoint. The two variations above are independent for the sampler in App. E.3.2. Although both kernels use the global parameter vector $\theta = (\Psi, \beta_K, \ldots, \beta_0)$, for the local transition pair at reverse step k, $q_\theta(x)$ depends only on (Ψ, β_k), whereas $\pi_\theta(x)$ depends only on β_{k-1}. Hence, the paired kernels have no shared learnable coordinate: varying the reverse kernel leaves the paired forward kernel fixed, and vice versa. The same coefficient can occur in the neighboring transition pair, and the sum over all (t, k) automatically accumulates its reverse- and forward-kernel contributions. If a paired forward and reverse kernel shared a parameter, the two variations would be coupled and the projection would require the corresponding cross-Fisher blocks; the independent derivations above could then not simply be combined.

Combining the forward- and reverse-kernel gradients. Because the reverse- and forward-kernel gradients have disjoint support in θ, Eqs. 217 and 226 combine as

$$\nabla_\theta \mathcal{L}_{\text{DA:WPO}}(\theta) = \nabla_\theta \mathcal{L}_q(\theta) + \nabla_\theta \mathcal{L}_\pi(\theta), \tag{228}$$

where each derivative acts only through its respective kernel score. Because those parameter blocks are disjoint, the two gradients are also obtained by differentiating the single local loss

$$\mathcal{L}_{\text{DA:WPO}}(\theta \mid \tilde{s}_{\tilde{t}}) = \frac{\mathcal{T}}{2} \mathbb{E}_{a_t^{k-1} \sim q_{\theta^*}(\cdot \mid a_t^k, s_t)} \left[\left\| \nabla_{a_t^{k-1}} \left(\log \frac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi_\theta(a_t^k \mid a_t^{k-1}, s_t)} - \alpha Q^{q_{\theta_{\text{roll}}}}_{\text{DA}}(\tilde{s}_{\tilde{t}}, a_t^{k-1}) \right) \right\|^2 \right]. \tag{229}$$

Finally, averaging the local loss over rollout augmented states gives

$$\mathcal{L}_{\text{DA:WPO}}(\theta) = \mathbb{E}_{\tilde{s}_{\tilde{t}} \sim d^{q_{\theta_{\text{roll}}}}_{\text{DA}}} \left[\mathcal{L}_{\text{DA:WPO}}(\theta \mid \tilde{s}_{\tilde{t}}) \right]. \tag{230}$$

This is Eq. 12. With the sampled action and critic held fixed, its gradient is

$$\begin{aligned}
\nabla_\theta \mathcal{L}_{\text{DA:WPO}}(\theta) = \mathbb{E}_{\substack{\tilde{s}_{\tilde{t}} \sim d^{q_{\theta_{\text{roll}}}}_{\text{DA}} \\ a_t^{k-1} \sim q_{\theta^*}(\cdot \mid a_t^k, s_t)}} \Bigg[& \left(\nabla_\theta \nabla_{a_t^{k-1}} \log \frac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi_\theta(a_t^k \mid a_t^{k-1}, s_t)} \right) \\
& \times \nabla_{a_t^{k-1}} \left(\mathcal{T} \log \frac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi_\theta(a_t^k \mid a_t^{k-1}, s_t)} - Q^{q_{\theta_{\text{roll}}}}_{\text{DA}}(\tilde{s}_{\tilde{t}}, a_t^{k-1}) \right) \Bigg].
\end{aligned} \tag{231}$$

The first factor contains the reverse-score derivative and the negative forward-score derivative, so Eq. 231 combines both projected kernel gradients without differentiating through the complete diffusion chain.

Parametric projection. The inverse-Fisher projection is obtained by applying Eqs. 190–192 separately to the disjoint parameter blocks of $q_\theta(a_t^{k-1} \mid a_t^k, s_t)$ and $\pi_\theta(a_t^k \mid a_t^{k-1}, s_t)$. In practice, constructing a Fisher matrix over all neural-network parameters would be prohibitively expensive. We therefore use the adjusted variance-preserving Gaussian preconditioner derived in Eqs. 42–43; without this option, DA: WPO applies the ordinary gradient in Eq. 231.

G COMMON HYPERPARAMETERS

Tables 6 and 7 summarize the common training configurations for MuJoCo and ManiSkill. Exact configurations for each experiment, including task-specific settings and ablation overrides, are provided with the accompanying code.

G.1 MUJOCO

Table 6 records the common JAX continuous-control presets. The PyTorch action-transform and G1 diffusion-step experiments use the settings specified in Apps. E.3.2 and E.3.6, respectively.

Table 6: Common MuJoCo hyperparameters for the JAX training presets.

Parameter	DA: PPO	DA: REPPO	DA: WPO
Rollout and training budget			
Environment interactions		50M	
Parallel environments	2048	1024	1024
Environment steps per rollout, per env.		64	
Local transitions per rollout, per env.		512	
Minibatches per epoch		64	
Epochs per rollout	12	8	8
Discount γ / trace parameter λ		$0.999/0.98$	
Observation / reward normalization		Yes / no	
Diffusion policy			
Control MLP hidden layers $\times$ width		3×256	
Activation / control LayerNorm		GeLU / no	
Time-encoder hidden / output width		$32/16$	
Diffusion steps K / prior std.		$8/3.0$	
Base time step / initial friction		$0.125/0.25$	
Learned per-step time increments		Yes	
Learned friction	Yes	Yes	No
Environment-action clipping bound	1.0	0.999	1.0
Critic			
Critic target	GAE (V)	TD-λ (Q)	TD-λ (Q)
Hidden width / LayerNorm		512 / yes	
Encoder / head linear layers		$2/2$	
Activation / loss		SiLU / HL-Gauss	
HL-Gauss bins	151	151	301
Critic support		$[-100, 200]$	
Auxiliary-loss weight	0.35	0.10	0.05
Optimization and regularization			
Optimizer / learning-rate schedule		Adam / constant	
Actor and critic learning rate	10^{-4}	10^{-3}	10^{-3}
Maximum gradient norm		0.5	
PPO clipping ϵ / chain-KL bound	0.2	0.04	0.02
Temperature	Fixed: 2×10^{-4}	Learned	Learned
Initial learned temperature	—	0.01	0.01
Target-entropy factor c	—	-5	-8
Temperature / KL-dual learning rate	—	3×10^{-4}	6×10^{-4}
Initial KL multiplier	—	0.3	0.3

The effective rollout length is $128 \times \frac{1}{2} = 64$ environment steps, and the configured eight minibatches become $8K = 64$ partitions of local transitions. For entropy tuning, $\mathcal{H}_{\text{tar}} = \mathcal{H}[q_{\text{prior}}] + cd$, where d is the action dimension. The presets request approximately 50 periodic evaluations over training; the continuous-control comparison uses the evaluation curves described in App. E.3.1.

G.2 PUSHT HYPERPARAMETERS

Table 7 records the DA:REPPO PushT configuration in Fig. 2c; the Gaussian baseline and matched evaluation protocol are described in App. E.3.4.

Table 7: Hyperparameters of the unchunked DA:REPPO PushT experiment.

Parameter	Value
Parallel environments / policy steps per rollout	$1024/128$
Training budget / episode horizon	250.35M interactions / 100 steps
Action chunk size / diffusion steps	$1/8$
Epochs / minibatches per epoch	$8/64$
Discount / trace parameter	$0.9/0.98$
Actor / critic hidden dimensions	$(512, 512, 512)/(512, 512)$
Actor architecture / activation	Residual MLP + LayerNorm / GeLU
Time-embedding frequencies	32
Prior standard deviation / base time step	$2.5/0.125$
Time grid / learned friction	Cosine / yes
Learned time increments / noise scale	No / no
Action transform	$\tanh$
Critic loss / bins / support	HL-Gauss / 301 / $[-10, 15]$
Auxiliary-loss weight	0.05
Optimizer / network learning rate	Adam / $3 \cdot 10^{-4}$
Maximum gradient norm / chain-KL bound	$0.5/0.08$
Initial temperature / entropy factor	$0.01/-4$
Entropy-reference standard deviation	3.0
Temperature / KL-dual learning rate	$2.12 \cdot 10^{-4}$ each
Initial KL multiplier / forward-kernel constraint	0.3 / disabled
Observation / running reward normalization	Yes / no

Training uses the task's normalized dense reward and bootstraps from successful final observations. The entropy target is $d[\frac{1}{2}\log(2\pi e\, 3^2) - 4]$, with d the action dimension. Its reference standard deviation differs from the sampling prior's 2.5. The evaluation protocol and results are specified in App. E.3.4.