A Diffusion Model Framework for Maximum Entropy Reinforcement Learning

Sebastian Sanokowski[1] **Kaustubh Patil**[2,3] **Alois Knoll**[1]

[1]Technical University Munich, Chair of Robotics, Artificial Intelligence and Embedded Systems
[2]Technical University Munich, Practical Project Student Exchange Program
[3]MIT World Peace University

Abstract

Diffusion models have achieved remarkable success in data-driven learning and in sampling from complex, unnormalized target distributions. Building on this progress, we reinterpret Maximum Entropy Reinforcement Learning (MaxEntRL) as a diffusion model-based sampling problem. We tackle this problem by minimizing the reverse Kullback-Leibler (KL) divergence between the diffusion policy and the optimal policy distribution using a tractable upper bound. By applying the policy gradient theorem to this objective, we derive a modified surrogate objective for MaxEntRL that incorporates diffusion dynamics in a principled way. This leads to simple diffusion-based variants of Soft Actor-Critic (SAC), Proximal Policy Optimization (PPO) and Wasserstein Policy Optimization (WPO), termed DiffSAC, DiffPPO and DiffWPO. All of these methods require only minor implementation changes to their base algorithm. We find that on standard continuous control benchmarks, DiffSAC, DiffPPO and DiffWPO achieve better returns and higher sample efficiency than SAC and PPO.

1. Introduction

Diffusion models (Song et al., 2021a; Sohl-Dickstein et al., 2015; Ho et al., 2020) have rapidly emerged as one of the most powerful tools for generative modeling, achieving state-of-the-art performance across image, video, and trajectory synthesis (Rombach et al., 2022; Ho et al., 2022; Chi et al., 2025). Beyond generative AI, recent work has begun applying diffusion samplers to *sampling from unnormalized target distributions*.

In this setting, one aims to sample from a distribution of the form

$$\pi(x) = \frac{\exp(-\alpha E(x))}{\mathcal{Z}} \qquad \text{where} \qquad \mathcal{Z} = \int_{\mathbb{R}^N} \exp(-\alpha E(x))\, dx$$

where $x \in \mathbb{R}^N$, $E(x)$ is an energy function and $\alpha = \frac{1}{\mathcal{T}}$ where $\mathcal{T} > 0$ is the temperature. Only the unnormalized density $\tilde{\pi}(x) := \exp(-\alpha E(x))$ is evaluatable, while exact sampling and the computation of the normalizing constant (partition function) $\mathcal{Z}$ is intractable. Importantly, there is no available data from this distribution. Diffusion samplers provide an expressive solution by transporting noise samples toward high-density regions of π via a learned reverse diffusion process. These approaches have been explored in both the continuous domain (Zhang & Chen, 2022; Berner et al., 2022; Vargas et al., 2023; 2024; Richter & Berner, 2024) and the discrete domain (Sanokowski et al., 2024; 2025a), demonstrating strong performance on challenging sampling problems.

In the context of reinforcement learning, diffusion models are particularly appealing because they can naturally represent complex, multimodal action distributions, which are often encountered in high-dimensional reinforcement learning problems. Even when the optimal action distribution is unimodal, diffusion models provide a flexible mechanism to capture non-Gaussian shapes, such as heavy-tailed or Laplace-like distributions, allowing for more expressive and accurate policy representations than standard Gaussian approximations. Thus, diffusion models might be a promising approach to improve exploration, robustness, and ultimately performance on challenging RL environments.

. Correspondence to: Sebastian Sanokowski <sebastian.sanokowski[at]tum.de>.

Preprint. December 4, 2025.

Reinforcement learning naturally fits the unnormalized-sampling paradigm. A long line of work has shown that RL can be formulated as probabilistic inference (Ziebart, 2010; Kappen et al., 2012; Todorov, 2008; Levine, 2018) and thus RL corresponds to sampling from an unnormalized distribution whose partition function is intractable—precisely the regime where diffusion samplers excel.

Our approach: In this work, we extend the sampling perspective to maximum entropy reinforcement learning (MaxEntRL) to diffusion models (see Sec. 2 and Sec. 3). Specifically, we introduce three diffusion-augmented RL algorithms: **DiffPPO**, **DiffSAC**, and **DiffWPO**. Each algorithm arises from a shared reverse-KL formulation and is obtained by augmenting an existing RL algorithm with a diffusion policy that samples actions through a learned reverse diffusion process. Concretely, **DiffSAC** is a diffusion extension of Soft Actor-Critic (SAC) (Haarnoja et al., 2018), **DiffPPO** extends Proximal Policy Optimization (PPO) (Schulman et al., 2017), and **DiffWPO** is a diffusion-based variant of Maximum Entropy Wasserstein Policy Optimization (WPO) (Pfau et al., 2025). This framework can also be viewed as a generalization of Sanokowski et al. (2025a), who train diffusion models using RL, whereas here RL itself is performed *through* diffusion.

Diffusion models have recently been applied to MaxEntRL, but existing methods face limitations. Many rely on forward-KL objectives (Dong et al., 2025; Ma et al., 2025) with importance weighting, which introduces bias, high variance, and *mode-covering* behavior misaligned with the Reinforcement Learning goals. Our DiffSAC method avoids possibly memory-intensive backpropagation through the whole diffusion chain, unlike DiME (Celik et al., 2025). Meanwhile, our DiffPPO method generalizes DPPO (Ren et al., 2025) to arbitrary temperatures and a broader family of reverse-KL objectives, reducing to DPPO exactly when $\mathcal{T} = 0$. For further discussion, see Sec. 5.

In summary, by formulating RL as sampling from a reward-dependent Boltzmann distribution and by using the data processing inequality, we derive a tractable upper bound on the reverse KL divergence between a diffusion policy and the target distribution (see Sec. 3). This yields a clean theoretical connection between diffusion models and MaxEntRL, which we coin *Diffusion-based Maximum Entropy Reinforcement Learning* (DMERL) and leads naturally to the DiffPPO, DiffSAC, and DiffWPO algorithms with minimal implementation overhead by a change of the corresponding MDP, the surrogate objective, and the value function (see Sec. 3.1).

Our main contributions are as follows:

- We establish in Sec. 2.4 a novel mathematical connection between the off-policy Maximum Entropy RL surrogate loss (e.g., used in **SAC**) and the Log-Variance (also known as Trajectory Balance) loss (Richter et al., 2020; Malkin et al., 2022a).
- In Sec. 2.5 we derive and propose a Maximum Entropy formulation of the recently introduced Wasserstein Policy Optimization (WPO) (Pfau et al., 2025).
- We introduce a unified, diffusion-based framework for RL derived from reverse KL minimization in Sec. 3, which generalizes and significantly simplifies prior diffusion-based RL approaches.
- Building on this unified framework, we introduce and instantiate three novel, practical algorithms: **DiffPPO**, **DiffSAC**, and **DiffWPO**. These methods are easily implementable, requiring only minor modifications to standard libraries such as Stable-Baselines3 (Raffin et al., 2021).
- Preliminary results in Sec. 4 demonstrate that our diffusion-based policies achieve substantially improved sample efficiency and superior average returns compared to their vanilla counterparts on standard continuous-control benchmarks.

2. Problem Description

Reinforcement learning (RL) can be interpreted as an inference or sampling problem, where the objective is to generate trajectories that maximize cumulative rewards. Let

$$\tau = (s_0, a_0, s_1, a_1, \ldots, a_T, s_{T+1})$$

denote a trajectory generated under transition dynamics $p(s_{t+1} \mid s_t, a_t)$ and initial-state distribution $p(s_0)$, with reward function $R_{\text{env}}(s_t, a_t)$.

Here, $s_t \in \mathcal{S}$ denotes the state at time t, $a_t \in \mathcal{A} = \mathbb{R}^N$ denotes the action at time t, and T is the finite time horizon of the trajectory.

Instead of directly maximizing expected rewards, we define an *unnormalized target distribution* over action sequences $a_{0:T}$:

$$\widetilde{\pi}(a_{0:T}) = \int_{s_{0:T+1}} \prod_{k=0}^{T} p(s_{k+1} \mid s_k, a_k)\, \widetilde{\pi}(a_k \mid s_k)\, p(s_0)\, ds_{0:T+1},$$

where $\widetilde{\pi}(a_k \mid s_k) = \exp(\alpha\, R_{\text{env}}(s_k, a_k))$, and $\alpha = \frac{1}{\mathcal{T}} > 0$, where $\mathcal{T}$ is the temperature. This defines a *reward-weighted trajectory distribution*, where trajectories with higher cumulative rewards receive exponentially more probability mass. The normalized target distribution is

$$\pi(a_{0:T}) = \frac{\widetilde{\pi}(a_{0:T})}{Z}, \qquad Z = \int \widetilde{\pi}(a_{0:T})\, da_{0:T}.$$

The learned policy $q_\theta(a_{0:T})$ serves as a *variational approximation* to this target, defined by

$$q_\theta(a_{0:T}) = \int_{s_{0:T+1}} \prod_{k=0}^{T} p(s_{k+1} \mid s_k, a_k)\, q_\theta(a_k \mid s_k)\, p(s_0)\, ds_{0:T+1}.$$

Both $q_\theta(a_{0:T})$ and $\widetilde{\pi}(a_{0:T})$ are intractable due to integration over state states s_t and thus directly minimizing any f-divergence (Csiszár, 1967) between $D_f(q_\theta(a_{0:T}) \,||\, \pi(a_{0:T}))$ is not a valid option. To overcome this, an tractable upper bound on the f-divergence between trajectory distributions using the *data processing inequality* can be used:

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

where

$$q_\theta(a_{0:T}, s_{0:T+1}) = \prod_{k=0}^{T} p(s_{k+1} \mid s_k, a_k)\, q_\theta(a_k \mid s_k)\, p(s_0) \quad \text{and} \quad \pi(a_{0:T}, s_{0:T+1}) = \prod_{k=0}^{T} p(s_{k+1} \mid s_k, a_k)\, \pi(a_k \mid s_k)\, p(s_0).$$

Importantly, the data processing inequality ensures that the divergence between the joint state-action distributions provides an upper bound on the divergence defined over actions alone. Optimizing the upper bound, therefore, constrains the original objective. While it does not guarantee that a decrease in the upper bound leads to a strict decrease in the left-hand side, a reduction of the upper bound tightens the gap between the two divergences. The left-hand side is provably minimized only when the optimization drives the right-hand side down to the point where the inequality becomes tight. In this way, minimizing the tractable bound optimizes the variational policy $q_\theta(a_{0:T})$ to approximate the target distribution $\pi(a_{0:T})$, even though both cannot be evaluated in a tractable way.

2.1. Reinforcement Learning as Reverse Kullback–Leibler Divergence Minimization

Choosing $f = \text{KL}$ yields the *reverse Kullback–Leibler divergence* objective which can be simplified to (see App. A.1):

$$D_{\text{KL}}^{\mathcal{T}}\big(q_\theta(a_{0:T}, s_{0:T+1}) \,\big\|\, \pi(a_{0:T}, s_{0:T+1})\big) = \sum_{t=0}^{T} \mathbb{E}_{s_t, a_t \sim q_\theta(a_t, s_t)}\big[\mathcal{T} \log q_\theta(a_t \mid s_t) - R_{\text{env}}(s_t, a_t)\big] + C, \tag{2}$$

where C is a constant independent of θ and $q_\theta(a_t, s_t) = q_\theta(a_t|s_t)q_\theta(s_t)$. In the following, we will often write $s_t, a_t \sim q_\theta$, which means that first s_t is sampled from $q_\theta(s_t)$ by unrolling the policy through the MDP and followed by sampling a_t from $q_\theta(a_t|s_t)$.
Next we define $D_{\text{KL}}^{\mathcal{T}} := \mathcal{T} D_{\text{KL}}$ and use it as a loss in order to make it evaluatable at $\mathcal{T} = 0$. In contrast to the original formulation, Eq. 2 is tractable and memory efficient gradient-based optimization is enabled by applying the *policy gradient theorem* (Sutton & Barto, 2018) (see App. A):

$$\nabla_\theta D_{\text{KL}}^{\mathcal{T}}\big(q_\theta(a_{0:T}, s_{0:T+1}) \,\big\|\, \pi(a_{0:T}, s_{0:T+1})\big) = \sum_{t=0}^{T} \mathbb{E}_{s_t, a_t \sim q_\theta}\Big[\big(\mathcal{T} \log q_\theta(a_t \mid s_t) - Q^{q_\theta}(s_t, a_t)\big) \nabla_\theta \log q_\theta(a_t \mid s_t)\Big],$$

where $Q^{q_\theta}(s_t, a_t) = R_{\text{env}}(s_t, a_t) + V(s_{t+1})$ and $V(s_t) = \mathbb{E}_{a_t \sim q_\theta(a_t|s_t)}\big[Q^{q_\theta}(s_t, a_t) - \mathcal{T} \log q_\theta(a_t|s_t)\big]$.

By applying the *log-derivative trick in reverse*, we can express the same objective as a surrogate loss that is more convenient to optimize in practice (see App. A.3):

$$\boxed{\mathcal{L}_{\mathrm{RL}}(\theta) = \mathcal{T}\sum_{t=0}^{T}\mathbb{E}_{s_t\sim q_{\theta^*}}\Big[D_{\mathrm{KL}}\Big(q_\theta(a_t \mid s_t)\,\Big\|\,\frac{\exp(\alpha\, Q^{q_{\theta^*}}(s_t,a_t))}{Z}\Big)\Big],} \tag{3}$$

where the $*$ in θ^* denotes the application of a $\mathrm{stop\_grad}$ operation, which means that gradients are not propagated through these terms. In practice, the outer summation over time $t = 0, \ldots, T$ does not need to be evaluated exhaustively at every optimization step. Instead, it can be approximated via *Monte Carlo integration* by randomly sampling a subset of time steps. This yields an unbiased estimator of the full objective while avoiding backpropagation through all T steps, thereby reducing computational cost. Concretely, we can estimate the surrogate loss as

$$\widehat{\mathcal{L}}(\theta) \;\approx\; \frac{T}{|\mathcal{U}|}\sum_{t\in\mathcal{U}}\mathbb{E}_{s_t\sim q_{\theta^*}}\Big[\mathcal{L}(\theta, s_t)\Big], \qquad \text{where} \qquad \mathcal{L}_{\mathrm{MaxEntRL}}(\theta, s_t) := \mathcal{T}D_{\mathrm{KL}}\Big(q_\theta(a_t \mid s_t)\,\Big\|\,\frac{\exp(\alpha\, Q^{q_{\theta^*}}(s_t,a_t))}{Z}\Big)$$

and $\mathcal{U}$ denotes a minibatch of $|\mathcal{U}|$ uniformly sampled time indices.

This surrogate loss recovers the structure used as a basis in **Soft Actor-Critic (SAC)** (Haarnoja et al., 2018) and **Proximal Policy Optimization (PPO)** (Schulman et al., 2017), and corresponds to the well-known *Maximum Entropy Reinforcement Learning* (MaxEntRL) objective, in which the expected reward is augmented with the policy entropy to encourage exploration and robustness.

2.2. Proximal Policy Optimization (PPO)

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

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

where $q_{\theta_{\mathrm{old}}}$ denotes the behavior policy used to collect trajectories.

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

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

where $r_t(\theta) = \frac{q_\theta(a_t|s_t)}{q_{\theta_{\mathrm{old}}}(a_t|s_t)}$ 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 $A_t = Q^{\theta_{\mathrm{old}}}(s_t, a_t) - V^{\theta_{\mathrm{old}}}(s_t) - \mathcal{T}\log q_\theta(a_t|s_t)$ where the subtraction of $V^{\theta_{\mathrm{old}}}(s_t)$ reduces the variance of the gradient estimates while leaving the average gradient unchanged.

Compared to the original surrogate loss derived from reverse-KL minimization, PPO samples states s_t from $q_{\theta_{\mathrm{old}}}(s_t)$ rather than from the current variational distribution $q_\theta(s_t)$. Importantly, for the first gradient step, the gradient of the PPO objective coincides exactly with the gradient of the original surrogate loss. For subsequent steps, the gradient begins to deviate, but the clipping mechanism in PPO ensures that this deviation remains controlled and does not become excessively large.

2.3. Soft Actor-Critic (SAC)

SAC instead learns a Q-function $Q_\phi^{q_{\theta^*}}(s_t, a_t)$ with parameters ϕ and can therefore leverage the *reparameterization trick* to compute gradients through sampled actions:

$$a_t = f_\theta(\xi_t; s_t), \quad \xi_t \sim \mathcal{N}(0, I), \tag{4}$$

allowing the usage of the identity:

$$\nabla_\theta \mathcal{L}_{\text{SAC}}(\theta, s_t) = \nabla_\theta \mathbb{E}_{s_t\sim\mathcal{B}, a_t\sim q_\theta}[\mathcal{T}\log q_\theta(a_t|s_t) - Q_\phi^{q_{\theta^*}}(s_t, a_t)] \tag{5}$$

$$= \mathbb{E}_{s_t\sim\mathcal{B}, \xi_t\sim\mathcal{N}(0,I)}\big[\mathcal{T}\nabla_\theta \log q_\theta(f_\theta(\xi_t; s_t)|s_t) - \nabla_\theta Q_\phi^{q_{\theta^*}}(s_t, f_\theta(\xi_t; s_t))\big]. \tag{6}$$

SAC deviates from the surrogate reverse-KL loss as the expectation is taken *off-policy* over samples from a *replay buffer* $\mathcal{B}$ rather than from the current variational distribution q_θ^*. This allows for more efficient reuse of past experience and improves sample efficiency, while still optimizing the MaxEntRL objective $\mathcal{L}_{\text{MaxEntRL}}(\theta, s_t)$.

2.4. Connection between the Log Variance Loss and Off-Policy Surrogate Objectives

In App. B we prove that the surrogate objective in Eq. 3 can also be derived from the *Log Variance (LV) loss* (Richter & Berner, 2024), also known as the *Trajectory Balance* loss in the context of GFlowNets (Malkin et al., 2022a), defined for an off-policy sampling distribution ω as

$$D_{LV}^{\omega}(q_\theta(a_{0:T}, s_{0:T+1})\|\pi(a_{0:T}, s_{0:T+1})) \;=\; \frac{1}{2}\,\mathbb{E}_{(a_{0:T}, s_{0:T+1})\sim\omega}\Big[\Big(\log\frac{q_\theta(a_{0:T}, s_{0:T+1})}{\pi(a_{0:T}, s_{0:T+1})} - b_\theta^{\omega}\Big)^2\Big],$$

where $b_\theta^{\omega} = \mathbb{E}_{(a_{0:T}, s_{0:T+1},)\sim\omega}[\log\frac{q_\theta(a_{0:T}, s_{0:T+1})}{\pi(a_{0:T}, s_{0:T+1})}]$. Under the assumption that ω is absolutely continuous with respect to π and q_θ, the LV loss is only zero if and only if $\log q_\theta(a_{0:T}, s_{0:T+1}) = \log \pi(a_{0:T}, s_{0:T+1})$. Under this assumption, ω may be *any* distribution over trajectories, such as from on-policy samples or from the replay buffer. Furthermore, the on-policy LV loss yields exactly the same gradient as the rKL loss when π does not contain learnable parameters (Richter et al., 2020; Malkin et al., 2022b; Sanokowski et al., 2025b).

We prove that by setting $\omega(s_t, a_t) \;=\; q_{\theta^*}(a_t \mid s_t)\,\mathcal{B}(s_t)$, i.e. for any time step t states are sampled from any off-policy distribution but the actions a_t are sampled on-policy, the LV gradient takes a policy-gradient-like form (see App. B):

$$\nabla_\theta D_{LV}^{\omega} = \sum_{t=0}^{T}\mathbb{E}_{(a_t,s_t)\sim(q_{\theta^*},\mathcal{B})}\Big[\Big(\log q_\theta(a_t \mid s_t) + \Big(\alpha Q^{\omega,q_{\theta^*}}(s_t, a_t) - V^{\omega,q_{\theta^*}}(s_t)\Big)\Big)\nabla_\theta \log q_\theta(a_t \mid s_t)\Big], \tag{7}$$

where $Q^{\omega,q_{\theta^*}}(s_t, a_t) = R_{\text{env}}(s_t, a_t) + V^{\omega,q_{\theta^*}}(s_{t+1})$ and $V^{\omega,q_{\theta^*}}(s_t) = \mathbb{E}_{a_t\sim\omega(a_t|s_t)}\big[Q^{\omega,q_{\theta^*}}(s_t, a_t) - \mathcal{T}\log q_{\theta^*}(a_t|s_t)\big]$. Since the actions are sampled from $q_{\theta^*}(a_t \mid s_t)$, we can again apply the *reverse log-derivative trick* (see App. A.3) to Eq. 7, which yields exactly the off-policy surrogate objective used in **SAC**.

$$\boxed{\mathcal{L}_{\text{MaxEntRL}}^{\text{Off-Policy}}(\theta) = \mathcal{T}\sum_{t=0}^{T}\mathbb{E}_{s_t\sim\mathcal{B}}\Big[D_{\text{KL}}\Big(q_\theta(a_t \mid s_t)\,\Big\|\,\frac{\exp(\alpha\, Q^{\mathcal{B},q_{\theta^*}}(s_t, a_t))}{Z}\Big)\Big].} \tag{8}$$

Two important caveats follow. First, the LV loss is not an f-divergence (Malkin et al., 2022b), and it *does not* generally satisfy the data-processing inequality (Sanokowski et al., 2025b); unlike the reverse-KL bound in Eq. 1, there is no information-theoretic guarantee that minimizing D_{LV}^{ω} upper-bounds a trajectory-level divergence over marginalized action sequences. This makes the LV objective less principled from an information-theoretic perspective. Second, **PPO** can be interpreted within the same framework at first order: the initial PPO update coincides with the reverse-KL surrogate gradient, while later updates deviate because PPO samples states from $\omega = q_{\theta_{\text{old}}}$ instead of from q_{θ^*}. The PPO clipping mechanism explicitly constrains this deviation, keeping the effective update close to the reverse-KL policy-gradient direction.

2.5. Maximum Entropy Wasserstein Policy Optimization

Starting from the reverse–KL surrogate objective in Eq. 3, we may interpret this quantity as a functional over action distributions and therefore apply Wasserstein gradient flow-based updates to the policy (Benamou & Brenier, 2000; Neklyudov et al., 2023). In contrast to the reward-only functional considered in (Pfau et al., 2025), the reverse–KL objective introduces the additional entropy-dependent term $\log q_\theta(a_t \mid s_t)$ inside the flow, leading to a slightly different velocity field and hence a modified parametric projection.

Using the functional derivative derived in App. C, projecting the Wasserstein flow onto the parameter space yields the update

$$\theta_{\tau+1} = \theta_\tau + \eta\,\mathcal{F}_{\theta\theta}^{-1}\,\mathbb{E}_{s_t\sim q_{\theta^*}(s_t),\, a_t\sim q_\theta(a_t|s_t)}[\nabla_{a_t}(Q^{q_{\theta^*}}(s_t, a_t) - \mathcal{T}\log q_{\theta^*}(a_t \mid s_t))\,\nabla_\theta \nabla_{a_t}\log q_\theta(a_t \mid s_t)]\,, \tag{9}$$

where $\mathcal{F}_{\theta\theta}$ is the Fisher information matrix arising from the KL projection, η is the learning rate, and τ is the gradient update iteration step. For practical implementation, we follow (Pfau et al., 2025) in replace the Fisher matrix using the following gradient transformations

$$\nabla_\mu \to \sigma^2 \nabla_\mu, \qquad \nabla_\sigma \to \tfrac{1}{2}\sigma^2 \nabla_\sigma,$$

yielding a simple preconditioner suitable for RL problems. We can also write donw a corresponding surrogate loss for WPO, which can be written as:

$$\boxed{\mathcal{L}_{\text{WPO}}(\theta, s_t) = \mathbb{E}_{a_t \sim q_{\theta^*}(a_t|s_t)}\Big[\nabla_{a_t}\big(\mathcal{T} \log q_{\theta^*}(a_t \mid s_t) - Q^{q_{\theta^*}}(s_t, a_t)\big)\, \nabla_{a_t} \log q_\theta(a_t \mid s_t)\Big].} \tag{10}$$

Importantly, the Fisher information matrix $\mathcal{F}_{\theta\theta}$ or its approximations must be applied after computing the gradient of Eq. 10 to perform the natural-gradient preconditioning.

2.6. Diffusion Samplers

We now introduce the diffusion process used to model the policy distribution $q_\theta(a^{(0)})$ with the aim of approximating an intractable target distribution $\pi(a^{(0)})$. For clarity, we drop the state-dependence and assume diagonal Gaussian noise.

Continuous-Time Formulation. Diffusion samplers define a stochastic differential equation (SDE) that gradually perturbs a target distribution into stationary distribution, along with a corresponding reverse process that aims to reconstruct it. For the variance-preserving (VP) SDE (Song et al., 2021b), the forward process is

$$da^{(k)} = -\tfrac{1}{2}\beta(k)\, a^{(k)}\, dk + \nu\sqrt{\beta(k)}\, dW_k, \quad k \in [0, 1], \tag{11}$$

where $\beta(k) > 0$ is a noise schedule and W_k a standard Wiener process. Starting from $k = 0$, this forward process diffuses the target distribution into a simple Gaussian prior $q(a^{(K)}) = \mathcal{N}(0, \nu)$ at $k = 1$. The corresponding reverse-time SDE is

$$da^{(k)} = \Big[-\tfrac{1}{2}\beta(k)\, a^{(k)} - \nu^2 \beta(k)\, \nabla_{a^{(k)}} \log q_k(a^{(k)})\Big] dk + \nu\sqrt{\beta(k)}\, d\bar{W}_k, \tag{12}$$

where $\bar{W}_s$ denotes a reverse Wiener process.

Discrete-Time Approximation. Applying Euler–Maruyama integration with step size Δ_k (from $k = 1$ to $k = 0$) gives the discrete updates.

The **forward diffusion** step from $a^{(k-1)}$ to $a^{(k)}$ is

$$a^{(k)} = (1 - \tfrac{1}{2}\beta_k \Delta_k)\, a^{(k-1)} + \nu\sqrt{\beta_k \Delta_k}\, \varepsilon_k, \qquad \varepsilon_k \sim \mathcal{N}(0, I),$$

so that

$$p(a^{(k)} \mid a^{(k-1)}) = \mathcal{N}\Big((1 - \tfrac{1}{2}\beta_k \Delta_k)\, a^{(k-1)}, \beta_k \Delta_k\, \nu\Big).$$

The **reverse diffusion** step, parameterized by θ, is

$$a^{(k-1)} = (1 + \tfrac{1}{2}\beta_k \Delta_k)\, a^{(k)} + \nu^2\, \beta_k \Delta_k\, s_\theta(a^{(k)}, k) + \nu\sqrt{\beta_k \Delta_k}\, \varepsilon_k,$$

with

$$q_\theta(a^{(k-1)} \mid a^{(k)}) = \mathcal{N}\Big((1 + \tfrac{1}{2}\beta_k \Delta_k)\, a^{(k)} + \beta_k \Delta_k\, s_\theta(a^{(k)}, k), \beta_k \Delta_k\, \nu\Big),$$

where $s_\theta(a^{(k)}, k)$ approximates the unknown score function $\nabla_{a^{(k)}} \log q_k(a^{(k)})$.

Intractability of the Marginal. The marginal of the learned model is obtained by integrating out all intermediate diffusion variables:

$$q_\theta(a^{(0)}) = \int \prod_{k=1}^{K} q_\theta(a^{(k-1)} \mid a^{(k)})\, p(a^{(K)})\, da^{(1:K)}.$$

This integral is, in general, intractable and consequently, the reverse-KL divergence $D_{\mathrm{KL}}(q_\theta(a^{(0)}) \,\|\, \pi(a^{(0)}))$ cannot be directly used as a loss function. However, by the *data processing inequality*, we can upper bound it via the joint divergence:

$$D_{\mathrm{KL}}\big(q_\theta(a^{(0)}) \,\|\, \pi(a^{(0)})\big) \leq D_{\mathrm{KL}}\big(q_\theta(a^{(0:K)}) \,\|\, \pi(a^{(0:K)})\big),$$

which is tractable as

$$q_\theta(a^{(0:K)}) = \prod_{k=1}^{K} q_\theta(a^{(k-1)} \mid a^{(k)})\, q(a^{(K)}) \quad \text{and} \quad \pi(a^{(0:K)}) = \prod_{k=1}^{K} \pi(a^{(k)} \mid a^{(k-1)})\, \pi(a^{(0)}).$$

As explained in Sec. 2, by optimizing the right-hand side, we implicitly optimize the left-hand side of the equation.

3. Method

When applying diffusion policies in RL, optimizing the right-hand side of Eq. 1 is intractable as the marginal $q_\theta(a_t \mid s_t)$ cannot be easily evaluated when the policy is parameterized by a diffusion model. Therefore, similarly to Sec. 2.6, we apply the data-processing inequality once again to Eq. 1 by including the joint probability of the whole reverse and forward diffusion processes over $a_t^{0:K}$. This yields

$$D_{\mathrm{KL}}\big(q_\theta(a_{0:T}, s_{0:T+1}) \,\big\|\, \pi(a_{0:T}, s_{0:T+1})\big) \leq D_{\mathrm{KL}}\big(q_\theta(a_{0:T}^{0:K}, s_{0:T+1}) \,\big\|\, \pi(a_{0:T}^{0:K}, s_{0:T+1})\big), \tag{13}$$

where $q_\theta(a_t^{0:K} \mid s_t)$ and $\pi(a_t^{0:K} \mid s_t)$ denote the joint distributions

$$q_\theta(a_{0:T}^{0:K}, s_{0:T+1}) = \prod_{t=0}^{T} p(s_{t+1} \mid s_t, a_t^0)\, q_\theta(a_t^{0:K} \mid s_t)\, p(s_0) \;\text{ and }\; \pi(a_{0:T}^{0:K}, s_{0:T+1}) = \prod_{t=0}^{T} p(s_{t+1} \mid s_t, a_t^0)\, \pi(a_t^{0:K} \mid s_t)\, p(s_0).$$

KL decomposition: The KL divergence between the joint diffusion policy q_θ and the reference trajectory distribution π decomposes over time t and diffusion index k as

$$D_{\mathrm{KL}}^{\mathcal{T}}\big(q_\theta(a_{0:T}^{0:K}, s_{0:T+1}) \,\big\|\, \pi(a_{0:T}^{0:K}, s_{0:T+1})\big) = \sum_{t=0}^{T}\sum_{k=1}^{K} \mathbb{E}_{s_t,\, a_t^k,\, a_t^{k-1} \sim q_\theta}\left[\mathcal{T} \log \frac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi(a_t^k \mid a_t^{k-1}, s_t)} - R_{\mathrm{env}}(s_t, a_t^0)\, \mathbf{1}_{\{k=0\}}\right] + C, \tag{14}$$

where $\mathbf{1}_{\{k=0\}}$ is the indicator function and C collects all terms independent of θ.

We prove in App. D that when taking the derivative of Eq. 14 the policy gradient theorem still holds and thus the KL objective can also be rewritten as a surrogate loss for *Diffusion-based Maximum Entropy Reinforcement Learning*:

$$\boxed{\mathcal{L}_{\mathrm{DiffRL}}(\theta, \tilde{s}_{\tilde{t}}) = \mathcal{T} \sum_{t=0}^{T}\sum_{k=1}^{K} \mathbb{E}_{a_t^k \sim q_{\theta^*}}\left[D_{\mathrm{KL}}\left(q_\theta(\cdot \mid \tilde{s}_{\tilde{t}}) \,\middle\|\, \pi(a_t^k \mid \cdot, s_t) \frac{\exp\left(\alpha\, Q_{\mathrm{Diff}}^{q_{\theta^*}}(\tilde{s}_{\tilde{t}}, \cdot)\right)}{Z(\tilde{s}_{\tilde{t}})}\right)\right].} \tag{15}$$

Where the corresponding Q- and value functions are

$$Q_{\mathrm{Diff}}^{q_{\theta^*}}(\tilde{s}_{\tilde{t}}, a_t^{k-1}) = \tilde{R}_{\mathrm{Diff}}(a_t^{k-1}, \tilde{s}_{\tilde{t}}) + V_{\mathrm{Diff}}(\tilde{s}_{\tilde{t}+1}),$$

$$V_{\mathrm{Diff}}(\tilde{s}_{\tilde{t}}) = \mathbb{E}_{a_t^{k-1} \sim q_{\theta^*}(\cdot \mid a_t^k, s_t)}\left[Q_{\mathrm{Diff}}^{q_{\theta^*}}(\tilde{s}_{\tilde{t}}, a_t^{k-1}) - \mathcal{T} \log \frac{q_{\theta^*}(a_t^{k-1} \mid a_t^k, s_t)}{\pi(a_t^k \mid a_t^{k-1}, s_t)}\right], \tag{16}$$

$\tilde{s}_{\tilde{t}} = \tilde{s}_{\tilde{t}(t,k)} = (s_t, a_t^k, k)$ and $\tilde{R}_{\mathrm{Diff}}$ is defined in Eq. 17 (see Sec. 3.1). Importantly, the value function includes the log ratio between forward and reverse diffusion transition probabilities.

3.1. Diffusion-Based Maximum Entropy Reinforcement Learning MDP

We can formulate the resulting Diffusion-based Maximum Entropy MDP by flattening the original time steps ($t = 0, \dots, T$) and the reverse diffusion steps ($k = K, \dots, 0$) into a single augmented time index ($\tilde{t}$):

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

The reward is defined such that only the actual environment action ($k = 0$) receives the environment reward:

$$\tilde{R}_{\text{Diff}}(\tilde{s}_{\tilde{t}(t,k)}, \tilde{a}_{\tilde{t}(t,k)}) = \begin{cases} 0, & k > 0, \\ R_{\text{env}}(s_t, a_t^0), & k = 0. \end{cases} \tag{17}$$

The augmented MDP transition kernel is

$$p(\tilde{s}_{\tilde{t}+1} \mid \tilde{s}_{\tilde{t}}, \tilde{a}_{\tilde{t}}) = \begin{cases} \delta\Big(\tilde{s}_{\tilde{t}+1} = (s_t, a_t^{k-1}, k-1)\Big), & k > 0, \\ p(s_{t+1} \mid s_t, a_t^0) \otimes q_{\text{prior}}(a_{t+1}^K) \otimes \delta(k-K), & k = 0 \end{cases}$$

This formulation explicitly captures the reverse diffusion steps as intermediate MDP states. For ($k > 0$), the MDP moves to the next lower diffusion step ($k-1$) while keeping the environment state (s_t) fixed. When ($k = 0$), the environment transitions forward to (s_{t+1}) and the next diffusion chain starts with (a_{t+1}^K) sampled from the prior, effectively resetting the diffusion step index to (K). This ensures that the augmented MDP correctly integrates both the environment dynamics and the diffusion-based policy structure. This MDP is the same as in DPPO (Ren et al., 2025), however, unlike in DPPO the definition of the value function is different (see Eq. 16). Thus, our formulation matches the DPPO formulation at $\mathcal{T} = 0$ and is otherwise different.

3.2. Diffusion-based Maximum Entropy Wasserstein Policy Optimization

We extend diffusion-based policies to *Wasserstein Policy Optimization (WPO)* (Pfau et al., 2025), which can be derived by projecting the Wasserstein Gradient Flow of the surrogate loss in Eq. 15 via the reverse-KL into parameter space (see App. C). Thus we obtain the following surrogate objective for **DiffWPO**:

$$\mathcal{L}_{\text{WPO}}(\theta, \tilde{s}_{\tilde{t}}) = \mathbb{E}_{a_t \sim q_{\theta^*}(a_t^{k-1} \mid a_t^k, s_t)} \Big[\nabla_{a_t^{k-1}} \Big(\mathcal{T} \log \frac{q_{\theta^*}(a_t^{k-1} \mid a_t^k, s_t)}{\pi(a_t^k \mid a_t^{k-1}, s_t)} - Q_{\text{Diff}}^{q_{\theta^*}}(\tilde{s}_{\tilde{t}}, a_t^{k-1}) \Big) \nabla_{a_t^{k-1}} \log q_\theta(a_t^{k-1} \mid a_t^k, s_t) \Big], \tag{18}$$

where the gradient of this loss should be scaled afterwards using the inverse Fisher Matrix or its approximations as in Pfau et al. (2025). For additional implementation details, see App. E.3.

4. Experiments

In this section, we evaluate the proposed diffusion-augmented RL algorithms (**DiffPPO**, **DiffSAC**, **DiffWPO**) on a suite of standard continuous control benchmarks. First, we present some ablation studies and then we compare our method against vanilla counterparts (PPO, SAC) to measure sample efficiency, stability, and final performance.

4.1. Ablation Studies

Methods improves with increasing amount of Diffusion Steps: We evaluate how the performance of DiffPPO, DiffWPO, and DiffSAC scales with the number of diffusion steps. These experiments are conducted with the **variance-preserving (VP) SDE** with prior variance $\nu = 2.2$ and a linear noise schedule β_k ranging from 3 to 0.05. A sweep is performed over three learning rates to get the best configuration in each setting. Additional hyperparameters are detailed in Appendix F.3. Figure 1 presents the results, where each diffusion-based RL algorithm is assessed using three distinct diffusion step configurations. Performance metrics are averaged across four independent seeds. Our findings indicate that increasing the number of diffusion steps enhances the efficiency of all methods, reducing the number of environment interactions required while simultaneously improving overall return.

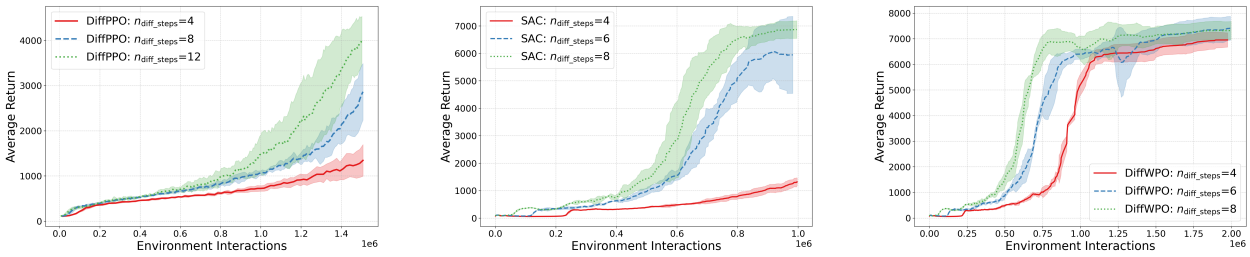
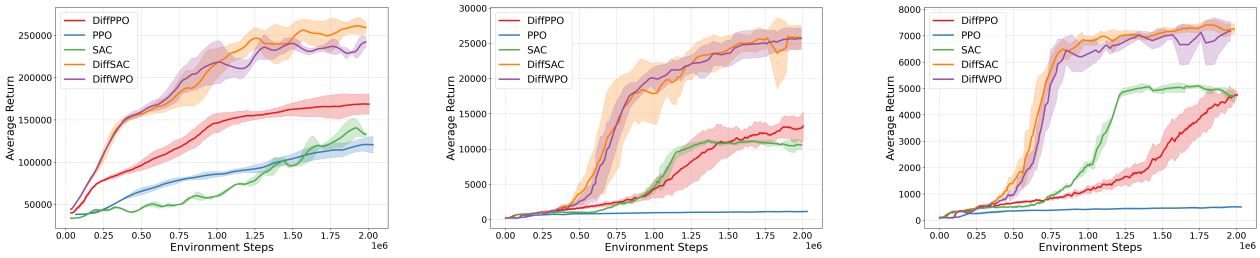


Figure 1. Effect of varying the number of diffusion steps K on the Humanoid-v4 environment for DiffPPO, DiffSAC and DiffWPO. The average and standard deviation are taken over four seeds.

4.2. Main Results

Figure 2 compares DiffPPO, DiffSAC, and DiffWPO against respective baselines, PPO and SAC, on the Humanoid, Humanoid-Run, and Humanoid-Standup benchmarks. Across all three tasks, diffusion-augmented policies exhibit consistently higher sample efficiency and achieve a higher average return. In particular, our experiments show that DiffPPO substantially increases sample efficiency in terms of environment interactions compared to PPO, enabling the agent to achieve higher returns with fewer base-environment interactions. Likewise, DiffSAC not only improves sample efficiency relative to SAC but also achieves a higher overall average return, demonstrating the benefit of integrating diffusion-based structure into off-policy training. Additionally, our results show that DiffWPO performs similarly to DiffSAC. Overall, these evaluations show that our proposed methods—DiffPPO, DiffSAC, and DiffWPO—achieve strong performance across challenging humanoid control tasks.



(a) Humanoid-standup-v4 *(b)* Humanoid-run-v4 *(c)* Humanoid-v4

Figure 2. Performance comparison of diffusion-augmented RL algorithms and their vanilla counterparts. Curves are averaged over four independent seeds; shaded areas denote standard deviation.

5. Related Work

Diffusion Models in Reinforcement Learning: Diffusion models have recently been explored in RL to represent complex action distributions. Other works often optimize forward-KL objectives (Dong et al., 2025; Ma et al., 2025) by replacing the reverse KL divergence in Eq. 3 with a forward KL divergence. However, the forward KL requires samples from the target distribution, which are not available, and thus neural importance sampling is used to address this challenge. However, this introduces bias, high variance, and *mode-covering* behavior, which is undesirable in RL because sampling suboptimal actions should be entirely avoided. DiME (Celik et al., 2025) integrates diffusion models with SAC, but in contrast to our method, does not evaluate a Q-function at each diffusion step. Their objective is based on applying the Data Processing Inequality directly on the surrogate loss in Eq. 3. Their method requires backpropagation through the entire diffusion chain for every environment action, which is memory-intensive for a large number of diffusion steps T. Furthermore, within the computation of the maximum entropy value function as in Sec. 2.1, they rely on estimating $\log q_\theta(a_t|s_t)$ using a lower bound. In contrast to this, our Diffusion-based RL approaches train the diffusion model itself using RL, as in (Sanokowski et al., 2025a), thus allowing for more memory-efficient training. DPPO (Ren et al., 2025) corresponds to a special case of DiffPPO when $\mathcal{T} = 0$. Our DiffPPO formulation generalizes this method to arbitrary temperatures.

6. Conclusion

In this work, we introduced a principled framework for generalizing reinforcement learning to diffusion-based policies, which we name **Diffusion-based Maximum Entropy Reinforcement Learning (DMERL)**. This unified approach, derived from reverse KL minimization, generalizes and simplifies prior diffusion-based RL methodologies.

We further make key theoretical findings: first, establishing a direct equivalence by showing that the off-policy Maximum Entropy RL surrogate loss (e.g., used in SAC) can be derived from the log-variance (trajectory balance) loss (Richter et al., 2020; Malkin et al., 2022a); and second, deriving a Maximum Entropy formulation of the recently proposed Wasserstein Policy Optimization (WPO) (Pfau et al., 2025).

Building on our DMERL framework, we proposed three novel diffusion-augmented algorithms: **DiffPPO**, **DiffSAC**, and **DiffWPO**. These methods are practical and easily implementable, requiring only minor modifications to standard libraries, such as Stable-Baselines3 (Raffin et al., 2021). We demonstrated that they achieve strong performance on challenging continuous-control benchmarks, substantially improving both sample efficiency and final average returns compared to their respective base algorithms (SAC and PPO). Our ablation studies show that performance improves as the number of diffusion steps increases.

While we show that DMERL methods already achieve promising results, their performance could be further enhanced by incorporating architectural and algorithmic improvements such as using the Bro Architecture (Nauman et al., 2024), CrossQ learning in the critic (Bhatt et al., 2019), and Distributional RL (Bellemare et al., 2017) as used in DiMES (Celik et al., 2025). Finally, we note that our current implementation is not optimized for maximum efficiency, as diffusion environment steps are executed through the `SubprocVecEnv` in Stable Baselines 3. This design choice introduces avoidable overhead as the intermediate diffusion MDP steps can, in principle, be vectorized, which would make the overall algorithm substantially more efficient.

Looking ahead, there are several promising directions for future work. First, our framework could be extended to use *diffusion bridge samplers* for policy parametrization (Vargas et al., 2024; Richter & Berner, 2024; Sanokowski et al., 2025b) for more efficient trajectory-level modeling. Second, this framework can in principle also be applied to discrete state and action spaces, particularly where the number of possible actions is very large, making direct sampling intractable. Examples of this include combinatorial optimization tasks such as those explored in NeuralCO (Karalias & Loukas, 2020). Notably, our framework offers a more general approach than some existing methods, such as those in (Sanokowski et al., 2025a), by inherently allowing for probabilistic transition environment states. Finally, such a diffusion-based RL framework in discrete state spaces can be applied to many other settings, such as reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022) for Diffusion Language Models (DiffusionLLMs) (Nie et al., 2025), supporting memory-efficient RLHF-based finetuning.

Impact Statement

Diffusion-based reinforcement learning has the potential to improve decision-making systems in real-world applications where sample efficiency and safe exploration are critical. By enabling policies that can model complex, multimodal action distributions, our framework could lead to more adaptive and reliable robotics, autonomous systems, and other AI agents that interact with dynamic environments.

At the same time, as with all powerful generative and decision-making models, there are societal risks. More capable RL agents could be misused in autonomous systems for harmful purposes or inadvertently reinforce biased or unsafe behaviors if trained inappropriately. Careful evaluation, robust safety constraints, and ethical deployment practices are essential to ensure these technologies benefit society responsibly.

Acknowledgements

This work utilized high-performance computing resources, which were indispensable for training and evaluating our diffusion-based reinforcement learning models. We gratefully acknowledge the extensive support and computational access provided by the EUROHPC Joint Undertaking. We would like to specifically thank the VEGA supercomputing facility for providing a reliable computing environment and a generous allocation of GPU hours. The Karolina cluster was essential for accelerating our experimentation and enabling the large-scale batch rollouts and ablation studies. Furthermore, we extend our thanks to the team operating Meluxina for their technical support and allowing us to run long-horizon experiments under

demanding memory and compute conditions. The results presented in this work would not have been possible without the computational power and infrastructure reliability provided by these institutions.

Author Contributions: S.S. developed the theory, implemented the algorithm, and conducted the primary experiments. K.P assisted with experiments. A.K provided funding support.

References

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

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

Berner, J., Richter, L., and Ullrich, K. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

Bhatt, A., Palenicek, D., Belousov, B., Argus, M., Amiranashvili, A., Brox, T., and Peters, J. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. *arXiv preprint arXiv:1902.05605*, 2019.

Celik, O., Li, Z., Blessing, D., Li, G., Palenicek, D., Peters, J., Chalvatzaki, G., and Neumann, G. Dime: Diffusion-based maximum entropy reinforcement learning. *arXiv preprint arXiv:2502.02316*, 2025.

Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Csiszár, I. On information-type measure of difference of probability distributions and indirect observations. *Studia Sci. Math. Hungar.*, 2:299–318, 1967.

Dong, X., Cheng, J., and Zhang, X. S. Maximum entropy reinforcement learning with diffusion policy. *arXiv preprint arXiv:2502.11612*, 2025.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. 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.

Hibat-Allah, M., Inack, E. M., Wiersema, R., Melko, R. G., and Carrasquilla, J. 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.

Ho, J., Jain, A., and Abbeel, P. 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.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

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

Karalias, N. and Loukas, A. 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.

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

Ma, H., Nabati, O., Rosenberg, A., Dai, B., Lang, O., Szpektor, I., Boutilier, C., Li, N., Mannor, S., Shani, L., et al. Reinforcement learning with discrete diffusion policies for combinatorial action spaces. *arXiv preprint arXiv:2509.22963*, 2025.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022a.

Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E., Everett, K., Zhang, D., and Bengio, Y. Gflownets and variational inference. *arXiv preprint arXiv:2210.00580*, 2022b.

Nauman, M., Ostaszewski, M., Jankowski, K., Miłoś, P., and Cygan, M. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. *Advances in neural information processing systems*, 37:113038–113071, 2024.

Neklyudov, K., Nys, J., Thiede, L., Carrasquilla, J., Liu, Q., Welling, M., and Makhzani, A. 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.

Nie, S., Zhu, F., You, Z., Zhang, X., Ou, J., Hu, J., Zhou, J., Lin, Y., Wen, J.-R., and Li, C. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Pfau, D., Davies, I., Borsa, D. L., Araújo, J. G. M., Tracey, B. D., and van Hasselt, H. Wasserstein policy optimization. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=oAKe7MG9GM.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Ren, A. Z., Lidard, J., Ankile, L. L., Simeonov, A., Agrawal, P., Majumdar, A., Burchfiel, B., Dai, H., and Simchowitz, M. Diffusion policy policy optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mEpqHvbD2h.

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

Richter, L., Boustati, A., Nüsken, N., Ruiz, F., and Akyildiz, O. D. Vargrad: a low-variance gradient estimator for variational inference. *Advances in Neural Information Processing Systems*, 33:13481–13492, 2020.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Sanokowski, S., Berghammer, W., Hochreiter, S., and Lehner, S. Variational annealing on graphs for combinatorial optimization. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/c9c54ac0dd5e942b99b2b51c297544fd-Abstract-Conference.html.

Sanokowski, S., Hochreiter, S., and Lehner, S. 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.

Sanokowski, S., Berghammer, W., Ennemoser, M., Wang, H. P., Hochreiter, S., and Lehner, S. 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.

Sanokowski, S., Gruber, L., Bartmann, C., Hochreiter, S., and Lehner, S. Rethinking losses for diffusion bridge samplers. *arXiv preprint arXiv:2506.10982*, 2025b.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. 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.

Song, Y., Durkan, C., Murray, I., and Ermon, S. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021a.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. 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.

Sun, H., Guha, E. K., and Dai, H. Annealed training for combinatorial optimization on graphs. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*, 2022. URL https://openreview.net/forum?id=fo3b0XjTkU.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

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

Vargas, F., Grathwohl, W., and Doucet, A. Denoising diffusion samplers. *arXiv preprint arXiv:2302.13834*, 2023.

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

Zhang, Q. and Chen, Y. 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.

Ziebart, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

A. Reverse Log-Derivative Trick and Derivation of the Surrogate KL Objective

In this appendix, we provide the complete derivation from the trajectory-level KL divergence

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

to the per-state surrogate KL objective used in maximum-entropy reinforcement learning. We first show how the environment dynamics cancel, then explain why the policy gradient theorem applies, and finally show how the reverse log-derivative trick leads to a tractable surrogate loss.

A.1. Trajectory Distributions and Cancellation of Dynamics

Recall the trajectory distributions:

$$q_\theta(a_{0:T}, s_{0:T+1}) = \prod_{k=0}^{T} p(s_{k+1} \mid s_k, a_k)\, q_\theta(a_k \mid s_k)\, p(s_0),$$

$$\pi(a_{0:T}, s_{0:T+1}) = \prod_{k=0}^{T} p(s_{k+1} \mid s_k, a_k)\, \pi(a_k \mid s_k)\, p(s_0).$$

Both distributions contain the same environment dynamics and initial state distribution. Inside log probability ratios $\log \frac{q_\theta(a_{0:T}, s_{0:T})}{\pi(a_{0:T}, s_{0:T})}$ within the KL divergence we have,

$$\log p(s_{k+1} \mid s_k, a_k) - \log p(s_{k+1} \mid s_k, a_k) = 0, \qquad \log p(s_0) - \log p(s_0) = 0,$$

so these terms cancel exactly. Thus the trajectory-level KL reduces to a sum of per-timestep policy KL terms:

$$\log q_\theta(a_{0:T}, s_{0:T}) - \log \pi(a_{0:T}, s_{0:T}) = \sum_{t=0}^{T} \big(\log q_\theta(a_t \mid s_t) - \log \pi(a_t \mid s_t)\big).$$

with $\log \pi(a_t \mid s_t) = R_{\mathrm{env}}(a_t, s_t) + C_t$ leading with $C := \sum_{t=0}^{T} C_t$ to

$$\boxed{D_{\mathrm{KL}}^{\mathcal{T}}\big(q_\theta(a_{0:T}, s_{0:T}) \,\big\|\, \pi(a_{0:T}, s_{0:T})\big) = \sum_{t=0}^{T} \mathbb{E}_{s_t, a_t \sim q_\theta(a_t, s_t)}\big[\mathcal{T} \log q_\theta(a_t \mid s_t) - R_{\mathrm{env}}(s_t, a_t)\big] + C.}$$

A.2. Why the Policy Gradient Theorem Applies

We begin with the reverse–KL objective

$$D_{\mathrm{KL}}(q_\theta \,\|\, \pi) = \mathbb{E}_{q_\theta}\left[\sum_{t=0}^{T} \big(\log q_\theta(a_t \mid s_t) - \log \pi(a_t \mid s_t)\big)\right].$$

To express this in the standard RL framework, we rewrite it as the expectation of a sum of *modified rewards*. Define

$$\tilde{r}(s_t, a_t) := R_{\mathrm{env}}(s_t, a_t) - \mathcal{T} \log q_\theta(a_t \mid s_t), \qquad \tilde{R}(\tau) = \sum_{t=0}^{T} \tilde{r}(s_t, a_t).$$

The key identity

$$\mathbb{E}_{a_t \sim q_\theta}[\nabla_\theta \log q_\theta(a_t \mid s_t)] = 0$$

implies that terms of the form $f(a_t, s_t)\, \nabla_\theta \log q_\theta(a_t \mid s_t)$ behave, for gradient purposes, exactly as if f were independent of θ. Thus, the reward dependence of $-\mathcal{T} \log q_\theta(a_t \mid s_t)$ on θ does not introduce any additional terms in the gradient, and we can treat $\tilde{r}(s_t, a_t)$ as a valid reward function for the purpose of applying the policy gradient theorem.

Hence, the KL-based objective is equivalent (up to a sign) to the standard RL objective

$$J(\theta) = \mathbb{E}_{\tau\sim q_\theta}\left[\sum_{t=0}^{T} \tilde{r}(s_t, a_t)\right], \qquad \text{with} \quad J(\theta) = -D_{\mathrm{KL}}(q_\theta \| \pi).$$

Since $J(\theta)$ is now the expected cumulative reward under policy q_θ, the policy gradient theorem (Sutton & Barto, 2018) applies directly:

$$\boxed{\nabla_\theta D_{\mathrm{KL}}(q_\theta \| \pi) = \sum_{t=0}^{T} \mathbb{E}_{s_t, a_t \sim q_\theta}[-\tilde{r}(s_t, a_t)\, \nabla_\theta \log q_\theta(a_t \mid s_t)]\,.}$$

Substituting $\tilde{r}(s_t, a_t) = R_{\mathrm{env}}(s_t, a_t) - \mathcal{T} \log q_\theta(a_t \mid s_t)$ yields the expression stated in the main text.

A.3. Reverse Log-Derivative Trick

Define the unnormalized Boltzmann policy:

$$\pi(a \mid s) \propto \exp(\beta Q^{q_\theta}(s, a)).$$

The KL divergence

$$D_{\mathrm{KL}}\big(q_\theta(a_t|s_t) \,\|\, \pi(a_t|s_t)\big)$$

has gradient

$$\nabla_\theta D_{\mathrm{KL}}^{\mathcal{T}} = \mathbb{E}_{q_\theta}\left[(\mathcal{T} \log q_\theta(a_t|s_t) - Q^{q_\theta}(s_t, a_t))\, \nabla_\theta \log q_\theta(a_t|s_t)\right],$$

which proves the identity.

The target distribution $\pi(a \mid s)$ depends on θ through the critic Q^{q_θ}. When applying the reverse log-derivative trick, the gradient must only act on the *explicit* dependence of the KL term on q_θ. To make the gradient match the true KL gradient, we must *treat the inside of the Boltzmann distribution as a constant* with respect to θ:

$$Q^{q_\theta}(s, a) \;\longrightarrow\; Q^{q_{\theta^*}}(s, a),$$

where θ^* indicates a stop-gradient.

This is not a heuristic modification: The stop-gradient arises directly from applying the log-derivative trick in reverse, ensuring that the gradient of the surrogate KL exactly matches the gradient of the original trajectory KL for the current iterate.

The resulting surrogate loss is

$$\boxed{\mathcal{L}(\theta) = \mathcal{T} \sum_{t=0}^{T} \mathbb{E}_{s_t \sim q_{\theta^*}}\left[D_{\mathrm{KL}}\left(q_\theta(a_t|s_t) \,\middle\|\, \frac{\exp(\beta Q^{q_{\theta^*}}(s_t, a_t))}{Z(s_t)}\right)\right].} \tag{19}$$

This surrogate objective satisfies the following property:

The gradient of the surrogate loss equals the gradient of the original trajectory KL exactly at the current iterate $\theta = \theta^*$.

After the first update, the gradients of the surrogate and true objectives deviate. This mirrors the behavior of most modern RL algorithms—including PPO or TRPO (Schulman et al., 2015), which optimize surrogate objectives that are locally exact but globally approximate. This approximation is crucial for computational tractability and is a necessary design choice.

B. Policy-gradient style derivation for the Log Variance loss

Notation and assumptions

Consider finite-horizon trajectories written explicitly as

$$(a_{0:T}, s_{0:T+1}) \equiv (s_0, a_0, s_1, a_1, \ldots, s_T, a_T, s_{T+1}).$$

Let $\rho(s_0)$ be the initial state distribution, $q_\theta(a_t \mid s_t)$ the parametric policy (the object we differentiate), and $p(s_{t+1} \mid s_t, a_t)$ the Markov environment dynamics. The reference trajectory distribution π is fixed and independent of θ. We will often assume compatible factorizations for q_θ and π, namely

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

$$\pi(a_{0:T}, s_{0:T+1}) = \rho(s_0) \prod_{t=0}^{T} \pi(a_t \mid s_t)\, p(s_{t+1} \mid s_t, a_t),$$

but the key requirement is only that π does not depend on θ. Let ω denote an arbitrary (possibly off-policy) distribution over full trajectories $(a_{0:T}, s_{0:T+1})$. We assume ω is fixed (does not depend on θ).

Define the per-trajectory log-ratio

$$\ell_\theta(a_{0:T}, s_{0:T+1}) := \log \frac{q_\theta(a_{0:T}, s_{0:T+1})}{\pi(a_{0:T}, s_{0:T+1})} = \sum_{t=0}^{T} \log \frac{q_\theta(a_t \mid s_t)}{\pi(a_t \mid s_t)}$$

The LV loss is defined under the sampling distribution ω as:

$$D_{LV}^{\omega}(q_\theta \| \pi) \;=\; \frac{1}{2}\, \mathbb{E}_{(a_{0:T}, s_{0:T+1}) \sim \omega} \big[\, (\ell_\theta(a_{0:T}, s_{0:T+1}) - b_\theta^{\omega})^2 \big],$$

with the baseline

$$b_\theta^{\omega} := \mathbb{E}_{(a_{0:T}, s_{0:T+1}) \sim \omega} \big[\ell_\theta(a_{0:T}, s_{0:T+1}) \big].$$

Thus, the gradient is given by:

$$\begin{aligned} \nabla_\theta D_{LV}^{\omega} &= \tfrac{1}{2}\, \nabla_\theta \mathbb{E}_\omega \big[(\ell_\theta - b_\theta^{\omega})^2 \big] \\ &= \mathbb{E}_\omega \Big[(\ell_\theta - b_\theta^{\omega}) \big(\nabla_\theta \ell_\theta - \nabla_\theta b_\theta^{\omega} \big) \Big] \\ &= \mathbb{E}_\omega \big[(\ell_\theta - b_\theta^{\omega}) \nabla_\theta \ell_\theta \big] - \mathbb{E}_\omega[\ell_\theta - b_\theta^{\omega}]\, \mathbb{E}_\omega[\nabla_\theta \ell_\theta]. \end{aligned}$$

By definition $\mathbb{E}_\omega[\ell_\theta - b_\theta^{\omega}] = 0$, so the second term vanishes. Thus we have the compact unbiased form

$$\boxed{\nabla_\theta D_{LV}^{\omega} = \mathbb{E}_{(a_{0:T}, s_{0:T+1}) \sim \omega} \big[(\ell_\theta(a_{0:T}, s_{0:T+1}) - b_\theta^{\omega})\, \nabla_\theta \ell_\theta(a_{0:T}, s_{0:T+1}) \big].}$$

Because π is fixed and the dynamics p and ρ are independent of θ,

$$\nabla_\theta \ell_\theta(a_{0:T}, s_{0:T+1}) = \nabla_\theta \log q_\theta(a_{0:T}, s_{0:T+1}) = \sum_{t=0}^{T} \nabla_\theta \log q_\theta(a_t \mid s_t).$$

Substituting into the boxed expression and interchanging sums and expectation yields

$$\begin{aligned} \nabla_\theta D_{LV}^{\omega} &= \mathbb{E}_{(a_{0:T}, s_{0:T+1}) \sim \omega} \left[\big(\ell_\theta(a_{0:T}, s_{0:T+1}) - b_\theta^{\omega} \big) \sum_{t=0}^{T-1} \nabla_\theta \log q_\theta(a_t \mid s_t) \right] \\ &= \sum_{t=0}^{T-1} \mathbb{E}_{(a_{0:T}, s_{0:T+1}) \sim \omega} \Big[\nabla_\theta \log q_\theta(a_t \mid s_t) \, \big(\ell_\theta(a_{0:T}, s_{0:T+1}) - b_\theta^{\omega} \big) \Big]. \end{aligned}$$

We can now split the total log-ratio ℓ_θ and the baseline b_θ^{ω} into "Past" (indices 0 to $t-1$) and "Future" (indices t to T).

$$\begin{aligned} \ell_\theta(a_{0:T}, s_{0:T+1}) &= \sum_{i=0}^{t-1} \log \frac{q_\theta(a_i \mid s_i)}{\pi(a_i \mid s_i)} + \sum_{i=t}^{T} \log \frac{q_\theta(a_i \mid s_i)}{\pi(a_i \mid s_i)} := \ell_\theta(a_{0:t-1}, s_{0:t-1}) + \ell_\theta(a_{t:T}, s_{t:T+1}) \\ b_\theta^{\omega} &= b_{0:t-1}^{\omega} + b_{t:T}^{\omega} := \mathbb{E}_{(a_{0:t}, s_{0:t}) \sim \omega} \big[\ell_\theta(a_{0:t-1}, s_{0:t-1}) \big] + \mathbb{E}_{(a_{t:T}, s_{t:T+1}) \sim \omega(\cdot \mid s_t)\omega(s_t)} \big[\ell_\theta(a_{t:T}, s_{t:T+1}) \big] \end{aligned}$$

Substitute this into the gradient expression for a single time step t:

$$\mathbb{E}_{(a_{0:T},s_{0:T+1})\sim\omega}\Big[\nabla_\theta \log q_\theta(a_t \mid s_t)\,\Big((\ell_\theta(a_{0:t-1},s_{0:t-1}) - b^\omega_{0:t-1}) + (\ell_\theta(a_{t:T},s_{t:T+1}) - b^\omega_{t:T})\Big)\Big].$$

Integrating over $(a_{0:t-1}, s_{0:t-1})$ then yields:

$$\mathbb{E}_{(a_{t:T},s_{t:T+1})\sim\omega(\cdot|s_t)\omega(s_t)}\Big[\nabla_\theta \log q_\theta(a_t \mid s_t)\,\Big((b^\omega_{0:t-1} - b^\omega_{0:t-1}) + (\ell_\theta(a_{t:T},s_{t:T+1}) - b^\omega_{t:T}\Big)\Big] \tag{20}$$

$$= \mathbb{E}_{(a_{t:T},s_{t:T+1})\sim\omega(\cdot|s_t)\omega(s_t)}\Big[\nabla_\theta \log q_\theta(a_t \mid s_t)\,\Big((\ell_\theta(a_{t:T},s_{t:T+1}) - b^\omega_{t:T}\Big) \tag{21}$$

With the past terms cancelled, we are left only with the future components:

$$\nabla_\theta D^\omega_{LV} = \sum_{t=0}^{T} \mathbb{E}_{(a_{t:T},s_{t:T+1})\sim\omega(\cdot|s_t)\omega(s_t)}\Big[\nabla_\theta \log q_\theta(a_t \mid s_t)\,\big((\ell_\theta(a_{t:T},s_{t:T+1}) - b^\omega_{t:T}\big)\Big].$$

Conditioning on (a_t, s_t) and defining $Q^{\omega,q_{\theta^*}}_\ell(a_t,s_t) = \log\pi(a_t \mid s_t) - \mathbb{E}_{(a_{t+1:T},s_{t+1:T+1})\sim\omega}[(\ell_\theta(a_{t+1:T},s_{t+1:T+1})] = R_{\text{env}}(s_t,a_t) - b^\omega_{t+1:T}$. By defining $V^{\omega,q_{\theta^*}}_t := \mathbb{E}_{s_{t+1}\sim\omega(s_{t+1})}[V^{\omega,q_{\theta^*}}(s_{t+1})] = -b^\omega_{t+1:T}(s_{t+1})$ and remember that $\log\pi(a_t \mid s_t) = R_{\text{env}}(s_t,a_t)$ we arrive at:

$$\mathcal{T}\,\nabla_\theta D^\omega_{LV} = \sum_{t=0}^{T} \mathbb{E}_{(a_t,s_t)\sim\omega}\Big[\nabla_\theta \log q_\theta(a_t \mid s_t)\,\Big(\mathcal{T}\log q_\theta(a_t \mid s_t) - \Big(Q^\omega_\ell(a_t,s_t) - V^{\omega,q_{\theta^*}}_t\Big)\Big)\Big].$$

where:

$$\begin{aligned} Q^{\omega,q_{\theta^*}}(s_t,a_t) &= R_{\text{env}}(s_t,a_t) + V^{\omega,q_{\theta^*}}(s_{t+1}),\\ V^{\omega,q_{\theta^*}}(s_t) &= \mathbb{E}_{a_t\sim\omega(a_t|s_t)}\Big[Q^{\omega,q_{\theta^*}}(s_t,a_t) - \mathcal{T}\log q_{\theta^*}(a_t \mid s_t)\Big]. \end{aligned}$$

When $\omega(s_t,a_t) = \mathcal{B}(s_t)\,q_{\theta^*}(a_t \mid s_t)$, i.e. at every step t sampling actions a_t is done on policy, any baseline can be used and we can pull the expectation in $V^{\omega,q_{\theta^*}}_t = \mathbb{E}_{s_{t+1}\sim\omega(s_{t+1})}[V^{\omega,q_{\theta^*}}(s_{t+1})]$ out and arrive at:

$$\boxed{\mathcal{T}\,\nabla_\theta D^\omega_{LV} = \sum_{t=0}^{T} \mathbb{E}_{\mathbb{E}_{(a_t,s_t)\sim(q_{\theta^*},\mathcal{B})}}\Big[\nabla_\theta \log q_\theta(a_t \mid s_t)\,\Big(\mathcal{T}\log q_\theta(a_t \mid s_t) - \Big(Q^\omega_\ell(a_t,s_t) - V^{\omega,q_{\theta^*}}(s_t)\Big)\Big)\Big].}$$

C. Derivation of Maximum Entropy Wasserstein Policy Optimization

Functional derivative. Expanding the KL gives (up to a constant independent of q)

$$\mathcal{L}(q,s) = \int q(a \mid s)\big(\log q(a \mid s) - \alpha Q^{q_{\theta^*}}(s,a)\big)\,da + \text{const.}$$

Hence the functional derivative w.r.t. the density q is

$$\frac{\delta\mathcal{L}(q,s)}{\delta q}(s,a) = \log q(a \mid s) - \alpha Q^{q_{\theta^*}}(s,a) + \text{const.} \tag{22}$$

The additive constant (including $\log Z(s)$ and the $+1$ from $\delta\int q\log q/\delta q$) does not affect spatial gradients in a and can therefore be dropped for the flow.

Wasserstein gradient flow (descent). We seek the steepest descent of $\mathcal{J}_s$ in the 2-Wasserstein metric; the corresponding Wasserstein gradient flow (continuity equation form) is

$$\frac{\partial q_\theta(a \mid s)}{\partial t} = -\nabla_a\cdot\Big(q_\theta(a \mid s)\,\nabla_a\frac{\delta\mathcal{L}(q,s)}{\delta q}(s,a)\Big) = -\nabla_a\cdot\Big(q_\theta(a \mid s)\,\nabla_a\big(\log q_\theta(a \mid s) - \alpha Q^{q_{\theta^*}}(s,a)\big)\Big). \tag{23}$$

For compactness define the velocity field

$$v(a) := \nabla_a\big(\log q_\theta(a \mid s) - \alpha Q^{q_{\theta^*}}(s,a)\big).$$

C.1. Projection of Wasserstein Flows onto a Parametric Policy Family

To convert the Wasserstein gradient flow in Eq. (23) into a practical update for a parametric policy $q_\theta(a \mid s)$, we project the induced flow onto the space of densities representable by θ. Concretely, we choose the parameter perturbation $\Delta\theta$ that minimizes the KL divergence between the infinitesimally flowed density and the perturbed parametric density:

$$\Delta\theta = \arg\min_{\delta\theta} D_{\mathrm{KL}}\Big[q_\theta \,\Big\|\, q_\theta + \frac{\partial q_\theta}{\partial t}dt - \nabla_\theta q_\theta\, \delta\theta\Big].$$

Locally, the KL can be approximated by a quadratic form defined by the Fisher information blocks:

$$\mathrm{D}_{\mathrm{KL}} \approx \begin{pmatrix} dt \\ -\Delta\theta \end{pmatrix}^T \begin{pmatrix} \mathcal{F}_{tt} & \mathcal{F}_{t\theta}^T \\ \mathcal{F}_{t\theta} & \mathcal{F}_{\theta\theta} \end{pmatrix} \begin{pmatrix} dt \\ -\Delta\theta \end{pmatrix},$$

$$\mathcal{F}_{tt} = \mathbb{E}_{q_\theta}[(\partial_t \log q_\theta)^2], \qquad \mathcal{F}_{t\theta} = \mathbb{E}_{q_\theta}[\partial_t \log q_\theta\, \nabla_\theta \log q_\theta], \qquad \mathcal{F}_{\theta\theta} = \mathbb{E}_{q_\theta}[\nabla_\theta \log q_\theta\, \nabla_\theta \log q_\theta^T].$$

Minimizing this quadratic form gives the optimal parameter update

$$\Delta\theta = \mathcal{F}_{\theta\theta}^{-1}\mathcal{F}_{t\theta},$$

where the mixed block $\mathcal{F}_{t\theta}$ captures the correlation between the flow in action space and the parametric gradients. For the reverse–KL functional, $\mathcal{F}_{t\theta}$ reduces to

$$\mathcal{F}_{t\theta} = \mathbb{E}_{a\sim q_\theta}\big[\nabla_\theta \nabla_a \log q_\theta(a \mid s)\; \nabla_a(\log q_\theta(a \mid s) - \alpha Q^{q_{\theta^*}}(s,a))\big],$$

which leads directly to the Maximum Entropy WPO update in Eq. (9).

Projection to parameter space. As explained in (Neklyudov et al., 2023; Pfau et al., 2025) the induced flow can be projected on the parametric family q_θ by minimizing the local KL between the flowed density and the parametric perturbation. The mixed Fisher block is

$$\mathcal{F}_{t\theta} \;=\; \int \nabla_\theta \log q_\theta(a \mid s)\; \frac{\partial q_\theta(a \mid s)}{\partial t}\; da. \tag{24}$$

Insert the flow from Eq. 23:

$$\mathcal{F}_{t\theta} = -\int \nabla_\theta \log q_\theta(a \mid s)\; \nabla_a \cdot \big(q_\theta(a \mid s)\, v(a)\big)\; da.$$

Expand the divergence and apply integration by parts. Using $\nabla_a \cdot (qv) = q\,\nabla_a \cdot v + v \cdot \nabla_a q$ we obtain

$$\begin{aligned} \mathcal{F}_{t\theta} &= -\int \nabla_\theta \log q_\theta \Big(q_\theta \nabla_a \cdot v + v \cdot \nabla_a q_\theta\Big) da \\ &= -\int \nabla_\theta q_\theta\, \nabla_a \cdot v\, da \;-\; \int \nabla_\theta \log q_\theta\, v \cdot \nabla_a q_\theta\, da. \end{aligned} \tag{25}$$

For the first integral we apply the divergence theorem (integration by parts):

$$-\int \nabla_\theta q_\theta(a \mid s)\; \nabla_a \cdot v(a)\; da = -\int_{\partial\Omega} \big(\nabla_\theta q_\theta(a \mid s)\, v(a)\big) \cdot n(a)\, dS(a) + \int \nabla_a\big(\nabla_\theta q_\theta(a \mid s)\big) \cdot v(a)\, da, \tag{26}$$

where $\Omega = \mathbb{R}^n$ is the action space and the surface integral is the boundary term. Under the standard regularity / tail-decay assumptions for parametric policies (e.g. Gaussian tails, or other densities for which $\nabla_\theta q_\theta$ vanishes sufficiently fast at $|a| \to \infty$) the surface integral vanishes. With the boundary term dropped we continue:

Thus we arrive at:

$$\mathcal{F}_{t\theta} = \int \nabla_a\big(\nabla_\theta q_\theta\big) \cdot v\; da - \int \nabla_\theta \log q_\theta\; v \cdot \nabla_a q_\theta\; da.$$

Using the identities

$$\nabla_a q_\theta = q_\theta \, \nabla_a \log q_\theta, \qquad \nabla_\theta q_\theta = q_\theta \, \nabla_\theta \log q_\theta,$$

we expand the first integrand:

$$\nabla_a(\nabla_\theta q_\theta) = \nabla_a\big(q_\theta \nabla_\theta \log q_\theta\big) = (\nabla_a q_\theta)\, \nabla_\theta \log q_\theta + q_\theta \, \nabla_a \nabla_\theta \log q_\theta.$$

Thus

$$\int \nabla_a(\nabla_\theta q_\theta) \cdot v \, da = \int \Big[(\nabla_a q_\theta)\, (\nabla_\theta \log q_\theta \cdot v) + q_\theta \, (\nabla_a \nabla_\theta \log q_\theta \cdot v)\Big] da.$$

Hence the terms including $(\nabla_a q_\theta)\, (\nabla_\theta \log q_\theta \cdot v)$ cancel and we arrive at:

$$\mathcal{F}_{t\theta} = \int q_\theta(a \mid s)\, (\nabla_a \nabla_\theta \log q_\theta(a \mid s) \cdot v(a))\, da.$$

Divide by q_θ and write the integral as an expectation:

$$\mathcal{F}_{t\theta} = \mathbb{E}_{a \sim q_\theta(\cdot|s)}[\nabla_\theta \nabla_a \log q_\theta(a \mid s) \;\cdot\; v(a)]\,.$$

Using $v(a) = \nabla_a(\log q_\theta(a \mid s) - \alpha Q^{q_{\theta^*}}(s,a))$, we obtain

$$\boxed{\mathcal{F}_{t\theta} = \mathbb{E}_{a \sim q_\theta(\cdot|s)}\big[\nabla_\theta \nabla_a \log q_\theta(a \mid s)\; \nabla_a\big(\log q_\theta(a \mid s) - \alpha Q^{q_{\theta^*}}(s,a)\big)\big]\,.}$$

Final Update Formula: Let $\mathcal{F}_{\theta\theta}$ denote the Fisher information matrix

$$\mathcal{F}_{\theta\theta} \;=\; \mathbb{E}_{a \sim q_\theta(\cdot|s)}\big[\nabla_\theta \log q_\theta \, \nabla_\theta \log q_\theta^\top\big].$$

Thus the parameter increment

$$\Delta\theta \;=\; \mathcal{F}_{\theta\theta}^{-1}\, \mathcal{F}_{t\theta}.$$

is given by:

$$\boxed{\theta_{t+1} = \theta_t + \eta\, \mathcal{F}_{\theta\theta}^{-1}\, \mathbb{E}_{s \sim q_{\theta^*},\, a \sim q_\theta}\Big[\nabla_\theta \nabla_a \log q_\theta(a \mid s)\; \nabla_a\big(\alpha Q^{q_{\theta^*}}(s,a) - \mathcal{T} \log q_{\theta^*}(a \mid s)\big)\Big].} \tag{27}$$

Where we have included the outer scaling $\mathcal{T}$ and averaging over states sampled from q_{θ^*} (the stop-gradient sampling distribution).

Practical approximations. Equation (27) requires the mixed derivative $\nabla_\theta \nabla_a \log q_\theta$ and a (possibly large) Fisher matrix inverse $\mathcal{F}_{\theta\theta}^{-1}$. In practice we (as in (Pfau et al., 2025)) approximate expectations with samples, and use tractable approximations to $\mathcal{F}_{\theta\theta}^{-1}$ (diagonal, block diagonal, K-FAC, or other). The heuristic scaling used in (Pfau et al., 2025) (e.g. scaling of ∇_μ and ∇_σ when backpropagating through a Gaussian policy) can be applied unchanged here. After these approximations the update reduces to the working form shown in the main text.

D. Policy Gradient Theorem for Diffusion Policies

To rewrite the KL objective in Eq. 14 in a form suitable for the direct application of the policy-gradient theorem, we introduce a modified reward that absorbs both the diffusion-consistency terms and, at $k = 0$, the environment reward:

$$\tilde{R}_{\text{MaxEntDiff}}(a_t^{k-1}, a_t^k, s_t) := \begin{cases} -\mathcal{T} \log \dfrac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi(a_t^k \mid a_t^{k-1}, s_t)} & \text{if } k > 0, \\[2ex] R_{\text{env}}(s_t, a_t^0) - \mathcal{T} \log \dfrac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi(a_t^k \mid a_t^{k-1}, s_t)} & \text{if } k = 0. \end{cases} \tag{28}$$

The reverse-diffusion distribution $\pi(a_t^k \mid a_t^{k-1}, s_t)$ does not depend on θ. Furthermore, for any q_θ we have the identity

$$\mathbb{E}_{a\sim q_\theta}[\nabla_\theta \log q_\theta(a)] = 0.$$

Therefore the term $\log q_\theta(a_t^{k-1} \mid a_t^k, s_t)$ that appears inside the modified reward may be replaced by

$$\log q_{\theta^*}(a_t^{k-1} \mid a_t^k, s_t),$$

where θ^* indicates a *stop–gradient* (i.e., treated as a constant). Under this convention the reward $\tilde{R}_{\mathrm{MaxEntDiff}}$ is independent of the policy parameters θ.

Consequently, the KL objective can be written as an expectation of a parameter-independent per-step reward. Thus $\tilde{R}_{\mathrm{MaxEntDiff}}$ is a valid reinforcement-learning reward signal, and the classical policy-gradient theorem applies directly without further modification or justification.

D.1. Resulting policy gradient

Applying the policy-gradient theorem to the parameter-independent reward $\tilde{R}_{\mathrm{MaxEntDiff}}$, we obtain

$$\nabla_\theta D_{\mathrm{KL}}^{\mathcal{T}} = \mathbb{E}_{q_\theta}\left[\sum_{t=0}^{T}\sum_{k=1}^{K} \tilde{R}_{\mathrm{MaxEntDiff}}(a_t^{k-1}, a_t^k, s_t)\, \nabla_\theta \log q_\theta(a_t^k \mid a_t^{k+1}, s_t)\right]. \tag{29}$$

Introducing the diffusion value and advantage functions defined in the main text, this can equivalently be written in advantage form:

$$\nabla_\theta D_{\mathrm{KL}}^{\mathcal{T}} = \mathbb{E}_{q_\theta}\left[\sum_{t,k}\left(\mathcal{T}\log\frac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi(a_t^k \mid a_t^{k-1}, s_t)} - R_{\mathrm{env}}(s_t, a_t^0)\,\mathbf{1}_{\{k=0\}}\right)\nabla_\theta \log q_\theta(a_t^k \mid a_t^{k+1}, s_t)\right].$$

D.2. Reverse Log-Derivative Trick and Diffusion Surrogate Loss

We now derive the surrogate loss used in Eq. 15.

Consider the KL divergence

$$D_{\mathrm{KL}}\big(q_\theta(a_t^{k-1} \mid a_t^k, s_t)\,\big\|\,\pi(a_t^k \mid a_t^{k-1}, s_t)\exp\left(\alpha\, Q_{\mathrm{Diff}}^{q_\theta}(a_t^{k-1}, a_t^k, s_t)\right)\big) = \mathbb{E}_{q_\theta}\left[\log\frac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi(a_t^k \mid a_t^{k-1}, s_t)} - \alpha\, Q_{\mathrm{Diff}}^{q_\theta}(a_t^{k-1}, a_t^k, s_t)\right] + C,$$

where $Z(s_t)$ is the normalizing partition function (independent of θ).

Differentiating w.r.t. θ and applying the log-derivative trick gives

$$\nabla_\theta D_{\mathrm{KL}} = \mathbb{E}_{q_\theta}\left[\left(\log\frac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi(a_t^k \mid a_t^{k-1}, s_t)} - \alpha\, Q_{\mathrm{Diff}}^{q_\theta}(a_t^{k-1}, a_t^k, s_t)\right)\nabla_\theta \log q_\theta(a_t^{k-1} \mid a_t^k, s_t)\right]. \tag{30}$$

This identity, applied at each diffusion index k and each timestep t, yields the surrogate loss expression in Eq. 15 of the main text:

$$D_{\mathrm{KL}}^{\mathcal{T}} = \mathcal{T}\sum_{t,k}\mathbb{E}_{q_\theta}\left[D_{\mathrm{KL}}\left(q_\theta(\cdot \mid a_t^k, s_t)\,\middle\|\,\pi(a_t^k \mid \cdot, s_t)\frac{\exp(\alpha Q_{\mathrm{Diff}}^{q_\theta}(s_t, \cdot))}{Z(s_t, a_t^k)}\right)\right] + C.$$

This completes the derivation.

E. Implementation Details

E.1. DiffPPO

Our Diffusion PPO implementation follows the standard PPO algorithm but operates on the maximum entropy diffusion objective introduced in Section 3. Specifically, instead of learning directly from the environment reward $R_{\mathrm{env}}(s_t, a_t)$,

we use the *maximum entropy diffusion reward* $\tilde{R}_{\text{MaxEntDiff}}$ as defined in Eq. 28. During rollout, we therefore treat each reverse-diffusion step k as part of an augmented trajectory, and importantly, since the diffusion log ratios are already incorporated in the reward, we do not use these log ratios within the value function.

The MDP is augmented following Section 3.1, but DiffPPO incorporates the diffusion terms directly into the reward rather than absorbing them into the value function. Thus, the PPO advantage estimator and clipped surrogate loss remain unchanged; the only modification is that the reward at each step is replaced with $\tilde{R}_{\text{MaxEntDiff}}$. Apart from this, reward reshaping and the diffusion rollout procedure, all optimization steps follow the Stable Baselines3 PPO implementation.

In DiffPPO, we apply variational annealing (Hibat-Allah et al., 2021; Sun et al., 2022; Sanokowski et al., 2023) by exponentially reducing the temperature from an initial value $\mathcal{T}_{\text{start}}$ down to $\mathcal{T} = 0$. We set $\mathcal{T}_{\text{start}} = \frac{k}{\dim(\mathcal{A})}$ and found that $k = [0.1, 0.3]$ is a well performing range for this hyperparameter. During training, the temperature is halved every 10% of the total training steps.

E.2. DiffSAC

Diffusion Soft Actor-Critic is implemented on top of the Stable Baselines3 SAC codebase, but replaces both the underlying MDP and the value-learning objective. As described in Section 3.1, we convert each environment step into a sequence of K diffusion-index transitions over the augmented state $\tilde{s}_{\tilde{t}(t,k)} = (s_t, a_t^k, k)$. The external reward remains unchanged and is assigned only at diffusion index $k = 0$, following the definition of $\tilde{R}_{\text{Diff}}$ in Section 3.1.

Crucially, DiffSAC uses the *Diffusion value function* and *Diffusion Q-function* defined in Section 3.1, which include the additional log-ratio term

$$-\mathcal{T} \log \frac{q_\theta(a_t^{k-1} \mid a_t^k, s_t)}{\pi(a_t^k \mid a_t^{k-1}, s_t)}.$$

The actor updates are performed by minimizing Eq. 15 and the target updates use V_{Diff} (see Eq. 16) instead of the value function as defined in Sec. 2.1. For automatic temperature tuning as in (Haarnoja et al., 2018), we follow (Celik et al., 2025) and use a lower bound of the entropy to control the temperature.

The entropy lower bound $H_{\text{lower}} \leq H(\pi(a_t|s_t))$ is given

$$H_{\text{lower}} = \mathbb{E}_{a_t^{0:K} \sim q_\theta}\Big[\sum_{k=1}^{K} \frac{\pi(a_t^k|a_t^{k-1}, s_t)}{q_\theta(a_t^{k-1}|a_t^k, s_t)} - \log q(a_t^{(K)}, s_t)\Big]. \tag{31}$$

For DiffWPO and DiffSAC we observed that for your configuration of the diffusion noise schedule, setting the target Entropy value to $-10\dim(\mathcal{A})$ yielded good results.

E.3. DiffWPO

The Diffusion Wasserstein Policy Optimization algorithm is implemented by modifying DiffSAC by replacing the Loss function in Eq. 15 by Eq. 18. Thus, in our experiments, DiffWPO is also an off-policy algorithm.

F. Experimental Details

F.1. Control Environments

F.2. Humanoid Environment Configuration

The `Humanoid-run-v4` environment was instantiated with the following reward parameters. We set:

$$\texttt{forward\_reward\_weight} = 5.0, \quad \texttt{ctrl\_cost\_weight} = 0.005, \quad \texttt{healthy\_reward} = 10.0.$$

The increased forward–reward weight emphasizes sustained forward velocity. The higher control–cost weight penalizes excessive torque usage, promoting smoother and more energy-efficient movements. The elevated healthy–reward value strengthens the incentive to maintain an upright posture. These adjustments together create a reward structure that prioritizes stable, efficient running behavior.

F.3. Hyperparameters

For each experiment in Sec. 4, we run a hyperparameter sweep over three learning rates. For SAC and PPO, we additionally run a grid search over two values of γ and for PPO also over the entropy coefficient.

F.4. Diffusion Hyperparameters

The integration of diffusion samplers into reinforcement learning introduces a new set of hyperparameters, including:

- the **diffusion schedule**,
- the **initial SDE noise level** ($\beta_{\max}$),
- the **final SDE noise level** ($\beta_{\min}$),
- the **number of diffusion steps** (K), where we consistently set $\Delta t = \frac{1}{K}$.

For all diffusion-based methods, we define $\gamma = 0.9991^{(1/K)}$ to eliminate the need for hyperparameter tuning in this context. Additionally, we fix the GAE-λ parameter in DiffPPO at 0.95, reserving a detailed investigation of its impact for future research.

In DiffPPO, we scale the number of update steps per environment collection step by a factor of K. The implications of this adjustment, as well as the potential need to increase the number of collection steps in PPO with K, remain topics for future exploration. Increasing K reduces the frequency of original environment observations in the buffer, and its broader effects—such as on the clipping parameter—have yet to be examined.

F.5. Score Network Initialization

The neural network parameterizing the score function $s_\theta(a^{(k)}, k)$ is initialized such that at the start of training, the model reproduces the prior distribution $p(a_K) = \mathcal{N}(0, \nu I)$.

F.6. Details on Action Generation:

For all experiments, actions are sampled in $\mathbb{R}^N$ space. The final action applied to the environment is obtained by scaling the last diffusion step through a $\tanh$ function to match the environment's action bounds. In principle, simple clipping of the unbounded action values to the valid action range could also be applied as an alternative to the $\tanh$ squashing function.