Diffusion-Based Maximum Entropy Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Maximum entropy reinforcement learning (MaxEnt-RL) has become the standard approach to RL due to its beneficial exploration properties. Traditionally, policies are parameterized using Gaussian distributions, which significantly limits their representational capacity. Diffusion-based policies offer a more expressive alternative, yet integrating them into MaxEnt-RL poses challenges—primarily due to the intractability of computing their marginal entropy. We propose Diffusion-Based Maximum Entropy RL (DIME). *DIME* leverages recent advances in approximate inference with diffusion models to derive a lower bound on the maximum entropy objective. Additionally, we propose a policy iteration scheme that provably converges to the optimal diffusion policy. Our method enables the use of expressive diffusion-based policies while retaining the principled exploration benefits of MaxEnt-RL, significantly outperforming other diffusion-based methods on challenging high-dimensional control benchmarks. It is also competitive with state-of-the-art non-diffusion based RL methods while requiring fewer algorithmic design choices and smaller update-to-data ratios, reducing computational complexity.

1 Introduction

The maximum entropy reinforcement learning (MaxEnt-RL) objective augments the task reward in each time step with the entropy of the policy [76, 60, 25, 26]. This objective has several favorable properties among which improved exploration [75, 25] is crucial for RL. Recent successful model-free RL algorithms leverage these favorable properties and build upon this framework [8, 46] improving sample efficiency and leading to remarkable results. However, the policies are traditionally parameterized using Gaussian distributions, significantly limiting their representational capacity. On the other hand, diffusion models [57, 30, 58, 36] are highly expressive generative models and have proven beneficial in representing complex behavior policies [54, 16]. However, important metrics such as the marginal entropy are intractable to compute [74] which restricts their usage in RL. Because of this shortcoming, recent methods propose different ways to train diffusion-based methods in off-policy RL. While these methods are discussed in more detail in the related work section (App. C), most of them require additional techniques to add artificial (in most cases Gaussian) noise to the generated actions to induce exploration in the behavior generation process. Hence, they do not leverage the diffusion model to generate potentially non-Gaussian exploration patterns but fall back to mainly Gaussian exploration. Nonetheless, there have been significant advances in training diffusion-based models for approximate inference [7, 55]. Since the policy improvement in MaxEnt-RL can also be cast as an approximate inference problem to the energy-based policy [25], it is a natural step to explore these parallels.

We propose Diffusion-Based Maximum Entropy Reinforcement Learning (DIME). *DIME* leverages recent advances in approximate inference with diffusion models [55] to derive a lower bound on the MaxEnt objective. We propose a policy iteration framework with monotonic policy improvement that converges to the optimal diffusion policy. Additionally, building on recent off-policy RL algorithms such as Cross-Q [8] and distributional RL [6], we propose a practical version of DIME that can be used for training diffusion-based RL policies. On 13 challenging continuous high-dimensional control benchmarks, we empirically validate that DIME significantly outperforms other diffusion-based baselines on all environments and consistently outperforms other state-of-the-art RL methods based on a Gaussian policy on 10 out of 13 environments, while being computationally more efficient and requiring less algorithmic design choices as the current state of the art baseline BRO [46].

2 Preliminaries

2.1 Maximum Entropy Reinforcement Learning

Notation We consider the task of learning a policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$, where $\mathcal{S}$ and $\mathcal{A}$ denote a continuous state and action space, respectively using reinforcement learning (RL). We formalize the RL problem using an infinite horizon Markov decision process consisting of the tuple $(\mathcal{S}, \mathcal{A}, r, p, \rho_\pi, \gamma)$, with bounded reward function $r : \mathcal{S} \times \mathcal{A} \to [r_{\text{min}}, r_{\text{max}}]$ and transition density $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ which denotes the likelihood for transitioning into a state $s' \in \mathcal{S}$ when being in $s \in \mathcal{S}$ and executing an action $a \in \mathcal{A}$. We follow [26] and slightly overload ρ_π which denotes the state and state-action marginals induced by a policy π. Moreover, $\gamma \in [0, 1)$ denotes the discount factor. For brevity, we use $r_t \triangleq r(s_t, a_t)$. We denote objective functions that we aim to maximize as J and minimize as $\mathcal{L}$.

Control as inference. The goal of maximum entropy reinforcement learning (MaxEnt-RL) is to jointly maximize the sum of expected rewards and entropies of a policy

$$J(\pi) = \sum_{t=l}^{\infty} \gamma^{t-l} \mathbb{E}_{\rho_\pi} \left[r_t + \alpha \mathcal{H}(\pi(a_t|s_t)) \right], \tag{1}$$

where $\mathcal{H}(\pi(a|s)) = -\int \pi(a|s) \log \pi(a|s) \mathrm{d}a$ is the differential entropy, and $\alpha \in \mathbb{R}^+$ controls the exploration exploitation trade-off [25]. To keep the notation uncluttered we absorb α into the reward function via $r \leftarrow r/\alpha$. Defining the Q-function of a policy π as

$$Q^\pi(s_t, a_t) = r_t + \sum_{l=1}^{\infty} \gamma^l \mathbb{E}_{\rho_\pi} \left[r_{t+l} + \mathcal{H}\left(\pi(a_{t+l}|s_{t+l})\right) \right], \tag{2}$$

with $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the MaxEnt objective can be cast as an approximate inference problem as

$$\mathcal{L}(\pi) = D_{\text{KL}} \left(\pi(a_t|s_t) \middle| \frac{\exp Q^\pi(s_t, a_t)}{\mathcal{Z}^\pi(s_t)} \right), \tag{3}$$

in a sense that $\max_\pi J(\pi) = \min_\pi \mathcal{L}(\pi)$. Here, D_{KL} denotes the Kullback-Leibler divergence and $\mathcal{Z}^\pi(s) = \int \exp Q^\pi(s, a) \mathrm{d}a$ is the state-dependent normalization constant.

Policy iteration is a two-step iterative update scheme that is, under certain assumptions, guaranteed to converge to the optimal policy with respect to the maximum entropy objective. The two steps include policy evaluation and policy improvement. Given a policy π, policy evaluation aims to evaluate the value of π. To that end, [26] showed that repeated application of the Bellman backup operator $\mathcal{T}^\pi Q^k$ with $\mathcal{T}^\pi Q(s_t, a_t) \triangleq r_t + \gamma \mathbb{E}\left[Q(s_{t+1}, a_{t+1}) + \mathcal{H}(a_{t+1}|s_{t+1})\right]$ converges to Q^π as $k \to \infty$, starting from any Q. To update the policy, that is, to perform the policy improvement step, the Q-function of the previous evaluation step, $Q^{\pi_{\text{old}}}$ is used to obtain a new policy according to

$$\pi_{\text{new}} = \arg\min_{\pi \in \Pi} D_{\text{KL}} \left(\pi(a_t|s_t) \middle| \frac{\exp Q^{\pi_{\text{old}}}(s_t, a_t)}{\mathcal{Z}^{\pi_{\text{old}}}(s_t)} \right), \tag{4}$$

where Π is a set of policies such as a family of parameterized distributions. Note that $\mathcal{Z}^{\pi_{\text{old}}}(s_t)$ is not required for optimization as it is independent of π. [26] showed that for all state-action pairs $(s, a) \in \mathcal{S} \times \mathcal{A}$ it holds that $Q^{\pi_{\text{new}}}(s, a) \geq Q^{\pi_{\text{old}}}(s, a)$ ensuring that policy iteration converges to the optimal policy π^* in the limit of infinite repetitions of policy evaluation and improvement.

2.2 Denoising Diffusion Policies

For a given state $s \in \mathcal{S}$, we consider a stochastic process on the time-interval $[0, T]$ given by an Ornstein-Uhlenbeck (OU) process [1] [56]

$$\mathrm{d}a_t = -\beta_t a_t \mathrm{d}t + \eta\sqrt{2\beta_t}\mathrm{d}B_t, \quad a_0 \sim \vec{\pi}_0(\cdot|s), \tag{5}$$

with diffusion coefficient $\beta : [0, T] \to \mathbb{R}^+$, standard Brownian motion $(B_t)_{t\in[0,T]}$, and some target policy $\vec{\pi}_0$. For $t, l \in [0, T]$, we denote the marginal density of Eq. 5 at t as $\vec{\pi}_t$ and the conditional density at time t given l as $\vec{\pi}_{t|l}$. Eq. 5 is commonly referred to as *forward* or *noising process* since, for a suitable choice of β, it holds that $\vec{\pi}_T \approx \mathcal{N}(0, \eta^2 I)$. Denoising diffusion models leverage the fact, that the time-reversed process of Eq. 5 is given by

$$\mathrm{d}a_t = \left(-\beta_t a_t \mathrm{d}t - 2\eta^2\beta_t \nabla \log \vec{\pi}_t(a_t|s)\right) + \eta\sqrt{2\beta_t}\mathrm{d}B_t, \tag{6}$$

starting from $\overleftarrow{\pi}_T = \vec{\pi}_T \approx \mathcal{N}(0, \eta^2 I)$ and running backwards in time [47, 4, 29]. For the *backward*, *generative* or *denoising process* (Eq. 6), we denote the density as $\overleftarrow{\pi}$. Here, time-reversal means that the marginal densities align, i.e., $\vec{\pi}_t = \overleftarrow{\pi}_t$ for all $t \in [0, T]$. Hence, starting from $a_T \sim \mathcal{N}(0, \eta^2 I)$, one can sample from the target policy $\vec{\pi}_0$ by simulating Eq. 6. However, for most densities $\vec{\pi}_0$, the scores $(\nabla \log \vec{\pi}_t(a_t|s))_{t\in[0,T]}$ are intractable, requiring numerical approximations. To address this, denoising score-matching objectives are commonly employed, that is,

$$\mathcal{L}_{\text{SM}}(\theta) = \mathbb{E}\left[\beta_t \|f_t^\theta(a_t, s) - \nabla \log \vec{\pi}_{t|0}(a_t|a_0, s)\|^2\right], \tag{7}$$

where t is sampled on $[0, T]$ and f^θ denotes a parameterized score network [32, 66]. For OU processes, the conditional densities $\nabla \log \vec{\pi}_{t|0}$ are explicitly computable, making the objective tractable for optimizing θ [58]. Once trained, the score network f^θ can be used to simulate the denoising process

$$\mathrm{d}a_t = \left(-\beta_t a_t \mathrm{d}t - 2\eta^2\beta_t f_t^\theta(a_t, s)\right) + \eta\sqrt{2\beta_t}\mathrm{d}B_t, \tag{8}$$

to obtain samples $a_0 \sim \pi_0^\theta$ that are approximately distributed according to $\vec{\pi}_0$. Here, π_t^θ denotes the marginal distribution of Eq. 8 at time t. While score-matching techniques work well in practice, they cannot be applied to maximum entropy reinforcement learning. This is because the expectation in Eq. 7 requires samples $a_0 \sim \vec{\pi}_0 \propto \exp Q^\pi$ which are not available. However, in the next section, we build on recent advances in approximate inference to optimize diffusion models without requiring samples from a_0, relying instead on evaluations of Q^π.

3 Diffusion-Based Maximum Entropy RL

Here, we express the maximum entropy objective as an approximate inference problem for diffusion models. We then use these results to introduce a policy iteration scheme that provably converges to the optimal policy. Lastly, we propose a practical algorithm for optimizing diffusion models.

3.1 Control as Inference for Diffusion Policies

Directly maximizing the maximum entropy objective

$$J(\overleftarrow{\pi}) = \sum_{t=l}^{\infty} \gamma^{t-l}\mathbb{E}_{\rho_\pi}\left[r_t(s_t, a_t^0) + \alpha\mathcal{H}(\overleftarrow{\pi}_0(a_t^0|s_t))\right],$$

for a diffusion model is difficult as the marginal entropy $\mathcal{H}(\overleftarrow{\pi}_0(a|s))$ of the denoising process in Eq. 6 is intractable. Please note that we use superscripts for the actions to indicate the diffusion step to avoid collisions with the time step used in RL. Moreover, we will again absorb α into the reward and use $r_t \triangleq r(s_t, a_t^0)$. To overcome this intractability, we propose to maximize a lower bound. We start by discretizing the stochastic processes introduced in Section 2.2 and use the results as a foundation to derive this lower bound. Note that while similar results can be derived from a continuous-time perspective (see e.g., [7, 55, 51]), such derivation would require a background in stochastic calculus,

[1]Please note, for clarity, we slightly abuse notation by using t to denote the time in the stochastic process, not be confused with the time step in RL. The distinction becomes clear when we discretize the processes.

making it less accessible to a broader audience. The Euler-Maruyama (EM) discretization [56] of the noising (Eq. 5) and denoising (Eq. 6) process is

$$a^{n+1} = a^n - \beta_n a^n \delta + \epsilon_n \quad \text{and} \tag{9}$$
$$a^{n-1} = a^n + \left(\beta_n a^n + 2\eta^2 \beta_n \nabla \log \vec{\pi}_n(a^n|s)\right)\delta + \xi_n, \tag{10}$$

respectively, with $\epsilon_n, \xi_n \sim \mathcal{N}(0, 2\eta^2\beta_n \delta I)$. Here, δ denotes a constant discretization step size such that $N = T/\delta$ is an integer. To simplify notation, we write a^n, instead of $a^{n\delta}$. Under the EM discretization, the noising and denoising process admit the following joint distributions

$$\vec{\pi}_{0:N}(a^{0:N}|s) = \vec{\pi}_0(a^0|s) \prod_{n=0}^{N-1} \vec{\pi}_{n+1|n}(a^{n+1}|a^n, s), \tag{11}$$
$$\overleftarrow{\pi}_{0:N}(a^{0:N}|s) = \overleftarrow{\pi}_N(a^N|s) \prod_{n=1}^{N} \overleftarrow{\pi}_{n-1|n}(a^{n-1}|a^n, s), \tag{12}$$

in a sense that $\vec{\pi}_{0:N}$ and $\overleftarrow{\pi}_{0:N}$ converge to the law of $(a_t)_{t\in[0,T]}$ in Eq. 5 and 6, as $\delta \to 0$, respectively [22]. Here, $\vec{\pi}_{n+1|n}$ and $\overleftarrow{\pi}_{n-1|n}$ are Gaussian transition densities from Eq. 9 and 10. To obtain a maximum entropy objective for diffusion models, we make use of the following lower bound on the marginal entropy, that is, $\mathcal{H}(\overleftarrow{\pi}_0(a_0|s)) \geq \ell_{\overleftarrow{\pi}}(a^0, s)$, where

$$\ell_{\overleftarrow{\pi}}(a^0, s) = \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\vec{\pi}_{1:N|0}(a^{1:N}|a^0, s)}{\overleftarrow{\pi}_{0:N}(a^{0:N}|s)}\right]. \tag{13}$$

Please note that similar bounds have been used, e.g., in [1, 61, 53, 42, 5], or, more generally, follow from the data processing inequality [17]. A derivation can be found in Appendix A. From Eq. 13, it directly follows that

$$J(\overleftarrow{\pi}) \geq \bar{J}(\overleftarrow{\pi}) = \sum_{t=l}^{\infty} \gamma^{t-l} \mathbb{E}_{\rho_\pi}\left[r_t + \ell_{\overleftarrow{\pi}}(a_t^0, s_t)\right]. \tag{14}$$

Next, we cast Eq. 14 as an approximate inference problem to make the objective more interpretable. To that end, let us define the Q-function of a denoising policy $\overleftarrow{\pi}$ with respect to the maximum entropy objective $\bar{J}$ as

$$Q^{\overleftarrow{\pi}}(s_t, a_t^0) = r_t + \sum_{l=1} \gamma^l \mathbb{E}_{\rho_\pi}\left[r_{t+l} + \ell_{\overleftarrow{\pi}}(a_{t+l}^0, s_{t+l})\right], \tag{15}$$

with $Q^{\overleftarrow{\pi}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. With Eq. 15 we identify the corresponding approximate inference problem as finding $\overleftarrow{\pi}$ which minimizes (please see Appendix A for derivation)

$$\bar{\mathcal{L}}(\overleftarrow{\pi}) = D_{\text{KL}}\left(\overleftarrow{\pi}_{0:N}(a^{0:N}|s)|\vec{\pi}_{0:N}(a^{0:N}|s)\right), \tag{16}$$

where the target policy, i.e., the marginal of the noising process in Eq. 11 is given by the exponentiated Q-function of the diffusion policy

$$\vec{\pi}_0(a^0|s) = \frac{\exp Q^{\overleftarrow{\pi}}(s, a^0)}{\mathcal{Z}^{\overleftarrow{\pi}}(s)}. \tag{17}$$

Recall from Section 2.2 that we aim to time-reverse the noising process, that is, to ensure for all states $s \in \mathcal{S}$, it holds that $\overleftarrow{\pi}_{0:N} = \vec{\pi}_{0:N}$. Please note that this is precisely what Eq. 16 is trying to accomplish, i.e., we aim to learn a diffusion model $\overleftarrow{\pi}$, such that the denoising process time-reverses the noising process, and, in particular, has a marginal distribution given by $\pi_0 = \exp Q^{\overleftarrow{\pi}}/\mathcal{Z}^{\overleftarrow{\pi}}$. Lastly, from the data processing inequality it directly follows that

$$D_{\text{KL}}\left(\overleftarrow{\pi}_0(a^0|s)\middle|\frac{\exp Q^{\overleftarrow{\pi}}(s, a^0)}{\mathcal{Z}^{\overleftarrow{\pi}}(s)}\right) \leq D_{\text{KL}}\left(\overleftarrow{\pi}(a^{0:N}|s)|\vec{\pi}(a^{0:N}|s)\right), \tag{18}$$

which shows the approximate inference problem in Eq. 16 indeed optimizes the same inference problem stated in Eq. 3.

3.2 Diffusion-based Policy Iteration

We propose a policy iteration scheme for learning an optimal maximum entropy policy, similar to [26]. However, here we restrict the family of stochastic actors to diffusion policies $\overleftarrow{\pi} \in \overleftarrow{\Pi} \subset \Pi$. Throughout this section, we assume finite action spaces to enable theoretical analysis, but relax this assumption in Section 3.3. All proofs of this section are deferred to Appendix A.

For policy evaluation, we aim to compute the value of a policy $\overleftarrow{\pi}$. We define the Bellman backup operator as

$$\mathcal{T}^{\overleftarrow{\pi}} Q(s_t, a_t^0) \triangleq r_t + \gamma \mathbb{E}\left[Q(s_{t+1}, a_{t+1}^0) + \ell_{\overleftarrow{\pi}}(a_{t+1}^0, s_{t+1})\right]. \tag{19}$$

Note that this equation contains the entropy-lower bound $\ell_{\overleftarrow{\pi}}$. By applying standard convergence results for policy evaluation [59] we can obtain the value of a policy by repeatedly applying $\mathcal{T}^{\overleftarrow{\pi}}$ as established in Proposition 3.1.

Proposition 3.1 (Policy Evaluation). *Let $\mathcal{T}^{\overleftarrow{\pi}}$ be the Bellman backup operator for a diffusion policy $\overleftarrow{\pi}$ as defined in Eq. 19. Further, let $Q^0 : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and $Q^{k+1} = \mathcal{T}^{\overleftarrow{\pi}} Q^k$. Then, it holds that $\lim_{k\to\infty} Q^k = Q^{\overleftarrow{\pi}}$ where $Q^{\overleftarrow{\pi}}$ is the Q value of $\overleftarrow{\pi}$.*

For the policy improvement step, we seek to improve the current policy based on its value using the Q-function. Formally, we need to solve the approximate inference problem

$$\overleftarrow{\pi}^{\text{new}} = \arg\min_{\overleftarrow{\pi}\in\overleftarrow{\Pi}} D_{\text{KL}}\left(\overleftarrow{\pi}_{0:N}(a^{0:N}|s) | \vec{\pi}_{0:N}^{\text{old}}(a^{0:N}|s)\right), \tag{20}$$

for all $s \in \mathcal{S}$, where $\vec{\pi}_{0:N}^{\text{old}}(a^{0:N}|s)$ is as in Eq. 11 with marginal density

$$\vec{\pi}_0^{\text{old}}(a^0|s) = \frac{\exp Q^{\overleftarrow{\pi}_{\text{old}}}(s, a^0)}{\mathcal{Z}^{\overleftarrow{\pi}_{\text{old}}}(s)}. \tag{21}$$

Indeed, solving Eq. 20 results in a policy with higher value as established below.

Proposition 3.2 (Policy Improvement). *Let $\overleftarrow{\pi}_{old}, \overleftarrow{\pi}_{new} \in \overleftarrow{\Pi}$ be defined as in Eq. 21 and 20, respectively. Then for all $(s, a) \in \mathcal{S} \times \mathcal{A}$ it holds that $Q^{\overleftarrow{\pi}_{new}}(s, a) \geq Q^{\overleftarrow{\pi}_{old}}(s, a)$.*

Combining these results leads to the policy iteration method which alternates between policy evaluation (Proposition 3.1) and policy improvement (Proposition 3.2) and provably converges to the optimal policy in $\overleftarrow{\Pi}$ (Proposition 3.3).

Proposition 3.3 (Policy Iteration). *Let $\overleftarrow{\pi}^0, \overleftarrow{\pi}^{i+1}, \overleftarrow{\pi}^i, \overleftarrow{\pi}_* \in \overleftarrow{\Pi}$ and let $\overleftarrow{\pi}^{i+1}$ be the policy obtained from $\overleftarrow{\pi}^i$ after a policy evaluation and improvement step. Then, for any starting policy $\overleftarrow{\pi}^0$ it holds that $\lim_{i\to\infty} \overleftarrow{\pi}^i = \overleftarrow{\pi}^*$, with $\overleftarrow{\pi}^*$ such that for all $\overleftarrow{\pi} \in \overleftarrow{\Pi}$ and $(s, a) \in \mathcal{S} \times \mathcal{A}$ it holds that $Q^{\overleftarrow{\pi}^*}(s, a) \geq Q^{\overleftarrow{\pi}}(s, a)$.*

However, performing policy iteration until convergence is, in practice often intractable, particularly for continuous control tasks. As such, we will introduce a practical algorithm next.

3.3 DIME: A Practical Diffusion RL Algorithm

For a practical algorithm, we use a parameterized function approximation for the Q-function and the policy, that is, Q_ϕ and π^θ, with parameters ϕ and θ. Here, π^θ is represented by a parameterized score network, see Eq. 8. For policy evaluation, we can minimize the Bellman residual,

$$J_Q(\phi) = \frac{1}{2}\mathbb{E}\left[\left(Q_\phi(s_t, a_t^0) - Q_{\text{target}}(s_t, a_t^0)\right)^2\right], \tag{22}$$

using stochastic gradients with respect to ϕ. We provide implementation details in Section 3.4. Moreover, the expectation is computed using state-action pairs collected from environment interactions and saved in a replay buffer. For policy improvement, we solve the approximate inference problem

$$\mathcal{L}(\theta) = D_{\text{KL}}\left(\pi_{0:N}^\theta(a^{0:N}|s) | \vec{\pi}_{0:N}(a^{0:N}|s)\right), \tag{23}$$

where the target policy, i.e., the marginal of the noising process in Eq. 11 is given by the approximate Q-function $\vec{\pi}_0(a^0|s) = \frac{\exp Q_\phi(s,a^0)}{\mathcal{Z}_\phi(s)}$, where states are again sampled from a replay buffer. Further expanding $\mathcal{L}(\theta)$ yields

$$\mathcal{L}(\theta) = \mathbb{E}_{\pi^\theta}\left[\log \pi_N^\theta(a^N|s) - Q_\phi(s, a^0) + \sum_{n=1}^{N} \log \frac{\pi_{n-1|n}^\theta(a^{n-1}|a^n, s)}{\vec{\pi}_{n|n-1}(a^n|a^{n-1}, s)}\right] + \log \mathcal{Z}_\phi(s), \tag{24}$$

showing that $\mathcal{Z}_\phi$ is not needed to minimize Eq. 24 as it is independent of θ. Moreover, contrary to the score-matching objective (see Eq. 7) that is commonly used to optimize diffusion models, stochastic optimization of $\mathcal{L}(\theta)$ does not need access to samples $a_0 \sim \exp Q_\phi/\mathcal{Z}_\phi$, instead relying on stochastic gradients obtained via reparameterization trick [37] using samples from the diffusion model π^θ.

3.4 Implementation Details

Autotuning Temperature. We follow implementations like SAC [27] where the reward scaling parameter α is not absorbed into the reward but scales the entropy term. Choosing α depends on the reward ranges and the dimensionality of the action space, which requires tuning it per environment. We instead follow prior works [27] for auto-tuning α by optimizing $J(\alpha) = \alpha\left(\mathcal{H}_{\text{target}} - \ell_{\mathcal{H}}^\theta\right)$, where $\mathcal{H}_{\text{target}}$ is a target value for the mismatch between the noising and denoising processes.

Autotuning Diffusion Coefficient. The objective function in Eq. 24 is fully differentiable with respect to parameters of the diffusion process. As such, we additionally treat the diffusion coefficient β as a learnable parameter that is optimized end-to-end, further reducing the need for manual hyperparameter tuning. Further details on the parameterization can be found in Appendices G and E.

Q-function. Following [8] we adopt the CrossQ algorithm, i.e., we use Batch Renormalization in the Q-function and avoid a target network for calculating Q_{target}. When updating the Q-function, the values for the current and next state-action pairs are queried in parallel. The next Q-values are used as target values where the gradients are stopped. Additionally, we employ distributional Q learning as proposed by [6]. The details are described in Appendix G.

4 Experiments

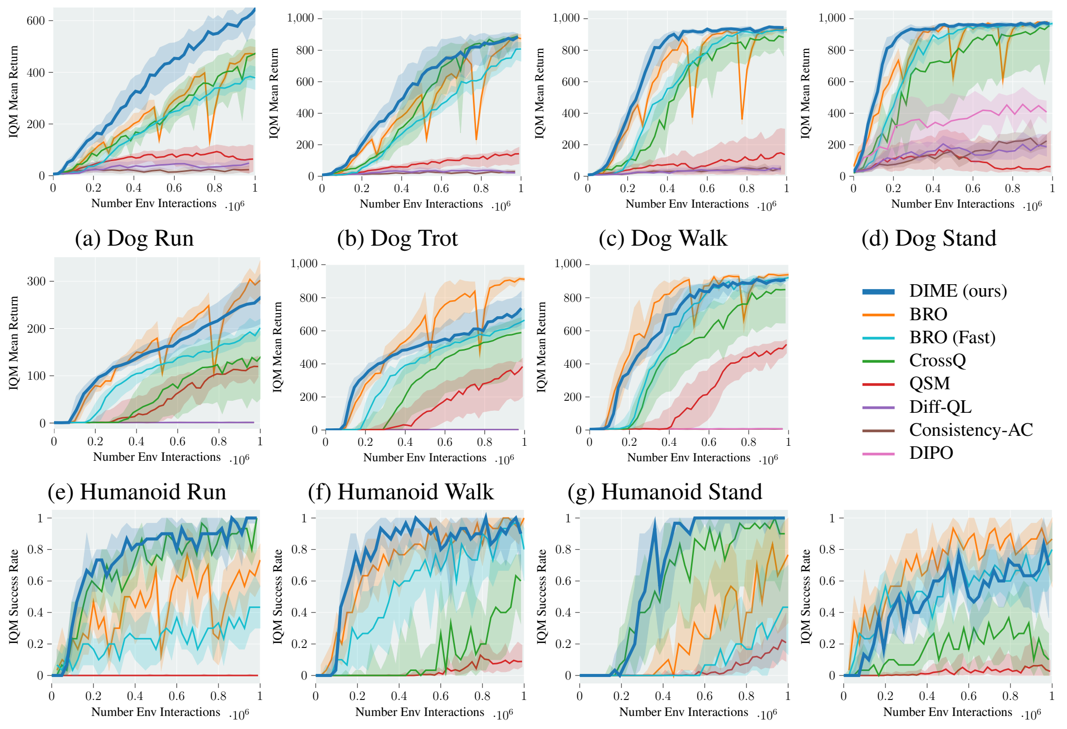
We analyze DIME's algorithmic features with an intensive ablation study where we clarify the role of the reward scaling parameter α, the effect of varying diffusion steps, the gained performance boost when using a diffusion policy representation over a Gaussian representation in Appendix D and we analyze employing distributional Q learning in Appendix E.

In a broad range of 13 sophisticated learning environments (see Appendix F) from different benchmark suits, ranging from mujoco gym [11], deepmind control suit (DMC) [62], and myo suite [12], we compare DIME's performance against state-of-the-art RL baselines that employ diffusion and Gaussian policy parameterizations. The considered environments are challenging locomotion and manipulation learning tasks with up to 39-dimensional action and 223-dimensional observation spaces. We consider QSM [52], Diffusion-QL [69], Consistency-AC [21], DIPO [71], QVPO [20], and DACER [68] as diffusion-based policy baselines. Additionally, we compare against the state-of-the-art RL methods CrossQ [8] and BRO [46], where we have used the provided learning curves from the latter. Both methods use a Gaussian parameterized policy and have shown remarkable results. We have run the learning curves for 10 seeds using the official code releases and report the *interquartile mean (IQM)* with a 95% stratified bootstrap confidence interval as suggested by [2].

4.1 Performance Comparisons

Gym Environments. Fig 3c and Fig. 3d show the learning curves for the *An-tv3* and *Humanoid-v3* tasks respectively. While the diffusion-based baselines perform reasonably well on the *Ant-v3* task with DIPO outperforming the rest, they are all outperformed by DIME and CrossQ which perform comparably. On the *Humanoid-v3* DIME achieves a significantly higher return than all baselines.

DMC: Dog and Humanoid Tasks (Fig. 1). We additionally consider BRO and BRO Fast (identical to BRO but different update-to-data (UTD) ratio) on DMC suit's *dog* and *humanoid tasks.* Please note that we used the online available learning curves provided by the official implementation for BRO. DIME outperforms all baselines significantly on the *dog-run* environment and converges faster to the same end performance on the remaining dog environments (see Fig. 1a - 1d). BRO has slightly higher



(a) Dog Run (b) Dog Trot (c) Dog Walk (d) Dog Stand

(e) Humanoid Run (f) Humanoid Walk (g) Humanoid Stand

(h) Object Hold Hard (i) Reach Hard (j) Key Turn Hard (k) Pen Twirl Hard

Figure 1: **Training curves on DMC's dog, humanoid tasks, and the hand environments from the MYO Suite.** DIME performs favorably on the high-dimensional dog tasks where it significantly outperforms all baselines (dog-run) or converges faster to the final performance. On the humanoid tasks, DIME outperforms all diffusion-based baselines, CrossQ and BRO Fast, and performs on par with BRO on the humanoid-stand task and slightly worse on the humanoid-run and humanoid-walk tasks. In the MYO Suite, DIME consistently outperforms the baselines or performs on par.

average performance on the *humanoid-run* and *humanoid-walk* (see Fig. 1f - 1e)) tasks indicating that DIME performs favorably on more high-dimensional tasks like the dog environments and tasks from the myo suite. However, DIME's asymptotic behavior in the *humanoid-run* achieves slightly higher aggregated performance than BRO, where we have run both algorithms for 3M steps (Fig. 8c). However, BRO requires full parameter resets, leading to performance drops during training, and it is run with a UTD ratio of 10, which is 5 times higher than DIME. This leads to longer training times. As reported in their paper [46], BRO needs an average training time of 8.5h, whereas DIME trains in approximately 4.5h with 16 diffusion steps on the humanoid-run on the same GPU (*Nvidia A100*).

MYO Suite (Fig. 1). Except for *pen twirl hard* (Fig. 1k), DIME consistently outperforms BRO, BRO Fast, and CrossQ in that it converges to a higher or faster end success rate.

5 Conclusion and Future Work

We introduced DIME, a method for learning diffusion models for maximum entropy reinforcement learning by leveraging connections to approximate inference. We view this work as a starting point for exciting future research. Specifically, we explored *denoising* diffusion models, where the forward process follows an OU process. However, approximate inference with diffusion models is an active and rapidly evolving field, with numerous recent advancements that consider alternative stochastic processes. For example, [55] proposed learning both the forward and backward processes, while [51] further enhanced exploration by incorporating the gradient of the target density into the diffusion process. Additionally, [14] combined learned diffusion models with Sequential Monte Carlo [19], resulting in a highly effective inference method. These approaches hold significant promise for further improving diffusion-based policies in RL. We have conducted preliminary experiments on the framework from [55] and provide them in Appendix I.

References

[1] Felix V Agakov and David Barber. An auxiliary variational method. In *Neural Information Processing: 11th International Conference, ICONIP 2004, Calcutta, India, November 22-25, 2004. Proceedings 11*, pages 561–566. Springer, 2004.

[2] 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.

[3] Tara Akhound-Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, et al. Iterated denoising energy matching for sampling from boltzmann densities. *arXiv preprint arXiv:2402.06121*, 2024.

[4] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[5] Oleg Arenz, Gerhard Neumann, and Mingjun Zhong. Efficient gradient-free variational inference using policy search. In *International conference on machine learning*, pages 234–243. PMLR, 2018.

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

[7] Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling. *Transactions on Machine Learning Research*.

[8] Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *The Twelfth International Conference on Learning Representations*, 2024.

[9] Denis Blessing, Julius Berner, Lorenz Richter, and Gerhard Neumann. Underdamped diffusion bridges with applications to sampling. In *The Thirteenth International Conference on Learning Representations*, 2025.

[10] Denis Blessing, Xiaogang Jia, and Gerhard Neumann. End-to-end learning of gaussian mixture priors for diffusion sampler. In *The Thirteenth International Conference on Learning Representations*, 2025.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[12] Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. Myosuite – a contact-rich simulation suite for musculoskeletal motor control, 2022. arXiv preprint arXiv:2205.00588.

[13] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[14] Junhua Chen, Lorenz Richter, Julius Berner, Denis Blessing, Gerhard Neumann, and Anima Anandkumar. Sequential controlled langevin diffusions. *arXiv preprint arXiv:2412.07081*, 2024.

[15] Junhua Chen, Lorenz Richter, Julius Berner, Denis Blessing, Gerhard Neumann, and Anima Anandkumar. Sequential controlled langevin diffusions. In *The Thirteenth International Conference on Learning Representations*, 2025.

[16] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[17] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[18] Paolo Dai Pra. A stochastic control approach to reciprocal diffusion processes. *Applied mathematics and Optimization*, 23(1):313–329, 1991.

[19] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

[20] Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[21] Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[22] Arnaud Doucet, Will Grathwohl, Alexander G Matthews, and Heiko Strathmann. Score-based diffusion meets annealed importance sampling. *Advances in Neural Information Processing Systems*, 35:21482–21494, 2022.

[23] Linjiajie Fang, Ruoxue Liu, Jing Zhang, Wenjia Wang, and Bingyi Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[24] Tomas Geffner and Justin Domke. Langevin diffusion variational inference. In *International Conference on Artificial Intelligence and Statistics*, pages 576–593. PMLR, 2023.

[25] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

[26] 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*, pages 1861–1870. PMLR, 2018.

[27] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

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

[29] Ulrich G Haussmann and Etienne Pardoux. Time reversal of diffusions. *The Annals of Probability*, pages 1188–1205, 1986.

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

[31] Jian Huang, Yuling Jiao, Lican Kang, Xu Liao, Jin Liu, and Yanyan Liu. Schrödinger-föllmer sampler: sampling without ergodicity. *arXiv preprint arXiv:2106.10880*, 1, 2021.

[32] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[33] Haque Ishfaq, Guangyuan Wang, Sami Nur Islam, and Doina Precup. Langevin soft actor-critic: Efficient exploration through uncertainty-driven critic learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[34] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022.

[35] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023.

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

[37] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[38] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pages 45–73. Springer, 2012.

[39] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[40] Zechu Li, Rickmer Krohn, Tao Chen, Anurag Ajay, Pulkit Agrawal, and Georgia Chalvatzaki. Learning multimodal behaviors from scratch with diffusion policy gradient. *arXiv preprint arXiv:2406.00681*, 2024.

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

[42] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International conference on machine learning*, pages 1445–1453. PMLR, 2016.

[43] Liyuan Mao, Haoran Xu, Xianyuan Zhan, Weinan Zhang, and Amy Zhang. Diffusion-dice: In-sample diffusion guidance for offline reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[44] Safa Messaoud, Billel Mokeddem, Zhenghai Xue, Linsey Pang, Bo An, Haipeng Chen, and Sanjay Chawla. S 2 ac: Energy-based reinforcement learning with stein soft actor critic. In *The Twelfth International Conference on Learning Representations*.

[45] Michal Nauman and Marek Cygan. On the theory of risk-aware agents: Bridging actor-critic and economics. In *ICML 2024 Workshop: Aligning Reinforcement Learning Experimentalists and Theorists*, 2023.

[46] Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[47] Edward Nelson. *Dynamical theories of Brownian motion*, volume 101. Princeton university press, 2020.

[48] Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

[49] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pages 16828–16847. PMLR, 2022.

[50] Maxence Noble, Louis Grenioux, Marylou Gabrié, and Alain Oliviero Durmus. Learned reference-based diffusion sampling for multi-modal distributions. *arXiv preprint arXiv:2410.19449*, 2024.

[51] Nikolas Nusken, Francisco Vargas, Shreyas Padhy, and Denis Blessing. Transport meets variational inference: Controlled monte carlo diffusions. In *The Twelfth International Conference on Learning Representations: ICLR 2024*, 2024.

[52] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *Forty-first International Conference on Machine Learning*, 2024.

[53] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International conference on machine learning*, pages 324–333. PMLR, 2016.

[54] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal conditioned imitation learning using score-based diffusion policies. In *Robotics: Science and Systems*, 2023.

[55] Lorenz Richter and Julius Berner. Improved sampling via learned diffusions. In *The Twelfth International Conference on Learning Representations*.

[56] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

[57] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[58] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[59] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. *Robotica*, 17(2):229–235, 1999.

[60] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pages 1049–1056, 2009.

[61] Dustin Tran, Rajesh Ranganath, and David M Blei. The variational gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.

[62] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[63] Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Conference on Learning Theory*, pages 3084–3114. PMLR, 2019.

[64] Francisco Vargas, Will Sussman Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. In *The Eleventh International Conference on Learning Representations*.

[65] Francisco Vargas, Andrius Ovsianas, David Fernandes, Mark Girolami, Neil D Lawrence, and Nikolas Nüsken. Bayesian learning via neural schrödinger–föllmer flows. *Statistics and Computing*, 33(1):3, 2023.

[66] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[67] Dilin Wang and Qiang Liu. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016.

[68] Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang WU, Jingliang Duan, and Shengbo Eben Li. Diffusion actor-critic with entropy regulator. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[69] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. International Conference on Learning Representations, 2023.

[70] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[71] Long Yang, Zhixiong Huang, Feng hao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

[72] Dinghuai Zhang, Ricky TQ Chen, Cheng-Hao Liu, Aaron Courville, and Yoshua Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *arXiv preprint arXiv:2310.02679*, 2023.

[73] Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling. *arXiv preprint arXiv:2111.15141*, 2021.

[74] Hongyi Zhou, Denis Blessing, Ge Li, Onur Celik, Xiaogang Jia, Gerhard Neumann, and Rudolf Lioutikov. Variational distillation of diffusion policies into mixture of experts. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

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

[76] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

A Derivations

Lower-Bound Derivation. $\mathcal{H}(\pi_0(a_0|s)) \geq \ell_{\overleftarrow{\pi}}(a^0, s)$

$$\begin{aligned}
\mathcal{H}(\pi_0(a_0|s)) &= -\mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\overleftarrow{\pi}_{0:N}(a^{0:N}|s)}{\overleftarrow{\pi}_{1:N|0}(a^{1:N}|s,a^0)}\right] \qquad (25)\\
&= -\mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\overleftarrow{\pi}_{0:N}(a^{0:N}|s)\vec{\pi}_{1:N|0}(a^{1:N}|s,a^0)}{\overleftarrow{\pi}_{1:N|0}(a^{1:N}|s,a^0)\vec{\pi}_{1:N|0}(a^{1:N}|s,a^0)}\right]\\
&= \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\vec{\pi}_{1:N|0}(a^{1:N}|s,a^0)}{\overleftarrow{\pi}_{0:N}(a^{0:N}|s)}\right] + \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\overleftarrow{\pi}_{1:N|0}(a^{1:N}|s,a^0)}{\vec{\pi}_{1:N|0}(a^{1:N}|s,a^0)}\right] \qquad (26)\\
&= \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\vec{\pi}_{1:N|0}(a^{1:N}|s,a^0)}{\overleftarrow{\pi}_{0:N}(a^{0:N}|s)}\right] + \mathbb{E}_{\pi_0}\left[D_{\mathrm{KL}}\left(\overleftarrow{\pi}_{1:N|0}(a^{1:N}|s,a^0) \| \, \vec{\pi}_{1:N|0}(a^{1:N}|s,a^0)\right)\right] \qquad (27)\\
&\geq \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\vec{\pi}_{1:N|0}(a^{1:N}|s,a^0)}{\overleftarrow{\pi}_{0:N}(a^{0:N}|s)}\right], \qquad (28)
\end{aligned}$$

where we have used the relation

$$\pi_0(a_0|s) = \frac{\overleftarrow{\pi}_{0:N}(a^{0:N}|s)}{\overleftarrow{\pi}_{1:N|0}(a^{1:N}|s,a^0)} \tag{29}$$

and the fact that the KL divergence is always non-negative

Approximate Inference Formulation. Recall the definition of the Q-function

$$Q^{\overleftarrow{\pi}}(s_t, a_t^0) = r_t + \sum_{l=1} \gamma^l \mathbb{E}_{\rho_\pi}\left[r_{t+l} + \ell_{\overleftarrow{\pi}}(a_{t+l}^0, s_{t+l})\right]. \tag{30}$$

and

$$\ell_{\overleftarrow{\pi}}(a^0, s) = \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\vec{\pi}_{1:N|0}(a^{1:N}|a^0,s)}{\overleftarrow{\pi}_{0:N}(a^{0:N}|s)}\right]. \tag{31}$$

We start reformulating the objective

$$\begin{aligned}
J(\overleftarrow{\pi}) \geq \bar{J}(\overleftarrow{\pi}) &= \sum_{t=l}^{\infty} \gamma^{t-l} \mathbb{E}_{\rho_\pi}\left[r_t + \ell_{\overleftarrow{\pi}}(a_t^0, s_t)\right]. \qquad (32)\\
&= \sum_{t=l+1}^{\infty} \gamma^{t-l} \mathbb{E}_{\rho_\pi}\left[r_t + \ell_{\overleftarrow{\pi}}(a_t^0, s_t)\right] + \mathbb{E}_{\rho^\pi}\left[r_l + \ell_{\overleftarrow{\pi}}(a_l^0, s_l)\right] \qquad (33)\\
&= \mathbb{E}_{\rho^\pi}\left[Q^{\overleftarrow{\pi}}(s_t, a_t^0)\right] + \mathbb{E}_{\rho^\pi}\left[\ell_{\overleftarrow{\pi}}(a_l^0, s_l)\right] \qquad (34)\\
&= \mathbb{E}_{\rho^\pi}\left[Q^{\overleftarrow{\pi}}(s_t, a_t^0) + \ell_{\overleftarrow{\pi}}(a_l^0, s_l)\right] \qquad (35)\\
&= \mathbb{E}_{\rho^\pi, \overleftarrow{\pi}_{0:N}}\left[Q^{\overleftarrow{\pi}}(s_t, a_t^0) + \log \frac{\vec{\pi}_{1:N|0}(a^{1:N}|a^0,s)}{\overleftarrow{\pi}_{0:N}(a^{0:N}|s)}\right] \qquad (36)\\
&= -\mathbb{E}_{\rho^\pi}\left[D_{\mathrm{KL}}\left(\overleftarrow{\pi}(a^{0:N}|s) \| \, \vec{\pi}(a^{0:N}|s)\right) - \log \mathcal{Z}^{\overleftarrow{\pi}}(s)\right], \qquad (37)
\end{aligned}$$

where we used

$$\vec{\pi}_0(a^0|s) = \frac{\exp Q^{\overleftarrow{\pi}}(s, a^0)}{\mathcal{Z}^{\overleftarrow{\pi}}(s)} \tag{38}$$

in the last step. When minimizing, the negative sign in front of the KL vanishes. Please note that the expectation over the marginal state distribution was ommited in the main text to avoid cluttered notation.

B Proofs

Proof of Proposition 3.1 (Policy Evaluation). Let's define the entropy-augmented reward of a diffusion policy as

$$r_{\overleftharpoon{\pi}}(s_t, a_t^0) \triangleq r_t(s_t, a_t^0) + \mathbb{E}_{\overleftharpoon{\pi}_{0:N}} \left[\log \frac{\vec{\pi}_{1:N|0}(a^{1:N}|a^0, s)}{\overleftharpoon{\pi}_{0:N}(a^{0:N}|s)} \right] \tag{39}$$

and the update rule for the Q-function as

$$Q(s_t, a_t^0) \leftarrow r_{\overleftharpoon{\pi}}(s_t, a_t^0) + \gamma \mathbb{E}_{s_{t+1} \sim p, a_{t+1}^0 \sim \overleftharpoon{\pi}} \left[Q(s_{t+1}, a_{t+1}^0) \right]. \tag{40}$$

This formulation allows us to apply the standard convergence results for policy evaluation as stated in [59].

Proof of Proposition 3.2 (Policy Improvement). It holds that

$$\overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s) = \frac{\exp Q^{\pi^{(i)}}(s, a^N)}{Z^{\pi^{(i)}}(s)} \vec{\pi}^{(i)}(a^{0:N-1}|a^N, s) \tag{41}$$

Moreover, using the fact that the KL divergence is always non-negative, we obtain

$$0 = D_{\mathrm{KL}} \left(\overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s) \| \overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s) \right) \leq D_{\mathrm{KL}} \left(\overleftharpoon{\pi}^{(i)}(a^{0:N}|s) \| \overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s) \right) \tag{42}$$

Rewriting the KL divergences yields

$$\mathbb{E}_{\overleftharpoon{\pi}^{(i+1)}} \left[\log \frac{\overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s)}{\overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s)} \right] \leq \mathbb{E}_{\overleftharpoon{\pi}^{(i)}} \left[\log \frac{\overleftharpoon{\pi}^{(i)}(a^{0:N}|s)}{\overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s)} \right] \tag{43}$$

$$\iff \quad \mathbb{E}_{\overleftharpoon{\pi}^{(i+1)}} \left[\log \overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s) \right] - \mathbb{E}_{\overleftharpoon{\pi}^{(i+1)}} \left[\log \overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s) \right] \tag{44}$$

$$\leq \mathbb{E}_{\overleftharpoon{\pi}^{(i)}} \left[\log \overleftharpoon{\pi}^{(i)}(a^{0:N}|s) \right] - \mathbb{E}_{\overleftharpoon{\pi}^{(i)}} \left[\log \overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s) \right]$$

$$\iff \quad \mathbb{E}_{\overleftharpoon{\pi}^{(i+1)}} \left[\log \overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s) \right] - \mathbb{E}_{\overleftharpoon{\pi}^{(i+1)}} \left[\log \frac{\exp Q^{\pi^{(i)}}(s, a^N)}{Z^{\pi^{(i)}}(s)} \vec{\pi}^{(i)}(a^{0:N-1}|a^N, s) \right] \tag{45}$$

$$\leq \mathbb{E}_{\overleftharpoon{\pi}^{(i)}} \left[\log \overleftharpoon{\pi}^{(i)}(a^{0:N}|s) \right] - \mathbb{E}_{\overleftharpoon{\pi}^{(i)}} \left[\log \frac{\exp Q^{\pi^{(i)}}(s, a^N)}{Z^{\pi^{(i)}}(s)} \vec{\pi}^{(i)}(a^{0:N-1}|a^N, s) \right]$$

$$\iff \quad \mathbb{E}_{\overleftharpoon{\pi}^{(i+1)}} \left[Q^{\pi^{(i)}}(s, a^N) \right] + \mathbb{E}_{\overleftharpoon{\pi}^{(i+1)}} \left[\log \frac{\vec{\pi}^{(i)}(a^{0:N-1}|a^N, s)}{\overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s)} \right] \tag{46}$$

$$\geq \mathbb{E}_{\overleftharpoon{\pi}^{(i)}} \left[Q^{\pi^{(i)}}(s, a^N) \right] + \mathbb{E}_{\overleftharpoon{\pi}^{(i)}} \left[\log \frac{\vec{\pi}^{(i)}(a^{0:N-1}|a^N, s)}{\overleftharpoon{\pi}^{(i)}(a^{0:N}|s)} \right]$$

To keep the notation uncluttered we use

$$d^{(i+1)}(s, a^N) = \mathbb{E}_{\overleftharpoon{\pi}^{(i+1)}} \left[\log \frac{\vec{\pi}^{(i)}(a^{0:N-1}|a^N, s)}{\overleftharpoon{\pi}^{(i+1)}(a^{0:N}|s)} \right] \quad \text{and} \quad d^{(i)}(s, a^N) = \mathbb{E}_{\overleftharpoon{\pi}^{(i)}} \left[\log \frac{\vec{\pi}^{(i)}(a^{0:N-1}|a^N, s)}{\overleftharpoon{\pi}^{(i)}(a^{0:N}|s)} \right] \tag{47}$$

$$
\begin{aligned}
Q^{\pi^{(i)}}(s,a^N) &= r_0 + \mathbb{E}\left[\gamma\left(d^{(i)}(s_1,a_1^N) + \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[Q^{\pi^{(i)}}(s_1,a_1^N)\right]\right)\right] && (48)\\
&\leq r_0 + \mathbb{E}\left[\gamma\left(d^{(i+1)}(s_1,a_1^N) + \mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[Q^{\pi^{(i)}}(s_1,a_1^N)\right]\right)\right] && (49)\\
&= r_0 + \mathbb{E}\left[\gamma\left(d^{(i+1)}(s_1,a_1^N) + r_1\right) + \gamma^2\left(d^{(i)}(s_2,a_2^N) + \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[Q^{\pi^{(i)}}(s_2,a_2^N)\right]\right)\right] && (50)\\
&\leq r_0 + \mathbb{E}\left[\gamma\left(d^{(i+1)}(s_1,a_1^N) + r_1\right) + \gamma^2\left(d^{(i+1)}(s_2,a_2^N) + \mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[Q^{\pi^{(i)}}(s_2,a_2^N)\right]\right)\right] && (51)\\
&\vdots && (52)\\
&\leq r_0 + \mathbb{E}\left[\sum_{t=1}^{\infty}\gamma^t\left(d^{(i+1)}(s_t,a_t^N) + r_t\right)\right] = Q^{\pi^{(i+1)}}(s,a^N) && (53)
\end{aligned}
$$

Since Q improves monotonically, we eventually reach a fixed point $Q^{(i+1)} = Q^{(i)} = Q^*$

Proof of Proposition 3.3 (Policy Iteration). From Proposition 3.2 it follows that $Q^{\overleftarrow{\pi}^{i+1}}(s,a) \geq Q^{\overleftarrow{\pi}^{i}}(s,a)$. If for $\lim_{k\to\infty}\overleftarrow{\pi}^k = \overleftarrow{\pi}^*$, then it must hold that $Q^{\overleftarrow{\pi}^*(s,a)} \geq Q^{\overleftarrow{\pi}}(s,a)$ for all $\overleftarrow{\pi} \in \overleftarrow{\Pi}$ which is guaranteed by Proposition 3.2.

C Related Work

Maximum Entropy RL. The maximum entropy RL framework uses the entropy of the policy at each time step as an additional objective, providing a principled way of inducing exploration in the RL policy. It is different from entropy regularized RL [48], where the entropy of the policy is maximized only for the current time step. [25] proposed Soft-Q Learning, where amortized Stein variational gradient descent [67] (SVGD) is used to train a parameterized sampler that can sample from the energy-based policy. SAC [26] proposes an actor-critic RL method but frames the policy update as an approximate inference problem to the energy-based policy using a Gaussian policy parameterization. SAC has been extended to energy-based policies using SVGD in [44], where the authors also propose a new method to estimate the entropy in closed form. While SVGD is a powerful method for learning an energy-based policy, it is harder to scale these approaches to high-dimensional control problems. For improving exploration, LSAC [33] proposes leveraging Langevin Monte Carlo [70] in conjunction with a distributed critic objective to sample a state-action value. Recent advances of SAC also define the state-of-the-art in off-policy RL in many domains, such as CrossQ [8] and BRO [46]. CrossQ proposed removing the target network by leveraging batch renormalization and BRO scales to large networks in RL by using several methods such as optimistic exploration [45], network resets [49], weight decay, and high update-to-data ratios.

Diffusion-Based Policies in RL. Early works have researched diffusion models in offline RL [38, 39] as trajectory generators [34] or as expressive policy representations [69, 35, 28, 13, 21, 43, 23, 41]. More recently, diffusion models in online RL have become more popular. DIPO [71] proposes training a diffusion-based policy using a behavior cloning loss. The actions in the replay buffer serve as target actions for the policy improvement step and are updated using the gradients of the Q-function $\nabla_a Q(s,a)$. DIPO has been extended to develop methods for learning multi-modal behaviors[40] by leveraging hierarchical clustering to isolate different behavior modes. DIPO relies on the stochasticity inherent to the diffusion model for exploration and does not explicitly control it via an objective. QSM [52] directly matches the policy's score with the gradient of the Q-function $\nabla_a Q(s,a)$. While their objective avoids differentiating through the whole diffusion chain, the proposed objective disregards the entropy of the policy and, therefore, exploration. Consequently, QSM needs to add noise to the final action of the diffusion chain. More recently, DACER [68] proposed using the data-generating process as the policy representation and backpropagating the gradients through the diffusion chain. However, they do not consider a backward process as we do, and their objective for updating the diffusion model is based on the expected Q-values only. To incentivize the exploration, DACER adds diagonal Gaussian noise to the sampled actions, where the variance of this noise is controlled by a scaling term that is updated automatically using an approximation of the marginal entropy by

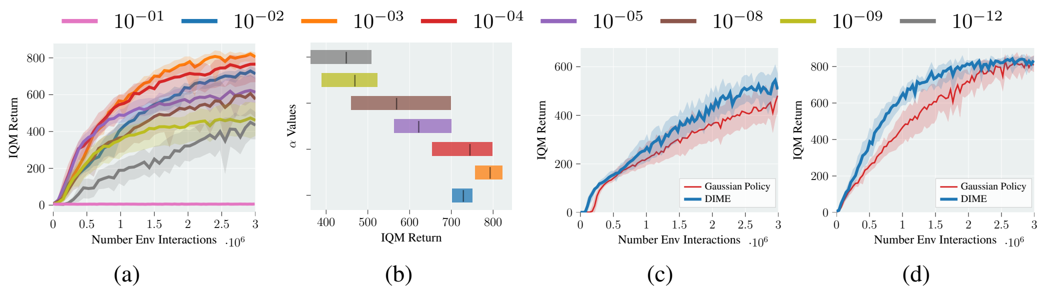


Figure 2: **Reward Scaling Sensitivity (a)-(b)**. The α parameter controls the exploration-exploitation trade-off. (a) shows the learning curves for varying values on DMC's dog-run task. Too high α values ($\alpha = 0.1$) do not incentivize learning whereas too small α values ($\alpha \leq 10^{-5}$) converge to suboptimal behavior. (b) shows the aggregated end performance for each learning curve in (a). For increasing α values, the end performance increases until it reaches an optimum at $\alpha = 10^{-3}$ after which the performance starts dropping. **Diffusion Policy Benefit (c) and (d).** We compare DIME to a Gaussian policy with the same implementation details as DIME on the (a) humanoid-run and (b) dog-run tasks. The diffusion-based policy reaches a higher return (a) and converges faster.

extracting a Gaussian Mixture Model from the diffusion policy. Concurrently, QVPO [20] proposed weighting their diffusion loss with their respective Q-values after applying transformations. However, QVPO relies on sampling actions from a uniform distribution to enforce exploration.

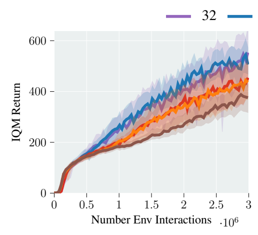
DIME distinguishes from prior works in that we use the maximum entropy RL framework for training the diffusion policy, which was not considered before. This allows direct control of the exploration-exploitation trade-off arising naturally through this objective without the need for additional approximations. DIME is leveraging the diffusion model to generate non-Gaussian exploration actions which is in contrast to most other diffusion RL approaches that still require including Gaussian or uniform exploration noise.

Approximate Inference with Diffusion Models. Early works on approximate inference with diffusion models were formalized as a stochastic optimal control problem using Schrödinger-Föllmer diffusions [18, 63, 31] and only recently realized with deep-learning based approaches [65, 73]. [64, 7] later extended these results to denoising diffusion models. A more general framework where both forward and backward processes of the diffusion model are learnable was concurrently proposed by [55, 51]. Recently, many extensions have been proposed, see e.g. [3, 50, 24, 72, 14, 10, 9, 15]. Our work can be seen as an instance of the sampler presented in [7]. However, our formulation allows using different diffusion samplers such as those presented in [55, 9], while we restrict ourselves in this work to the sampler presented in [7].

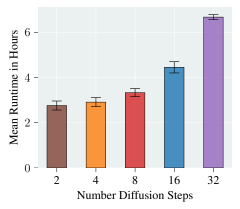
D Ablation Studies

Exploration Control. The parameter α balances the exploration-exploitation trade-off by scaling the reward signal. We analyze the effect of this parameter by comparing DIME's learning curves with different α values on the dog-run task from the DMC (see Fig. 2a). Additionally, we show the performance of the last return measurements for each learning curve in Fig. 2b. Too high α values ($\alpha = 0.1$) do not incentivize maximizing the task's return, leading to no learning at all, whereas small values ($\alpha \leq 10^{-5}$) lead to suboptimal performance because the policy does not explore sufficiently. We can also see a clear trend that starting from $\alpha = 10^{-12}$, the performance gradually increases until the best performance is reached for $\alpha = 10^{-3}$.

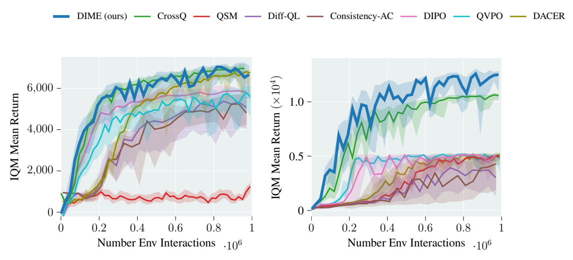
Diffusion Policy Benefit. We aim to analyze the performance benefits of the diffusion-parameterized policy compared to a Gaussian parameterization in the same setup by only exchanging the policy and the corresponding policy update. This comparison ensures that the Gaussian policy is trained with the identical implementation details from DIME as described in Sec. 3.4 and showcases the performance benefits of a diffusion-based policy. Fig. 2c and 2d show the learning curves of both versions on DMC's humanoid-run and dog-run environments. The diffusion policy's expressivity leads to a higher aggregated return in the humanoid-run and to significantly faster convergence in the high-dimensional dog-run task. We attribute this performance benefit to an improved exploration behavior.



(a) Varying the Diffusion Steps

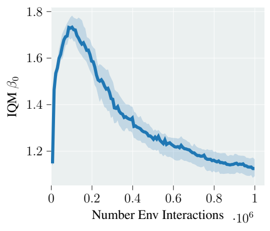


(b) Runtime for 1M Steps

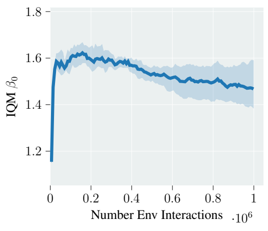


(c) Ant-v3 (d) Humanoid-v3

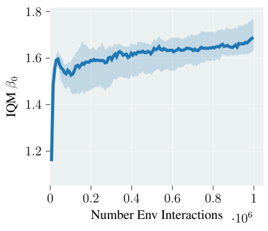
Figure 3: **Varying the Number of diffusion steps (a)-(b).** The number of diffusion steps might affect the performance and the computation time. (a) shows DIME's learning curves for varying diffusion steps. *Two* diffusion steps perform badly, whereas *four* and *eight* diffusion steps perform similar but still worse than *16* and *32* diffusion steps which perform similarly. (b) shows the computation time for 1MIO steps of the corresponding learning curves. The smaller the diffusion steps, the less computation time is required. **Learning Curves on Gym Benchmark Suite (c)-(d).** We compare DIME against various diffusion baselines and CrossQ on the (c) *Ant-v3* and (d) *Humanoid-v3* from the Gym suite. While all diffusion-based methods are outperformed by DIME, DIME performs on par with CrossQ on the Ant environment. DIME performs favorably on the high-dimensional *Humanoid-v3* environment, where it also outperforms CrossQ.



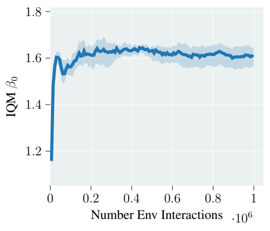
(a) β_0 on the Dog Run



(b) β_{10} on the Dog Run



(c) β_{20} on the Dog Run



(d) β_{30} on the Dog Run

Figure 4: **Learned β parameters.** DIME's policy improvement objective (Eq. 24) allows to train various parameters end-to-end, such as the scaling for the diffusion coefficient β. More concretely, we train a scaling parameter β_k per dimension k, that scales the cosine schedule. We visualize the adaptation of the parameter for the dimension $k = 0, 10, 20, 30$ over the training, averaged over 10 seeds for the dog-run task. Clearly, DIME first increases the parameter at the beginning of the training phase. Depending on the dimension, it either converges to a rather high value ($k = 20$ and $k = 30$), or keeps being reduced for other dimensions $k = 0$ and $k = 10$.

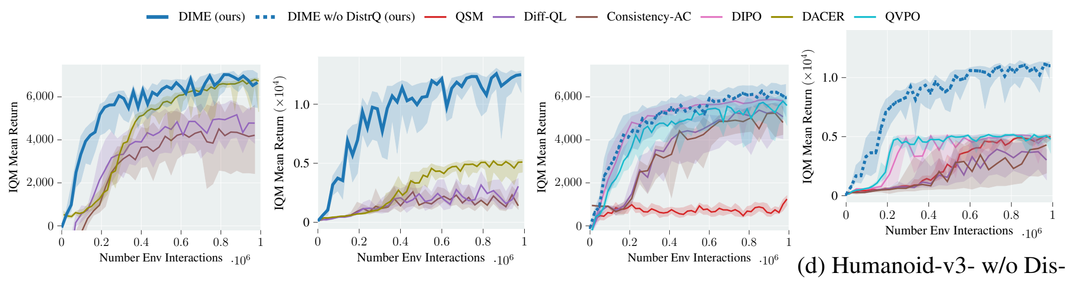
Number of Diffusion Steps. The number of diffusion steps determines how accurately the stochastic differential equations are simulated and is a hyperparameter that affects the performance. Usually, the higher the number of diffusion steps the better the model performs at the burden of higher computational costs. In Fig. 3a we plot DIME's performance for varying diffusion steps on DMC's humanoid-run environment and report the corresponding runtimes for 1 Mio environment steps in Fig. 3b on an *Nvidia A100* GPU machine. With an increasing number of diffusion steps, the performance and runtime increases. However, from 16 diffusion steps on, the performance stays the same.

E Additional Experiments

End-To-End Learning of β. We visualize the adaptation of the scaling for the diffusion coefficient β in Fig. 4 during learning on DMC's dog-run environment.

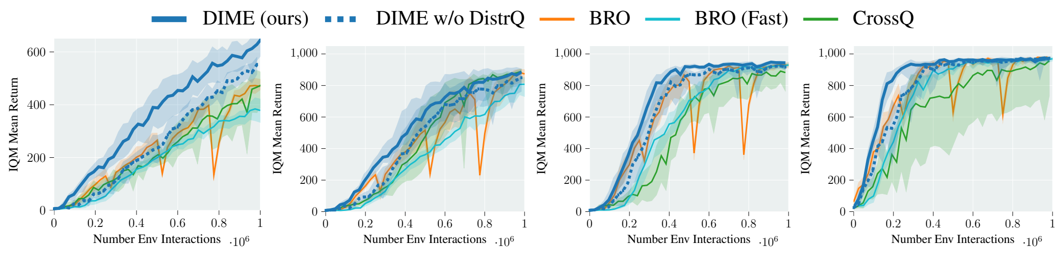
Extended Analysis on Distributional Q Learning. DIME employs distributional Q Learning [6] to represent the Q-function as a distribution over bins. We compare DIME to baselines when using distributional Q Learning and when using the well-known Bellman residual (see Eq. 22) for updating the parameters of the Q-function.

We start by comparing DIME with distributional Q learning against diffusion-based baselines that employ distributional Q learning. Fig. 5a and Fig. 5b show the learning curves on the *Ant-v3*



(a) Ant-v3 (b) Humanoid-v3 (c) Ant-v3 - w/o DistrQ (d) Humanoid-v3- w/o DistrQ

Figure 5: **Comparison to Diffusion Baselines with (a)-b)) and without Distributional Q (c)-d)) on the Ant-v3 and Humanoid-v3 tasks.** We provide the learning curves for distributional versions for Diff-QL and Consistency-AC alongside DACER, which employs distributional Q by default on the Ant-v3 (a) and Humanoid-v3 (b) tasks. DIME converges faster on the Ant-v3 (a) task to the same performance achieved by DACER and outperforms all baselines on the more high-dimensional Humanoid-v3 (b) task. Additionally, we compare DIME without distributional Q against the diffusion baselines without distributional Q on the Ant-v3 (c) and Humanoid-v3 (d) tasks. DIME without distributional Q performs on par with the baselines DIPO and QVPO on the Ant-v3 (c) and outperforms all baselines on the Humanoid-v3 (d).



(a) Dog Run (b) Dog Trot (c) Dog Walk (d) Dog Stand

Figure 6: **Ablation on Distributional Q.** Comparison of DIME and DIME without employing distributional Q (dashed line). While there is a small improvement when using distributional Q, DIME w/o Distributional Q still performs on par, or better than BRO, which employs quantile distributional RL. DIME w/o DistrQ outperforms CrossQ and BRO (Fast).

and *Humanoid-v3*, respectively, where we compare against DACER, a distributional Q variant of Diff-QL, and Consistency-AC. DIME converges faster to the same performance as DACER on the *Ant-v3* task and outperforms the baselines on the *Humanoid-v3* task. In the setting without distributional Q Learning, i.e., when updating the parameters using the residual Bellman function, DIME performs similarly to DIPO and QVPO on the *Ant-v3* task and outperforms all baselines on the higher-dimensional *Humanoid-v3* task (Fig. 5c and Fig. 5d).

Additionally, we compare DIME with and without distributional Q Learning on the four dog environments from the DMC suite (Fig. 5), where we concentrate on the strong baselines BRO [46] and CrossQ [8]. BRO employs quantile distributional Q learning, whereas CrossQ uses the Bellman residual loss function for updating the Q-function's parameters. In the main text, we have already observed that DIME with distributional Q performs favorably over the baselines. Fig. 5 shows a small improvement when using distributional Q. However, DIME without distributional Q (dashed line) still performs on par, or better than BRO and consistently performs better than BRO (Fast) and CrossQ. Please note that BRO and BRO (Fast) employ quantile distributional RL [46].

F Environments

All environments are visualized in Fig. 7. We consider the *Ant-v3* and the *Humanoid-v3* environments from mujoco gym [11]. The *humanoid-stand*, *humanoid-walk* , *humanoid-run*, *dog-stand*, *dog-walk*, *dog-trot* and *dog-run* environments from the deepmind control suite (DMC) [62]. The hand environments from myo suite are the *object-hold-random*,*reach-random*, *key-turn-random* and *pen-*

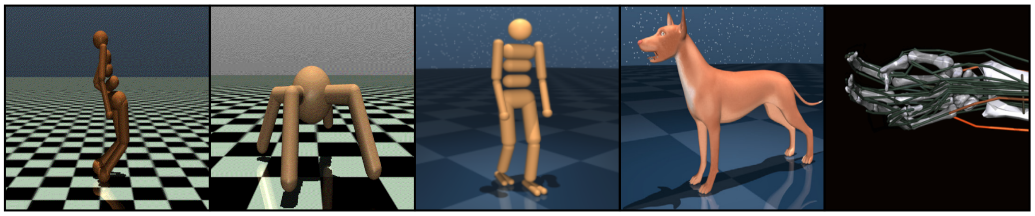


Figure 7: **Considered environments.** The *Humanoid-v3* and the *Ant-v3* are environments from the mujoco gym benchmark [11]. The three environments*humanoid-run*,*humanoid-walk* and *humanoid-stand* are from the deepmind control suite (DMC) benchmark [62]. The dog environments consist of *dog-run*, *dog-walk*, *dog-stand*, *dog-trot* and are also from the DMC sutie benchmark. Finally, the myo suite hand environments *object-hold-hard*,*reach-hard*, *key-turn-hard*, *pen-twirl-hard* are from the myo suite [12].

twirl-random environments [12]. The action and observation spaces of the respective environments are shown in Table 1.

Training Environment	**Observation Space Dim.**	**Action Space Dim.**
Ant-v3	111	8
Humanoid-v3	376	17
dog-run	223	38
dog-walk	223	38
dog-trot	223	38
dog-stand	223	38
humanoid-run	67	24
humanoid-walk	67	24
humanoid-stand	67	24
myoHandObjHoldRandom-v0	91	39
myoHandReachRandom-v0	115	39
myoHandKeyTurnRandom-v0	93	39
myoHandPenTwirlRandom-v0	83	39

Table 1: Observation and Action Space Dimensions for Various Training Environments

G Implementation Details

We consider a score network with *3* layers and a *256* dimensional hidden layer with gelu activation function. We use Fourier features to encode the timestep and scale the embedding using a feed-forward neural network with two layers, with a hidden dimension of 256. For the diffusion coefficient, we use a cosine schedule and additionally optimize a scaling parameter for the diffusion coefficient per dimension end-to-end (i.e,. we learn the parameter β (please see Appendix E).

We employ distributional Q following [6], where the Q-model outputs probabilities q over b bins. Using the bellman backup operator for diffusion models from Eq. 19 and the bin values b we follow [6] and calculate the target probabilities q_{target}. Using the entropy-regularized cross-entropy loss $\mathcal{L}(\phi) = -\sum q_{target} \log q_\phi - 0.005 \sum q_\phi \log q_\phi$ we update the parameters ϕ of the Q-function. Please note that the entropy regularization was not proposed in the original paper from [6], however, we noticed that a small regularization helps improve the performance in the early learning stages but does not change the asymptotic performance. Additionally, we follow [46] and use the *mean* of the two Q-values instead of the *min* as it has usually been used in RL so far.

The expected Q-values for updating the actor can be easily calculated using the expectation $Q(s, a_t^0) = \sum_i q_i(s_t, a_t^0) b_i$

Action Scaling. Practical applications have a bounded action space that can usually be scaled to a fixed range. However, the action range of the diffusion policy $\vec{\pi}$ is unbounded. Therefore, we follow recent works [26] and propose applying the change of variables with a *tanh* squashing function at the last diffusion step $n = 0$. For the backward process $\overleftarrow{q}_{0:N}(u^{0:N}|s)$ with unbounded action space $u \in \mathbb{R}^D$ we can squash the action $a^0 = \tanh u^0$ such that $a^0 \in (-1, 1)$ and its density is given by

$$\overleftarrow{\pi}_{0:N}(a^{0:N}|s) = \overleftarrow{q}_{0:N}(u^{0:N}|s) \det\left|\left(\frac{\mathrm{d}a^0}{\mathrm{d}u^0}\right)\right|^{-1}, \tag{54}$$

with the corresponding log-likelihood

$$\log \overleftarrow{\pi}_{0:N}(a^{0:N}|s) = \log \overleftarrow{q}_N(u^N) + \sum_{n=1}^{N} \log \overleftarrow{q}_{n-1}(u^{n-1}|u^n, s) - \sum_{i=1}^{D} \log\left(1 - \tanh^2\left(u_i^N\right)\right). \tag{55}$$

This means that the Gaussian kernels of the diffusion chain have the same log probabilities except for the correction term of the last step at $n = 0$.

Algorithm 1 DIME: Diffusion-Based Maximum Entropy Reinforcement Learning

```
Input: Initialized parameters $\theta, \phi, \alpha$, learningrates $\lambda$
 1: for $k = 1$ to M do
 2:     if k % UTD then
 3:         $a_t^{0:T} \sim \pi_{0:N}^{\theta}(a^{0:N}|s_t)$
 4:         $s_{t+1} \sim p(s_{t+1}|a_t^0, s_t)$
 5:         $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{s_t, a_t^0, r_t, s_{t+1}\}$
 6:     end if
 7:     $\phi \leftarrow \phi - \lambda_\phi \nabla_\phi J_Q(\phi)$ (Eq. 22)
 8:     if k % POLICYDELAY then
 9:         $\theta \leftarrow \theta - \lambda_\theta \nabla_\theta \mathcal{L}(\theta)$ (Eq. 23)
10:         $\alpha \leftarrow \alpha - \lambda_\alpha J(\alpha)$
11:     end if
12: end for
```

Algorithm 1 shows the learning procedure of DIME. Note that policy delay refers to the number of delayed updates of the policy compared to the critic. UTD is the update to data ratio.

H List of Hyperparameters

	DIME	**QSM**	**Diff-QL**	**Consistency-AC**	**DIPO**	**DACER**	**QVPO**
Update-to-data ratio	2	1	1	1	1	1	1
Discount	0.99	0.99	0.99	0.99	0.99	0.99	0.99
batch size	256	256	256	256	256	256	256
Buffer size	1e6	1e6	1e5	1e5	1e6	1e6	1e6
$\mathcal{H}_{target}$	4dim($\mathcal{A}$)	N/A	N/A	N/A	N/A	-0.9dim$\mathcal{A}$	N/A
Critic hidden depth	2	2	2	3	3	3	2
Critic hidden size	2048	2048	256	256	256	256	256
Actor/Score depth	3	3	4	4	4	3	2
Actor/Score size	256	256	256	256	256	256	256
Num. Bins/Quantiles	100	N/A	N/A	N/A	N/A	2	N/A
Temp. Learn. Rate	1e-3	N/A	N/A	N/A	N/A	3e-2	N/A
Learn. Rate Critic	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4
Learn. Rate Actor/Score	3e-4	3e-4	1e-5	1e-5	3e-4	3e-4	3e-4
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Diffusion Steps	16	15	5	N/A	100	20	20
Prior Distr.	$\mathcal{N}(0, 2.5)$	$\mathcal{N}(0, 1)$	N/A	N/A	N/A	$\mathcal{N}(0, 1)$	$\mathcal{N}(0, 1)$
Exploration Steps	5000	1e4	1e4	1e4	1e4	1e4	1e4
Score-Q align. factor	N/A	50	N/A	N/A	N/A	N/A	N/A

Table 2: Hyperparameters of DIME and all diffusion-based algorithms for all benchmark suits. Varying hyperparameters for different benchmark suits are described in the text.

DIME. For DIME, we use distributional Q, where the maximum and minimum values for the bins have been chosen per benchmark suite. We have used $v_{min} = -1600$ and $v_{max} = 1600$ for the

	DIME	**BRO**	**BRO Fast**	**CrossQ**
Polyak weight	N/A	0.005	0.005	N/A
Update-to-data ratio	2	10	2	2
Discount	0.99	0.99	0.99	0.99
batch size	256	128	128	256
Buffer size	1e6	1e6	1e6	1e6
$\mathcal{H}_{target}$	4dim$(\mathcal{A})$	dim$(\mathcal{A})/2$	dim$(\mathcal{A})/2$	dim$(\mathcal{A})$
Critic hidden depth	2	BRONET	BRONET	2
Critic hidden size	2048	512	512	2048
Actor/Score depth	3	BRONET	BRONET	3
Actor/Score size	256	256	256	256
Num. Bins/Quantiles	100	100	100	N/A
Temp. Learn. Rate	1e-3	3e-4	3e-4	3e-4
Learn. Rate Critic	3e-4	3e-4	3e-4	7e-4
Learn. Rate Actor/Score	3e-4	3e-4	3e-4	7e-4
Optimizer	Adam	AdamW	AdamW	Adam
Diffusion Steps	16	N/A	N/A	N/A
Prior Distr.	$\mathcal{N}(0, 2.5)$	N/A	N/A	N/A
Exploration Steps	5000	2500	2500	5000
Score-Q align. factor	N/A	N/A	N/A	N/A

Table 3: Hyperparameters of DIME and Gaussian-based algorithms for all benchmark suits. Varying hyperparameters for different benchmark suits are described in the text.

gym environments, $v_{min} = -200$ and $v_{max} = 200$ for the DMC suite and $v_{min} = -3600$ and $v_{max} = 3600$ for the myo suite.

QSM. In certain environments, we observed that QSM with default hyperparameters performed poorly, particularly in several DMC tasks and the Gym Ant-v3 tasks. To address this, we fine-tuned the hyperparameters for QSM in each of these underperforming tasks. For the DMC tasks, we found that QSM often requires an α value—representing the alignment factor between the score and the Q-function [52]—in the range of 100-200, rather than the default value of 50 reported in QSM's original implementation. In the Ant-v3 task, we determined that α needs to be set to 1. In the original implementation, the number of diffusion steps is set to be 5, however, we found using more steps, such as 10 and 15, can significantly improve the performance in these under performed tasks.

CrossQ. We used the hyperparameters from the original paper [8] for the gym benchmark suite. However, we used a different set of hyperparameters for the DMC and MYO suites for improved performance. More precisely, we increased the policy size to *3 layers* with *256 hidden size*. Additionally, we reduced the learning rate to *7e-4*.

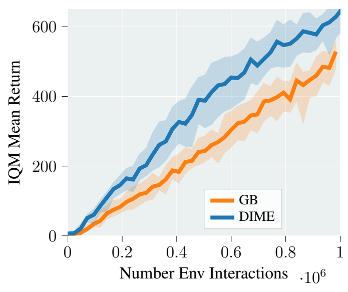
I General Diffusion Policies

DIME's maximum entropy reinforcement learning framework for training diffusion policies is not specifically restricted to denoising diffusion policies but can be extended to general diffusion policies. This can be realized using the General Bridges framework as presented in [55]. In this case, we can write the forward and backward process as

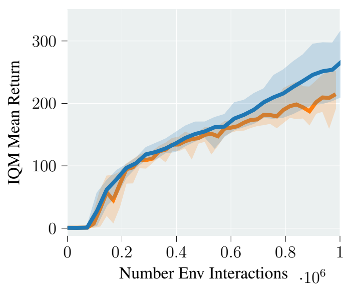
$$\mathrm{d}a_t = [f(a_t, t) + \beta u(a_t, s, t)]\,\mathrm{d}t + \sqrt{2\beta_t}\mathrm{d}B_t, \qquad a_0 \sim \vec{\pi}_0(\cdot|s), \tag{56}$$

$$\mathrm{d}a_t = [f(a_t, t) - \beta v(a_t, s, t)]\,\mathrm{d}t + \sqrt{2\beta_t}\mathrm{d}B_t, \qquad a_T \sim \mathcal{N}(0, I), \tag{57}$$

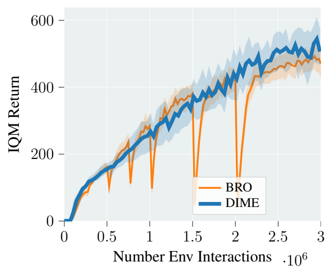
with the drift and control functions $f, u, v : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$, the diffusion coefficient $\beta : [0, T] \to \mathbb{R}^+$, standard Brownian motion $(B_t)_{t\in[0,T]}$ and some target policy $\vec{\pi}_0$. Again we denote the marginal density of the forward process as $\vec{\pi}_t$ and the conditional density at time t given l as $\vec{\pi}_{t|l}$ for $t, l \in [0, T]$. The backward process starts from $\overleftarrow{\pi}_T = \vec{\pi}_T \sim \mathcal{N}(0, I)$ and runs backward in time where we denote its density as $\overleftarrow{\pi}$.



(a) DIME and GB on Dog Run



(b) DIME and GB on Humanoid Run Run



(c) DIME and BRO on Humanoid

Figure 8: **Preliminary results for the GB sampler on the dog run (a) and humanoid run (b) environments from DMC. Comparison to BRO on the humanoid run for 3 million steps.**

The respective discretization using the Euler Maruyama (EM) [56] method are given by

$$a^{n+1} = a^n + [f(a^n, n) + \beta u(a^n, s, n)]\,\delta + \epsilon_n, \tag{58}$$

$$a^{n-1} = a^n - [f(a^n, n) - \beta v(a^n, s, n)]\,\delta + \xi_n, \tag{59}$$

where $\epsilon_n, \xi_n \sim \mathcal{N}(0, 2\beta\delta I)$, with the constant discretization step size δ such that $N = T/\delta$ is an integer. We have used the simplified notation where we write a^n instead of $a^{n\delta}$. The discretizations admit the joint distributions

$$\vec{\pi}_{0:N}(a^{0:N}|s) = \pi_0(a^0|s) \prod_{n=0}^{N-1} \vec{\pi}_{n+1|n}(a^{n+1}\big|a^n, s), \tag{60}$$

$$\overleftarrow{\pi}_{0:N}(a^{0:N}|s) = \overleftarrow{\pi}_N(a^N|s) \prod_{n=1}^{N} \overleftarrow{\pi}_{n-1|n}(a^{n-1}\big|a^n, s), \tag{61}$$

with Gaussian kernels

$$\vec{\pi}_{n+1|n}(a^{n+1}\big|a^n, s) = \mathcal{N}(a^{n+1}|a^n + [f(a^n, n) + \beta u(a^n, s, n)]\,\delta, 2\beta\delta I) \tag{62}$$

$$\overleftarrow{\pi}_{n-1|n}(a^{n-1}\big|a^n, s) = \mathcal{N}(a^{n-1}|a^n - [f(a^n, n) - \beta v(a^n, s, n)]\,\delta, 2\beta\delta I) \tag{63}$$

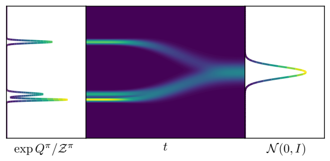
Following the same framework presented in the main text, we can now optimize the controls u and v using the same objective

$$\bar{\mathcal{L}}(u, v) = D_{\mathrm{KL}}\left(\overleftarrow{\pi}_{0:N}(a^{0:N}|s)|\vec{\pi}_{0:N}(a^{0:N}|s)\right), \tag{64}$$

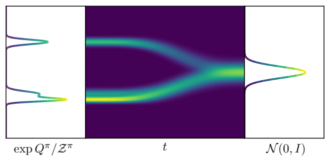
where the target policy at time step $n = 0$ is given as

$$\pi_0(a^0|s) = \frac{\exp Q^{\overleftarrow{\pi}}(s, a^0)}{\mathcal{Z}^{\overleftarrow{\pi}}(s)}. \tag{65}$$

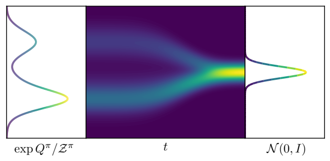
In practice, we optimize the control functions u and v using parameterized neural networks. We have run preliminary results using the general bridge framework within the maximum entropy objective as suggested in our work. The learning curves can be seen in Fig. 8.



(a) $\alpha < 1$



(b) $\alpha = 1$



(c) $\alpha > 1$

Figure 9: **The effect of the reward scaling parameter** α. The figures in (a)-(b) show diffusion processes for different α values starting at a prior distribution $\mathcal{N}(0, I)$ and going backward in time to approximate the target distribution $\exp{(Q^\pi/\alpha)}/Z^\pi$. Small values for α (a) lead to concentrated target distributions with less noise in the diffusion trajectories especially at the last time steps. The higher α becomes (b) and (c), the more the target distribution is smoothed and the distribution of the samples at the last time steps becomes more noisy. Therefore, the parameter α directly controls the exploration by enforcing noisier samples the higher α becomes.