D2 Actor Critic: Diffusion Actor Meets Distributional Critic

Lunjun Zhang
University of Toronto, Vector Institute
lunjun@cs.toronto.edu

Bradly Stadie
Northwestern University
bstadie@northwestern.edu

Abstract

We develop a new model-free reinforcement learning (RL) algorithm: **D2AC**. Motivated by recent advances in model-based RL, we make two adjustments to the typical actor critic RL pipeline. First, we learn distributional critics with a novel fusion of distributional RL and clipped double Q-learning. Second, we use a diffusion model to parameterize the policy, and derive an efficient method for aligning the diffusion process with policy improvement. These changes are highly effective, resulting in highly performant model-free policies on a benchmark of eighteen hard RL tasks, including Humanoid, Dog, and Shadow Hand domains, spanning both dense-reward and goal-conditioned RL scenarios. D2AC significantly closes the performance gap between model-free and model-based RL methods, while being at least four times faster to run than model-based counterparts.

1 Introduction

Model-based reinforcement learning (RL) has mastered challenging domains where model-free RL methods struggled [52, 39, 19]. Yet, what precisely leads to the superior performance of model-based methods has long been a curious question. For instance, following the success of model-free Atari agents such as DQN [38] and Rainbow [24], model-based Dreamer-v2 [18] surpassed model-free counterparts by using a learned world model to plan online and generate synthetic data. Not long after did model-free agent BBF [50] outperform Dreamer-v2 in the absence of any synthetic data by focusing on design choices that enable sample-efficient scaling. In continuous control and robotics, Soft Actor-Critic [17] outperformed all prior model-based methods [67], but a new generation of model-based algorithms built on top of it and eventually surpassed its success [21]. The two paradigms of RL, model-based versus model-free, are often not a dichotomy but rather carefully intertwined, continually benefiting from each other's progress.

In this paper, we take inspiration from the model-predictive control (MPC) literature to enhance model-free RL algorithms for continuous control, with the goal of significantly narrowing the performance gap between model-based and model-free RL in robotics domains. We focus on two intriguing properties of a popular MPC method called Model Predictive Path Integral Control (MPPI) [68, 70], which has been widely used in model-based RL [37, 20]:

1. Through random shooting, MPC models a distribution rather than a point estimate of future returns;
2. MPC uses an iterative refinement process on action proposals akin to denoising diffusion for policy improvement.

Those two insights allow us to comprehensively enhance critic and actor learning, and propose a new model-free RL method called **D2 Actor Critic** (D2AC). D2AC employs distributional critics [6] to model return distributions, and additionally applies the clipped double Q-learning technique [16] to make Q-function estimates more accurate. On the actor learning side, D2AC uses a denoising

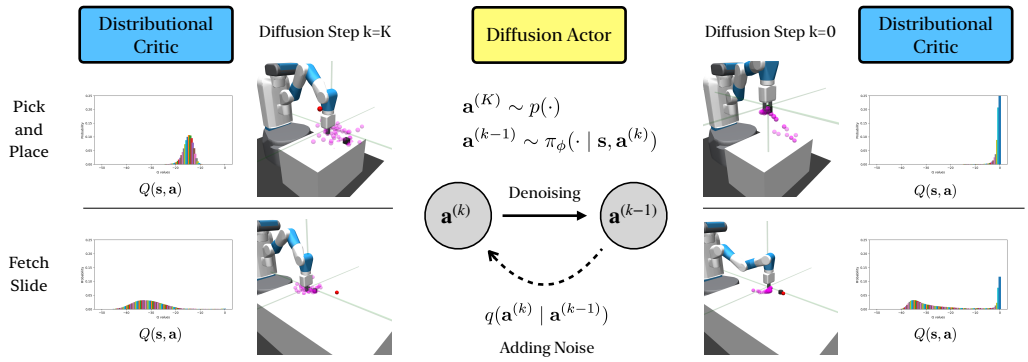

Figure 1: **D2 Actor Critic** uses a distributional Q-function to estimate a distribution over possible returns. A diffusion actor uses this Q-function to help align the denoising process with policy improvement. Above are visualizations on the Pick-and-Place and Fetch Slide environments.

diffusion model for the policy network under the formulation of EDM [31], and derives a novel and simplified policy improvement objective to supervise each step of the denoising process so that action proposals can be iteratively refined at inference. This derivation takes inspiration from the monotonic policy improvement theory [48] and applies additional simplifications to the policy improvement bound based on properties of diffusion, yielding a conceptually simple yet fast and performant diffusion algorithm for policy improvement.

D2AC achieves excellent performance on a variety of hard robotics environments, including locomotion tasks such as Humanoid and Dog domains in DeepMind Control Suite [60], and manipulation tasks such as Pick-and-Place and Shadow-Hand Manipulate in multi-goal RL environments with sparse rewards [42]. Importantly, D2AC consistently performs well across difficult domains where prior model-free RL methods struggle or completely fail. Moreover, it significantly closes the performance gap between model-free RL and SOTA model-based methods such as TD-MPC2 [20], despite using an order-of-magnitude less compute compared to TD-MPC2 and being considerably faster to run. Our results show that D2AC can serve as a strong base for policy optimization in continuous control. Videos of our policy in action can be found here `https://d2ac-actor-critic.github.io/`

2 Related Works

Model-free RL has two families of approaches: policy gradient methods and value-based methods. Policy gradient methods typically require on-policy data and trust-region regularization, as showcased in TRPO [48], PPO [49], and IMPALA [14]. Value-based methods hold the promise of utilizing any off-policy data as a means to achieving greater generalization. Since DQN was introduced [38], many important techniques have been found to help deep Q learning, including double Q-learning [61, 62], clipped double Q-learning [16], distributional RL [6, 11], soft Q learning [17], and Munchausen RL [64]. Some works have also explored a synthesis of policy gradients and Q learning. In particular, Deterministic Policy Gradient (DPG) [51] uses SARSA [44] to tackle continuous action space [35]. D2AC builds upon past progress in model-free RL: it combines distributional RL with clipped double Q learning, and uses DPG to train a single denoising step rather than directly find the optimal policy.

Model Prediction Control (MPC) requires a learned dynamics model to generate virtual rollouts and plan its action accordingly. Popular methods include Random Shooting (RS) [43], Cross-Entropy Method (CEM) [12], and Model Predictive Path Integral Control (MPPI) [68]. MPC is also closely related to model-based RL framework Dyna [59]. Traditionally, the pain point of MPC is the quality of the learnt dynamics: model-based methods had lower asymptotic performance on hard domains [67], where learning good world models requires more expressive generative models [30, 72]. MuZero [46] managed to circumvent this issue by only modeling future rewards and values without the states. Sampled MuZero [27], TD-MPC [21, 20] all combine a model-free agent, MuZero dynamics, and an off-the-shelf planning module to achieve impressive results on hard tasks in continuous control. Our work is inspired by MPC, but only tackles model-free RL. D2AC is orthogonal to model-based planning, and can be freely combined with any dynamics learning and MPC method.

Diffusion Policy for RL has attracted interest following the success of denoising diffusion models [53, 25] and its generalized formulation, denoising score matching (DSM) [58, 31, 56]. The most straightforward application of such models is conditional generative modeling of actions, or conditional behaviour cloning (BC), using a denoising policy [29, 9]. However, making denoising policies compatible with the online RL has been a challenge, because policy improvement necessarily means that the target data distribution is non-stationary. Existing works have studied how to apply diffusion policies to offline RL settings [22, 13] and online RL settings [71, 8]. Our work uses the monotonic improvement theory from policy optimization literature [48] to derive a new method of training diffusion policies in online RL setting, which significantly improves upon prior art [71].

3 Background

Reinforcement Learning A Markov Decision Process (MDP) is defined by the state space $\mathcal{S}$, action space $\mathcal{A}$, the initial state distribution $\rho_0(\mathbf{s})$, the transition probability $p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$, and the reward function $r(\mathbf{s}, \mathbf{a})$. The policy $\pi(\mathbf{a} \mid \mathbf{s})$ generates a trajectory τ with horizon T within the environment: $\tau = \{\mathbf{s}_0, \mathbf{a}_0 \cdots \mathbf{s}_T\}$, where $\mathbf{s}_0 \sim \rho_0(\cdot)$, and $\mathbf{a}_t \sim \pi(\cdot \mid \mathbf{s}_t)$, $\mathbf{s}_{t+1} \sim p(\cdot \mid \mathbf{s}_t, \mathbf{a}_t)$ for $t \in \{0, \cdots, T-1\}$. We denote the trajectory distribution starting from state s under policy π as $p_\pi(\tau \mid \mathbf{s})$. With a discount factor $\gamma \in (0, 1)$, the goal is for the policy π to maximize the discounted cumulative rewards: $\mathcal{J}(\pi) = \mathbb{E}_{\rho_0(\mathbf{s}_0) p_\pi(\tau|\mathbf{s}_0)}[\sum_{t=0}^{T-1} \gamma^{t-1} r(\mathbf{s}_t, \mathbf{a}_t)]$. Value-based RL methods optimize this objective by estimating a Q function for the current policy π: $Q(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) p_\pi(\tau|\mathbf{s}_{t+1})}[\sum_{\Delta=0}^{T-t} \gamma^{\Delta} r(\mathbf{s}_{t+\Delta}, \mathbf{a}_{t+\Delta})]$ and then training the policy π to maximize Q.

Denoising Diffusion Models are generative models that estimate the gradients of data distribution [65, 57, 54]. Diffusion models train a denoiser D_ϕ to reconstruct data $\mathbf{x} \sim p_{\text{data}}$ from noised data $\mathbf{x}^{(k)} \sim \mathcal{N}(\mathbf{x}, \sigma_k^2 \mathbf{I})$ under various noise levels $\sigma_{\max} = \sigma_K > \cdots > \sigma_0 = \sigma_{\min}$. Let $p(\mathbf{x}^{(k)}) = \int p_{\text{data}}(\mathbf{x}) \mathcal{N}(\mathbf{x}^{(k)} | \mathbf{x}, \sigma_k^2 \mathbf{I}) \mathrm{d}\mathbf{x}$. The min $\sigma_{\min}$ and max $\sigma_{\max}$ of noise levels are selected such that $p(\mathbf{x}^{(0)}) \approx p_{\text{data}}(\mathbf{x})$ and $p(\mathbf{x}^{(K)}) \approx \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$. The diffusion loss is typically: $\mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), k \sim \text{Unif}(1 \cdots K)}[\lambda(k) \| D_\phi(\mathbf{x} + \boldsymbol{\epsilon} \sigma_k; \sigma_k) - \mathbf{x} \|^2]$, where $\lambda(k)$ is the loss weighting based on noise level. Then the score for arbitrary input $\mathbf{x}$ is $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma) = (D_\phi(\mathbf{x}; \sigma) - \mathbf{x}) / \sigma^2$, which can be used to sample data by starting from noise and solving either an SDE or ODE [58]. We follow EDM [31] to define the denoiser as $D_\phi(\mathbf{x}, \sigma) = c_{\text{skip}}(\sigma) \mathbf{x} + c_{\text{out}}(\sigma) F_\phi(c_{\text{in}}(\sigma) \mathbf{x}, c_{\text{noise}}(\sigma))$.

4 Clipped Double Distributional Q-Learning

Distributional reinforcement learning is the idea that, rather than modeling a point estimate of future returns, we should instead model the entire distribution of returns. This has been shown to be highly effective [6]. Yet, issues from non-distributional RL tend to carry over to the distributional case as well. In particular, a major pain-point of Q-learning is the over-estimation of Q-values. It has been shown [61, 62] that under mild assumptions Q-values will always be systemically over-estimated, leading to poor behaviors when agents become confident about incorrect actions. In the distributional case, this manifests as regions where the return distribution are systemically over-estimated.

Below, we will show how we can combine distributional RL with clipped double Q-learning (CDQ) [16], a trick that is known to be highly effective at regularizing over-estimated Q-values.

4.1 Categorical Representation

In categorical distributional RL, Q-functions are designed like a classifier (softmax over logits) [6] to output discrete probabilities $\mathbf{q}_\theta(\mathbf{s}, \mathbf{a})$ over a list of possible values $\mathbf{z}_q$. Suppose that the range of possible values is $[V_{\min}, V_{\max}]$. This range is divided into N intervals with $N+1$ *bins*, and the values that the $N+1$ bins represents are often called the *support*, denoted as $\mathbf{z}_q$, starting with $V_{\min}$ and ending with $V_{\max}$. Each bin corresponds to a discrete probability outputted by the Q function. For a categorical Q function $\mathbf{q}_\theta(\mathbf{s}, \mathbf{a})$ in distributional RL, the *expected* Q value is:

$$Q_\theta(\mathbf{s}, \mathbf{a}) = \mathbf{q}_\theta(\mathbf{s}, \mathbf{a})^\top \mathbf{z}_q \tag{1}$$

An important fact about such a categorical parameterization is that it can represent any continuous value within its range using a linear combination of two adjacent bins. For instance, given the support $\mathbf{z}_q = [1, 2, 3, 4]$, the value 3.7 can simply be represented as $h_{\mathbf{z}_q}(3.7) = [0, 0, 0.3, 0.7]$, since $3.7 = 3 \times 0.3 + 4 \times 0.7$. The function $h_{\mathbf{z}_q}$ is called the *two-hot* function.

4.2 Categorical Projection

The key of distributional RL is to derive the categorical labels $\boldsymbol{\ell}$ for directly supervising the logits of the Q function via a cross-entropy loss. This is computed via one-step bootstrapping.

Given the transition tuple $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, the return distribution under the next state $\mathbf{s}'$ is used to supervise the logits under the current state $\mathbf{s}$. We can use the target network $\mathbf{q}_{\bar{\theta}}$ and the policy $\mathbf{a}' \sim \pi_\phi(\cdot \mid \mathbf{s}')$ to obtain $\mathbf{q}_{\bar{\theta}}(\mathbf{s}', \mathbf{a}')$, the probabilities of returns under $\mathbf{s}'$. For the current state, the return distribution of the next state happens after a reward r and a discount factor γ. From the perspective of $\mathbf{s}$, the support for the return distribution under $\mathbf{s}'$ has shifted to $\mathbf{z}_p = (r + \gamma \mathbf{z}_q)$, in contrast to the support $\mathbf{z}_q$ for $\mathbf{s}$.

To compute the learning target for $\mathbf{q}_\theta(\mathbf{s}, \mathbf{a})$, the probabilities $\mathbf{p} = \mathbf{q}_{\bar{\theta}}(\mathbf{s}', \mathbf{a}')$ under support $\mathbf{z}_p$ need to be projected back to $\mathbf{z}_q$. Intuitively, such a projection needs to project each value of the bins $\mathbf{z}_p[j]$ (under $\mathbf{s}'$) back to the original support $\mathbf{z}_q$ (under $\mathbf{s}$) using the two-hot function $h_{\mathbf{z}_q}(\cdot)$, and sum across the bins $j = 0 \cdots N$ using the probabilities $\mathbf{p}[j]$. This projection operator $\boldsymbol{\Phi}_{\text{dist}}$ is defined as:

$$\boldsymbol{\Phi}_{\text{dist}}(\mathbf{z}_p, \mathbf{p}, \mathbf{z}_q)[k] = \sum_{j=0}^{N} h_{\mathbf{z}_q}(\mathbf{z}_p[j])[k] \cdot \mathbf{p}[j] \tag{2}$$

4.3 Clipped Double Distributional RL

To apply double Q-learning in the distributional setting, we learn two Q-functions with parameters $\theta = \{\theta_1, \theta_2\}$. Let $\mathbf{q}_{\theta_i}(\mathbf{s}, \mathbf{a}) = \text{softmax}(\text{logits}_{\theta_i}(\mathbf{s}, \mathbf{a}))$, for $i \in \{1, 2\}$. We have:

$$Q_{\theta_i}(\mathbf{s}, \mathbf{a}) = \mathbf{q}_{\theta_i}(\mathbf{s}, \mathbf{a})^\top \mathbf{z}_q \quad Q_\theta^{\min}(\mathbf{s}, \mathbf{a}) = \min_{i=1,2} Q_{\theta_i}(\mathbf{s}, \mathbf{a}) \tag{3}$$

We then define clipped distributional target as the label from the critic with the lower expected value:

$$\mathbf{q}_\theta^{\text{clip}}(\mathbf{s}, \mathbf{a}) = \begin{cases} \mathbf{q}_{\theta_1}(\mathbf{s}, \mathbf{a}) & Q_{\theta_1}(\mathbf{s}, \mathbf{a}) \le Q_\theta^{\min}(\mathbf{s}, \mathbf{a}) \\ \mathbf{q}_{\theta_2}(\mathbf{s}, \mathbf{a}) & Q_{\theta_2}(\mathbf{s}, \mathbf{a}) \le Q_\theta^{\min}(\mathbf{s}, \mathbf{a}) \end{cases} \tag{4}$$

The critic learning loss is a cross-entropy loss applied to two critics with the same target:

$$\begin{aligned} &\mathbf{a}' \sim \pi_\phi(\cdot \mid \mathbf{s}') \quad \boldsymbol{\ell} = \boldsymbol{\Phi}_{\text{dist}}(r + \gamma \mathbf{z}_q, \mathbf{q}_{\bar{\theta}}^{\text{clip}}(\mathbf{s}', \mathbf{a}'), \mathbf{z}_q) \\ &L_{\text{critic}}(\theta) = -\boldsymbol{\ell}^\top (\log \mathbf{q}_{\theta_1}(\mathbf{s}, \mathbf{a}) + \log \mathbf{q}_{\theta_2}(\mathbf{s}, \mathbf{a})) \end{aligned} \tag{5}$$

In practice, only using a single sample from $\mathbf{a}'$ to evaluate the Q-function under $\mathbf{s}'$ is sufficient for stable learning of the distributional critics. The critic $Q_\theta^{\min}$ is the one we use to supervise the actor.

5 Policy Improvement for Diffusion Actors

When the policy π in actor-critic algorithms is a diffusion model [53, 25], it starts from the noise distribution $\mathcal{N}(0, \sigma_{\text{max}}^2 \mathbf{I})$ and goes through K diffusion steps $\mathbf{a}^{(K)} \to \mathbf{a}^{(K-1)} \cdots \to \mathbf{a}^{(0)}$ using a sequence of noise levels $\sigma_{\text{max}} = \sigma_K > \cdots > \sigma_1 > \sigma_0 = \sigma_{\text{min}}$ to sample an action. If we assume a sampler based on the Euler-Maruyama method [58], then one step of the diffusion process $\mathbf{a}^{(k)} \to \mathbf{a}^{(k-1)}$ for policy π_ϕ can be written as:

$$\pi_\phi(\mathbf{a}^{(k-1)} \mid \mathbf{s}, \mathbf{a}^{(k)}) = \mathcal{N}(\mathbf{a}^{(k-1)} \mid \boldsymbol{\mu}_\phi(\mathbf{a}^{(k)}, k; \mathbf{s}), (\sigma_k^2 - \sigma_{k-1}^2)\mathbf{I})$$

Where $\boldsymbol{\mu}_\phi$ is a learned denoising process that takes actions away from the noise and towards the desired action distribution. It is parameterized as:

$$\begin{aligned} \boldsymbol{\mu}_\phi(\mathbf{a}^{(k)}, k; \mathbf{s}) &= \mathbf{a}^{(k)} + (\sigma_k^2 - \sigma_{k-1}^2) \nabla_{\mathbf{a}} \log p(\mathbf{a}^{(k)} \mid \sigma_k, \mathbf{s}) \\ \nabla_{\mathbf{a}} \log p(\mathbf{a} \mid \sigma, \mathbf{s}) &= (D_\phi(\mathbf{a}, \sigma; \mathbf{s}) - \mathbf{a})/\sigma^2 \end{aligned}$$

where $D_\phi(\mathbf{a}, \sigma; \mathbf{s}) = c_{\text{skip}}(\sigma)\mathbf{a} + c_{\text{out}}(\sigma) F_\phi(\mathbf{s}, c_{\text{in}}(\sigma)\mathbf{a}, c_{\text{noise}}(\sigma))$. The functions $c_{\text{skip}}(\sigma)$, $c_{\text{out}}(\sigma)$, and $c_{\text{noise}}(\sigma)$ are flexible design choices [31]. Thus, taking an action $\mathbf{a}^{(0)}$ from the denoising process $\mathbf{a}^{(k)} \to \cdots \mathbf{a}^{(0)}$ can be written as an integral over the denoising steps:

$$\pi_\phi(\mathbf{a} \mid \mathbf{s}) = \int p(\mathbf{a}^{(K)}) \prod_{k=1}^{K} \pi_\phi(\mathbf{a}^{(k-1)} \mid \mathbf{s}, \mathbf{a}^{(k)}) \mathrm{d}\mathbf{a}_{1:K} \tag{6}$$

We would like a way to guarantee the diffusion process actually provides us with a better action. Define the optimal policy to be $p^*(\mathbf{a} \mid \mathbf{s}) = (\exp Q(\mathbf{s}, \mathbf{a})/\alpha)/Z(\mathbf{s})$, where α is the temperature. The goal of entropy-regularized policy improvement is to minimize the following [47, 17, 7]:

$$\arg\max_{\phi} - D_{KL}(\pi_\phi(\mathbf{a} \mid \mathbf{s}) \parallel p^*(\mathbf{a} \mid \mathbf{s})) = \mathbb{E}_{\mathbf{a} \sim \pi_\phi(\cdot|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})] + \alpha H(\pi_\theta(\mathbf{a} \mid \mathbf{s})) \tag{7}$$

Diffusion Policy Gradients: In the case of diffusion policies, our main challenge with optimizing for policy improvement is that accessing $\mathbf{a} \sim \pi_\phi$ requires K intermediate sampling steps. To optimize a lower bound on the policy gradient objective, the variational lower bound in [32] can be applied with a positive transformation of the Q-values [41, 1], with $\lambda_{\text{pg}}(k) = (1/\sigma_{k-1}^2 - 1/\sigma_k^2) \cdot (K/2)$:

$$\mathbb{E}_\pi \, \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), k \sim \text{Unif}(1 \cdots K)}[\lambda_{\text{pg}}(k) \| \mathbf{a} - D_\phi(\mathbf{a} + \sigma_k \boldsymbol{\epsilon}, \sigma_k; \mathbf{s}) \|^2 \cdot \exp Q(\mathbf{s}, \mathbf{a})] \tag{8}$$

On the one hand, policy gradients tend to suffer from high variance and instability. On the other hand, value gradients [23] for diffusion require backpropagation through time (BPTT), which is known to be difficult [40]. Those challenges have been previously noted in [66, 7, 10]. Next, we will show that by re-examining the MDP view of diffusion process and making mild assumptions, we can circumvent those challenges. The main insight is that we can convert a classical policy improvement result from the literature into an easier-to-manage form by isolating the effect of one-step denoising.

Diffusion MDP: Towards a policy improvement result for a diffusion actor, let's first define a diffusion MDP as taking K steps of actions from $p(\mathbf{a}^{(K)})$ to $\pi(\mathbf{a}^{(k-1)} \mid \mathbf{s}, \mathbf{a}^{(k)})$, reaching the final state $\mathbf{a}^{(0)}$. After this diffusion process (under discount factor γ), the policy receives a reward $Q(\mathbf{s}, \mathbf{a}^{(0)})/\gamma^K$ at the final timestep of diffusion (reward is 0 elsewhere). The objective of diffusion is to maximize:

$$\eta(\mathbf{s}, \pi) = \mathbb{E}_{\mathbf{a}^{(K)} \cdots \mathbf{a}^{(0)} \sim \pi}[Q(\mathbf{s}, \mathbf{a}^{(0)})] \tag{9}$$

Where we have made the dependence on the intermediate diffusion steps $\mathbf{a}^{(K)} \cdots \mathbf{a}^{(1)}$ explicit. In the diffusion MDP, the Q-function, value function, and advantage function are given by:

$$\begin{aligned}
V_\pi^{\text{diffusion}}((\mathbf{s}, \mathbf{a}^{(k)})) &\triangleq \mathbb{E}_{\mathbf{a}^{(k-1)} \cdots \mathbf{a}^{(0)} \sim \pi}[(\gamma^{k-1}/\gamma^K) Q(\mathbf{s}, \mathbf{a}^{(0)})] \\
Q_\pi^{\text{diffusion}}((\mathbf{s}, \mathbf{a}^{(k+1)}), \mathbf{a}^{(k)})) &\triangleq \mathbb{E}_{\mathbf{a}^{(k-1)} \sim \pi}[\gamma V_\pi^{\text{diffusion}}((\mathbf{s}, \mathbf{a}^{(k-1)}))] \\
A_\pi^{\text{diffusion}}((\mathbf{s}, \mathbf{a}^{(k+1)}), \mathbf{a}^{(k)})) &\triangleq Q_\pi^{\text{diffusion}}((\mathbf{s}, \mathbf{a}^{(k+1)}), \mathbf{a}^{(k)})) - V_\pi^{\text{diffusion}}((\mathbf{s}, \mathbf{a}^{(k+1)}))
\end{aligned}$$

Monotonic improvement theory, and in particular the policy improvement objective from Trust Region Policy Optimization [48], can now be applied naively to the diffusion MDP framework. Doing so will give us gives us the following proximal objective for policy improvement:

$$\begin{aligned}
\eta(\mathbf{s}, \pi) &\geq J_{\pi_{\text{ref}}}(\mathbf{s}, \pi) - \frac{4\epsilon\gamma}{(1-\gamma)^2} \left\{ \max_{k, \mathbf{a}^{(k)}} D_{KL}(\pi_{\text{ref}}(\cdot \mid \mathbf{s}, \mathbf{a}^{(k)}), \pi(\cdot \mid \mathbf{s}, \mathbf{a}^{(k)})) \right\} \\
J_{\pi_{\text{ref}}}(\mathbf{s}, \pi) &\triangleq \eta(\mathbf{s}, \pi_{\text{ref}}) + \mathbb{E}_{\substack{k \sim \text{Unif}\{1, K\}, \mathbf{a}^{(k)} \sim \pi_{\text{ref}} \\ \mathbf{a}^{(k-1)} \sim \pi(\cdot|\mathbf{a}^{(k)}, \mathbf{s})}}[\gamma^k A_{\pi_{\text{ref}}}^{\text{diffusion}}((\mathbf{s}, \mathbf{a}^{(k)}), \mathbf{a}^{(k-1)})]
\end{aligned}$$

These equations say that we can lower bound our policy objective (9), by a loss that depends on the Diffusion MDP advantage of the reference policy $A_{\pi_{\text{ref}}}^{\text{diffusion}}$, and the KL between the reference policy π_{ref} and the policy we are learning π. Essentially, if π and π_{ref} are sufficiently close, then if π_{ref} makes an improvement, so does π. Since this is a lower bound on the target objective (9), we can move in the direction of improvement by taking the gradient of $J_{\pi_{\text{ref}}}(\mathbf{s}, \pi)$ in the lower bound

$$\nabla_\phi J_{\pi_{\text{ref}}}(\mathbf{s}, \pi_\phi) = \nabla_\phi \underbrace{\mathbb{E}_{k \sim \text{Unif}\{1, K\}, \mathbf{a}^{(k)} \sim \pi_{\text{ref}}} \left[\gamma^k \, \mathbb{E}_{\mathbf{a}^{(k-1)} \sim \pi_\phi}[Q_{\pi_{\text{ref}}}^{\text{diffusion}}((\mathbf{s}, \mathbf{a}^{(k)}), \mathbf{a}^{(k-1)})] \right]}_{J_{\pi_{\text{ref}}}^{\text{proximal}}(\mathbf{s}, \pi_\phi)} \tag{10}$$

Note that in practice this is still difficult, because we need to learn a separate Q function $Q^{\text{diffusion}}$.

Proposed simplification: Rather than learning $Q_{\pi_{\text{ref}}}^{\text{diffusion}}$ for diffusion MDP, we utilize the fact that the denoising function $\hat{\mathbf{a}} = D_\phi(\mathbf{a}^{(k)}, \sigma; \mathbf{s})$ directly aims to predict a strictly cleaner signal with each passing step [54]. The diffusion literature has shown that [25, 58] for a well-trained diffusion model, more sampling steps result in smaller truncation error in the SDE solver and higher quality of $\hat{\mathbf{a}}$. Since Q-value is analogous to unnormalized log probability [34], it is plausible to reason that, under mild assumptions, *more sampling steps should lead to higher Q-value*, giving us a lower bound on the Q-value based on a single denoising step. We formalize this intuition in the following proposition.

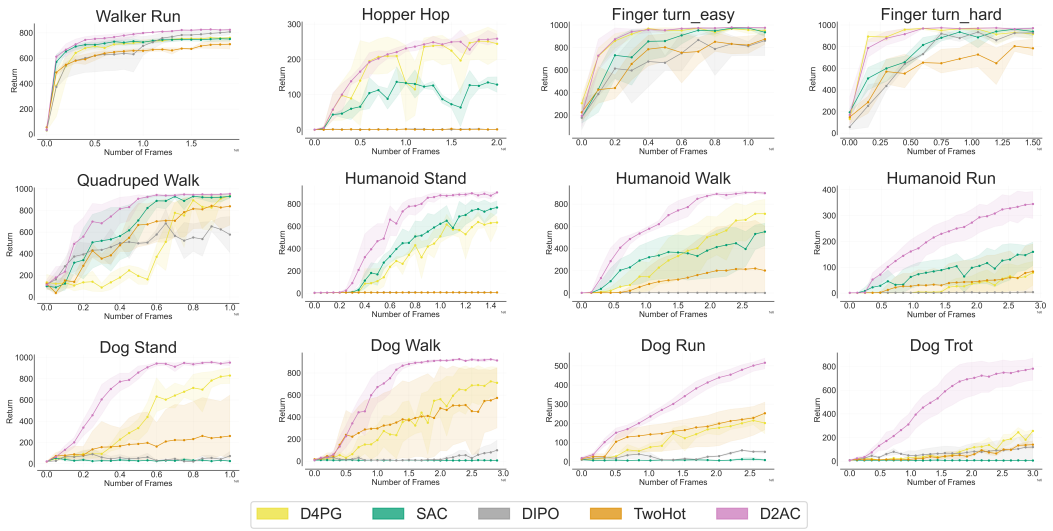

Figure 2: Experiments on **DeepMind Control Suite**. Results over 5 seeds. In model-free RL, D2AC achieves much better sample efficiency and asymptotic performance compared to all other baselines.

Proposition 1. *Define the data distribution under diffusion step $k \geq 0$ to be (with $\hat{\sigma} \geq \sigma_{min}$):*

$$p_k(\mathbf{a}|\mathbf{s}) = \int \pi(\mathbf{a}^{(0)}|\mathbf{s})\mathcal{N}(\mathbf{a}|\mathbf{a}^{(0)}, \sigma_k^2\mathbf{I})\mathrm{d}\mathbf{a}^{(0)} \quad \hat{p}_k(\mathbf{a}|\mathbf{s}) = \int p_{k+1}(\mathbf{u}|\mathbf{s})\mathcal{N}(\mathbf{a}|D(\mathbf{u}, \sigma_{k+1}; \mathbf{s}), \hat{\sigma}^2\mathbf{I})\mathrm{d}\mathbf{u}$$

Under the assumptions that (i) entropy of p_k and $\hat{p}_k$ strictly increases with k; (ii) the KL distance from p_k and $\hat{p}_k$ to optimal policy p^* strictly decreases with k; (iii) $\hat{\sigma}$ is sufficiently large such that $D_{KL}(\hat{p}_0 \parallel p^*) \geq D_{KL}(p_0 \parallel p^*)$. Then for any state $\mathbf{s}$ and diffusion step k, we have

$$\mathbb{E}_{p_0}[Q] \geq \mathbb{E}_{\hat{p}_k}[Q] \tag{11}$$

Which results in the following lower bound of equation (10) based on one-step generation:

$$\begin{aligned}
&\gamma^k \, \mathbb{E}_{\mathbf{a}^{(k-1)} \sim \pi_\phi}[Q^{\text{diffusion}}_{\pi_{\text{ref}}}((\mathbf{s}, \mathbf{a}^{(k)}), \mathbf{a}^{(k-1)})] \\
=&\gamma^{2k-K} \, \mathbb{E}_{\mathbf{a}^{(k-1)} \sim \pi_\phi} \underbrace{\mathbb{E}_{\mathbf{a}^{(k-2)} \cdots \mathbf{a}^{(0)} \sim \pi_{\text{ref}}}}_{\text{Additional } k-1 \text{ SDE steps}} [Q(\mathbf{s}, \mathbf{a}^{(0)})] \geq \gamma^{2k-K} \underbrace{\mathbb{E}_{\mathcal{N}(\hat{\mathbf{a}}|D_\phi(\mathbf{a}^{(k)}, \sigma_k; \mathbf{s}); \hat{\sigma}^2)}}_{\text{One step generation}} [Q(\mathbf{s}, \hat{\mathbf{a}})]
\end{aligned}$$

Intuitively, having additional refinement steps on action $\hat{\mathbf{a}}$ should result in higher Q-value on average.

Diffusion Value Gradients: This lower bound based on one-step generation is much easier to optimize, since it avoids the requirement for separate value functions $Q^{\text{diffusion}}_{\pi_{\text{ref}}}$ in diffusion MDP.

$$J^{\text{proximal}}_{\pi_{\text{ref}}}(\mathbf{s}, \pi_\phi) \geq \mathbb{E}_{k \sim \text{Unif}\{1,K\}, \mathbf{a}^{(k)} \sim \pi_{\text{ref}}, \boldsymbol{\epsilon} \sim \mathcal{N}(0,\mathbf{I})}[(\gamma^{2k}/\gamma^K) \cdot Q(\mathbf{s}, D_\phi(\mathbf{a}^{(k)}, \sigma_k; \mathbf{s}) + \hat{\sigma}\boldsymbol{\epsilon})] \tag{12}$$

This simplified policy objective can be written in a form similar to the L_2 loss commonly used in diffusion models [32]. In essence, the loss takes a noised version of on-policy action, and learns how to iteratively improve upon this noised action via denoising guided by the critic Q. Each intermediate diffusion step is aligned towards policy improvement. Setting $\pi_{\text{ref}} = \pi_\phi$, the D2AC policy loss is:

$$\nabla_\phi L^{\text{simple}}_\pi(\phi) = \mathbb{E}_{\substack{\mathbf{a} \sim \pi_\phi(\cdot|\mathbf{s}) \\ \boldsymbol{\epsilon} \sim \mathcal{N}(0,I), k \sim \text{Unif}\{1,K\} \\ \mathcal{N}(\hat{\mathbf{a}}|D_\phi(\mathbf{a}+\sigma_k\boldsymbol{\epsilon}, \sigma_k; \mathbf{s}), \hat{\sigma}^2)}} \nabla_\phi \left[\lambda(k) \, \| \hat{\mathbf{a}} + \nabla_{\hat{\mathbf{a}}} Q(\mathbf{s}, \hat{\mathbf{a}}) - D_\phi(\mathbf{a} + \sigma_k\boldsymbol{\epsilon}, \sigma_k; \mathbf{s}) \|^2 \right] \tag{13}$$

The function $\lambda(k)$ is the weighting according to the noise schedule. $\lambda(k) = \gamma^{2k}/\gamma^K$ in our derivation; in practice we find that simply setting $\lambda(k) = 1$ works well. Compared to diffusion policy gradients (15), our diffusion value gradients are designed to directly align the noisy predictions based on value gradients from Q without backprop-through-time, thus providing cleaner supervision signal.

6 Experiments

In these experiments, we aim to investigate the performance of D2AC across a variety of settings. These settings seek to stress test the algorithm by changing both the task modality (locomotion,

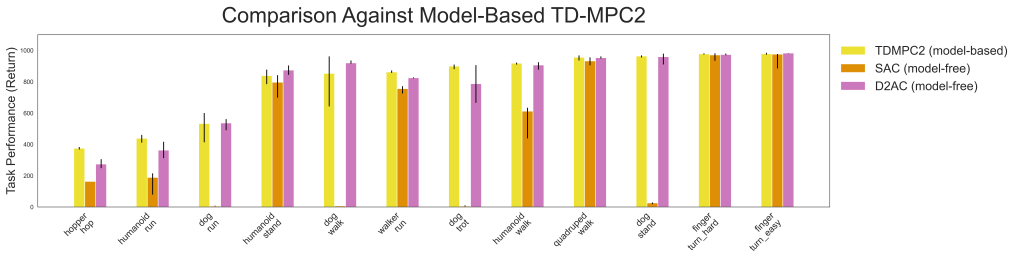

Figure 3: Comparison between model-based TD-MPC2 [20], SAC [17], and our method D2AC on DeepMind Control Suite. **D2AC without planning** can achieve results on-par with TD-MPC2.

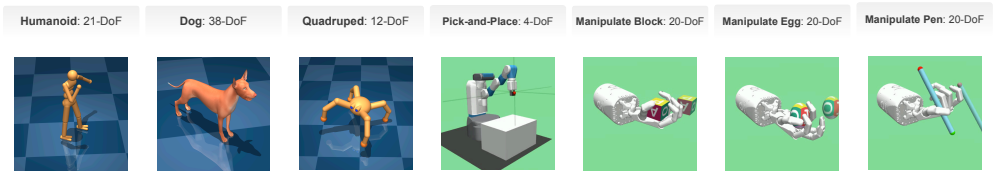

Figure 4: D2AC works out of the box across a wide range of environments, including locomotion and manipulation with sparse and dense rewards.

grasping, manipulation), and the reward modality (dense vs sparse). We take special care to investigate model-based RL in many of these settings, trying to understand how much we can close the gap between the two paradigms. Finally, we consider the practical scalability of D2AC by investigating the algorithm's wall clock run time and the relationship between available compute and expected model performance. We use rliable [2] library for rigorous statistical evaluation.

Evaluation environments: We focus on two primary settings: (i) DeepMind Control Suite [60], which mostly comprises locomotion tasks with dense rewards; (ii) Multi-Goal RL environments proposed in [42], which are robotic manipulation tasks with sparse rewards. A list of the environments we use are shown in Figure 4. Those environments include 21-DoF Humanoid, 38-DoF Dog, 12-DoF Quadruped, a 4-DoF Pick-and-Place task, and a series of 20-DoF Shadow Hand tasks aiming to manipulate Block, Egg, and Pen to target orientations.

6.1 Benchmarking on dense-reward environments

In Figure 2, we showcase our benchmarking results on 12 tasks in DeepMind control suite. We compare our method to a variety of baselines, including: Soft Actor Critic [17], D4PG [5], and the model-free component of TD-MPC2 [20], which we denote as *Two-hot + CDQ*, because it essentially combines SAC, two-hot critic representation, and clipped double Q-learning. We also compare against DIPO [71], which is a model-free RL method based on diffusion policy. D2AC achieves higher asymptotic performance across all 12 environments, and generally learns faster.

When compared to TD-MPC2 [20] trained with the same number of environment interactions, we see that D2AC is very competitive despite being model-free. In Figure 3, we see that D2AC achieves performance within five percent of TD-MPC2 on 8 out of 12 benchmark environments. On 4 of the environments, the performance of D2AC exceeds TD-MPC2. The two standout environments were Dog Walk and Humanoid Stand, where D2AC significantly outpaced TD-MPC2 performance. The relatively poor performance of *Two-hot + CDQ* baseline shows that the model-free component of TD-MPC2 is not in isolation a strength of the algorithm, and the strong performance of TD-MPC2 relies on its ability to pair this model free algorithm with its planning module. Our proposed model-free learning algorithm is orthogonal to any planning algorithm, and can be used as a drop-in replacement in any model-based method such as TD-MPC2.

D2AC Is highly performant in a wide variety of settings: D2AC achieves significantly better sample efficiency and asymptotic performance compared to all other model-free baselines we tested. This is true across multiple control modalities (grasping, locomotion, manipulation). We use 5 seeds for the results shown in Figure 2 and Figure 3; we also use the rliable [2] evaluation under Interquartile Mean (IQM) with interval estimates via stratified bootstrap confidence intervals.

Thanks to being model-free, **our method is considerably faster to run compared to TD-MPC2**: see Figure 7 for a clear comparison. In particular, D2AC is able to make significant process on the Dog Trot task within 24 hours, where TD-MPC2 has barely started increasing its episodic return; SAC completely fails on this task.

Overall, the strong performance of D2AC seems to suggest that model-based RL's recent success does not stem entirely from planning with rollouts. Rather, a major component of this success is model-based RL's ability to model a distribution of returns and iteratively refine action proposals.

6.2 Benchmarking on goal-conditioned RL environments

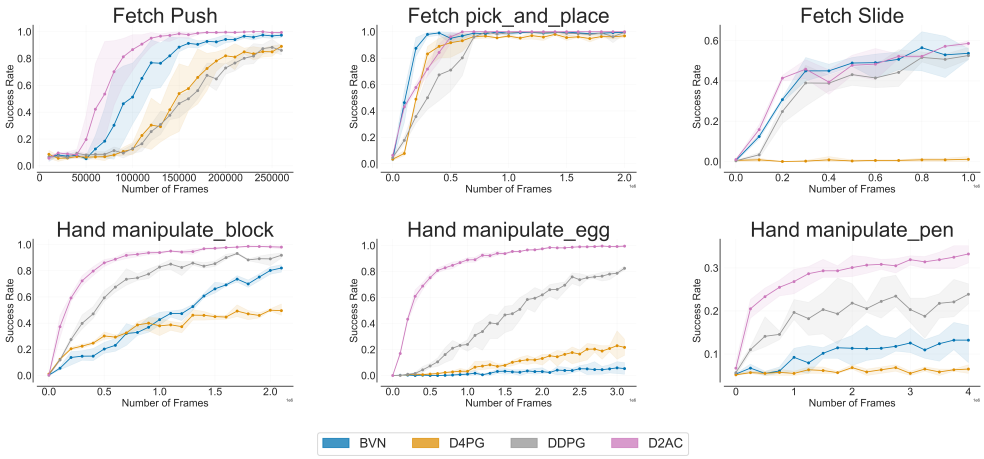

Figure 5: Experiments on **Multi-Goal RL environments with sparse rewards**. Results over 5 seeds.

In Figure 5, we present results on robotic manipulation tasks with sparse rewards. All methods are trained with Hindsight Experience Replay (HER) [3]. The baselines include: DDPG [35], a distributional version of HER [15], and recently proposed Bilinear-Value Network (BVN) [26] found to be effective in goal-conditioned RL. We use a centralized learner and 20 sampling workers for all methods. We find that on Fetch tasks, D2AC performs better or on-par with previous SOTA method BVN. On 20-DoF Hand Manipulate tasks, the second-best method is still DDPG, and our method is able to outperform it by a large margin, often both in terms of learning speed and policy performance at convergence. **This performance shows that D2AC is a strong base algorithm that can work out-of-the-box in a variety of settings.**

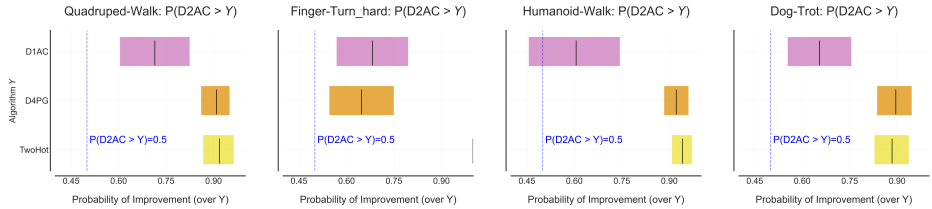

Figure 6: Ablation Studies based on Probability of improvement [2] using the recommended protocol from rliable. The X-axis gives the probability D2AC improves upon the performance of the baseline algorithm on the Y-axis. The boxes represent 95% confidence intervals for each baseline.

Interestingly, the previously proposed distributional critic function in [15] achieves worst performance than non-distributional critics. We hypothesize that this is because its specialized type of goal-conditioned distributional Bellman backup uses a one-hot target the moment a goal is reached, so it becomes more difficult for the agent to learn how to consistently stay at the goal. Our distributional critics do not make any particular assumptions about the problem structure of goal-conditioned RL, but still achieve very strong results in this setting.

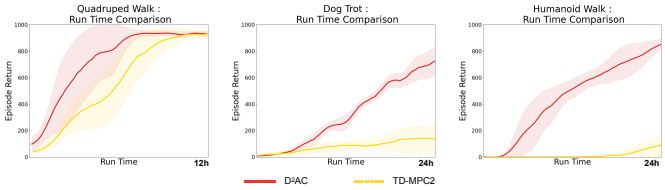

Figure 7: Wall-clock runtime comparison against TD-MPC2 on a single GPU. Thanks to being model-free algorithm without the need for dynamics unrolling and planning, D2AC is a lot faster.

6.3 Distributional Critic vs. Diffusion Actor

One natural question is the following: what is the relative importance of the diffusion actor vs the distributional critic? Are both components necessary? We test this by introducing an additional method called **D1AC**. This method implements the clipped double distributional critics proposed in our paper, but uses a standard Tanh-Gaussian actor similar to SAC. In Figure 6, we present the results of D1AC alongside D4PG and D2AC to ablate how much which each modification helps. D2AC has a high probability of improvement [2] on a variety of ablation environments considered. Note that P(D2AC > Y) is higher than 0.5 on all choices of Algorithm Y. The improvement of D1AC compared to D4PG is smaller depending on the environment; our hypothesis is that it might be important to increase the representation power of critic and actor at the same time. Furthermore, we hypothesize some environments might be more bottlenecked by the policy rather than the critic function. In addition, the fact that D1AC has much stronger performance than *Two-hot + CDQ* means that combining distributional RL with CDQ is significantly more effective than its counterpart with two-hot discrete critic, which was widely adopted in other methods [46, 19].

With respect to the diffusion actor, we are also interested in understanding the relationship between the number of diffusion steps at training vs the final model performance. In particular, we can consider both K^{train}, the number of diffusion steps used during training, and K, the number of diffusion steps used at inference. In Table 1, we see that there is some moderate improvements to the final results when training on longer diffusion horizons. Our results agree with the common practice in diffusion models [54, 31] of using more diffusion steps for training and for inference. In addition, we also find that setting $\pi_{\text{ref}} = \pi_\phi$ as done in D2AC policy loss (13) rather than just using the actions from the replay buffer $\pi_{\text{ref}} \neq \pi_\phi$ slightly improves results.

Table 1: Performance Statistics at 500K Environment Steps

Configuration	Quadruped Walk	Humanoid Walk
$K^{train} = K = 2$	778.6 ± 227.6	378.7 ± 52.5
$K^{train} = K = 5$	$\mathbf{941.5 \pm 9.7}$	$\mathbf{425.1 \pm 39.9}$
$K^{train} = 5, K = 2$	928.8 ± 15.9	386.9 ± 59.8
$K^{train} = K = 5$ where $\pi_{\text{ref}} \neq \pi_\phi$	829.9 ± 127.5	314.2 ± 76.5

6.4 Limitation of Our Method

Fundamentally, our method is still a model-free RL method; even a more expressive policy class based on denoising score matching cannot fully substitute for the role of planning. Although the denoising diffusion policy improvement we propose already has a lot of similarity to MPPI [68, 70] in terms of optimization objective, MPPI is explicitly guided by the critic function during the sampling process, and simultaneously considers multiple possible trajectories. We expect that explicit value-based planning should be able to further enhance diffusion performance, but leave it for future work.

7 Conclusion

We have presented D2AC, a model free RL algorithm that takes inspiration from recent advances in model-based RL (MBRL). By making careful use of a distributional critic and a denoising actor, we are able to capture two salient properties from MBRL algorithms: the ability to optimize over a distribution of possible returns, and the ability to iteratively refine our action selection. Several careful design choices were required to make this algorithm work at its full potential, most notably

using a double-Q learning trick with distributional RL and careful formulation of the actor's denoising loss function. The resulting algorithm, D2AC, achieves state of the art model-free performance on a variety of sparse- and dense-reward RL environments, and is significantly faster to run than model-based methods.

There exist a few exciting avenues for future work. D2AC can be plugged into any model-based methods as the model-free RL component; we hope to see how simple planning modules can boost its performance. Our method also did not consider exploration in a direct manner; for some environments, the bottleneck might be exploration. Finally, we are interested in further improving the computational efficiency of D2AC by investigating faster diffusion models [45, 56, 55] and beyond.

8 Funding Acknowledgment

Financial support was provided by National Institute for Mathematics and Theory in Biology (Simons Foundations award MP-TMPS-00005320 and National Science Foundation award DMS-2235451).

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

[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] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 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] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

[8] Yuhui Chen, Haoran Li, and Dongbin Zhao. Boosting continuous control with consistency policy. *arXiv preprint arXiv:2310.06343*, 2023.

[9] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[10] Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards. *arXiv preprint arXiv:2309.17400*, 2023.

[11] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pages 1096–1105. PMLR, 2018.

[12] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134:19–67, 2005.

[13] Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. *arXiv preprint arXiv:2309.16984*, 2023.

[14] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

[15] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[16] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[17] 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.

[18] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[19] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[20] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

[21] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

[22] 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.

[23] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in neural information processing systems*, 28, 2015.

[24] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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

[26] Zhang-Wei Hong, Ge Yang, and Pulkit Agrawal. Bilinear value networks. *arXiv preprint arXiv:2204.13695*, 2022.

[27] Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces. In *International Conference on Machine Learning*, pages 4476–4486. PMLR, 2021.

[28] Ehsan Imani and Martha White. Improving regression performance with distributional losses. In *International conference on machine learning*, pages 2157–2166. PMLR, 2018.

[29] 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.

[30] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[31] 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.

[32] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

[33] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

[35] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[37] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[39] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[40] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

[41] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007.

[42] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[43] Anil V Rao. A survey of numerical methods for optimal control. *Advances in the astronautical Sciences*, 135(1):497–528, 2009.

[44] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[45] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[46] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[47] John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

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

[49] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[50] Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pages 30365–30380. PMLR, 2023.

[51] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

[52] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[53] 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.

[54] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[55] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

[56] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[57] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[58] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[59] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[60] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[61] Hado Van Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[62] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[64] Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4235–4246, 2020.

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

[66] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. *arXiv preprint arXiv:2311.12908*, 2023.

[67] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

[68] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

[69] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

[70] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[71] Long Yang, Zhixiong Huang, Fenghao 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] Lunjun Zhang, Yuwen Xiong, Ze Yang, Sergio Casas, Rui Hu, and Raquel Urtasun. Learning unsupervised world models for autonomous driving via discrete diffusion. *arXiv preprint arXiv:2311.01017*, 2023.

A On Two-hot Functions

The function $h_{\mathbf{z}_q}$ is called the *two-hot* function. Mathematically, $h_{\mathbf{z}_q}(z)$ is defined as:

$$h_{\mathbf{z}_q}(z)[k] = \begin{cases} 1 & z \leq V_{\min} \text{ and } k = 0, \\ 0 & z \leq \mathbf{z}_q[k-1], \\ \frac{z - \mathbf{z}_q[k-1]}{\mathbf{z}_q[k] - \mathbf{z}_q[k-1]} & \mathbf{z}_q[k-1] \leq z \leq \mathbf{z}_q[k], \\ \frac{\mathbf{z}_q[k+1] - z}{\mathbf{z}_q[k+1] - \mathbf{z}_q[k]} & \mathbf{z}_q[k] \leq z \leq \mathbf{z}_q[k+1], \\ 0 & z \geq \mathbf{z}_q[k+1], \\ 1 & z \geq V_{\max} \text{ and } k = N. \end{cases}$$

Which is a piece-wise linear function with the following properties. $\forall z$, if $V_{\min} \leq z \leq V_{\max}$,

$$\sum_{j=0}^{N} h_{\mathbf{z}_q}(z)[j] = 1 \quad \sum_{j=0}^{N} h_{\mathbf{z}_q}(z)[j] \cdot \mathbf{z}_q[j] = z \tag{14}$$

Which means that for any z value, the two-hot function will distribute probability mass in the $[V_{\min}, V_{\max}]$ region that sums up to 1, with a weighted mean equal to z itself. Note that the two-hot function is not the only function that satisfies those properties; for instance, the HL-Gauss parameterization proposed in [28] can also satisfy those two conditions.

Lately, a different type of discrete critic based on *two-hot* representation has gained popularly from the success of MuZero [46], Dreamer-v3 [19], and TD-MPC2 [20]. To draw the connection between the two types of discrete critics, we first notice the following:

$$\sum_{j=0}^{N} \mathbf{p}[j] = 1 \qquad r + \gamma \sum_{j=0}^{N} \mathbf{z}_q[j] \cdot \mathbf{p}[j] = \sum_{j=0}^{N} (r + \gamma \mathbf{z}_q)[j] \cdot \mathbf{p}[j]$$

It follows that: two-hot discrete critic learning can be reformulated as **simply swapping the order between the sum and the two-hot function in the projection operator of distributional RL**:

$$\mathbf{\Phi}_{\text{dist}}(\mathbf{z}_p, \mathbf{p}, \mathbf{z}_q)[k] = \sum_{j=0}^{N} h_{\mathbf{z}_q}(\mathbf{z}_p[j])[k] \cdot \mathbf{p}[j]$$

$$\mathbf{\Phi}_{\text{twohot}}(\mathbf{z}_p, \mathbf{p}, \mathbf{z}_q)[k] = h_{\mathbf{z}_q}\left(\sum_{j=0}^{N} \mathbf{z}_p[j] \cdot \mathbf{p}[j] \right)[k]$$

And the two-hot label is:

$$\mathbf{a}' \sim \pi_\phi(\cdot \mid \mathbf{s}') \quad \boldsymbol{\ell}_{\text{twohot}} = \boldsymbol{\Phi}_{\text{twohot}}(r + \gamma \mathbf{z}_q, \mathbf{q}_{\bar{\theta}}(\mathbf{s}', \mathbf{a}'), \mathbf{z}_q)$$
$$L(\theta) = -\boldsymbol{\ell}_{\text{twohot}}^\top \log \mathbf{q}_\theta(\mathbf{s}, \mathbf{a})$$

In comparison, distributional RL can be seen as using soft labels rather than hard labels; it allows the uncertainty about future returns to propagate through Q-learning. Two-hot discrete critic learning does not allow such uncertainty to propagate between different timesteps through Bellman backup.

B Policy Gradients for Variance-Exploding (VE) Diffusion

Let one step of the diffusion process $\mathbf{a}^{(k)} \to \mathbf{a}^{(k-1)}$ for policy π_ϕ be:

$$\pi_\phi(\mathbf{a}^{(k-1)} \mid \mathbf{a}^{(k)}, \mathbf{s}) = \mathcal{N}(\mathbf{a}^{(k-1)} \mid \boldsymbol{\mu}_\phi(\mathbf{a}^{(k)}, k; \mathbf{s}), (\sigma_k^2 - \sigma_{k-1}^2)\mathbf{I})$$
$$\boldsymbol{\mu}_\phi(\mathbf{a}^{(k)}, k; \mathbf{s}) = \mathbf{a}^{(k)} + (\sigma_k^2 - \sigma_{k-1}^2)\nabla_{\mathbf{a}} \log p(\mathbf{a}^{(k)} \mid \sigma_k, \mathbf{s})$$
$$\nabla_{\mathbf{a}} \log p(\mathbf{a} \mid \sigma, \mathbf{s}) = (D_\phi(\mathbf{a}, \sigma; \mathbf{s}) - \mathbf{a})/\sigma^2$$

where we can see that

$$\begin{aligned}\boldsymbol{\mu}_\phi(\mathbf{a}^{(k)}, k; \mathbf{s}) &= \mathbf{a}^{(k)} + \frac{(\sigma_k^2 - \sigma_{k-1}^2)}{\sigma_k^2}\Big(D_\phi(\mathbf{a}^{(k)}, \sigma_k; \mathbf{s}) - \mathbf{a}^{(k)}\Big) \\ &= \frac{\sigma_{k-1}^2}{\sigma_k^2}\mathbf{a}^{(k)} + \frac{(\sigma_k^2 - \sigma_{k-1}^2)}{\sigma_k^2} D_\phi(\mathbf{a}^{(k)}, \sigma_k; \mathbf{s})\end{aligned}$$

In addition, let the latent variables $\mathbf{a}^{(1)} \cdots \mathbf{a}^{(K)}$ be:

$$q(\mathbf{a}^{(k)} \mid \mathbf{a}) = \mathcal{N}(\mathbf{a}^{(k)} \mid \mathbf{a}, \sigma_k^2 \mathbf{I})$$

Then we have the marginal distribution denoted as:

$$p(\mathbf{a}^{(k)}) = \int p_{\text{data}}(\mathbf{a}) q(\mathbf{a}^{(k)} \mid \mathbf{a}) \mathrm{d}\mathbf{a}$$

Under the assumption that $\sigma_{\max}$ is sufficiently large, the initial noise distribution is pre-defined as:

$$p(\mathbf{x}^{(K)}) = \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$$

The full policy distribution is:

$$\pi_\phi(\mathbf{a} \mid \mathbf{s}) = \int p(\mathbf{a}^{(K)}) \prod_{k=1}^{K} \pi_\phi(\mathbf{a}^{(k-1)} \mid \mathbf{s}, \mathbf{a}^{(k)}) \mathrm{d}\mathbf{a}_{1:K}$$

The distribution of $\mathbf{a}^{(k)}$ given $\mathbf{a}^{(i)}$, for $k > i$, is given by:

$$q(\mathbf{a}^{(k)} \mid \mathbf{a}^{(i)}) = \mathcal{N}(\cdot \mid \mathbf{a}^{(i)}, (\sigma_k^2 - \sigma_i^2)\mathbf{I})$$

Since the forward process is Markov, we have

$$q(\mathbf{a}^{(k)}, \mathbf{a}^{(i)} \mid \mathbf{a}) = q(\mathbf{a}^{(k)} \mid \mathbf{a}) \cdot q(\mathbf{a}^{(i)} \mid \mathbf{a})$$

The posterior distribution can be written as:

$$q(\mathbf{a}^{(i)} \mid \mathbf{a}^{(k)}, \mathbf{a})$$

Which can be computed in closed-form using the conjugate prior of Gaussian distributions:

$$q(\mathbf{a}^{(i)} \mid \mathbf{a}^{(k)}, \mathbf{a}) = \mathcal{N}\left(\cdot \,\middle|\, \frac{\sigma_i^2}{\sigma_k^2}\mathbf{a}^{(k)} + \frac{(\sigma_k^2 - \sigma_i^2)}{\sigma_k^2}\mathbf{a}, \sigma_i^2 \frac{(\sigma_k^2 - \sigma_i^2)}{\sigma_k^2} \right)$$

Then Jensen's inequality gives us the following lower bound [53, 25]:

$$-\log \pi_\phi(\mathbf{a} \mid \mathbf{s}) \le L_{\text{prior}} + \sum_{k=1}^{K} \mathbb{E}_{q(\mathbf{a}^{(k)} \mid \mathbf{a})} D_{KL}\Big[q(\mathbf{a}^{(k-1)} \mid \mathbf{a}^{(k)}, \mathbf{a}) \parallel \pi_\phi(\mathbf{a}^{(k-1)} \mid \mathbf{a}^{(k)}, \mathbf{s})\Big]$$
$$L_{\text{prior}} = D_{KL}(q(\mathbf{a}^{(K)} \mid \mathbf{a}) \parallel p(\mathbf{a}^{(K)}))$$

From the result of [32], we know that

$$D_{KL}(q(\mathbf{a}^{(k-1)}|\mathbf{a}^{(k)},\mathbf{a}) \parallel \pi_\phi(\mathbf{a}^{(k-1)}|\mathbf{a}^{(k)},\mathbf{s})) = \frac{1}{2\sigma_{k-1}^2}\frac{\sigma_k^2-\sigma_{k-1}^2}{\sigma_k^2}\|\mathbf{a}-D_\phi(\mathbf{a}^{(k)},\sigma_k;\mathbf{s})\|^2$$
$$= \frac{1}{2}\left(\frac{1}{\sigma_{k-1}^2}-\frac{1}{\sigma_k^2}\right)\|\mathbf{a}-D_\phi(\mathbf{a}^{(k)},\sigma_k;\mathbf{s})\|^2$$

Allowing us to arrive at the following result:

$$\log \pi_\phi(\mathbf{a}|\mathbf{s}) \exp Q(\mathbf{s},\mathbf{a}) \geq -\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}),k\sim\text{Unif}(1\cdots K)}[\lambda_{\text{pg}}(k)\|\mathbf{a}-D_\phi(\mathbf{a}+\sigma_k\boldsymbol{\epsilon},\sigma_k;\mathbf{s})\|^2 \exp Q(\mathbf{s},\mathbf{a})]$$

where the weightings for policy gradient objective λ_{pg} are:

$$\lambda_{\text{pg}}(k) = (1/\sigma_{k-1}^2 - 1/\sigma_k^2)\cdot(K/2)$$

C Proof of Proposition 1

Proposition 1. *Define the data distribution under diffusion step $k \geq 0$ to be (with $\hat{\sigma} \geq \sigma_{min}$):*

$$p_k(\mathbf{a}|\mathbf{s}) = \int \pi(\mathbf{a}^{(0)}|\mathbf{s})\mathcal{N}(\mathbf{a}|\mathbf{a}^{(0)},\sigma_k^2\mathbf{I})\mathrm{d}\mathbf{a}^{(0)} \tag{15}$$

$$\hat{p}_k(\mathbf{a}|\mathbf{s}) = \int p_{k+1}(\mathbf{u}|\mathbf{s})\mathcal{N}(\mathbf{a}|D(\mathbf{u},\sigma_{k+1};\mathbf{s}),\hat{\sigma}^2\mathbf{I})\mathrm{d}\mathbf{u} \tag{16}$$

Under the assumptions that (i) entropy of p_k and $\hat{p}_k$ strictly increases with k; (ii) the KL distance from p_k and $\hat{p}_k$ to optimal policy p^* strictly decreases with k; (iii) $\hat{\sigma}$ is sufficiently large such that $D_{KL}(\hat{p}_0 \parallel p^*) \geq D_{KL}(p_0 \parallel p^*)$. Then for any state $\mathbf{s}$ and diffusion step k, we have

$$\mathbb{E}_{p_0}[Q] \geq \mathbb{E}_{\hat{p}_k}[Q]$$

Proof.

D Diffusion Value Gradients

E MPPI

Model Predictive Path Integral Control (MPPI) applies an iterative process of sampling and return-weighted refinement [68, 70, 69, 39, 21]. Initial trajectories are sampled from a noise distribution:

$$\mathbf{x}_{i=0}^{k=0\cdots K-1} \sim \mathcal{N}(\cdot \mid \boldsymbol{\mu}_{i=0},\boldsymbol{\sigma}_{i=0}^2)$$
$$\boldsymbol{\mu}_{i=0} = \mathbf{0}$$
$$\boldsymbol{\sigma}_{i=0}^2 = \sigma_{\text{max}}^2\mathbf{I}$$

At the i-th iteration, MPPI samples K action sequences $\mathbf{x}_i^{k=0\cdots K-1}$ from

$$\mathcal{N}(\cdot \mid \boldsymbol{\mu}_i,\boldsymbol{\sigma}_i^2)$$

uses stochastic (virtual) rollouts to estimate the K empirical returns of each action sequences $\{R\}_{k=0}^{K-1}$. It then updates the sampling distribution according to

$$\boldsymbol{\mu}_{i+1} = \left(\sum_{k=0}^{K-1}\exp(R_k/\lambda)\mathbf{x}_i^k\right)\Big/\left(\sum_{k=0}^{K-1}\exp(R_k/\lambda)\right)$$
$$\boldsymbol{\sigma}_{i+1}^2 = \left(\sum_{k=0}^{K-1}\exp(R_k/\lambda)(\mathbf{x}_i^k-\boldsymbol{\mu}_{i+1})^2\right)\Big/\left(\sum_{k=0}^{K-1}\exp(R_k/\lambda)\right)$$

MPPI uses a distribution of returns to update the action proposals to higher-return regions.

Algorithm 1 Policy Optimization (*under Tanh Action Squashing*)

procedure UPDATE($\phi, \alpha \mid \mathbf{s}, \mathbf{a}, \sigma, Q$)
Input: policy parameters ϕ, entropy coefficient α.
Input: state $\mathbf{s}$, continuous action $\mathbf{a} \in (-\mathbf{1}, \mathbf{1})$.
Input: differentiable $Q(\mathbf{s}, \mathbf{a})$, noise level σ.
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \quad \tilde{\mathbf{u}} = \tanh^{-1}(\mathbf{a}) + \boldsymbol{\epsilon} \cdot \sigma$
$\hat{\boldsymbol{\mu}}_\phi = \boldsymbol{\mu}_\phi(\mathbf{s}, \tilde{\mathbf{u}}, \sigma), \quad \hat{\boldsymbol{\sigma}}_\phi = \boldsymbol{\sigma}_\phi(\mathbf{s}, \tilde{\mathbf{u}}, \sigma)$
$\mathbf{x}_\phi \sim \mathcal{N}(\cdot \mid \hat{\boldsymbol{\mu}}_\phi, \hat{\boldsymbol{\sigma}}^2_\phi)$
$\mathbf{x}^{\text{target}} = \mathbf{x} + \nabla_{\mathbf{x}} Q(\mathbf{s}, \tanh(\mathbf{x}))\big|_{\mathbf{x}=\mathbf{x}_\phi}$ ▷ Eq (13)
$\hat{f}^{\text{ent}}_\phi = f^{\text{ent}}(\mathbf{x}_\phi, \hat{\boldsymbol{\mu}}_\phi, \hat{\boldsymbol{\sigma}}_\phi)$ ▷ Eq (18)
$\phi \leftarrow \text{SGD}(\phi, \nabla_\phi[\|\mathbf{x}_\phi - \mathbf{x}^{\text{target}}\|^2 + \alpha \hat{f}^{\text{ent}}_\phi])$
$\alpha \leftarrow \text{SGD}(\alpha, \nabla_\alpha[-\alpha(\hat{f}^{\text{ent}}_\phi - \lambda_{\text{ent}}|\mathcal{A}|)])$ ▷ Update temperature
end procedure

F Entropy Regularization of Diffusion Policy

G On Tanh Squashing of Actions

The most commonly used action distribution for continuous control is a Gaussian distribution with tanh squashing [17]. The policy π_ϕ outputs the mean $\boldsymbol{\mu}_\phi$ and log sigma, which we exponentiate to get $\boldsymbol{\sigma}_\phi$.

$$\mathbf{u} \sim \mathcal{N}(\cdot \mid \boldsymbol{\mu}_\phi(\mathbf{s}), \boldsymbol{\sigma}^2_\phi(\mathbf{s})) \quad \mathbf{a} = \tanh(\mathbf{u}) \tag{17}$$

where $\mathbf{u}$ is sampled using the reparameterization trick [33]: $\mathbf{u} = \boldsymbol{\mu}_\phi + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}_\phi, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.

Applying the change of variable formula, The change of variable formula says

$$\log \pi(\mathbf{a} \mid \mathbf{s}) = \log \mathcal{N}(\mathbf{u} \mid \boldsymbol{\mu}_\phi, \boldsymbol{\sigma}^2_\phi) - \sum_i \log(1 - \tanh^2(\mathbf{u}_i))$$

And the second part can be written as:

$$\begin{aligned}
\log(1 - \tanh^2(\mathbf{u}))) &= 2 \log \operatorname{sech} \mathbf{u} \\
&= 2 \log 2 - 2 \log(e^{\mathbf{u}} + e^{-\mathbf{u}}) \\
&= 2 \log 2 - 2 \log(e^{\mathbf{u}}(1 + e^{-2\mathbf{u}})) \\
&= 2 \log 2 - 2\mathbf{u} - 2 \log(1 + e^{-2\mathbf{u}}) \\
&= 2 \log 2 - 2\mathbf{u} - 2 \operatorname{softplus}(-2\mathbf{u}) \\
&= 2[\log 2 - \mathbf{u} - \operatorname{softplus}(-2\mathbf{u})]
\end{aligned}$$

Thus, we use an easy-to-compute and numerically stable form of the log likelihood of Tanh Gaussian distribution:

$$f^{\text{ent}}(\mathbf{u}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \log \mathcal{N}(\mathbf{u} \mid \boldsymbol{\mu}, \boldsymbol{\sigma}^2) - 2 \cdot \mathbf{1}^\top [\log 2 - \mathbf{u} - \operatorname{softplus}(-2\mathbf{u})] \tag{18}$$

Thus, we can use the squashing to enforce the action boundary, and apply entropy regularization [17]:

$$L(\alpha) = -\alpha[f^{\text{ent}}_\phi - \lambda_{\text{ent}}|\mathcal{A}|]$$

Which we apply to policy network similar to other model-free baselines.

H Policy Network Parameterization

Let $p_\sigma(\tilde{\mathbf{u}}) = \int p_{\text{data}}(\mathbf{u}) \mathcal{N}(\tilde{\mathbf{u}} \mid \mathbf{u}, \sigma^2 \mathbf{I}) \mathrm{d}\mathbf{u}$. The min $\sigma_{\min}$ and max $\sigma_{\max}$ of noise levels are selected such that $p_{\sigma_{\min}}(\tilde{\mathbf{u}}) \approx p_{\text{data}}(\mathbf{u})$ and $p_{\sigma_{\max}}(\tilde{\mathbf{u}}) \approx \mathcal{N}(\mathbf{0}, \sigma^2_{\max} \mathbf{I})$. We adopt the EDM denoiser parameterization [31]:

$$\begin{aligned}
\boldsymbol{\mu}_\phi(\mathbf{s}, \mathbf{u}, \sigma) = c_{\text{skip}}(\sigma)\mathbf{u} \\
+ c_{\text{out}}(\sigma) F_\phi(\mathbf{s}, c_{\text{in}}(\sigma)\mathbf{u}, c_{\text{noise}}(\sigma))
\end{aligned} \tag{19}$$

Algorithm 2 : **D2 Actor Critic**

procedure ACTORCRITIC($\mathcal{D}, \theta, \phi, \alpha$)
Input: $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r) \sim \mathcal{D}$
Input: critic parameters θ, policy parameters ϕ.
Input: entropy coefficient α.
$\mathbf{a}' \sim \pi_\phi(\cdot \mid \mathbf{s}')$ ▷ Diffusion Sampling
Optimize θ with clipped double distributional RL (5)
$i \sim \text{Unif}\{1 \cdots K^{\text{train}}\}, \quad j \sim \text{Unif}\{1 \cdots K\}$
$\sigma_i \leftarrow \sigma(\frac{i-1}{K^{\text{train}}-1}), \quad \sigma_j \leftarrow \sigma(\frac{j-1}{K-1})$ ▷ Select noise levels in EDM
Update ϕ, α on $(\mathbf{s}, \mathbf{a}, \sigma_i)$ and $(\mathbf{s}', \mathbf{a}', \sigma_j)$. ▷ Alg 1
Target network update: $\overline{\theta} \leftarrow \tau\overline{\theta} + (1-\tau)\theta$ ▷ Exponential moving average.
end procedure

With the specific functions being:

$$
\begin{aligned}
c_{\text{skip}}(\sigma) &= \sigma_{\text{data}}^2/(\sigma^2 + \sigma_{\text{data}}^2) \\
c_{\text{out}}(\sigma) &= \sigma \cdot \sigma_{\text{data}}/\sqrt{\sigma^2 + \sigma_{\text{data}}^2} \\
c_{\text{in}}(\sigma) &= 1/\sqrt{\sigma^2 + \sigma_{\text{data}}^2} \\
c_{\text{noise}}(\sigma) &= \log \sigma
\end{aligned}
$$

This parameterization offers powerful inductive bias for an iterative denoising process that has been shown to work well. $\boldsymbol{\sigma}_\phi(\mathbf{s}, \mathbf{a}, \sigma)$ uses the same network F_ϕ except for the last linear layer. The conditioning of $c_{\text{noise}}(\sigma)$ is done via positional embedding [63].

A sequence of noise levels is used $\sigma_{\text{min}} = \sigma_1 < \sigma_2 < \cdots < \sigma_M = \sigma_{\text{max}}$. We follow EDM in setting $\sigma_i = \sigma(\frac{i-1}{M-1})$ for $i = \{1, \cdots, M\}$ and $\rho = 7$:

$$
\sigma(\eta) = \left(\sigma_{\text{min}}^{1/\rho} + \eta\left(\sigma_{\text{max}}^{1/\rho} - \sigma_{\text{min}}^{1/\rho}\right)\right)^\rho \tag{20}
$$

I Hyperparameters

Table 2: Hyperparameters for D2AC

Parameters	Value
Batch size	256
Optimizer	AdamW [36]
Learning rate for policy	0.001
Learning rate for critic	0.001
Learning rate for temperature α	0.0001
α initialization	0.2
Weight Decay	0.0001
Number of hidden layers (all networks)	2
Number of hidden units per layer	256
Non-linearity	Layer Normalization [4] + ReLU
γ	0.99
λ_{ent}	0.0
Polyak for target network	0.995
Target network update interval	1 for dense rewards, 10 for sparse rewards
Ratio between env vs optimization steps	1 for dense rewards, 2 for sparse rewards
Initial random trajectories	200
Number of parallel workers	4 for dense rewards, 20 for sparse rewards
Update every number of steps in environment	same as Number of parallel workers
Replay buffer size	10^6 for dense rewards, 2.5×10^6 for sparse rewards
$[V^{\text{min}}, V^{\text{max}}]$	$[-1000, 1000]$ for DM control $[-50, 0]$ for Multi-Goal RL
Number of bins (size of the support $\mathbf{z}_q$)	201 for DM control, 101 for Multi-Goal RL
σ_{min}	0.05
σ_{max}	2.0
σ_{data}	1.0
ρ	7
M	2
M^{train}	5
Noise-level conditioning	Position encoding on $(10^3 \log \sigma)/4$
Embedding size for noise-level conditioning	32