GenPO: Generative Diffusion Models Meet On-Policy Reinforcement Learning

Shutong Ding[1,5,*] **Ke Hu**[1,*] **Shan Zhong**[2] **Haoyang Luo**[1] **Weinan Zhang**[3]
Jingya Wang[1,5] **Jun Wang**[4] **Ye Shi**[1,5,†]

[1]ShanghaiTech University [2]University of Electronic Science and Technology of China
[3]Shanghai Jiao Tong University [4]University College London
[5]MoE Key Laboratory of Intelligent Perception and Human Machine Collaboration

{dingsht, huke2024, luohy12024}@shanghaitech.edu.cn
202211040927@std.uestc.edu.cn
wnzhang@sjtu.edu.cn
jun.wang@cs.ucl.ac.uk
{wangjingya, shiye}@shanghaitech.edu.cn

Abstract

Recent advances in reinforcement learning (RL) have demonstrated the powerful exploration capabilities and multimodality of generative diffusion-based policies. While substantial progress has been made in offline RL and off-policy RL settings, integrating diffusion policies into on-policy frameworks like PPO remains underexplored. This gap is particularly significant given the widespread use of large-scale parallel GPU-accelerated simulators, such as IsaacLab, which are optimized for on-policy RL algorithms and enable rapid training of complex robotic tasks. A key challenge lies in computing state-action log-likelihoods under diffusion policies, which is straightforward for Gaussian policies but intractable for flow-based models due to irreversible forward-reverse processes and discretization errors (e.g., Euler-Maruyama approximations). To bridge this gap, we propose GenPO, a generative policy optimization framework that leverages exact diffusion inversion to construct invertible action mappings. GenPO introduces a novel doubled dummy action mechanism that enables invertibility via alternating updates, resolving log-likelihood computation barriers. Furthermore, we also use the action log-likelihood for unbiased entropy and KL divergence estimation, enabling KL-adaptive learning rates and entropy regularization in on-policy updates. Extensive experiments on eight IsaacLab benchmarks, including legged locomotion (Ant, Humanoid, Anymal-D, Unitree H1, Go2), dexterous manipulation (Shadow Hand), aerial control (Quadcopter), and robotic arm tasks (Franka), demonstrate GenPO's superiority over existing RL baselines. Notably, GenPO is the first method to successfully integrate diffusion policies into on-policy RL, unlocking their potential for large-scale parallelized training and real-world robotic deployment. The official implementation of GenPO is provided in `https://github.com/wadx2019/genpo/`.

1 Introduction

In recent years, generative diffusion models, such as DDPM [25], DDIM [52], and flow matching [36, 35], have attracted considerable attention from reinforcement learning researchers due

[*]Equal contribution. †Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

 https://doi.org/10.52202/085713-4347

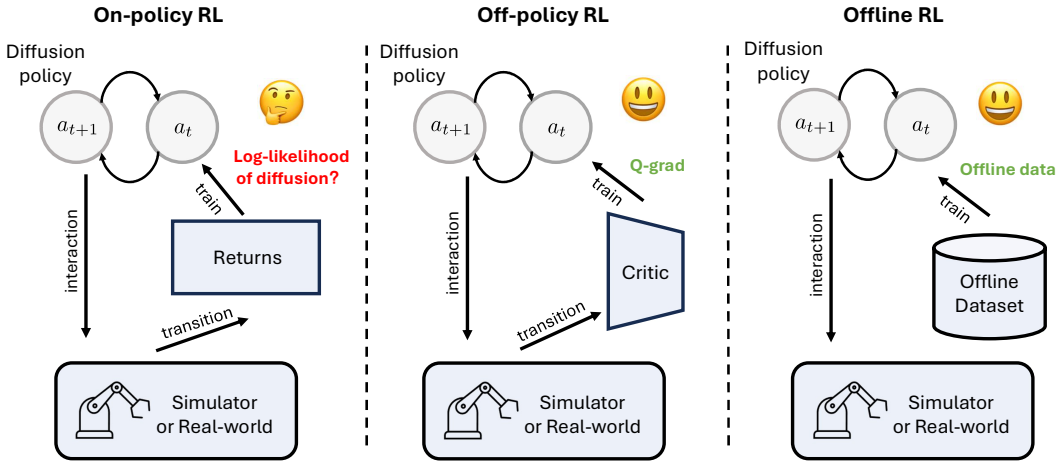

Figure 1: Existing diffusion-based reinforcement learning algorithms mainly focus on the off-policy (middle) and offline (right) RL. This is because we can generally obtain the gradient of the Q function to update the diffusion policy in off-policy RL and utilize the offline data to train the agent in offline RL. However, as for diffusion-based RL in the on-policy (left) algorithm, there still exists a challenge that we cannot obtain the log-likelihood of diffusion.

to their powerful exploration capabilities and multimodality compared with traditional parameterized Gaussian policies. However, existing works in this field almost merely focus on offline RL [60, 1, 27, 7, 8, 24, 67, 40] and off-policy RL [62, 13, 43, 41, 59] methods, and the integration of diffusion or flow-based policies into on-policy reinforcement learning like PPO [51] remains largely unexplored. However, existing massively parallel GPU-accelerated simulators, such as Isaac Gym [38] and its successor, IsaacLab, mainly [32] benefit the training of on-policy reinforcement learning algorithms, especially PPO, which are capable of achieving substantial performance in a relatively short wall-clock time. This has resulted in a fundamental dilemma, hindering the practical deployment of existing diffusion/flow-based RL algorithms in real-world applications [10, 31, 42, 45, 14].

Different from diffusion models [13] applied in off-policy RL, the primary challenge in integrating flow policies with on-policy reinforcement learning is how to compute the log-likelihood of state-action pairs, while it is straightforward for Gaussian policies. The underlying reason here lies in the inconsistency and non-reversibility between the forward and reverse processes of the generative diffusion policy, caused by the Euler-Maruyama (EM) discretization [48, 5]. Nevertheless, the optimization objective in on-policy RL necessitates the probability density of the given state-action pair. In this context, estimating the probability density of a given state-action pair in flow or diffusion models is nontrivial, and further utilizing this estimate for diffusion or flow policy improvement in the on-policy RL paradigm presents additional challenges.

To address these challenges, we propose GenPO, motivated by the exact diffusion inversion [57], which constructs an invertible flow mapping and thus avoids the mismatch of forward and reverse processes of the diffusion policy. Specifically, we first construct a new Markov process with the action space twice the size of the original in GenPO due to the limitation of exact diffusion inversion, and then enable reversible flow processes by alternately updating the two parts of the dummy action. In that case, we can calculate the exact probability density of the given action in this new Markov decision problem via change of variables theory like normalizing flow [46, 15]. Moreover, the two parts of the dummy action will be averaged as one and then mapped to the original action space during inference. Hence, we can perform GenPO to update the generative diffusion policy in the reformulated Markov decision problem, but finally obtain the optimal solution to the original task.

In addition, with the exact probability density, we estimate [11] the entropy of the diffusion policy and the Kullback-Leibler (KL) divergence with respect to the behavior diffusion policy. This allows for the combination of entropy regularization and KL-adaptive learning rate adjustment into GenPO, which has been confirmed effective in PPO. To demonstrate the superiority of GenPO, we conduct experiments on 8 IsaacLab benchmarks [38], which cover robot ant, humanoid, quadcopter, Franka

https://doi.org/10.52202/085713-4347

robot arm [23], Shadow dexterous hand [56], ANYbotics anymal-D, and Unitree legged robots [68]. The final results show that the proposed GenPO outperforms previous RL baselines in cumulative returns, while maintaining comparable sample efficiency and faster convergence. It is worth noting that GenPO is the first trial to incorporate the generative diffusion policy into on-policy RL algorithms and opens brand-new avenues for diffusion-type policies applied in large-scale parallel simulators and real-world robot control. Our contributions are summarized as follows:

- **Bridging generative diffusion models and on-policy RL.** GenPO achieves the first stable integration of diffusion-based policies with on-policy reinforcement learning, successfully deploying the inherent exploration capabilities and multimodality of diffusion models in large-scale GPU parallel simulators (IsaacLab).
- **Closing the likelihood gap in diffusion policies.** Unlike Gaussian policies with closed-form densities, diffusion policies lack tractable likelihoods. GenPO overcomes this by using invertible diffusion dynamics, enabling 1) exact log-likelihood computation, 2) unbiased entropy estimation, and 3) analytical KL divergence-bringing Gaussian-style advantages to expressive diffusion models.
- **Comprehensive experiments on IsaacLab benchmarks.** We evaluated GenPO on IsaacLab benchmarks. The experimental results show that in a massively parallel environment, previous diffusion-based reinforcement learning algorithms are almost ineffective, while GenPO achieves the best performance in terms of sample utilization efficiency and episodic rewards, far surpassing other algorithms.

2 Related Works

In this section, we review the literature of generative models such as VAE [28], GAN [12], normalizing flow [46], diffusion [65], and flow-based model [36, 34] for policy learning, and generally divide them into two classes according to their learning paradigm.

Generative Policy for Online Reinforcement Learning. The goal of online RL [54] is to learn an optimal policy with the interactions of the given environment. It is challenging to train a generative model policy within an online RL paradigm due to the absence of the action label. Generative policy learning methods in online RL can be broadly grouped into two paradigms. The first paradigm views the generative policies as a black-box function approximation like neural networks, and optimizes them via the policy gradient. In this paradigm, representative methods include DACER [59], CPQL [9], FlowPG [4], SAC-NF [39], and TRPO-NF [55], which directly apply deterministic or policy gradient on the generative model policies. By contrast, the second category leverages the internal structure and mechanisms of the generative model itself to perform the policy improvement. For instance, DIPO [62] and QVPO [13] respectively utilize the gradient and the value magnitude of the Q-function to identify optimal actions, which are then applied to the variational loss of the diffusion model. Additionally, QSM [43] and MaxEntDP [16] regard the noise network in the diffusion model as the score function of the target policy distribution and then employ the gradient of the Q-function and its variants to train the noise network. As to normalizing flow-based approaches, MEow [6] exploits the layer-wise nonlinear property of the flow network to train the Q and V functions, which yields the corresponding maximum-entropy normalizing flow policy based on SAC [21, 22]. Overall, current approaches that employ generative-model policies in online RL have predominantly focused on integrating these models with off-policy RL algorithms like SAC, with few efforts to combine them with on-policy RL methods.

Generative Policy for Other Learning Paradigms. Compared with generative policy for online RL, more existing research works on generative policy have been devoted to offline RL [30] and imitation learning paradigms, wherein policies are learned from offline datasets without environment interaction. For offline RL, BCQ [18], PLAS [66], SPOT [61] adopt VAE [28] policy regularizer to prevent the policy from optimizing outside the support of the dataset. Besides, CPED [64] utilizes flow GAN [20] to achieve more accurate policy regularization. Furthermore, diffusion-QL [60], EDP [27], SRDP [1], CEP [37], and FQL [41] leverage diffusion [25] and flow [36, 34] policies to accurately model the policy distribution in offline datasets, thereby obtaining policies with superior generalization capability. Regarding imitation learning, diffusion policy [10] and diffuser [26] serve as canonical examples of using diffusion models to fit the state-action mapping and state-trajectory mapping, respectively. The latter one is also referred to as the diffusion planner. Subsequently, FMIL [47], AVDC [29], and decision diffuser [2] follow their steps and demonstrate enhanced

 https://doi.org/10.52202/085713-4347

performance in different robots. Furthermore, DPPO [44] applies policy gradient to fine-tune the diffusion policy trained with the offline dataset. Nonetheless, DPPO merely explores the use of the RL technique in finetuning an offline-pretrained diffusion policy, rather than representing a diffusion-based online RL algorithm.

However, all of the above generative policy learning approaches have not been effectively integrated with on-policy reinforcement learning algorithms, making them unable to sufficiently leverage modern massively parallel GPU-accelerated simulators. Specifically, off-policy algorithms often struggle with convergence in such simulators, while offline RL and imitation learning methods are fundamentally incompatible with them due to the lack of online interaction. In contrast, our proposed Generative diffusion Policy Optimization (GenPO) seamlessly integrates with on-policy algorithms, enabling efficient training of high-performing generative policies within a short wall-clock time using large-scale GPU-parallelized simulation platforms.

3 Preliminaries

3.1 On-policy Reinforcement Learning

Reinforcement learning problems [54] are formalized as Markov decision processes (MDPs), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p(s' \mid s, a) : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ denotes the transition dynamics, $r(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho_0(s) : \mathcal{S} \to [0, \infty)$ is the distribution of the initial state, and $\gamma \in [0, 1)$ is the discount factor for the value estimation. The goal of RL is to maximize the expected discounted return as $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where $\pi_\theta(a \mid s)$ is the behavior of the agent parametrized by θ, and induces a corresponding distribution over behavior trajectories τ. For convenience, RL also defines two different value functions: the state value function $V_\pi(s) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0} \gamma^t r_t | s_0 = s]$ and the state-action value function $Q_\pi(s, a) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0} \gamma^t r_t | a_0 = a, s_0 = s]$. Besides, the advantage function, which represents the benefit of taking one specific action compared with the expected return at the given state, is defined as $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$.

On-policy RL algorithms train the policy with the transitions generated by the current policy itself, while off-policy RL algorithms use the transitions from a different policy as training samples. While on-policy RL algorithms preserve the consistency between the behavior and target policies, but often suffer from high variance and sample inefficiency. Proximal Policy Optimization (PPO) [51] addresses these challenges by introducing the clipped surrogate objective:

$$\mathcal{L}^{\text{CLIP}}(\theta) := \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}} \left[\min\big(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big) \right], \tag{1}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ and $\hat{A}_t$ is the estimation of the advantage function via GAE [50]. By leveraging the importance sampling technique and trust-region insights [49] without second-order complexity for approximation of the KL divergence constraint, PPO can achieve stable and efficient policy updates. The simplicity, robustness, and strong empirical performance on continuous-control benchmarks [38, 68] of PPO have made it the first algorithm to be adopted in many domains.

3.2 Generative Diffusion Models and Flow Matching

Generative diffusion models [25, 53, 52] are a kind of powerful latent variable generative model that can transform samples from the standard Gaussian distribution $x_T \sim \mathcal{N}(0, I)$ to the data distribution $x_0 \sim p_{data}$ via the denoising procedure with the noise network. The relationship between x_0 and x_T is defined by the forward SDE of diffusion

$$\mathrm{d}x_t = f(x_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}W_t, \tag{2}$$

where W_t the standard Wiener process (a.k.a., Brownian motion), $f(x_t, t)$ is a vector-valued function called the drift coefficient of x_t, and $g(t)$is a scalar function known as the diffusion coefficient of x_t. According to [53], the reverse-time SDE of diffusion (i.e., denoising procedure) is

$$\mathrm{d}x_t = \left[f(x_t, t) - g(t)^2 \nabla_x \log p_t(x_t)\right] \mathrm{d}t + g(t)\,\mathrm{d}\overline{W}_t, \tag{3}$$

where $\overline{W}_t$ is a standard Wiener process when time flows backwards from T to 0. In practice, DDPM [25] discretizes the SDE and approximates the forward step as $q(x_t \mid x_{t-1}) :=$

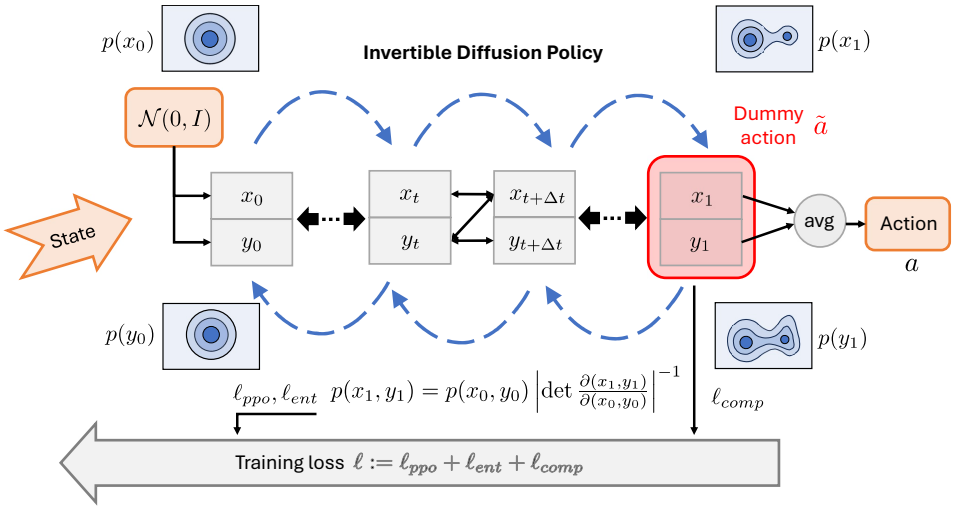

Figure 2: Forward and reverse process of GenPO. The forward process is to sample actions with the given state; the reverse process is to compute the probability density of the given state-action pair. Notably, the forward and reverse processes are invertible.

$\mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$, where β_t is the variance schedule. The noise network $\boldsymbol{\epsilon}_\theta(x_t, t)$ in DDPM can be trained by the variational lower bound loss:

$$\mathbb{E}_{t\sim[1,T],x_0,\boldsymbol{\epsilon}_t}\left[||\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)||^2\right], \tag{4}$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$, and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, I)$.

The flow matching model [36, 34] is developed for faster generation compared with classical diffusion models like DDPM, and can be viewed as a special type of generative diffusion model [19]. Similar to DDPM, flow matching model learns a time-dependent vector field $v_\theta(x, t)$, i.e., noise network in DDPM, that transports samples along a predefined family of conditional distributions $\{p_t\}_{t\in[0,1]}$ satisfying $x_1 \sim p_{\text{data}}$ and $x_0 \sim \mathcal{N}(0, I)$. Hence, the sample from the target distribution p_{data} can be achieved via solving the ode $x_1 = x_0 + \int_0^1 v(x_t, t)dt$.

Besides, The loss of flow matching is given as $\mathcal{L}(\theta) = \int_0^1 \mathbb{E}_{x\sim p_t} \left\|v_\theta(x, t) - v^*(x, t)\right\|^2 \mathrm{d}t$, where $v^*(x_t, t)$ is the true velocity field under the chosen path. When the path is Gaussian diffusion, $v^*(x, t)$ coincides with the score-drift term of the reverse SDE in diffusion. However, the log-likelihood of the given sample cannot be calculated directly in the generative diffusion framework.

4 GenPO: An On-policy RL Method based on Diffusion Models

As mentioned in Section 3.2, the log-likelihood of the given samples cannot be computed directly in generative models. However, to integrate diffusion policy into on-policy RL, we must present the explicit formula of log-likelihood under the given state-action pair for the policy update. Hence, we first elucidate the intrinsic reasons that make the computation of the log-likelihood in diffusion models challenging. Then, we address these challenges by (1) applying the exact diffusion inversion technique [57] and utilizing the change of variables technique like normalizing flow [46], (2) constructing a reformulated Markov decision process with a modified action space, along with a trick that maps the generated actions back to the original action space.

Besides, based on this framework, we also introduce an unbiased estimation method for both the entropy of the diffusion policy and the Kullback-Leibler (KL) divergence between different generative diffusion policies. Notably, this has not yet been accomplished by previous diffusion-based RL methods. In addition, since policy update is actually performed on a reformulated MDP, extraneous exploration is induced, thereby degrading RL learning efficiency. To address this issue, we also introduce an auxiliary compression loss term. Leveraging these techniques, we finally present the practical implementation of the proposed GenPO algorithm, which successfully incorporates the generative policy to PPO [51] in an effective manner. Figure 2 shows the training and inference process of the proposed GenPO.

https://doi.org/10.52202/085713-4347

4.1 Challenges in Calculating Diffusion Probability

Recalling existing generative models, such as VAE [28], GAN [12], normalizing flow [46], and diffusion model [53], it can be found that only normalizing flow and its variants like flow GAN [20] allow the exact likelihood computation via the change of variables theorem [46]. In contrast, for other generative models, probability densities or related statistics can only be obtained approximately via special designs [13, 5, 59]. Applying such approximate probability density estimates for policy update is unacceptable in on-policy RL algorithms.

According to Lemma 1, if we want to compute the probability density of a generative model exactly, the generative model must be invertible between the sampling distribution, like the standard Gaussian distribution, and the target distribution. Thus, we consider designing a reversible generative diffusion model and employing the change of variables theorem (5) to compute its probability density.

Lemma 1. ***(Change of Variables [3])*** *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ is an invertible and smooth mapping. If we have the random variable $X \sim q(x)$ and the random variable $Y = f(X)$ transformed by function f, the distribution $p(y)$ of Y is*

$$p(y) = q(x) \left| \det \frac{\partial f}{\partial x} \right|^{-1} \tag{5}$$

However, as mentioned in [57, 58, 63], it is nontrivial to realize an exact invertible diffusion model. This is because of the inconsistency between the forward and reverse processes of the generative diffusion model. (6) shows the forward and reverse process of DDIM [52]. It can be observed that this inconsistency is caused by the approximation of x_t with x_{t-1} in the forward process, since the x_t is unavailable in the forward step $t-1$.

$$\begin{aligned} \text{Reverse:} \quad & x_{t-1} = \sqrt{\alpha_{t-1}} \frac{x_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_\theta(x_t, t)}{\sqrt{\alpha_t}} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_\theta(x_t, t) \\ \text{Forward:} \quad & x_t = \frac{x_{t-1} - b_t \boldsymbol{\epsilon}_\theta(x_t, t)}{a_t} \approx \frac{x_{t-1} - b_t \boldsymbol{\epsilon}_\theta(x_{t-1}, t)}{a_t} \end{aligned} \tag{6}$$

where $a_t = \sqrt{\alpha_{t-1}/\alpha_t}, b_t = -\sqrt{\alpha_{t-1}(1-\alpha_t)/\alpha_t} + \sqrt{1-\alpha_t}$.

4.2 Exact Diffusion Inversion in Reformulated MDP

Consequently, to realize an invertible diffusion model, this issue must be addressed. Motivated by EDICT [57], we realize the invertible diffusion model via maintaining two coupled noise vectors and updating them alternately in the forward and reverse processes of the generative diffusion model, and then extend it into flow matching [35, 36] for fast generation as shown in (7, 8). However, the two coupled noise vectors also lead to a doubled sample space. This implies we cannot directly apply this technique to diffusion policy when the dimension of the action space is odd.

To resolve this problem, we reformulate the original MDP problem with a doubled dummy action space $\tilde{\mathcal{A}}$ in GenPO. Each dummy action $\tilde{a} = (x, y)$ consists of two components, which are subsequently averaged to produce a single action $a = \frac{x+y}{2}$ in the original space. It is obvious that, when the policy in the reformulated MDP is optimal, it is also optimal with the average mapping in the original MDP problem.

$$\begin{aligned} \text{Reverse:} \quad & \tilde{x}_{t+\Delta t} = x_t + v_\theta(y_t, t)\Delta t, && \tilde{y}_{t+\Delta t} = y_t + v_\theta(\tilde{x}_{t+\Delta t}, t)\Delta t \\ \text{Mixing:} \quad & x_{t+\Delta t} = p \cdot \tilde{x}_{t+\Delta t} + (1-p) \cdot \tilde{y}_{t+\Delta t}, && y_{t+\Delta t} = p \cdot \tilde{y}_{t+\Delta t} + (1-p) \cdot x_{t+\Delta t} \end{aligned} \tag{7}$$

Notably, different from EDICT, which just applies exact diffusion inversion for the inference of diffusion, GenPO independently samples the standard Gaussian noise for x_0, y_0 rather than samples one noise ϵ and sets $x_0 = y_0 = \epsilon$ to allow for the probability calculation using changes of variables (5). Moreover, due to the doubled action space in the modified MDP, there exists a meaningless exploration problem in the diffusion policy. For instance, if the optimal action $a^\star = 0$, we have $\tilde{a} = (-1, 1), (-2, 2)$ are both the optimal dummy action. Hence, to prevent unnecessary exploration in the modified MDP, a mixing scheme (7, 8) is adopted so that the two parts x, y of dummy action components remain closely aligned. In (7, 8), the mixing scheme is employed to facilitate information exchange between the x and y components of $\tilde{a}$, thereby ensuring the diffusion policy converges to

dummy actions with minimal discrepancy between x and y parts. Here the coefficient p is used to control the intensity of the interchanged information.

$$\begin{array}{lll} \text{Unmixing:} & \tilde{y}_{t+\Delta t} = \frac{y_{t+\Delta t} - (1-p)x_{t+\Delta t}}{p} & \tilde{x}_{t+\Delta t} = \frac{x_{t+\Delta t} - (1-p)\tilde{y}_{t+\Delta t}}{p} \\ \text{Forward:} & y_t = \tilde{y}_{t+\Delta t} - v_\theta(\tilde{x}_{t+\Delta t}, t)\Delta t, & x_t = \tilde{x}_{t+\Delta t} - v_\theta(y_t, t)\Delta t \end{array} \tag{8}$$

4.3 Practical Implementation

To enhance GenPO's exploration and stabilize training, we estimate policy entropy and introduce an adaptive KL-divergence-based learning rate schedule for flow policies. Unlike Gaussian policies, flow policies lack closed-form expressions for entropy and KL divergence, preventing direct computation. As a result, prior diffusion RL algorithms [13, 59, 5] have relied on heuristic approximations to encourage exploration. In contrast, GenPO is exactly invertible (Section 4.2), allowing precise computation of the log-likelihood of any state-action pair via the change-of-variables (5) in Section 4.1. This enables unbiased estimation of both entropy and KL divergence. Accordingly, we define the entropy loss in Eq. 9.

$$\mathcal{L}^{ENT}(\pi_\theta) := \mathbb{E}_{s,\tilde{a}\sim\pi_\theta}\left[\log\left(\pi_\theta(\tilde{a} \mid s)\right)\right]. \tag{9}$$

Besides, to enable adaptive scheduling of the learning rate, we also present an unbiased estimation of the KL divergence as shown in Algorithm 1, and the estimation formula for computing the KL divergence between diffusion policies is

$$\widehat{\mathrm{KL}}(\pi_{\theta_{old}} \mid \pi_\theta) := \mathbb{E}_{s,\tilde{a}\sim\pi_{\theta_{old}}}\left[\log\left(\pi_{\theta_{old}}(\tilde{a} \mid s)\right) - \log\left(\pi_\theta(\tilde{a} \mid s)\right)\right]. \tag{10}$$

We also present an instantiation of our algorithm within the PPO framework. The surrogate loss of PPO is modified as (11).

$$\mathcal{L}^{\mathrm{PPO}}(\theta) := \mathbb{E}_{(s_t,\tilde{a}_t)\sim\pi_{\theta_{\mathrm{old}}}}\left[\min\left(\frac{\pi_\theta(\tilde{a}_t \mid s_t)}{\pi_{\theta_{old}}(\tilde{a}_t \mid s_t)}\hat{A}_t,\ \mathrm{clip}(\frac{\pi_\theta(\tilde{a}_t \mid s_t)}{\pi_{\theta_{old}}(\tilde{a}_t \mid s_t)}, 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]. \tag{11}$$

Notably, to mitigate this issue and prevent ineffective exploration of the action space, we introduce a compression loss as $\mathbb{E}_{x_1,y_1\sim\pi_\theta}\left[(x_1 - y_1)^2\right]$, which diminishes the mean square error between x and y components, in the policy loss. The final diffusion policy loss is shown in (12).

$$\mathcal{L}(\theta) := \mathcal{L}^{PPO} + \lambda\mathcal{L}^{ENT} + \nu\mathbb{E}_{x_1,y_1\sim\pi_\theta}\left[(x_1 - y_1)^2\right], \tag{12}$$

where λ and ν are the coefficients of the entropy and compression loss, respectively.

5 Experiments

In this section, we empirically evaluate the proposed method on several control tasks within IsaacLab. We also conduct ablation studies to investigate the impact of key hyperparameters. These experiments aim to compare our approach with popular online RL algorithms and analyze how hyperparameter choices affect performance.

5.1 Comparative Evaluation

Setup. To evaluate our method, we conducted experiments on a suite of benchmark tasks provided by IsaacLab. The benchmark environments used in this study include Isaac-Ant-v0, Isaac-Humanoid-v0, Isaac-Lift-Cube-Franka-v0, Isaac-Repose-Cube-Shadow-Direct-v0, Isaac-Velocity-Flat-Anymal-D-v0, Isaac-Velocity-Rough-Unitree-Go2-v0, Isaac-Velocity-Rough-H1-v0, and Isaac-Quadcopter-Direct-v0. For brevity, we refer to these environments respectively as **Ant**, **Humanoid**, **Franka Arm**, **Shadow Hand**, **Anymal-D**, **Unitree-Go2**, **Unitree-H1**, and **Quadcopter** throughout the remainder of the paper. Our algorithm GenPO is compared and evaluated against several well-known model-free algorithms, including DDPG, TD3, SAC, PPO, as well as two diffusion-based off-policy algorithms: DACER and QVPO. To assess statistical robustness, all experiments are repeated across five random seeds. More details related to the experiments can be found in the supplementary materials.

Figure 3 presents the mean episodic return (solid line) and one standard deviation (shaded area). GenPO and PPO achieve near-optimal performance with fewer environment interactions compared to

 https://doi.org/10.52202/085713-4347

Algorithm 1 Generative Diffusion Policy Optimization

Input: generative diffusion policy $\pi_\theta(\tilde{a} \mid s)$, value network $V_\omega(s)$.

$\theta' \leftarrow \theta$
for t **in** $1, 2, \cdots, T$ **do**
  for each actor **do**
    Sample dummy actions $\tilde{a} = (x, y)$ from $\pi_\theta(a \mid s)$
    Use $a = \frac{x+y}{2}$ to interact with the environment for N timesteps
    Calculate the probability of $\pi_\theta(\tilde{a} \mid s)$ of the executed dummy actions
    Use GAE to estimate advantages $\hat{A}_1, \cdots, \hat{A}_N$ and returns $\hat{R}_1, \cdots, \hat{R}_n$
    Store transitions, dummy actions, probability, and advantage value in rollout
  end for
  for k **in** $1, 2, \cdots, K$ **do**
    Estimate KL divergence according to (10) and adjust the learning rate α

$$\alpha \leftarrow \begin{cases} \frac{\alpha}{2}, & D_{KL} \geq \bar{\epsilon}; \\ 2\alpha, & D_{KL} \leq \underline{\epsilon}. \end{cases}$$

    Update $\pi_\theta(\tilde{a} \mid s)$ with the surrogate loss (12) with and mini-batches from rollout
    Update the value network $V_\omega(s)$ with loss $\mathbb{E}_{s,a\sim\pi_\theta}\left[\left\|V_\omega(s) - \hat{R}(s,a)\right\|^2\right]$
  end for
end for

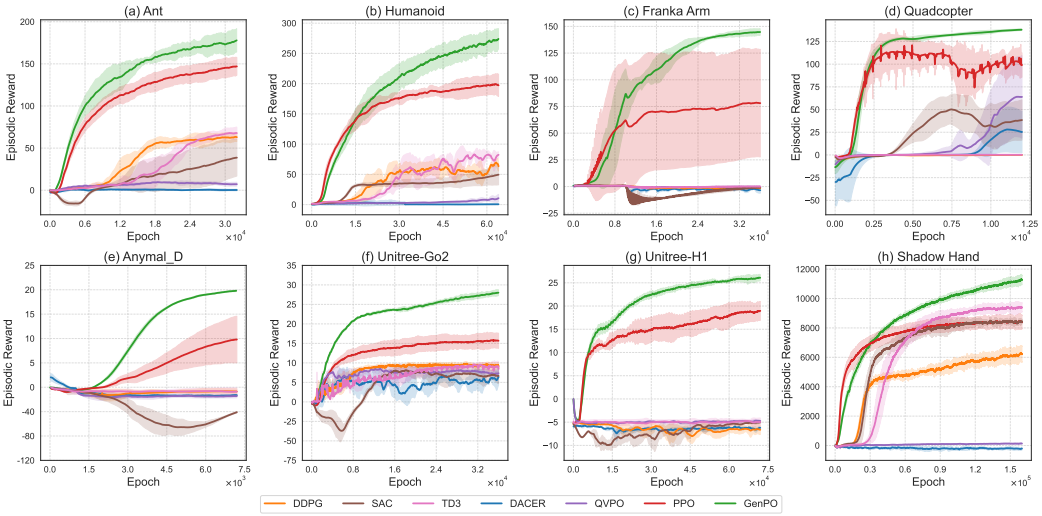

Figure 3: Learning curves across 8 IsaacLab benchmarks. Results are averaged over 5 runs. The x-axis denotes training epochs, and the y-axis shows average episodic return with one standard deviation shaded.

off-policy baselines, benefiting from synchronized data collection that enhances training stability and convergence speed. In contrast, off-policy methods struggle in large-scale parallel settings. This issue is particularly pronounced in diffusion-based algorithms, whose policies have difficulty tracking the rapidly shifting data distribution in the replay buffer. Additionally, GenPO's generative diffusion policy enhances exploration over standard Gaussian policies, contributing to more efficient and robust policy optimization. These factors together explain GenPO's superior performance across diverse tasks.

Table 1 reports the average episodic return (with standard deviation) across eight benchmark tasks. GenPO consistently achieves the highest mean return across all environments, significantly outperforming all algorithms. Notably, on the most challenging tasks, such as Unitree robots locomotion

Table 1: Comparison of the average final rewards of GenPO with some prevalent RL methods in IsaacLab Benchmarks. The maximum value for each task is shown in bold. The standard deviation of the five runs is enclosed in parentheses.

Algorithm	Ant	Humanoid	Franka-Arm	Quadcopter	Anymal-D	Unitree-Go2	Unitree-H1	Shadow-Hand
DDPG [33]	62.96(5.18)	63.34(7.39)	-1.61(0.76)	0.03(0.13)	-0.86(0.55)	9.09(0.81)	-6.63(1.04)	6209.59(559.44)
TD3 [17]	67.80(7.05)	82.36(4.70)	0.03(0.12)	0.17(0.15)	-0.89(0.77)	8.44(1.05)	-4.97(0.87)	9386.64(314.37)
SAC [22]	38.68(21.84)	49.52(16.89)	-1.10(0.95)	38.46(22.20)	-5.12(0.14)	6.67(2.02)	-5.05(0.18)	8459.74(135.79)
DACER [59]	0.29(0.98)	0.58(0.01)	-3.43(2.82)	25.39(22.99)	-1.62(0.09)	6.37(2.07)	-6.21(0.17)	-207.43(88.82)
QVPO [13]	7.19(2.32)	10.59(6.30)	-0.12(0.14)	63.99(45.42)	-1.81(0.36)	7.34(0.62)	-4.72(0.16)	134.36(39.05)
PPO [51]	146.94(10.61)	197.25(18.26)	78.14(50.43)	99.08(13.49)	9.80(4.78)	15.67(1.92)	18.97(2.02)	8402.21(435.64)
GenPO(*)	**177.90(13.87)**	**273.94(16.96)**	**144.78(3.10)**	**137.95(0.84)**	**19.80(0.16)**	**28.01(0.76)**	**26.09(0.68)**	**11282.35(322.94)**

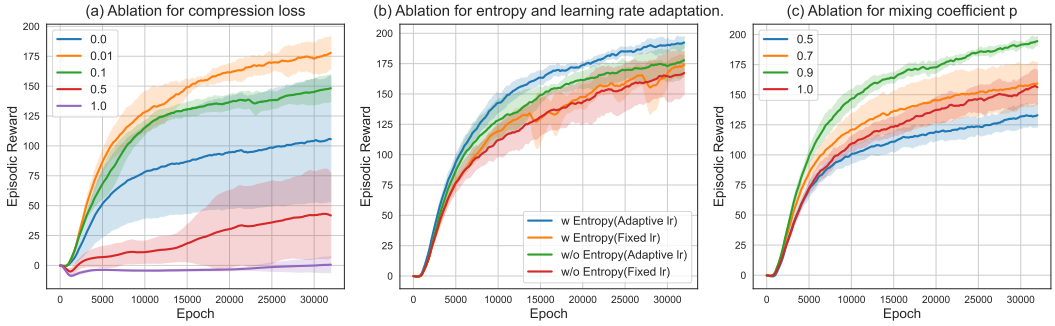

Figure 4: Ablation study results. (a) Effect of varying the compression loss coefficient ν on training stability and final performance. (b) Impact of entropy and learning rate adaptation on exploration and convergence. (c) Performance under different mixing coefficients p in flow policies.

and Shadow Hand manipulation, GenPO demonstrates not only higher returns but also lower variance, indicating greater training stability.

5.2 Ablation Study

To better understand the significance of each component in GenPO and explain its superior performance on the IsaacLab benchmark, we conducted the ablation study focusing on three key design choices: 1) the effect of the compression loss in the training objective; 2) the impact of entropy and KL divergence estimation; and 3) the choice of the mixing coefficient p. We present results on the Isaac-Ant-v0 task as a representative example, as similar trends are observed across other environments. Additional ablation results are provided in the supplementary materials.

Effect of Compression Loss. To validate the impact of the compression loss on final performance, we vary its coefficient ν in (12) and analyze the resulting training curves in Figure 4(a). We evaluate $\nu \in \{0, 0.01, 0.1, 0.5, 1.0\}$ and find that $\nu = 0.01$ provides the best balance between stability and regularization. Larger values overly constrain the policy and lead to poor performance, while excluding the loss ($\nu = 0$) slows convergence. Based on this tradeoff, we set $\nu = 0.01$ in all experiments.

Entropy and Learning rate adaptation. Figure 4(b) illustrates the effect of entropy regularization and learning rate adaptation. Adaptive adjustment of the learning rate leads to faster and more stable convergence, while entropy regularization enhances exploration and yields higher returns. Empirical results confirm that both components, implemented as described in Section 4.3, substantially improve overall performance. In all tasks, we employ an adaptive learning rate scheme and include an entropy regularization term in the loss defined in Eq. (12).

Mixing coefficient. As shown in Figure 4(c), GenPO performs best when the mixing coefficient is set to $p = 0.9$. This is because setting p too low will cause the forward mixing process (8) to be numerically unstable, making the value too large and affecting the stability of training. On the contrary, too high p will introduce redundant exploration in the doubled dummy action space, thus slowing down the convergence. Therefore, choosing $p = 0.9$ can achieve a practical trade-off between stability and exploration efficiency, so we set p to 0.9 for all experiments.

https://doi.org/10.52202/085713-4347

6 Conclusion, Limitation and Future Works

This paper proposes Generative Diffusion Policy Optimization (GenPO), a novel approach that integrates generative diffusion policies into the on-policy RL framework. GenPO enables tractable log-likelihood computation of actions in the diffusion model, which is unavailable in prior works, and further unbiasedly estimates the entropy and KL divergence of the diffusion policy, thereby achieving entropy regularization and adaptive learning rate adjustment during policy update. Finally, GenPO achieves superior performance compared with existing RL baselines, including diffusion-based RL methods, on eight IsaacLab benchmarks, with comparable sample efficiency and faster convergence, highlighting the potential of generative diffusion policies in on-policy RL for large-scale simulation and real-world robotics control.

Despite its excellent performance, GenPO faces the problem of relatively high computational and memory overhead to be resolved in the future. While the GPU's parallelism helps to mitigate this issue, GenPO still incurs considerable complexity in computing the Jacobian determinant. Hence, our future work will focus on exploring how to optimize the computational and memory efficiency of GenPO to facilitate its deployment in more real-world applications.

Acknowledgement

This work was supported by National Natural Science Foundation of China (62303319, 62406195), Shanghai Local College Capacity Building Program (23010503100), ShanghaiTech AI4S Initiative SHTAI4S202404, HPC Platform of ShanghaiTech University, and MoE Key Laboratory of Intelligent Perception and Human-Machine Collaboration (ShanghaiTech University), Shanghai Engineering Research Center of Intelligent Vision and Imaging. This work was also supported in part by computational resources provided by Fcloud CO., LTD.

References

[1] Suzan Ece Ada, Erhan Oztop, and Emre Ugur. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *IEEE Robotics and Automation Letters*, 9(4):3116âĂŞ3123, April 2024.

[2] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[3] Joseph K Blitzstein and Jessica Hwang. *Introduction to probability*. Chapman and Hall/CRC, 2019.

[4] Janaka Brahmanage, Jiajing Ling, and Akshat Kumar. Flowpg: action-constrained policy gradient with normalizing flows. *Advances in Neural Information Processing Systems*, 36:20118–20132, 2023.

[5] Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palanicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. Dime: Diffusion-based maximum entropy reinforcement learning. *arXiv preprint arXiv:2502.02316*, 2025.

[6] Chen-Hao Chao, Chien Feng, Wei-Fang Sun, Cheng-Kuang Lee, Simon See, and Chun-Yi Lee. Maximum entropy reinforcement learning via energy-based normalizing flow. *arXiv preprint arXiv:2405.13629*, 2024.

[7] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

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

[9] Yuhui Chen, Haoran Li, and Dongbin Zhao. Boosting continuous control with consistency policy. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 335–344, 2024.

[10] 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.

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

[12] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[13] 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. *arXiv preprint arXiv:2405.16173*, 2024.

[14] Shutong Ding, Yimiao Zhou, Ke Hu, Xi Yao, Junchi Yan, Xiaoying Tang, and Ye Shi. Diopt: Self-supervised diffusion for constrained optimization. *arXiv preprint arXiv:2502.10330*, 2025.

[15] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[16] Xiaoyi Dong, Jian Cheng, and Xi Sheryl Zhang. Maximum entropy reinforcement learning with diffusion policy. *arXiv preprint arXiv:2502.11612*, 2025.

[17] 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.

[18] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.

[19] Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin P Murphy, and Tim Salimans. Diffusion meets flow matching: Two sides of the same coin. 2024. *URL https://diffusionflow. github. io*, 2024.

[20] A Grover, M Dhar, and S Ermon. Flow-gan: combining maximum likelihood and adversarial learning in generative models. arxiv e-prints. *arXiv preprint arXiv:1705.08868*, 2017.

[21] 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.

[22] 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.

[23] Sami Haddadin. The franka emika robot: A standard platform in robotics research. *IEEE Robotics & Automation Magazine*, 2024.

[24] 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.

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

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

[27] 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, 2024.

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

 https://doi.org/10.52202/085713-4347

[29] Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B. Tenenbaum. Learning to act from actionless videos through dense correspondences, 2023.

[30] 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.

[31] Xiang Li, Varun Belagali, Jinghuan Shang, and Michael S Ryoo. Crossway Diffusion: Improving diffusion-based visuomotor policy via self-supervised learning. *arXiv preprint arXiv:2307.01849*, 2023.

[32] Zechu Li, Tao Chen, Zhang-Wei Hong, Anurag Ajay, and Pulkit Agrawal. Parallel q-learning: Scaling off-policy reinforcement learning under massively parallel simulation. In *International Conference on Machine Learning*, pages 19440–19459. PMLR, 2023.

[33] 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.

[34] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[35] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[36] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[37] 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.

[38] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[39] Bogdan Mazoure, Thang Doan, Audrey Durand, Joelle Pineau, and R Devon Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. In *Conference on Robot Learning*, pages 430–444. PMLR, 2020.

[40] Fei Ni, Jianye Hao, Yao Mu, Yifu Yuan, Yan Zheng, Bin Wang, and Zhixuan Liang. Metadiffuser: Diffusion model as conditional planner for offline meta-rl. In *International Conference on Machine Learning*, pages 26087–26105. PMLR, 2023.

[41] Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

[42] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models, 2023.

[43] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

[44] Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

[45] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

[46] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[47] Quentin Rouxel, Andrea Ferrari, Serena Ivaldi, and Jean-Baptiste Mouret. Flow matching imitation learning for multi-support manipulation. In *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, pages 528–535. IEEE, 2024.

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

[49] 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.

[50] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

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

[53] 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.

[54] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[55] Yunhao Tang and Shipra Agrawal. Boosting trust region policy optimization by normalizing flows policy. *arXiv preprint arXiv:1809.10326*, 2018.

[56] Paul Tuffield and Hugo Elias. The shadow robot mimics human actions. *Industrial Robot: An International Journal*, 30(1):56–60, 2003.

[57] Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22532–22541, 2023.

[58] Fangyikang Wang, Hubery Yin, Yue-Jiang Dong, Huminhao Zhu, Hanbin Zhao, Hui Qian, Chen Li, et al. Belm: Bidirectional explicit linear multi-step sampler for exact inversion in diffusion models. *Advances in Neural Information Processing Systems*, 37:46118–46159, 2024.

[59] Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, et al. Diffusion actor-critic with entropy regulator. *Advances in Neural Information Processing Systems*, 37:54183–54204, 2024.

[60] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[61] Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:31278–31291, 2022.

[62] 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.

[63] Guoqiang Zhang, Jonathan P Lewis, and W Bastiaan Kleijn. Exact diffusion inversion via bidirectional integration approximation. In *European Conference on Computer Vision*, pages 19–36. Springer, 2024.

[64] Jing Zhang, Chi Zhang, Wenjia Wang, and Bingyi Jing. Constrained policy optimization with explicit behavior density for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:5616–5630, 2023.

 https://doi.org/10.52202/085713-4347

[65] Qinsheng Zhang and Yongxin Chen. Diffusion normalizing flow. *Advances in neural information processing systems*, 34:16280–16291, 2021.

[66] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, pages 1719–1735. PMLR, 2021.

[67] Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. Madiff: Offline multi-agent learning with diffusion models, 2023.

[68] Ziwen Zhuang, Shenzhe Yao, and Hang Zhao. Humanoid parkour learning. *arXiv preprint arXiv:2406.10759*, 2024.

A Environmental Details

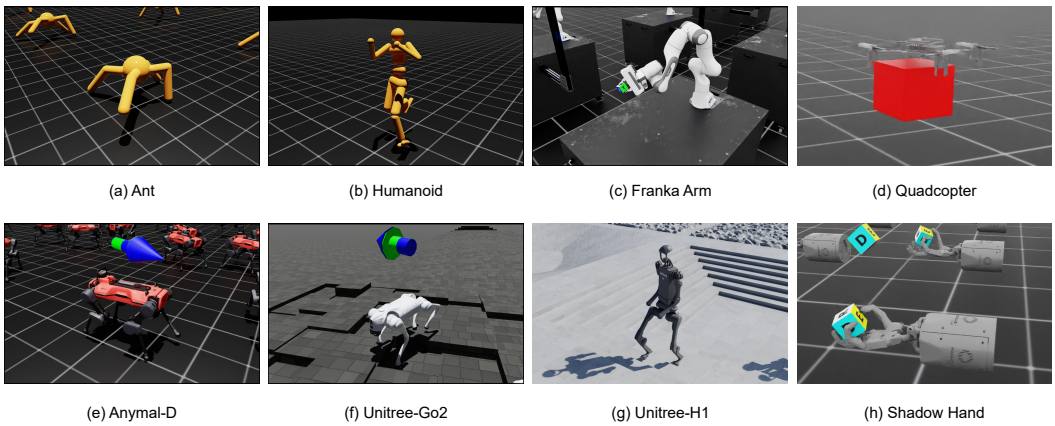

Figure 5: Eight Isaaclab benchmark visualizations, eight images from `https://isaac-sim.github.io/IsaacLab/main/source/overview/environments.html`. From (a) to (h) are Isaac-Ant-v0, Isaac-Humanoid-v0, Isaac-Lift-Cube-Franka-v0, Isaac-Quadcopter-Direct-v0, Isaac-Velocity-Flat-Anymal-D-v0, Isaac-Velocity-Rough-Unitree-Go2-v0, Isaac-Velocity-Rough-H1-v0, and Isaac-Repose-Cube-Shadow-Direct-v0.

IsaacLab [2] is a unified and modular framework for robot learning, designed to streamline common workflows in robotics research, including reinforcement learning, imitation learning, and motion planning. The framework leverages NVIDIA Isaac Sim for high-fidelity simulation and benefits from PhysX's GPU-accelerated physics engine as well as a tile-based rendering API for vectorized rendering.

As displayed in Figure 5, we selected 8 representative and challenging environments, which can be roughly divided into the following categories according to the official manual of Isaaclab `https://isaac-sim.github.io/IsaacLab/main/source/overview/environments.html`:

1. Classic: These classic tasks are derived from the IsaacGymEnv implementation of MuJoCo-style environments, providing standardized benchmarks for locomotion and control in continuous action spaces.
   - **Isaac-Ant-v0.** The state and action spaces are $(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{60\times 8}$. The task involves controlling a MuJoCo Ant robot to move in a specified direction.
   - **Isaac-Humanoid-v0.** The state and action spaces are $(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{87\times 21}$. The objective is to control a MuJoCo Humanoid robot to move in a specified direction.
2. Manipulation: These environments involve manipulation tasks performed by a fixed-base robotic arm or a dexterous hand, such as object reaching, grasping, or in-hand rotation.
   - **Isaac-Lift-Cube-Franka-v0.** The state and action spaces are $(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{36\times 8}$. The task involves controlling a Franka Emika robot arm to pick up a cube and transport it to a randomly sampled target position.
   - **Isaac-Repose-Cube-Shadow-Direct-v0.** The state and action spaces are $(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{157\times 20}$. The task requires using a Shadow Dexterous Hand to perform in-hand reorientation of a cube to match a target orientation.
3. Locomotion: This category includes legged locomotion environments that challenge agents to coordinate multiple degrees of freedom to achieve stable and directed movement. Tasks include forward walking, turning, and terrain adaptation, typically implemented with humanoid robots.
   - **Isaac-Velocity-Flat-Anymal-D-v0.** The state and action spaces are $(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{48\times 12}$. The task requires controlling an ANYmal D quadruped robot to follow a commanded velocity trajectory on flat terrain

[2] https://isaac-sim.github.io/IsaacLab/main/index.html

https://doi.org/10.52202/085713-4347

- **Isaac-Velocity-Rough-Unitree-Go2-v0.** The state and action spaces are $(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{235\times12}$. The task involves controlling a Unitree Go2 quadruped robot to follow a commanded velocity trajectory over rough terrain.
- **Isaac-Velocity-Rough-H1-v0.** The state and action spaces are $(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{256\times19}$. The task involves controlling a Unitree H1 humanoid robot to follow a commanded velocity trajectory over rough terrain.

4. Others:

   - **Isaac-Quadcopter-Direct-v0.** The state and action spaces are $(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{12\times4}$. The task is to control quadrotors to fly and hover at a designated goal position by applying thrust commands.

Moreover, based on the IsaacLab Engine, the an anonymous link for the visualization of our experimental results is shown in `anonymous-project365.github.io`.

B Training setups

B.1 Hardware Configurations

All experiments were carried out on a server equipped with two Intel Xeon Gold 6430 CPUs (32 cores per socket, 64 threads total per CPU, 128 threads total), with a base frequency of 2.1 GHz and a maximum turbo frequency of 3.4 GHz. The system supports 52-bit physical and 57-bit virtual addressing. For GPU acceleration, we used 8 NVIDIA GeForce RTX 4090 D GPUs, each with 24 GB of GDDR6X memory, connected via PCIe. The GPUs support CUDA 12.8 and were operating under the NVIDIA driver version 570.124.04. The machine is configured with NUMA topology across 2 nodes, each handling 64 logical CPUs. No GPU MIG (Multi-Instance GPU) or ECC was enabled during the experiments.

B.2 Reinforcement Learning Framework in IsaacLab

IsaacLab provides native integration with several reinforcement learning libraries, including RSL-RL `https://github.com/leggedrobotics/rsl_rl`, RL-Games `https://github.com/Denys88/rl_games`, SKRL `https://skrl.readthedocs.io/en/latest/`, and Stable-Baselines3 `https://stable-baselines3.readthedocs.io/en/master/index.html`, each of which exposes a distinct API for agent-environment interaction. In this work, we adopt SKRL as our primary framework for implementing baseline algorithms. SKRL is an open-source, modular, and extensible library that provides standardized implementations of widely used reinforcement learning methods. Specifically, we utilize its implementations of PPO `https://skrl.readthedocs.io/en/latest/api/agents/ppo.html`, SAC `https://skrl.readthedocs.io/en/latest/api/agents/sac.html`, TD3 `https://skrl.readthedocs.io/en/latest/api/agents/td3.html`, and DDPG `https://skrl.readthedocs.io/en/latest/api/agents/ddpg.html` for baseline evaluation. For algorithms such as QVPO and DACER, we design custom environment wrappers to adapt the existing interface without modifying the underlying simulation environments. Our proposed method, GenPO, is implemented using RSL-RL, a lightweight and high-performance framework tailored for robotics and continuous control, which emphasizes computational efficiency and ease of deployment.

B.3 Hyperparameters

Tables 2 and Table 10 summarize the hyperparameter configurations used across our experiments. For the baseline algorithms: PPO, SAC, TD3, and DDPG, we adopt the default hyperparameter settings provided by the SKRL library. For our proposed method, GenPO, we align its hyperparameter configuration with that of PPO to ensure a fair and controlled comparison.

Table 2: Hyper-parameters used in the Isaaclab-Ant-v0.

Hyperparameters	GenPO	PPO	QVPO	DACER	SAC	TD3	DDPG
hidden layers in actor network	[400,200,100]	[400,200,100]	[256,256,256]	[256,256,256]	[400,200,100]	[400,200,100]	[400,200,100]
hidden layers in critic network	[400,200,100]	[400,200,100]	[256,256,256]	[256,256,256]	[400,200,100]	[400,200,100]	[400,200,100]
Activation	mish	elu	mish	relu	mish	mish	mish
Batch size	4096	4096	4096	4096	4096	4096	4096
Use GAE	True	True	N/A	N/A	N/A	N/A	N/A
Discount for reward γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99
GAE Smoothing Parameter λ	0.95	0.95	N/A	N/A	N/A	N/A	N/A
Learning rate for actor	1×10^{-3}	1×10^{-3}	3×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Learning rate for critic	1×10^{-3}	1×10^{-3}	3×10^{-4}	3×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Actor Critic grad norm	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Memory size	N/A	N/A	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}
Entropy coefficient	0.01	0.01	N/A	N/A	N/A	N/A	N/A
Value loss coefficient	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Noise clip	N/A	N/A	N/A	0.5	N/A	N/A	N/A
Surrogate clip	0.2	0.2	N/A	N/A	N/A	N/A	N/A
Diffusion steps	5	N/A	20	20	N/A	N/A	N/A
Desired KL	0.01	0.01	N/A	N/A	N/A	N/A	N/A

Table 3: Hyper-parameters used in the Isaaclab-Humanoid-v0.

Hyperparameters	GenPO	PPO	QVPO	DACER	SAC	TD3	DDPG
hidden layers in actor network	[400,200,100]	[400,200,100]	[256,256,256]	[256,256,256]	[400,200,100]	[400,200,100]	[400,200,100]
hidden layers in critic network	[400,200,100]	[400,200,100]	[256,256,256]	[256,256,256]	[400,200,100]	[400,200,100]	[400,200,100]
Activation	mish	elu	mish	relu	mish	mish	mish
Batch size	4096	4096	4096	4096	4096	4096	4096
Use GAE	True	True	N/A	N/A	N/A	N/A	N/A
Discount for reward γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99
GAE Smoothing Parameter λ	0.95	0.95	N/A	N/A	N/A	N/A	N/A
Learning rate for actor	5×10^{-4}	5×10^{-4}	3×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Learning rate for critic	5×10^{-4}	5×10^{-4}	3×10^{-4}	3×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Actor Critic grad norm	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Memory size	N/A	N/A	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}
Entropy coefficient	0.01	0.01	N/A	N/A	N/A	N/A	N/A
Value loss coefficient	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Noise clip	N/A	N/A	N/A	0.5	N/A	N/A	N/A
Surrogate clip	0.2	0.2	N/A	N/A	N/A	N/A	N/A
Diffusion steps	5	N/A	20	20	N/A	N/A	N/A
Desired KL	0.01	0.01	N/A	N/A	N/A	N/A	N/A

Table 4: Hyper-parameters used in the Isaac-Lift-Cube-Franka-v0.

Hyperparameters	GenPO	PPO	QVPO	DACER	SAC	TD3	DDPG
hidden layers in actor network	[256,128,64]	[256,128,64]	[256,256,256]	[256,256,256]	[256, 128, 64]	[256, 128, 64]	[256, 128, 64]
hidden layers in critic network	[256,128,64]	[256,128,64]	[256,256,256]	[256,256,256]	[256, 128, 64]	[256, 128, 64]	[256, 128, 64]
Activation	mish	elu	mish	relu	mish	mish	mish
Batch size	4096	4096	4096	4096	4096	4096	4096
Use GAE	True	True	N/A	N/A	N/A	N/A	N/A
Discount for reward γ	0.98	0.98	0.99	0.99	0.99	0.99	0.99
GAE Smoothing Parameter λ	0.95	0.95	N/A	N/A	N/A	N/A	N/A
Learning rate for actor	1×10^{-4}	1×10^{-4}	3×10^{-4}	5×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
Learning rate for critic	1×10^{-4}	1×10^{-4}	3×10^{-4}	3×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
Actor Critic grad norm	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Memory size	N/A	N/A	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}
Entropy coefficient	0.006	0.006	N/A	N/A	N/A	N/A	N/A
Value loss coefficient	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Noise clip	N/A	N/A	N/A	0.5	N/A	N/A	N/A
Surrogate clip	0.2	0.2	N/A	N/A	N/A	N/A	N/A
Diffusion steps	5	N/A	20	20	N/A	N/A	N/A
Desired KL	0.01	0.01	N/A	N/A	N/A	N/A	N/A

 https://doi.org/10.52202/085713-4347

Table 5: Hyper-parameters used in the Isaac-Quadcopter-Direct-v0.

Hyperparameters	GenPO	PPO	QVPO	DACER	SAC	TD3	DDPG
hidden layers in actor network	[64,64]	[64,64]	[256,256,256]	[256,256,256]	[64, 64]	[64, 64]	[64, 64]
hidden layers in critic network	[64,64]	[64,64]	[256,256,256]	[256,256,256]	[64, 64]	[64, 64]	[64, 64]
Activation	mish	elu	mish	relu	mish	mish	mish
Batch size	4096	4096	4096	4096	4096	4096	4096
Use GAE	True	True	N/A	N/A	N/A	N/A	N/A
Discount for reward γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99
GAE Smoothing Parameter λ	0.95	0.95	N/A	N/A	N/A	N/A	N/A
Learning rate for actor	5×10^{-4}	5×10^{-4}	3×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Learning rate for critic	5×10^{-4}	5×10^{-4}	3×10^{-4}	3×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Actor Critic grad norm	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Memory size	N/A	N/A	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}
Entropy coefficient	0.01	0.01	N/A	N/A	N/A	N/A	N/A
Value loss coefficient	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Noise clip	N/A	N/A	N/A	0.5	N/A	N/A	N/A
Surrogate clip	0.2	0.2	N/A	N/A	N/A	N/A	N/A
Diffusion steps	5	N/A	20	20	N/A	N/A	N/A
Desired KL	0.01	0.01	N/A	N/A	N/A	N/A	N/A

Table 6: Hyper-parameters used in the Isaac-Velocity-Flat-Anymal-D-v0.

Hyperparameters	GenPO	PPO	QVPO	DACER	SAC	TD3	DDPG
hidden layers in actor network	[128,128,128]	[128,128,128]	[256,256,256]	[256,256,256]	[128, 128, 128]	[128, 128, 128]	[128, 128, 128]
hidden layers in critic network	[128,128,128]	[128,128,128]	[256,256,256]	[256,256,256]	[128, 128, 128]	[128, 128, 128]	[128, 128, 128]
Activation	mish	elu	mish	relu	mish	mish	mish
Batch size	4096	4096	4096	4096	4096	4096	4096
Use GAE	True	True	N/A	N/A	N/A	N/A	N/A
Discount for reward γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99
GAE Smoothing Parameter λ	0.95	0.95	N/A	N/A	N/A	N/A	N/A
Learning rate for actor	1×10^{-3}	1×10^{-3}	3×10^{-4}	5×10^{-4}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Learning rate for critic	1×10^{-3}	1×10^{-3}	3×10^{-4}	3×10^{-4}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Actor Critic grad norm	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Memory size	N/A	N/A	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}
Entropy coefficient	0.005	0.005	N/A	N/A	N/A	N/A	N/A
Value loss coefficient	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Noise clip	N/A	N/A	N/A	0.5	N/A	N/A	N/A
Surrogate clip	0.2	0.2	N/A	N/A	N/A	N/A	N/A
Diffusion steps	5	N/A	20	20	N/A	N/A	N/A
Desired KL	0.01	0.01	N/A	N/A	N/A	N/A	N/A

Table 7: Hyper-parameters used in the Isaac-Velocity-Rough-Unitree-Go2-v0.

Hyperparameters	GenPO	PPO	QVPO	DACER	SAC	TD3	DDPG
hidden layers in actor network	[512, 256, 128]	[512, 256, 128]	[256,256,256]	[256,256,256]	[512, 256, 128]	[512, 256, 128]	[512, 256, 128]
hidden layers in critic network	[512, 256, 128]	[512, 256, 128]	[256,256,256]	[256,256,256]	[512, 256, 128]	[512, 256, 128]	[512, 256, 128]
Activation	mish	elu	mish	relu	mish	mish	mish
Batch size	4096	4096	4096	4096	4096	4096	4096
Use GAE	True	True	N/A	N/A	N/A	N/A	N/A
Discount for reward γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99
GAE Smoothing Parameter λ	0.95	0.95	N/A	N/A	N/A	N/A	N/A
Learning rate for actor	1×10^{-3}	1×10^{-3}	3×10^{-4}	5×10^{-4}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Learning rate for critic	1×10^{-3}	1×10^{-3}	3×10^{-4}	3×10^{-4}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Actor Critic grad norm	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Memory size	N/A	N/A	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}
Entropy coefficient	0.01	0.01	N/A	N/A	N/A	N/A	N/A
Value loss coefficient	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Noise clip	N/A	N/A	N/A	0.5	N/A	N/A	N/A
Surrogate clip	0.2	0.2	N/A	N/A	N/A	N/A	N/A
Diffusion steps	5	N/A	20	20	N/A	N/A	N/A
Desired KL	0.01	0.01	N/A	N/A	N/A	N/A	N/A

Table 8: Hyper-parameters used in the Isaac-Velocity-Rough-H1-v0.

Hyperparameters	GenPO	PPO	QVPO	DACER	SAC	TD3	DDPG
hidden layers in actor network	[512, 256, 128]	[512, 256, 128]	[256,256,256]	[256,256,256]	[512, 256, 128]	[512, 256, 128]	[512, 256, 128]
hidden layers in critic network	[512, 256, 128]	[512, 256, 128]	[256,256,256]	[256,256,256]	[512, 256, 128]	[512, 256, 128]	[512, 256, 128]
Activation	mish	elu	mish	relu	mish	mish	mish
Batch size	4096	4096	4096	4096	4096	4096	4096
Use GAE	True	True	N/A	N/A	N/A	N/A	N/A
Discount for reward γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99
GAE Smoothing Parameter λ	0.95	0.95	N/A	N/A	N/A	N/A	N/A
Learning rate for actor	1×10^{-3}	1×10^{-3}	3×10^{-4}	5×10^{-4}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Learning rate for critic	1×10^{-3}	1×10^{-3}	3×10^{-4}	3×10^{-4}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Actor Critic grad norm	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Memory size	N/A	N/A	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}
Entropy coefficient	0.01	0.01	N/A	N/A	N/A	N/A	N/A
Value loss coefficient	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Noise clip	N/A	N/A	N/A	0.5	N/A	N/A	N/A
Surrogate clip	0.2	0.2	N/A	N/A	N/A	N/A	N/A
Diffusion steps	5	N/A	20	20	N/A	N/A	N/A
Desired KL	0.01	0.01	N/A	N/A	N/A	N/A	N/A

Table 9: Hyper-parameters used in the Isaac-Repose-Cube-Shadow-Direct-v0.

Hyperparameters	GenPO	PPO	QVPO	DACER	SAC	TD3	DDPG
hidden layers in actor network	[512, 512, 256, 128]	[512, 512, 256, 128]	[256,256,256]	[256,256,256]	[512, 512, 256, 128]	[512, 512, 256, 128]	[512, 512, 256, 128]
hidden layers in critic network	[512, 512, 256, 128]	[512, 512, 256, 128]	[256,256,256]	[256,256,256]	[512, 512, 256, 128]	[512, 512, 256, 128]	[512, 512, 256, 128]
Activation	mish	elu	mish	relu	mish	mish	mish
Batch size	4096	4096	4096	4096	4096	4096	4096
Use GAE	True	True	N/A	N/A	N/A	N/A	N/A
Discount for reward γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99
GAE Smoothing Parameter λ	0.95	0.95	N/A	N/A	N/A	N/A	N/A
Learning rate for actor	5×10^{-4}	5×10^{-4}	3×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Learning rate for critic	5×10^{-4}	5×10^{-4}	3×10^{-4}	3×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Actor Critic grad norm	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Memory size	N/A	N/A	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}	1×10^{6}
Entropy coefficient	0.005	0.005	N/A	N/A	N/A	N/A	N/A
Value loss coefficient	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Noise clip	N/A	N/A	N/A	0.5	N/A	N/A	N/A
Surrogate clip	0.2	0.2	N/A	N/A	N/A	N/A	N/A
Diffusion steps	5	N/A	20	20	N/A	N/A	N/A
Desired KL	0.016	0.016	N/A	N/A	N/A	N/A	N/A

Table 10: Hyper-parameters used in GenPO.

Hyperparameter	Ant	Humanoid	Franka Arm	Quadcopter	Anymal-D	Unitree-Go2	Unitree-H1	Shadow Hand
Compress coefficient λ	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Time embedding dimension	32	32	32	16	32	32	32	32
Hidden layers in time embedding	[256,256]	[256,256]	[256,256]	[128,128]	[256,256]	[256,256]	[256,256]	[256,256]
Mixing coefficient p	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

C Details of Model Architecture

C.1 Sinusoidal Positional Embedding

During training, instead of directly concatenating the time step t, state s_t, and action a_t as raw input features, we incorporate sinusoidal positional embeddings to encode the temporal component. This approach, inspired by the positional encoding technique commonly used in denoising diffusion probabilistic models (DDPMs), maps each timestep $t \in \{0, 1, \ldots, T\}$to a fixed-dimensional vector $e_t \in \mathbb{R}^d$ as follows:

$$e_t^{(2i)} = \sin\left(\frac{t}{10000^{2i/d}}\right), \quad e_t^{(2i+1)} = \cos\left(\frac{t}{10000^{2i/d}}\right), \quad \text{for } i = 0, 1, \ldots, \frac{d}{2} - 1. \tag{13}$$

The resulting embedding e_t is then passed through a multi-layer perceptron (MLP) to allow for non-linear transformation and projection into the model's feature space. The MLP output is subsequently concatenated with the state s_t and action a_t to form the final input representation:

$$\tilde{x}_t = \text{concat}(s_t, a_t, \text{MLP}(e_t)), \tag{14}$$

which enriches the model input with a smooth, continuous representation of temporal progression. Unlike raw scalar timestep concatenation, sinusoidal embeddings provide a structured encoding that helps the model infer temporal relationships and ordering, even in the absence of explicit recurrence or attention mechanisms.

https://doi.org/10.52202/085713-4347

C.2 Actor and Critic Model Architecture

In the Isaac-Ant-v0 environment, the actor network, corresponding to the flow policy, is implemented as a multi-layer perceptron (MLP) with hidden layer dimensions [400,200,100]. The activation function used is Mish. The input to the actor consists of the concatenation of the current state, action, and sinusoidal time embedding vectors, resulting in an input dimension of $|\mathcal{A}| + |\mathcal{S}| + |\mathcal{T}|$, The output dimension is equal to the action dimension $|\mathcal{A}|$, representing the flow-based action refinement. The critic network is also implemented as an MLP with the same hidden layer structure [400,200,100] and Mish activation. It receives only the current state as input and outputs a scalar value corresponding to the estimated state value.

D Additional Experiments

D.1 Effect of Different Parallel Environments Size

To assess the effect of parallelization on the learning efficiency of GenPO, we conducted an experiment varying the number of parallel environments. We trained GenPO agents using 512, 1024, 2048, 4096, and 8192 parallel environments, respectively. All other hyperparameters were kept consistent across these runs.

As illustrated in Figure 6, increasing the number of parallel environments generally leads to improved learning performance. Agents trained with a larger number of environments exhibit faster convergence and achieve higher final rewards compared to those trained with fewer environments. However, the performance gain from using 8192 environments over 4096 is marginal. To balance performance and computational efficiency, we fix the number of parallel environments to 4096 in all experiments.

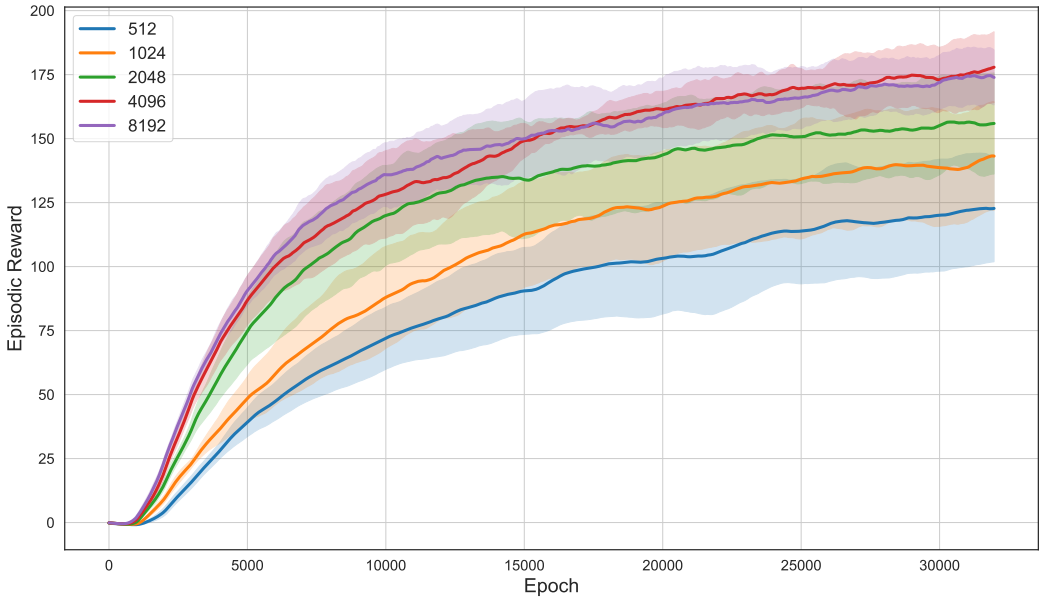

Figure 6: Learning curves for the Isaaclab-Ant-v0 benchmark with different numbers of parallel environments. Results are averaged over 5 runs. The x-axis denotes training epochs, and the y-axis shows average episodic return with one standard deviation shaded.

D.2 Effect of Different Flow Policy Steps

We investigate the impact of the number of flow steps, denoted by T, on learning performance. To this end, we train agents using different values of $T \in \{1, 2, 5, 10, 20\}$, while keeping all other hyperparameters fixed. As shown in Figure 7, the number of flow steps influences convergence

speed and performance stability. Notably, as long as T is not too small (e.g., $T \geq 2$), the overall performance remains robust.

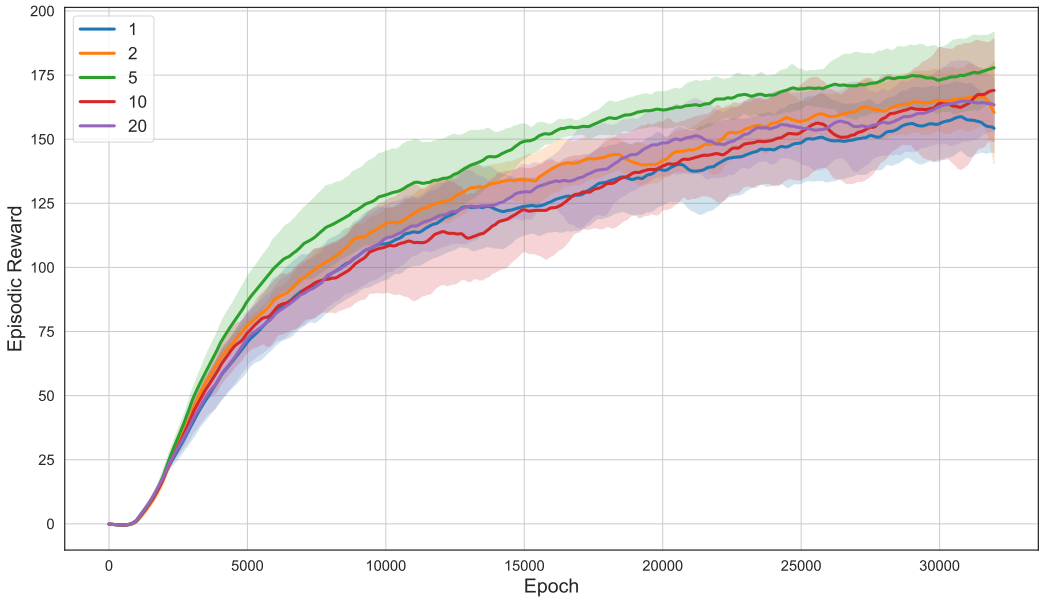

Figure 7: Learning curves for different flow policy time steps on the Isaaclab-Ant-v0 benchmark. Results are averaged over 5 runs. The x-axis denotes training epochs, and the y-axis shows average episodic return with one standard deviation shaded.

D.3 Effect of Different Operations on Dummy Action Space

We study the impact of recovering a single action from the dummy action $\tilde{a} = (x, y)$ by interpolating between the two partial components x, y. Specifically, we construct the real action as $\alpha x + (1 - \alpha)\, y$ and $\alpha \in \{0.0, 0.3, 0.5, 0.7, 1.0\}$ respectively. As illustrated in Figure 8, the best performance is achieved when $\alpha = 0.5$. This observation highlights the benefit of equally leveraging both components of the dummy action. When $\alpha = 0.0$ or $\alpha = 1.0$, only one side of the dummy action space is utilized, leading to limited exploration and suboptimal performance. In contrast, interpolating between both components enhances exploration diversity and improves sample efficiency, which is crucial for effective policy optimization in high-dimensional or multimodal action spaces.

D.4 Effect of Time Steps Embedding

We investigate the effect of sinusoidal time embeddings on the performance of the flow policy. As shown in Figure 9, incorporating sinusoidal embeddings significantly improves the policy's ability to utilize temporal information across different stages of the diffusion process. The structured, high-dimensional representation provided by the sinusoidal encoding facilitates better discrimination of time steps, which in turn enhances the model's capacity to learn effective denoising functions. In contrast, using simple concatenation of the scalar timestep with the state and action fails to provide sufficient temporal structure, resulting in degraded performance. Based on these observations, we adopt sinusoidal positional embeddings to encode time steps in all subsequent experiments.

D.5 Training and Inference Time

To validate the computational efficiency, we also conduct comparison experiments on the proposed GenPO with 6 other baselines, which respectively evaluate the inference, training and parallel training time on single/multiple GPUs, as shown in Figure 10, Figure 11 and Figure 12. It can be observed that, while the GenPO's inference time is slightly longer than that of Gaussian policies due to multiple

https://doi.org/10.52202/085713-4347

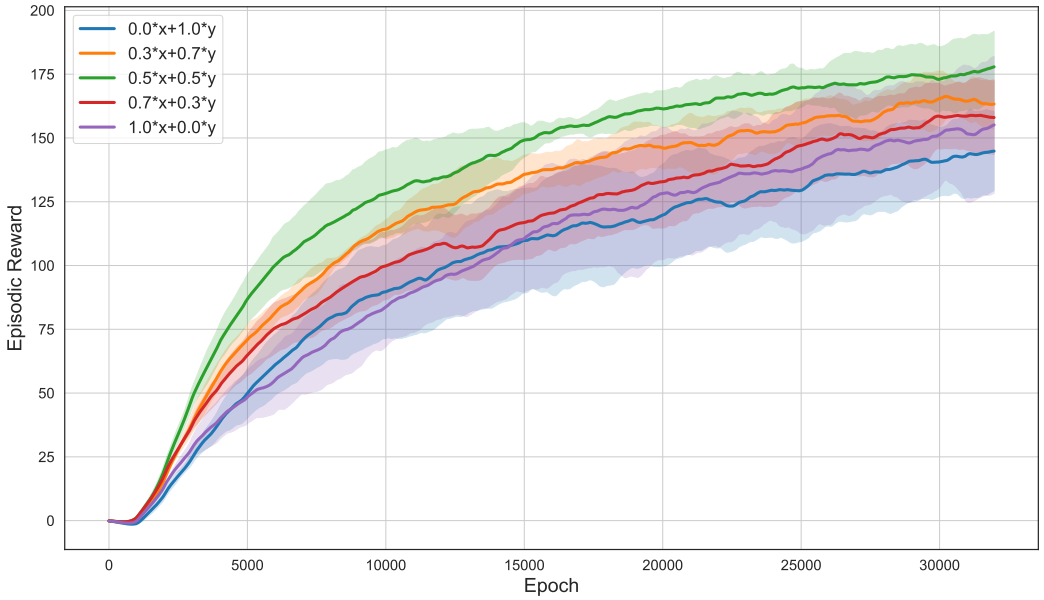

Figure 8: Learning curves for different operations on dummy action space on the Isaaclab-Ant-v0 benchmark. Results are averaged over 5 runs. The x-axis denotes training epochs, and the y-axis shows average episodic return with one standard deviation shaded.

denoising iterations of diffusion, it remains significantly more efficient than previous diffusion-based RL algorithms such as QVPO and DACER. Besides, to our knowledge, the inference time of 2.577ms is more than sufficient for the real-time decision in robot control. Furthermore, in terms of training time, although the performance of GenPO on a single GPU is less than ideal (yet still superior to previous diffusion-based RL methods), it can effectively leverage multi-GPU parallelism to significantly accelerate training. In contrast, the traditional RL algorithm PPO cannot obviously benefit from that.

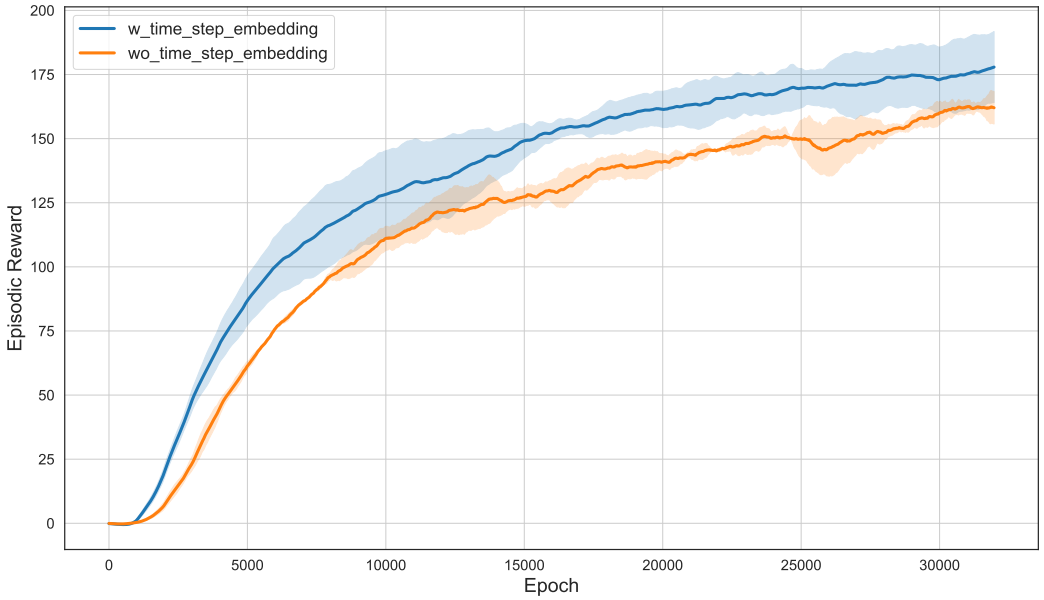

Figure 9: Learning curves on the Isaaclab-Ant-v0 benchmark with or without time step sinusoidal positional embedding. Results are averaged over 5 runs. The x-axis denotes training epochs, and the y-axis shows average episodic return with one standard deviation shaded.

E Proof for Lemma 1

Lemma 1. ***(Change of Variables [3])*** *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ is an invertible and smooth mapping. If we have the random variable $X \sim q(x)$ and the random variable $Y = f(X)$ transformed by function f, the distribution $p(y)$ of Y is*

$$p(y) = q(x) \left| \det \frac{\partial f}{\partial x} \right|^{-1}. \tag{15}$$

Proof. Here, we take the one-dimensional random variables X, Y as an example. For a general n-dimensional case, we can obtain the same result by applying multivariable calculus.

Firstly, we can divide f into two categories: (1) f is strictly increasing, (2) f is strictly decreasing, since f is an invertible and smooth mapping.

Strictly increasing. For strictly increasing f, the CDF of Y is

$$P(y) = P(Y \leq y) = P(f(X) \leq y) = P\left(X \leq f^{-1}(y)\right) = Q\left(f^{-1}(y)\right) = Q(x).$$

Then, according to $p(x) = \lim_{\epsilon \to 0} \frac{P(x+\epsilon) - P(x)}{\epsilon}$ and the chain rule, the PDF of the random variable Y is

$$p(y) = q(x)\frac{dx}{dy} = q(x)\left(\det \frac{\partial f}{\partial x}\right)^{-1}.$$

Strictly decreasing. The proof for f strictly decreasing is analogous. In that case, the PDF of Y is $p(y) = -q(x)\frac{dx}{dy}$ since $\frac{dx}{dy} < 0$. Hence, we need to add an absolute sign for the final formula, i.e., $p(y) = q(x)\left|\det \frac{\partial f}{\partial x}\right|^{-1}$. □

https://doi.org/10.52202/085713-4347

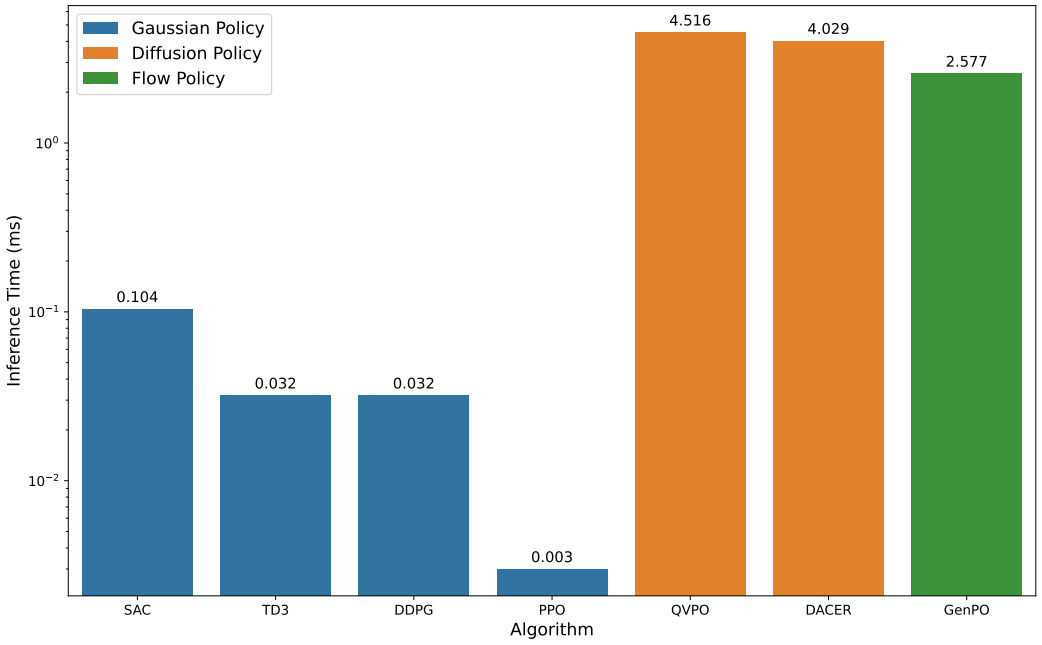

Figure 10: Inference time comparison on Issac-Ant-v0 environment.

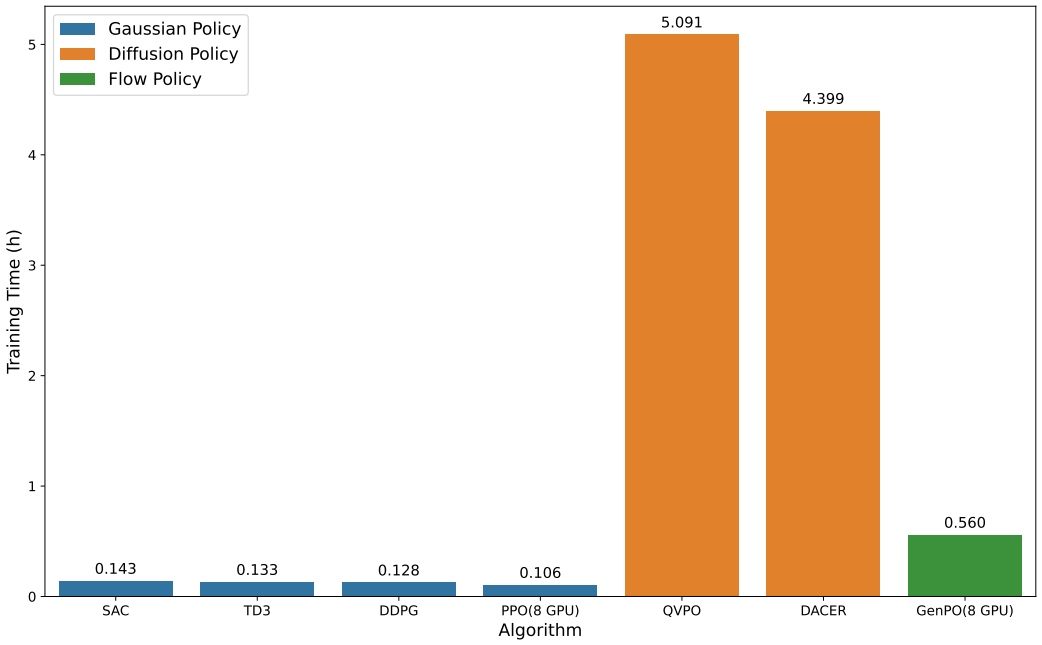

Figure 11: Training time comparison on Isaac-Ant-v0 environment.

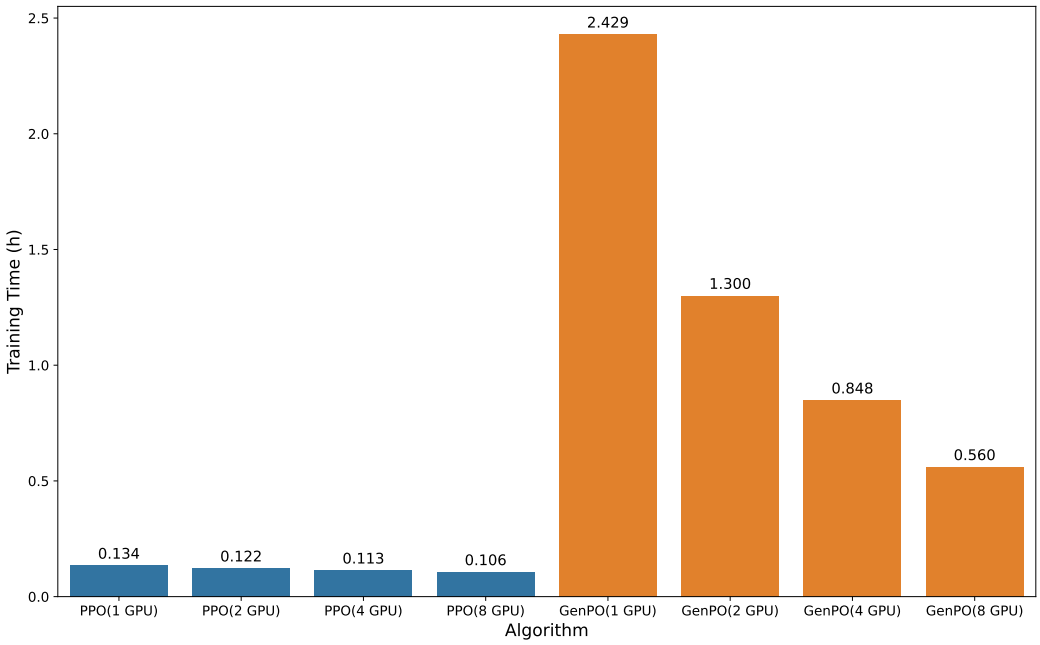

Figure 12: Comparison of training time between PPO and GenPO on the Isaac-And-v0 environment for scaling training on multiple GPUs.

https://doi.org/10.52202/085713-4347

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: As stated in the abstract and introduction, this paper bridges the gap between on-policy reinforcement learning and generative diffusion models, makes it possible to calculate the log-likelihood in diffusion policies, and finally enables large-scale parallel simulation environments to leverage the powerful multimodal capabilities of diffusion models. They can accurately reflect the contributions and scope of this paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The analysis of limitations is provided in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The proofs are provided in the supplementary materials.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: These details are provided in the supplementary materials.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

 https://doi.org/10.52202/085713-4347

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will release the code once the paper is accepted.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so âĂIJNoâĂİ is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: These details are provided in Section 5 and supplementary materials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The standard deviation in our experiment results (Figure 3, Figure 4 and Table 1) shows the statistical significance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The information on the computer resources is provided in the supplementary materials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research conformed with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our paper primarily focuses on theoretical research in how to employ the diffusion model in on-policy reinforcement learning, with no consideration of societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

https://doi.org/10.52202/085713-4347

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our experiment of this paper was conducted on 8 robot control tasks in IsaacLab, which poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The implementation of SAC, TD3, DDPG, PPO, DACER, and QVPO, and IsaacLab simulator are cited properly in the supplementary materials.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not introduce new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

 https://doi.org/10.52202/085713-4347

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We merely use the LLM for editing the paper.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.