Self-Adjoint Flow Policy Optimization

Guojian Zhan[1,2,†], Feihong Zhang[1,†], Xiangteng Zhang[1], Likun Wang[1], Letian Tao[1]
Tianyi Zhang[1], Wenxin Zhao[1], Yinuo Wang[4], Tianze Zhu[1], Jingliang Duan[3]
Yang Guan[1,*], Shengbo Eben Li[1,*]

[1] College of AI & School of Vehicle and Mobility, Tsinghua University
[2] ByteDance Seed
[3] School of Mechanical Engineering, University of Science & Technology Beijing
[4] 2030Lab, Yinwang Intelligent Technology

[†] Equal contribution. [*] Corresponding authors.

Abstract

Generative policies offer a promising alternative to Gaussian policies in reinforcement learning (RL) due to their ability to represent expressive and multimodal action distributions. However, integrating such policies into likelihood-based policy optimization remains challenging, since accurate and differential action likelihoods are intractable for standard generative policies. In this paper, we propose *Self-Adjoint Flow Policy Optimization* (SAFLOW), a likelihood-tractable generative policy with self-adjoint structure. SAFLOW parameterizes the policy as an invertible flow in a doubled action space and uses a Verlet-style self-adjoint composition for generation. This structure makes the inverse map obtainable by stepsize reversal, aligning forward sampling and backward likelihood evaluation under the same time-symmetric numerical rule. It also provides a second-order approximation accuracy to the learned continuous flow, improving the numerical consistency of likelihood and policy-ratio computation. We instantiate SAFLOW within an on-policy optimization framework and evaluate it on eight IsaacLab continuous-control tasks. SAFLOW consistently outperforms Gaussian-policy baselines and recent generative-policy methods, highlighting the value of self-adjoint numerical structure for likelihood-based generative policy optimization.

1 Introduction

Expressive stochastic policies are crucial for reinforcement learning (RL) in high-dimensional continuous-control problems. While standard Gaussian policies remain widely used because they provide tractable likelihoods and admit simple reparameterized sampling [1, 2, 3], their unimodal form can be restrictive in settings involving multimodal exploration, contact-rich manipulation, locomotion, or coordinated whole-body control [4, 5]. Generative policies, including diffusion and flow-based policies, offer a more flexible alternative by representing complex action distributions through learned state-conditioned generative dynamics [6, 7, 8].

Despite this expressiveness, deploying generative policies in likelihood-based policy optimization remains challenging because they often lack tractable and numerically reliable likelihood evaluation [9]. In particular, the finite-step discretized generation process is typically not an explicitly invertible map with a tractable density. As a result, the forward sampling procedure and the backward likelihood computation may become inconsistent, leading to unstable likelihood ratios and unreliable policy-gradient updates.

*Corresponding authors: guany17@tsinghua.org.cn; lishbo@tsinghua.edu.cn.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Recent work has explored several ways to make expressive generative policies compatible with policy-gradient optimization. FPO incorporates flow matching [10] into a PPO-style policy-gradient framework by deriving an advantage-weighted surrogate ratio from the conditional flow-matching loss, thereby avoiding explicit likelihood computation while retaining the expressiveness of flow-based policy parameterizations [9]. GenPO instead directly addresses likelihood tractability by introducing a doubled action space and an alternating update rule, making the discretized generative process explicitly invertible and enabling tractable change-of-variables likelihood evaluation [11]. However, this alternating update is first-order and asymmetric: although an inverse map can be computed explicitly, it is not, in general, obtained by applying the same update rule with the opposite stepsize. Consequently, forward sampling and backward likelihood evaluation are not governed by a single time-symmetric numerical scheme.

We argue that this numerical structure matters for likelihood-based generative policy optimization [12]. A generative policy is not merely a sampler; it also defines the likelihoods used to compute policy gradients, entropy estimates, KL penalties, and PPO-style likelihood ratios [13]. Exact invertibility ensures that a density can be evaluated, but it does not ensure that the forward sampler and backward likelihood evaluator are numerically aligned. If these two computations are induced by asymmetric discretizations, the resulting likelihood ratios may inherit discretization artifacts. We therefore propose to strengthen invertibility into self-adjoint numerical consistency: the inverse should be obtained by the same update rule run backward in time, and the discrete generative map should accurately approximate the learned continuous flow.

In this work, we propose *Self-Adjoint Flow Policy Optimization* (SAFLOW), a generative policy optimization framework built around a self-adjoint flow policy. SAFLOW operates in a doubled action space and replaces first-order alternating updates with a Verlet-style self-adjoint composition. As shown in Figure 1, this design preserves exact numerical inversion and makes forward and backward computations time-symmetric: $(\Psi_h^\theta)^{-1} = \Psi_{-h}^\theta$. Consequently, sampling and likelihood evaluation are implemented by the same update rule in opposite time directions. Moreover, the symmetric composition provides a second-order approximation to the learned continuous flow: $\left\|(\Psi_h^\theta)^K(z_0) - \varphi_T^\theta(z_0)\right\| = \mathcal{O}(h^2), \quad T = Kh = 1$, where φ_T^θ denotes the exact time-T flow induced by the learned continuous velocity field on the variable z.

Our contributions are as follows.

- We propose SAFLOW, a likelihood-tractable generative policy optimization framework that strengthens exact invertibility into self-adjoint numerical consistency. By operating in a doubled action space and using a Verlet-style kick–drift–kick composition, SAFLOW implements forward sampling and backward likelihood evaluation with the same update rule in opposite time directions.
- We prove that the proposed self-adjoint flow admits exact inversion by stepsize reversal and provides a second-order approximation to the learned continuous flow. We further show that this second-order consistency propagates to log-likelihoods, policy ratios, and PPO-style clipped surrogate objectives.
- We evaluate SAFLOW on eight IsaacLab continuous-control tasks spanning locomotion, manipulation, dexterous control, and aerial control. Across these diverse domains, SAFLOW consistently outperforms Gaussian-policy baselines and recent generative-policy methods, achieving higher final returns and more stable learning.

2 Preliminaries

Likelihood-Based Policy Optimization in RL. We consider a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s,a)$ is the transition kernel, $r(s,a)$ is the reward function, ρ_0 is the initial state distribution, and $\gamma \in [0,1)$ is the discount factor [14]. A stochastic policy $\pi_\theta(a|s)$ induces a trajectory distribution over $\tau = (s_0, a_0, s_1, a_1, \ldots)$, and the standard RL objective is $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. Many policy optimization algorithms rely on evaluating the action likelihood $\pi_\theta(a|s)$ or its logarithm for encouraging exploration or computing policy gradients. For example, policy-gradient methods with importance sampling use likelihood ratios of the form $r_\theta(s,a) = \frac{\pi_\theta(a|s)}{\mu(a|s)}$, where μ denotes the behavior policy that generated

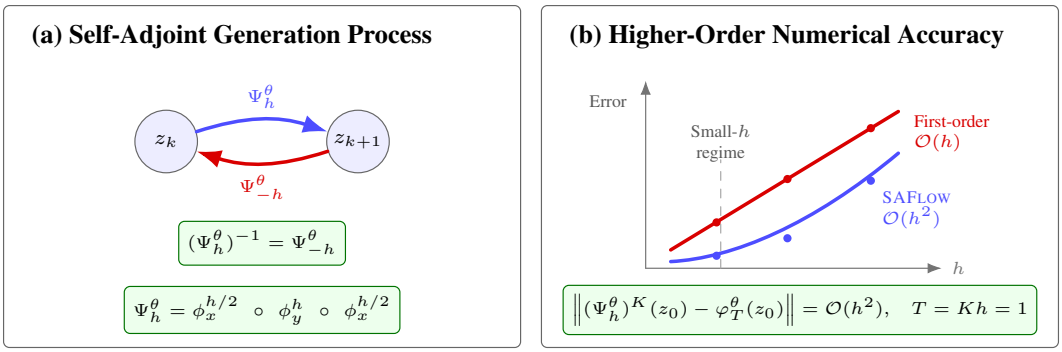


Figure 1: Key properties of the proposed SAFLOW. **Left:** the numerical update is exactly invertible by reversing the stepsize, so forward sampling and backward likelihood evaluation are implemented by the same rule in opposite time directions. **Right:** SAFLOW achieves second-order accuracy, leading to significantly smaller discretization error in the practically relevant small-stepsize regime.

the data [13, 2]. In on-policy PPO-style optimization, the clipped surrogate objective is

$$\mathcal{L}_{\text{clip}}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right], \tag{1}$$

where $\hat{A}_t$ is advantage estimate and $\epsilon > 0$ is the clipping threshold. Action log-likelihoods also appear in KL-based trust-region constraints and entropy-regularized objectives [2, 15]. Overall, accurate and differentiable likelihood evaluation is a fundamental requirement for a broad class of RL algorithms.

Generative Policies for RL. A generative policy samples an action by transforming a simple base random variable. Let $z_0 \sim p_0(z)$ be a base noise variable, usually a standard Gaussian. A state-conditioned generative policy defines a transformation $z_K = F_\theta(z_0; s)$, where $F_\theta(\cdot; s)$ may be implemented by a diffusion model [16] or flow matching velocity [4, 17]. The generated action is obtained as z_K. When $F_\theta(\cdot; s)$ is invertible, the density of the generated variable can be evaluated by the change-of-variables formula:

$$\log \pi_\theta(z_K|s) = \log p_0(z_0) - \log\left|\det \frac{\partial F_\theta(z_0; s)}{\partial z_0}\right|. \tag{2}$$

This formula is the basis of change of variables. However, standard diffusion/flow policies are not directly likelihood-tractable under finite-step discretization, because the sampling process is generally not written as an exactly invertible map with a tractable Jacobian [11]. This creates a gap between the expressiveness of diffusion/flow policies and the likelihood requirements of policy optimization.

3 Method

3.1 SAFLOW: Self-Adjoint Flow Policy

SAFLOW defines a state-conditioned invertible policy in an augmented action space. Let the environment action space be $\mathcal{A} \subseteq \mathbb{R}^d$. We adopt a doubled latent action $z = (x, y) \in \mathbb{R}^{2d}$, which provides sufficient degrees of freedom for invertible density modeling. Specifically, given a state s, SAFLOW samples a base variable $z_0 = (x_0, y_0) \sim p_0$, where p_0 is a standard Gaussian, and transform it with a K-step state-conditioned flow $z_K = F_\theta(z_0; s)$, $z_K = (x_K, y_K)$. The dummy action is $\tilde{a} = z_K$, and the action executed in the environment is $a = \mathcal{M}(z_K) = (x_K + y_K)/2$. The resulting stochastic policy is therefore induced by the invertible map F_θ in the augmented space.

The core design of SAFLOW is the construction of F_θ. Let $z_k = (x_k, y_k)$ denote the augmented variable at step k, and let $v_\theta(\cdot, t; s)$ be a state-conditioned velocity field. Rather than using a one-sided update, SAFLOW advances the two dummy-action components through a symmetric Verlet-style

kick–drift–kick technique:

$$\begin{aligned} \phi_x^{h/2}: \quad & x_{k+\frac{1}{2}} = x_k + \frac{h}{2} v_\theta(y_k, t_k; s), \\ \phi_y^{h}: \quad & y_{k+1} = y_k + h\, v_\theta\left(x_{k+\frac{1}{2}}, t_k + \frac{h}{2}; s\right), \\ \phi_x^{h/2}: \quad & x_{k+1} = x_{k+\frac{1}{2}} + \frac{h}{2} v_\theta(y_{k+1}, t_{k+1}; s). \end{aligned} \tag{3}$$

We denote the resulting one-step map by $z_{k+1} = \Psi_h^\theta(z_k; s)$. Equivalently, the step can be written as the symmetric composition $\Psi_h^\theta = \phi_x^{h/2} \circ \phi_y^h \circ \phi_x^{h/2}$, where ϕ_x and ϕ_y denote partial shear updates of the two dummy-action components. The full generative policy flow is the composition of K such self-adjoint steps, $F_\theta(\cdot; s) = (\Psi_h^\theta)^K(\cdot; s)$.

This construction has two immediate benefits. First, each partial update is explicitly invertible, so the whole generative policy admits tractable backward evaluation. Second, because the composition is symmetric, the inverse step is obtained by applying the same rule with the opposite stepsize. Thus, sampling and likelihood evaluation are aligned by the same numerical scheme in opposite time directions. In this sense, SAFLOW upgrades invertible generative policies from merely having an inverse map to having a time-symmetric self-adjoint flow.

We next describe likelihood evaluation. Given a dummy action z_K, we recover its base variable by self-adjoint inversion: $z_0 = F_\theta^{-1}(z_K; s) = (\Psi_{-h}^\theta)^K(z_K; s)$. By the change-of-variables formula,

$$\log \pi_\theta(z_K|s) = \log p_0(z_0) - \sum_{k=0}^{K-1} \log \left|\det D\Psi_h^\theta(z_k; s)\right|. \tag{4}$$

The likelihood depends on θ through the inverse map F_θ^{-1}, which determines the base point assigned to the observed dummy action. This enables tractable likelihood ratios for on-policy optimization while preserving an expressive state-conditioned generative policy.

3.2 SAFLOW Policy Optimization in the Augmented MDP

SAFLOW can be interpreted as performing policy optimization in an augmented MDP $\widetilde{\mathcal{M}} = (\mathcal{S}, \widetilde{\mathcal{A}}, \widetilde{P}, \widetilde{r}, \rho_0, \gamma)$, where $\widetilde{P}(s'|s, \widetilde{a}) = P(s'|s, M(\widetilde{a})), \widetilde{r}(s, \widetilde{a}) = r(s, M(\widetilde{a}))$. This formulation makes the likelihood ratio used by PPO well-defined in the augmented action space: $r_\theta(s, \widetilde{a}) = \widetilde{\pi}_\theta(\widetilde{a}|s)/\widetilde{\pi}_{\theta_{\text{old}}}(\widetilde{a}|s)$. Thus, SAFLOW does not require the marginal density of the projected action $a = M(\widetilde{a})$. Instead, it performs standard likelihood-ratio policy optimization in the augmented MDP, whose rewards and transitions depend on $\widetilde{a}$ only through $M(\widetilde{a})$.

Given rollout data generated by the old policy $\widetilde{\pi}_{\theta_{\text{old}}}$, SAFLOW optimizes the clipped PPO surrogate

$$L_{\text{clip}}(\theta) = \mathbb{E}\left[\min\left(r_\theta(s_t, \widetilde{a}_t)\widehat{A}_t, \text{clip}\big(r_\theta(s_t, \widetilde{a}_t), 1-\epsilon, 1+\epsilon\big)\widehat{A}_t\right)\right], \tag{5}$$

where $\widehat{A}_t$ is estimated from the actual rewards obtained after executing $M(\widetilde{a}_t)$.

The doubled action space introduces redundant degrees of freedom because many dummy actions induce the same environment action. To regularize this projection null space, SAFLOW adds a compression penalty

$$L_{\text{comp}}(\theta) = \mathbb{E}_{\widetilde{a}=(x,y)\sim\widetilde{\pi}_\theta(\cdot|s)}\left[\|x - y\|_2^2\right]. \tag{6}$$

The final actor objective is

$$L_{\text{policy}}(\theta) = L_{\text{clip}}(\theta) - \nu L_{\text{comp}}(\theta), \tag{7}$$

where $\nu > 0$ controls the compression strength. The compression term is not required for the correctness of the augmented policy-gradient objective; rather, it selects compact representatives among equivalent dummy-action lifts. The complete pseudocode is provided in Algorithm 1.

3.3 Theoretical Analysis

We analyze SAFLOW from two complementary perspectives. First, we show that optimizing dummy-action likelihoods is principled because the doubled action space defines an augmented MDP that preserves the original optimal control problem. Second, we show that the self-adjoint numerical structure yields exact inversion and second-order consistency of likelihood ratios and PPO surrogates.

Algorithm 1 SAFLOW Policy Optimization

Require: policy $\widetilde{\pi}_\theta$, critic V_ψ, flow steps K, step size $h = 1/K$, compression coefficient ν

for each training iteration **do**
  Set $\theta_{\text{old}} \leftarrow \theta$ and collect buffer $\mathcal{B}$
  for each environment step **do**
    Sample $z_0 = (x_0, y_0) \sim p_0$ and generate $\widetilde{a} = (x, y) = (\Psi_h^{\theta_{\text{old}}})^K(z_0|s)$
    Compute and store old log-prob $\ell_{\text{old}} = \log \widetilde{\pi}_{\theta_{\text{old}}}(\widetilde{a}|s)$
    Execute $a = M(\widetilde{a}) = (x + y)/2$ and store $(s, \widetilde{a}, r, s', \ell_{\text{old}})$
  end for
  for each minibatch **do**
    Evaluate $\log \widetilde{\pi}_\theta(\widetilde{a}|s)$ by (4)
    Update policy $\widetilde{\pi}_\theta$ by (7)
    Update critic V_ψ by standard TD error
  end for
end for

3.3.1 Principled Optimization in the Augmented Action Space

Theorem 1 (Optimality Preservation under Augmented Actions)**.** *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$ be the original MDP. Define the augmented action space $\widetilde{\mathcal{A}} = \mathcal{A} \times \mathcal{A}$ and the projection $M(x, y) = (x + y)/2$. Let $\widetilde{\mathcal{M}} = (\mathcal{S}, \widetilde{\mathcal{A}}, \widetilde{P}, \widetilde{r}, \rho_0, \gamma)$ be the augmented MDP. Then optimizing over augmented-action policies does not change the optimal control value:* $\sup_{\widetilde{\pi}} J_{\widetilde{\mathcal{M}}}(\widetilde{\pi}) = \sup_\pi J_{\mathcal{M}}(\pi)$.

Proof sketch. For any augmented policy $\widetilde{\pi}$, the projected policy $M_\#\widetilde{\pi}$ induces the same distribution over environment states and executed actions, since both $\widetilde{P}$ and $\widetilde{r}$ depend on $\widetilde{a}$ only through $M(\widetilde{a})$. Therefore $J_{\widetilde{\mathcal{M}}}(\widetilde{\pi}) = J_{\mathcal{M}}(M_\#\widetilde{\pi})$. Conversely, any original policy π can be lifted to the diagonal policy $\widetilde{\pi}^\Delta$ that samples $a \sim \pi(\cdot|s)$ and sets $(x, y) = (a, a)$. This lift satisfies $J_{\widetilde{\mathcal{M}}}(\widetilde{\pi}^\Delta) = J_{\mathcal{M}}(\pi)$. Combining both directions proves the result. See Appendix A.1 for more details. □

Theorem 2 (Policy Gradient in the Augmented Action Space)**.** *Assume that $\widetilde{\pi}_\theta(\widetilde{a}|s)$ is differentiable in θ and has support independent of θ. Then the policy gradient in the augmented MDP satisfies*

$$\nabla_\theta J_{\widetilde{\mathcal{M}}}(\theta) = \mathbb{E}_{(s,\widetilde{a})\sim d^{\widetilde{\pi}_\theta}}\left[\nabla_\theta \log \widetilde{\pi}_\theta(\widetilde{a}|s) A^{\widetilde{\pi}_\theta}(s, \widetilde{a})\right],$$

where $d^{\widetilde{\pi}_\theta}$ denotes the discounted state–action occupancy measure in $\widetilde{\mathcal{M}}$. Consequently, for samples collected by $\widetilde{\pi}_{\theta_{\text{old}}}$, the corresponding one-step importance ratio is $r_\theta(s, \widetilde{a}) = \widetilde{\pi}_\theta(\widetilde{a}|s)/\widetilde{\pi}_{\theta_{\text{old}}}(\widetilde{a}|s)$, which is the likelihood ratio used in PPO-style optimization over the augmented action space.

Proof sketch. The result follows directly from the standard policy-gradient theorem applied to the augmented MDP $\widetilde{\mathcal{M}}$. In this MDP, the policy acts on the dummy action $\widetilde{a}$, and the corresponding policy density is $\widetilde{\pi}_\theta(\widetilde{a}|s)$. Therefore, the score term in the policy gradient is $\nabla_\theta \log \widetilde{\pi}_\theta(\widetilde{a}|s)$, yielding the stated gradient expression. See Appendix A.2 for the full derivation. □

3.3.2 Numerical Advantages of Self-Adjoint Flow

Theorem 3 (Exact Time-Symmetric Inversion of the Self-Adjoint Flow)**.** *Let Ψ_h^θ denote one* SAFLOW *integration step with stepsize h, defined by the symmetric composition $\Psi_h^\theta = \phi_x^{h/2} \circ \phi_y^h \circ \phi_x^{h/2}$. Then the inverse of one step is obtained by applying the same update with the opposite stepsize: $(\Psi_h^\theta)^{-1} = \Psi_{-h}^\theta$. Consequently, for the K-step generative map $F_{\theta,h} = (\Psi_h^\theta)^K$, the inverse map used for likelihood evaluation is $F_{\theta,h}^{-1} = (\Psi_{-h}^\theta)^K$.*

Proof sketch. The result follows from the symmetric, self-adjoint structure of the update. Each split map is exactly inverted by negating its stepsize: $(\phi_x^{h/2})^{-1} = \phi_x^{-h/2}, (\phi_y^h)^{-1} = \phi_y^{-h}$. Using the inverse rule for compositions, we obtain $(\Psi_h^\theta)^{-1} = \Psi_{-h}^\theta$. Applying this one-step inverse repeatedly yields $F_{\theta,h}^{-1} = (\Psi_{-h}^\theta)^K$. See Appendix A.3 for more details. □

Theorem 4 (Second-order approximation in C^1)**.** *Assume that the split vector fields are sufficiently smooth on a compact domain and that both the exact and numerical trajectories remain in this domain over the time horizon T. Then the* SAFLOW *discretization satisfies* $\sup_{z_0} \left\| F_{\theta,h}(z_0) - \varphi_T^\theta(z_0) \right\| = \mathcal{O}(h^2)$*. If, in addition, the vector fields have one higher order of smoothness, then the Jacobians of the numerical and exact flows satisfy the same second-order bound:* $\sup_{z_0} \left\| DF_{\theta,h}(z_0) - D\varphi_T^\theta(z_0) \right\| = \mathcal{O}(h^2)$*. Together, these two estimates imply,* $\|F_{\theta,h} - \varphi_T^\theta\|_{C^1} = \mathcal{O}(h^2)$.

Proof sketch. The SAFLOW step is a symmetric Strang splitting $\Psi_h^\theta = \phi_x^{h/2} \circ \phi_y^h \circ \phi_x^{h/2}$. By the Baker–Campbell–Hausdorff expansion [18], the leading asymmetric error terms cancel, giving local error $\mathcal{O}(h^3)$; over $K = T/h$ steps this yields global error $\mathcal{O}(h^2)$. Applying the same argument to the variational dynamics gives the Jacobian error $\mathcal{O}(h^2)$, hence the C^1 bound. See Appendix A.4 for more details. □

Theorem 5 (Second-order likelihood consistency)**.** *Let φ_T^θ be the exact continuous flow corresponding to the split dynamics, and with $\|F_{\theta,h} - \varphi_T^\theta\|_{C^1} = \mathcal{O}(h^2)$. Consider that the base density p_0 is positive and continuously differentiable on the relevant compact domain, and that the Jacobian determinant of φ_T^θ is bounded away from zero. Let $\ell_{\theta,h}(s,z)$ be the log-likelihood induced by the discretized flow. Let $\ell_{\theta,\mathrm{cont}}(s,z)$ be the corresponding log-likelihood under the exact continuous flow. Then $|\ell_{\theta,h}(s,z) - \ell_{\theta,\mathrm{cont}}(s,z)| = \mathcal{O}(h^2)$ uniformly holds on the generated compact domain.*

Proof sketch. The C^1 closeness of $F_{\theta,h}$ and φ_T^θ, together with nondegeneracy of the exact flow, implies C^1 closeness of their inverse maps. The base-density term is then second-order accurate by Lipschitz continuity of $\log p_0$, and the log-determinant term is second-order accurate because $A \mapsto \log|\det A|$ is locally Lipschitz on nonsingular matrices. Combining the two terms gives the result. See Appendix A.5 for more details. □

Corollary 6 (Second-order consistency of policy-ratio and PPO surrogate)**.** *Let $r_{\theta,h}(s,z) = \exp\left(\ell_{\theta,h}(s,z) - \ell_{\theta_{\mathrm{old}},h}(s,z)\right)$ be the* SAFLOW *policy ratio, and let $r_{\theta,\mathrm{cont}}(s,z)$ be the corresponding exact-flow ratio. With Theorem 5 for both θ and θ_{old}, $r_{\theta,h}(s,z) = r_{\theta,\mathrm{cont}}(s,z)\exp(\mathcal{O}(h^2))$. If the exact ratio is bounded and $|\widehat{A}_t| \le A_{\max}$, then the clipped PPO surrogate computed with the discretized likelihood satisfies $|L_{\mathrm{clip},h}(\theta) - L_{\mathrm{clip},\mathrm{cont}}(\theta)| = \mathcal{O}(h^2)$.*

Proof sketch. The ratio result follows by subtracting the second-order log-likelihood errors for θ and θ_{old} and exponentiating. The PPO result follows because the clipped surrogate is Lipschitz in the policy ratio with constant at most $|\widehat{A}_t|$ for each sample. See Appendix A.6 for more details. □

Together, these results show that SAFLOW is principled at the RL formulation level and consistent at the numerical likelihood level: the doubled action space defines an optimality preserved augmented control problem and valid dummy-action likelihood ratios; the self-adjoint flow yields exact time-symmetric inversion structure, second-order approximation accuracy of learned continuous-time flow, likelihood, policy-ratio and PPO surrogates.

4 Experiments

Benchmarks. We assess the performance of our approach using a collection of benchmark environments from IsaacLab [19]. Specifically, we evaluate on eight tasks: Ant, Humanoid, Franka Arm, Shadow Hand, Anymal-D, Unitree-Go2, Unitree-H1, and Quadcopter, detailed in Appendix B.1.

Baselines. We benchmark SAFLOW against representative policy optimization algorithms. The comparison includes widely used Gaussian-policy methods: PPO [13] and TRPO [2], together with generative-policy approaches: FPO [9], GenPO [11]. To improve the reliability of the reported results, each experiment is carried out with five different random seeds. Additional experimental configurations and implementation details are provided in the Appendix B.2 and B.3.

Main Results. Figure 2 shows the learning curves of all methods, where SAFLOW generally learns faster and reaches higher final returns than the baselines. As summarized in Table 1, SAFLOW achieves the best mean return across all eight benchmark tasks, demonstrating consistent improvements over Gaussian-policy, and generative-policy baselines. The gains are especially pronounced

on challenging Humanoid, Unitree locomotion and Franka Arm, Shadow Hand manipulation tasks, where high-dimensional control and complex contact dynamics make optimization more difficult. These results suggest that SAFLOW improves both sample efficiency and training stability across diverse IsaacLab environments.

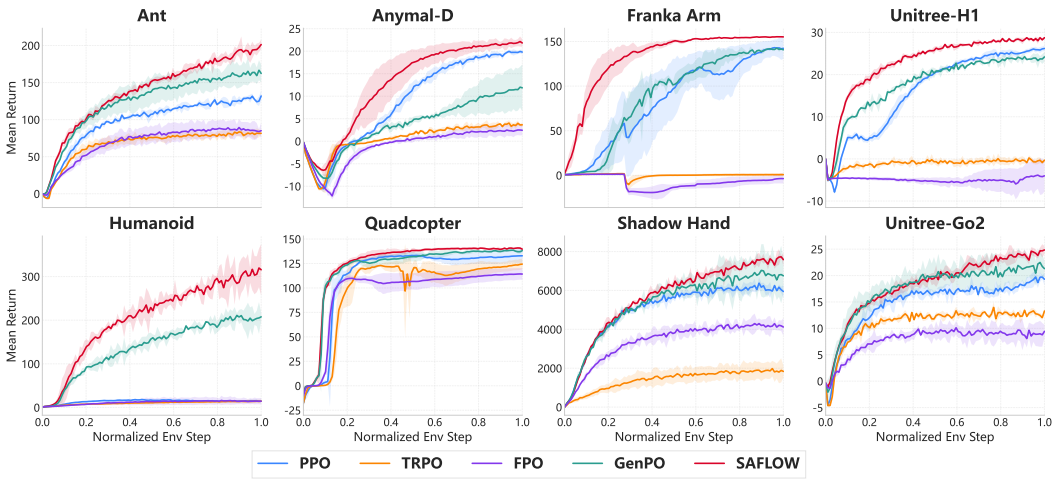


Figure 2: Learning curves across 8 IsaacLab tasks. Results are averaged over 5 runs. The x-axis denotes normalized env steps, and the y-axis shows average episodic return.

Table 1: Numerical results across 8 IsaacLab tasks.

Environment	PPO	TRPO	FPO	GenPO	SAFLOW
Ant	134.48 $\pm$ 1.72	83.98 $\pm$ 2.51	89.98 $\pm$ 11.54	168.92 $\pm$ 17.74	**201.34 $\pm$ 5.63**
Humanoid	18.91 $\pm$ 0.09	14.83 $\pm$ 2.70	16.59 $\pm$ 9.57	218.44 $\pm$ 8.23	**329.60 $\pm$ 21.72**
Franka Arm	143.53 $\pm$ 7.60	1.87 $\pm$ 0.98	1.22 $\pm$ 0.39	143.25 $\pm$ 1.41	**155.57 $\pm$ 0.03**
Anymal-D	20.13 $\pm$ 0.26	4.09 $\pm$ 0.11	2.70 $\pm$ 0.98	12.02 $\pm$ 4.61	**22.15 $\pm$ 1.00**
Unitree-Go2	20.44 $\pm$ 1.24	14.13 $\pm$ 0.70	10.56 $\pm$ 1.80	23.15 $\pm$ 2.57	**25.51 $\pm$ 1.49**
Unitree-H1	26.36 $\pm$ 0.42	0.65 $\pm$ 0.23	-0.02 $\pm$ 0.01	24.85 $\pm$ 0.45	**29.21 $\pm$ 0.16**
Quadcopter	133.39 $\pm$ 2.43	124.45 $\pm$ 4.61	114.44 $\pm$ 6.10	138.76 $\pm$ 1.08	**140.93 $\pm$ 0.99**
Shadow Hand	6430.09 $\pm$ 56.78	2035.10 $\pm$ 424.20	4427.83 $\pm$ 335.09	7188.21 $\pm$ 996.65	**7984.18 $\pm$ 161.96**

Scaling with Parallel Environments. We evaluate the scalability of SAFLOW on Ant by varying the number of GPU-parallel environments from 128 to 4096, with 2048 used by default in our main experiments. For clarity, Figure 3 reports the learning curves of three representative methods, PPO, GenPO, and SAFLOW; the complete set of curves is provided in Appendix C.5. SAFLOW outperforms these baselines when the number of parallel environments is relatively large (≥ 1024), and its performance further improves as more environments are used. These results suggest that SAFLOW effectively benefits from larger on-policy batches, where more diverse fresh rollouts reduce the variance of advantage and policy-ratio estimates within each update. This scaling behavior is enabled by the combination of an expressive generative policy, which exploits increased batch diversity, and a self-adjoint numerical structure, which provides stable and accurate likelihood evaluation for optimization.

Ablation Study. To further examine which parts of SAFLOW are responsible for its performance gains, we perform a set of ablation experiments using the Ant task.

(1) Flow Steps. Figure 4(a) shows the influence of the number of flow steps K. Increasing K slightly improves performance, as a finer discretization provides a more accurate approximation to the learned continuous flow. However, the overall differences across different choices of K are modest, and the learning curves and final returns remain similar. This indicates that SAFLOW is relatively robust to this hyperparameter. We use $K = 5$ as the default setting, which provides a good balance between performance and computational cost.

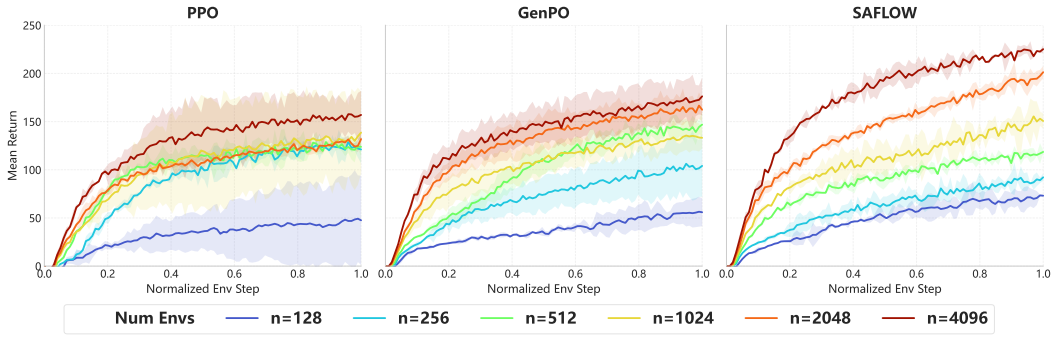


Figure 3: Learning curves of scaling experiment.

(2) Compress Coefficient. Figure 4(b) studies the effect of the compression coefficient ν. The results show that a small compression weight is beneficial, with $\nu = 0.01$ achieving the best performance. Larger values overly restrict the policy and significantly reduce the return, while removing the compression term also performs worse. This confirms the importance of mild compression.

(3) Time Embedding Dimension. During training, rather than using the denoising time step t directly as a raw input feature, we encode this temporal variable with sinusoidal positional embeddings to better capture temporal relationships and ordering, as detailed in Appendix B.2. Figure 4(c) evaluates different time embedding dimensions e_t. Larger embedding dimensions generally improve performance, while too small a dimension limits the representation ability of the policy. Among the tested settings, $e_t = 32$ achieves the strongest final return, and is therefore used as default setting.

(4) Time Embedding Hidden Size. Figure 4(d) compares different hidden sizes h_{emb} in the time embedding network. The results suggest that increasing the hidden size improves the expressiveness of the embedding module and leads to better performance. We choose $h_{\text{emb}} = 256$ as the default configuration, since it obtains the best final return among the tested values.

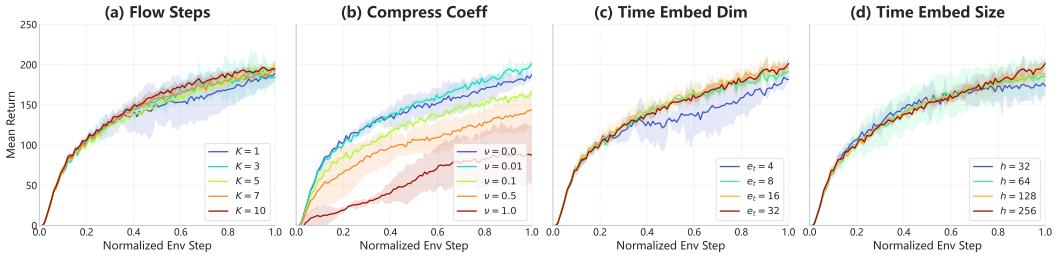


Figure 4: Learning curves of ablation study. Default setting: $K = 5, \nu = 0.01, e_t = 32, h_{\text{emb}} = 256$.

Additional Experiments. We conduct four additional experiments on Ant and Humanoid to further examine the efficiency, integration structure, and numerical behavior of SAFLOW.

(1) Training-time efficiency. Appendix C.1 reports wall-clock training time and performance-versus-time curves. Due to self-adjoint flow sampling and backward likelihood evaluation, SAFLOW requires approximately 32% and 23% more training time than GenPO on Ant and Humanoid, respectively. Despite this additional cost, its performance-versus-time curves remain competitive as shown in Figure 7, showing that the accuracy gains from the self-adjoint integration scheme are not solely a consequence of substantially increased computation.

(2) Controlled comparison of integration schemes. GenPO differs from SAFLOW not only in the numerical integration rule, but also in several auxiliary design choices, such as action mixing and customized entropy regularization. To isolate the effect of the integration scheme, Appendix C.2 provides a controlled comparison within our SAFLOW implementation, where we replace the self-adjoint update with the alternating integration scheme used by GenPO while keeping the remaining components fixed. The resulting comparison in Figure 8 shows that the self-adjoint Verlet-style

update provides consistent gains over the alternating scheme, supporting the role of time-symmetric integration in likelihood-based generative policy optimization.

(3) Likelihood stability during training. Appendix C.3 compares the standard deviation of policy likelihood ratios during training for GenPO and SAFLOW. SAFLOW exhibits smaller fluctuations as shown in Figure 9, suggesting that its self-adjoint discretization yields more stable and accurate forward–backward likelihood evaluation during policy updates.

(4) Toy validation of discretization and likelihood accuracy. Finally, Appendix C.4 evaluates both methods on a toy problem with an analytic solution. The results verify the expected second-order convergence of SAFLOW in both state discretization error and likelihood estimation error, complementing the empirical reinforcement-learning results with a controlled numerical validation.

5 Related Work

Expressive Generative Policies. Gaussian policies are widely used in policy-gradient algorithms because they provide tractable densities, efficient sampling, and stable likelihood-ratio estimation [2, 13]. However, their limited expressiveness can be restrictive in high-dimensional continuous-control tasks, especially when effective behavior requires multimodal exploration, contact-rich interaction, or coordinated whole-body motion [20, 21]. This limitation has motivated more expressive stochastic policy classes, including normalizing-flow policies, implicit push-forward policies, energy-based policies, and diffusion- or flow-based policies [22, 23, 24, 25, 26]. Most of these methods are developed in off-policy settings, where policy improvement is driven by learned value functions [21], entropy-regularized objectives [27], or value-weighted generative training [25], rather than exact on-policy likelihood ratios. While these works demonstrate the benefits of expressive generative actors, they do not directly address the likelihood-tractability and numerical-consistency requirements of likelihood-based on-policy optimization.

Likelihood-Based Generative Policy Optimization. On-policy methods such as TRPO and PPO rely on accurate action likelihoods to compute importance ratios and enforce trust-region-style updates [2, 13]. This creates a central challenge for generative policies: expressive diffusion or flow-based samplers do not automatically provide stable and differentiable likelihood evaluation [6]. Recent work addresses this challenge either by avoiding explicit likelihoods through surrogate ratios [6, 9, 28], or by constructing invertible generative updates in augmented action spaces [11]. These approaches narrow the gap between generative policies and likelihood-based RL, but exact invertibility alone does not ensure numerical consistency. In particular, if forward sampling and backward likelihood evaluation are based on asymmetric discretizations, the resulting likelihood ratios may inherit discretization artifacts. SAFLOW addresses this issue by introducing a self-adjoint flow in a doubled action space. Its update is inverted by applying the same rule with the opposite stepsize, aligning sampling and likelihood evaluation under a time-symmetric numerical scheme while providing a second-order approximation to the learned continuous flow.

6 Conclusion, Limitations and Future Work

We presented SAFLOW, a likelihood-tractable self-adjoint generative policy framework for on-policy reinforcement learning. SAFLOW defines an invertible flow in a doubled action space and uses a Verlet-style self-adjoint composition to make forward action sampling and backward likelihood evaluation consistent under stepsize reversal. This structure preserves exact numerical inversion while providing a second-order approximation to the underlying learned continuous flow. Across eight IsaacLab continuous-control tasks, SAFLOW consistently improves over Gaussian-policy baselines and recent generative-policy methods, highlighting the importance of numerically consistent likelihood evaluation for on-policy generative policy optimization.

We acknowledge that several limitations remain. First, SAFLOW uses a fixed number of flow steps across tasks. While this choice simplifies implementation and evaluation, different environments may benefit from different accuracy–efficiency trade-offs. Adaptive step allocation or early-stopping criteria for flow evaluation may further improve computational efficiency. Second, the compression coefficient is treated as a fixed hyperparameter. Although compression helps regularize redundant directions in the doubled action space, an adaptive schedule could reduce task-specific tuning and

better balance expressiveness with stability during training. Third, the current evaluation focuses on state-based simulated continuous-control benchmarks. Extending SAFLOW to dexterous manipulation, vision-based control, and real-world robotic systems would provide a broader assessment of its scalability and robustness. Future work will investigate adaptive numerical schemes for generative policy evaluation, automatic regularization schedules for augmented action spaces, and applications of self-adjoint likelihood-tractable policies to more complex settings.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 92582205), the Beijing Natural Science Foundation (Grant No. L257002), and the Tsinghua University (State Key Laboratory of Intelligent Green Vehicle and Mobility)–Xiaomi EV Technology Co., Ltd Joint Research Center for Autonomous Driving Safety and Intelligence. The authors declare no competing interests.

References

[1] Guojian Zhan, Likun Wang, Xiangteng Zhang, Jiaxin Gao, Masayoshi Tomizuka, and Shengbo Eben Li. Bootstrap off-policy with world model. *Advances in Neural Information Processing Systems*, 38:134093–134121, 2025.

[2] 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.

[3] 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.

[4] Guojian Zhan, Letian Tao, Pengcheng Wang, Yixiao Wang, Yuxin Chen, Yiheng Li, Hongyang Li, Masayoshi Tomizuka, and Shengbo Eben Li. Mean flow policy with instantaneous velocity constraint for one-step action generation. In *The Fourteenth International Conference on Learning Representations*, 2026.

[5] Guojian Zhan, Likun Wang, Pengcheng Wang, Feihong Zhang, Jingliang Duan, Masayoshi Tomizuka, and Shengbo Eben Li. Mind your entropy: From maximum entropy to trajectory entropy-constrained rl. *arXiv preprint arXiv:2511.11592*, 2025.

[6] 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.

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

[8] Guojian Zhan, Letian Tao, Pengcheng Wang, Yixiao Wang, Yuxin Chen, Yiheng Li, Hongyang Li, Masayoshi Tomizuka, and Shengbo Li. Mean flow policy with instantaneous velocity constraint for one-step action generation. In *International Conference on Learning Representations*, volume 2026, pages 157132–157154, 2026.

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

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

[11] Shutong Ding, Ke Hu, Shan Zhong, Haoyang Luo, Weinan Zhang, Jingya Wang, Jun Wang, and Ye Shi. Genpo: Generative diffusion models meet on-policy reinforcement learning. *arXiv preprint arXiv:2505.18763*, 2025.

[12] Christian Lubich and Gerhard Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer, 2006.

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

[14] Guojian Zhan, Xiangteng Zhang, Feihong Zhang, Letian Tao, and Shengbo Eben Li. Bicriteria policy optimization for high-accuracy reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

[15] Guojian Zhan, Likun Wang, Feihong Zhang, Yang Guan, and Shengbo Eben Li. Harmonized dual policy improvement for modelic reinforcement learning. In *Forty-third International Conference on Machine Learning*.

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

[17] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[18] Alexander Van-Brunt and Matt Visser. Explicit baker–campbell–hausdorff expansions. *Mathematics*, 6(8):135, 2018.

[19] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

[20] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

[21] Yinuo Wang, Likun Wang, Mining Tan, Wenjun Zou, Xujie Song, Wenxuan Wang, Tong Liu, Guojian Zhan, Tianze Zhu, Shiqi Liu, et al. Enhanced dacer algorithm with high diffusion efficiency. *arXiv preprint arXiv:2505.23426*, 2025.

[22] 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.

[23] Lingwei Peng, Hui Qian, Zhebang Shen, Chao Zhang, and Fei Li. Generative actor-critic: An off-policy algorithm using the push-forward model. *arXiv preprint arXiv:2105.03733*, 2021.

[24] 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. *Advances in Neural Information Processing Systems*, 37:56136–56165, 2024.

[25] Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. *arXiv preprint arXiv:2502.00361*, 2025.

[26] Lunjun Zhang, Shuo Han, Hanrui Lyu, and Bradly C Stadie. D2 actor critic: Diffusion actor meets distributional critic. *arXiv preprint arXiv:2510.03508*, 2025.

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

[28] Zechu Li, Rickmer Krohn, Tao Chen, Anurag Ajay, Pulkit Agrawal, and Georgia Chalvatzaki. Learning multimodal behaviors from scratch with diffusion policy gradient. *Advances in Neural Information Processing Systems*, 37:38456–38479, 2024.

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

A Theoretical Analysis

This appendix provides full proofs for the theoretical results used in the main text. We first prove that SAFLOW's doubled action space defines a valid augmented MDP whose optimal value matches that of the original MDP. We then prove that dummy-action likelihood ratios are valid policy-gradient ratios in the augmented MDP. Then, we prove numerical consistency results for the self-adjoint flow, including exact inversion, second-order likelihood consistency, and second-order perturbation bounds for policy ratios and PPO surrogates.

A.1 Optimality Preservation under Augmented Actions

Proof. We prove that the augmented MDP and the original MDP have the same optimal value, $\sup_{\widetilde{\pi}} J_{\widetilde{\mathcal{M}}}(\widetilde{\pi}) = \sup_{\pi} J_{\mathcal{M}}(\pi)$, by establishing the two inequalities separately:

First, fix any augmented policy $\widetilde{\pi}$. Define the original policy $\pi = M_{\#}\widetilde{\pi}$ by

$$\pi(B|s) = \widetilde{\pi}(M^{-1}(B)|s)$$

for every measurable set $B \subseteq \mathcal{A}$. Under $\widetilde{\pi}$ in the augmented MDP, the environment receives the projected action $a = M(\widetilde{a})$. Since

$$\widetilde{P}(s'|s,\widetilde{a}) = P(s'|s, M(\widetilde{a}))$$

and

$$\widetilde{r}(s,\widetilde{a}) = r(s, M(\widetilde{a})),$$

the joint distribution over state trajectories and projected actions under $\widetilde{\pi}$ in $\widetilde{\mathcal{M}}$ is identical to the joint distribution over state-action trajectories under π in $\mathcal{M}$. Therefore,

$$J_{\widetilde{\mathcal{M}}}(\widetilde{\pi}) = J_{\mathcal{M}}(M_{\#}\widetilde{\pi}) \leq \sup_{\pi} J_{\mathcal{M}}(\pi).$$

Taking the supremum over $\widetilde{\pi}$ gives

$$\sup_{\widetilde{\pi}} J_{\widetilde{\mathcal{M}}}(\widetilde{\pi}) \leq \sup_{\pi} J_{\mathcal{M}}(\pi).$$

Second, fix any original policy π. Define its diagonal lift $\widetilde{\pi}^{\Delta}$ by sampling $a \sim \pi(\cdot|s)$ and setting

$$\widetilde{a} = (x, y) = (a, a).$$

Equivalently,

$$\widetilde{\pi}^{\Delta}(dx, dy|s) = \int_{\mathcal{A}} \delta_a(dx)\delta_a(dy)\pi(da|s).$$

For this lifted policy,

$$M(x, y) = a$$

almost surely. Hence the state-action trajectory in $\widetilde{\mathcal{M}}$ under $\widetilde{\pi}^{\Delta}$ coincides with the trajectory in $\mathcal{M}$ under π, and

$$J_{\widetilde{\mathcal{M}}}(\widetilde{\pi}^{\Delta}) = J_{\mathcal{M}}(\pi).$$

Taking the supremum over π gives

$$\sup_{\widetilde{\pi}} J_{\widetilde{\mathcal{M}}}(\widetilde{\pi}) \geq \sup_{\pi} J_{\mathcal{M}}(\pi).$$

Combining both inequalities proves the claim. □

A.2 Policy Gradient in the Augmented Action Space

Proof. We apply the standard policy-gradient theorem to the augmented MDP

$$\widetilde{\mathcal{M}} = (\mathcal{S}, \widetilde{\mathcal{A}}, \widetilde{P}, \widetilde{r}, \rho_0, \gamma).$$

In this MDP, the action variable is the dummy action $\widetilde{a}$, and the policy density is $\widetilde{\pi}_\theta(\widetilde{a}|s)$. Hence, the score function appearing in the policy gradient is

$$\nabla_\theta \log \widetilde{\pi}_\theta(\widetilde{a}_t|s_t).$$

The reward in the augmented MDP is induced by the projected environment action:

$$\widetilde{r}(s_t, \widetilde{a}_t) = r(s_t, M(\widetilde{a}_t)).$$

Therefore, the corresponding advantage $A^{\widetilde{\pi}_\theta}(s_t, \widetilde{a}_t)$ is the advantage of taking the dummy action $\widetilde{a}_t$ in the augmented MDP, with its effect on the environment determined by $M(\widetilde{a}_t)$. The policy-gradient theorem then gives

$$\nabla_\theta J_{\widetilde{\mathcal{M}}}(\theta) = \mathbb{E}_{(s,\widetilde{a})\sim d^{\widetilde{\pi}_\theta}} \left[\nabla_\theta \log \widetilde{\pi}_\theta(\widetilde{a}|s) A^{\widetilde{\pi}_\theta}(s, \widetilde{a}) \right],$$

where $d^{\widetilde{\pi}_\theta}$ denotes the discounted state–action occupancy measure in $\widetilde{\mathcal{M}}$.

When rollout samples are collected from $\widetilde{\pi}_{\theta_{\text{old}}}$ and reused to update $\widetilde{\pi}_\theta$, the change of measure is also performed in the augmented action space. Thus, the corresponding one-step importance ratio is

$$r_\theta(s, \widetilde{a}) = \frac{\widetilde{\pi}_\theta(\widetilde{a}|s)}{\widetilde{\pi}_{\theta_{\text{old}}}(\widetilde{a}|s)},$$

which is exactly the likelihood ratio used by SAFLOW for PPO-style optimization. □

A.3 Exact Inversion of the Self-Adjoint Flow

Proof. Let $z = (x, y)$ denote the dummy action. SAFLOW uses a symmetric split update of the form

$$\Psi_h^\theta = \phi_x^{h/2} \circ \phi_y^h \circ \phi_x^{h/2},$$

where ϕ_x^h and ϕ_y^h are the two sub-updates associated with the split dynamics. The K-step map is

$$F_{\theta,h} = (\Psi_h^\theta)^K, \qquad T = Kh = 1.$$

Each split update is inverted by changing the sign of its stepsize:

$$(\phi_x^{h/2})^{-1} = \phi_x^{-h/2}, \qquad (\phi_y^h)^{-1} = \phi_y^{-h}.$$

Using the inverse rule for compositions,

$$(A \circ B \circ C)^{-1} = C^{-1} \circ B^{-1} \circ A^{-1},$$

we obtain

$$\begin{aligned}(\Psi_h^\theta)^{-1} &= \left(\phi_x^{h/2} \circ \phi_y^h \circ \phi_x^{h/2}\right)^{-1} \\ &= \phi_x^{-h/2} \circ \phi_y^{-h} \circ \phi_x^{-h/2} \\ &= \Psi_{-h}^\theta.\end{aligned}$$

Since

$$F_{\theta,h} = \underbrace{\Psi_h^\theta \circ \cdots \circ \Psi_h^\theta}_{K \text{ times}},$$

its inverse is obtained by composing the inverse one-step map K times:

$$F_{\theta,h}^{-1} = \underbrace{\Psi_{-h}^\theta \circ \cdots \circ \Psi_{-h}^\theta}_{K \text{ times}} = (\Psi_{-h}^\theta)^K.$$

□

A.4 Second-Order Approximation of the Flow

Proof. The SAFLOW one-step update is the symmetric Strang-type composition

$$\Psi_h^\theta = \phi_x^{h/2} \circ \phi_y^h \circ \phi_x^{h/2}.$$

For sufficiently smooth vector fields, the Baker–Campbell–Hausdorff expansion of symmetric splitting methods shows that even-order local error terms cancel. As a result, the local truncation error of one step is

$$\Psi_h^\theta(z) - \varphi_h^\theta(z) = \mathcal{O}(h^3),$$

uniformly on compact sets. Over a fixed horizon $T = Kh$, the number of steps is $K = T/h$, so accumulation of the local truncation error gives the global state error

$$\sup_{z_0} \left\| F_{\theta,h}(z_0) - \varphi_T^\theta(z_0) \right\| = K\mathcal{O}(h^3) = \mathcal{O}(h^2).$$

For the C^1 estimate, consider the variational equation of the exact flow and the corresponding linearized numerical flow. Under one additional derivative of smoothness, the same symmetric splitting argument applies to the variational dynamics. Thus the local Jacobian error is $\mathcal{O}(h^3)$, and accumulation over $K = T/h$ steps gives

$$\sup_{z_0} \left\| DF_{\theta,h}(z_0) - D\varphi_T^\theta(z_0) \right\| = \mathcal{O}(h^2).$$

Combining the state and Jacobian estimates yields the C^1 bound. □

A.5 Second-Order Likelihood Consistency

Proof. Let ρ_0 be the base density over the initial dummy action z_0. The discretized SAFLOW density is computed by change of variables:

$$\ell_{\theta,h}(s,z) = \log \widetilde{\pi}_{\theta,h}(z|s) = \log p_0(F_{\theta,h}^{-1}(z)) + \log \left| \det DF_{\theta,h}^{-1}(z) \right|.$$

Similarly, the exact continuous-flow likelihood is

$$\ell_{\theta,\mathrm{cont}}(s,z) = \log \rho_0((\varphi_T^\theta)^{-1}(z)) + \log \left| \det D(\varphi_T^\theta)^{-1}(z) \right|.$$

Because φ_T^θ is a diffeomorphism with Jacobian determinant bounded away from zero, the inverse map is locally stable under C^1 perturbations. Since

$$\|F_{\theta,h} - \varphi_T^\theta\|_{C^1} = \mathcal{O}(h^2),$$

we have

$$\|F_{\theta,h}^{-1} - (\varphi_T^\theta)^{-1}\|_{C^1} = \mathcal{O}(h^2)$$

on the generated compact domain.

We decompose the log-likelihood error into the base-density term and the log-determinant term:

$$\begin{aligned}
&|\ell_{\theta,h}(s,z) - \ell_{\theta,\mathrm{cont}}(s,z)| \\
&\le \left| \log p_0(F_{\theta,h}^{-1}(z)) - \log \rho_0((\varphi_T^\theta)^{-1}(z)) \right| \\
&\quad + \left| \log \left| \det DF_{\theta,h}^{-1}(z) \right| - \log \left| \det D(\varphi_T^\theta)^{-1}(z) \right| \right|.
\end{aligned}$$

For the first term, since ρ_0 is positive and continuously differentiable on a compact set, $\log \rho_0$ is Lipschitz on that set. Therefore,

$$\left| \log p_0(F_{\theta,h}^{-1}(z)) - \log \rho_0((\varphi_T^\theta)^{-1}(z)) \right| = \mathcal{O}(h^2).$$

For the second term, the C^1 inverse closeness gives

$$DF_{\theta,h}^{-1}(z) = D(\varphi_T^\theta)^{-1}(z) + \mathcal{O}(h^2).$$

Because the determinant of $D(\varphi_T^\theta)^{-1}(z)$ is bounded away from zero, the map

$$A \mapsto \log |\det A|$$

is Lipschitz in a neighborhood of this Jacobian matrix. Hence

$$\left| \log \left| \det DF_{\theta,h}^{-1}(z) \right| - \log \left| \det D(\varphi_T^\theta)^{-1}(z) \right| \right| = \mathcal{O}(h^2).$$

Combining the two estimates proves the result. □

A.6 Policy-Ratio and PPO-Surrogate Consistency

Proof. By Theorem 5 and its proof in Appendix A.5,

$$\ell_{\theta,h}(s,z) = \ell_{\theta,\mathrm{cont}}(s,z) + \mathcal{O}(h^2),$$

and

$$\ell_{\theta_{\mathrm{old}},h}(s,z) = \ell_{\theta_{\mathrm{old}},\mathrm{cont}}(s,z) + \mathcal{O}(h^2).$$

Subtracting the two identities gives

$$\ell_{\theta,h}(s,z) - \ell_{\theta_{\mathrm{old}},h}(s,z) = \ell_{\theta,\mathrm{cont}}(s,z) - \ell_{\theta_{\mathrm{old}},\mathrm{cont}}(s,z) + \mathcal{O}(h^2).$$

Exponentiating both sides yields

$$r_{\theta,h}(s,z) = r_{\theta,\mathrm{cont}}(s,z)\exp(\mathcal{O}(h^2)).$$

If $r_{\theta,\mathrm{cont}}$ is bounded, then

$$\exp(\mathcal{O}(h^2)) - 1 = \mathcal{O}(h^2),$$

which implies the additive bound.

For a fixed advantage value A, define

$$g_A(r) = \min\left(rA, \mathrm{clip}(r, 1-\epsilon, 1+\epsilon)A\right).$$

The map $r \mapsto rA$ is $|A|$-Lipschitz. The clipped map $r \mapsto \mathrm{clip}(r, 1-\epsilon, 1+\epsilon)A$ is also $|A|$-Lipschitz. The pointwise minimum of two functions with Lipschitz constant $|A|$ is also $|A|$-Lipschitz. Therefore,

$$|g_A(r) - g_A(r')| \le |A||r - r'|.$$

Using $A = \widehat{A}_t$, $r = r_{\theta,h}$, and $r' = r_{\theta,\mathrm{cont}}$, we get

$$|g_{\widehat{A}_t}(r_{\theta,h}) - g_{\widehat{A}_t}(r_{\theta,\mathrm{cont}})| \le |\widehat{A}_t||r_{\theta,h} - r_{\theta,\mathrm{cont}}|.$$

By the bounded-advantage assumption and policy-ratio consistency,

$$|g_{\widehat{A}_t}(r_{\theta,h}) - g_{\widehat{A}_t}(r_{\theta,\mathrm{cont}})| = \mathcal{O}(h^2).$$

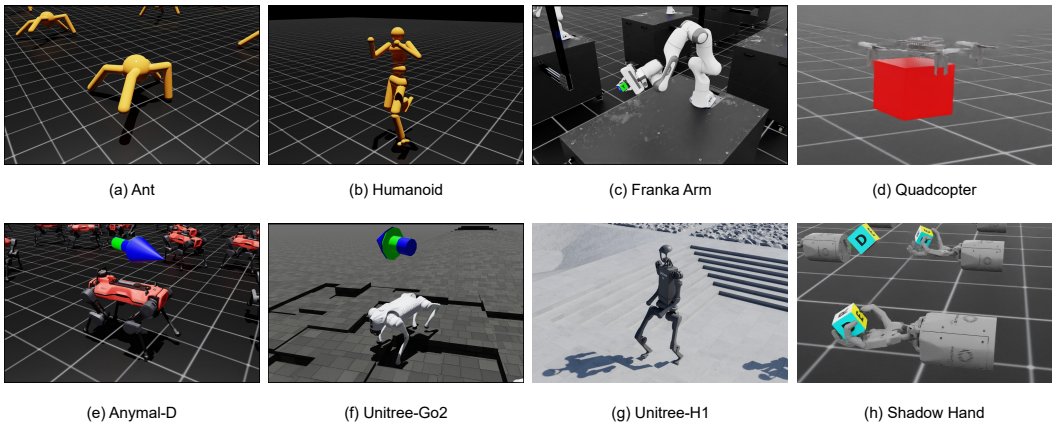
Taking expectation over rollout samples preserves the order, yielding

$$|L_{\mathrm{clip},h}(\theta) - L_{\mathrm{clip},\mathrm{cont}}(\theta)| = \mathcal{O}(h^2).$$

□

B Environmental Details

B.1 Benchmark Introduction



(a) Ant (b) Humanoid (c) Franka Arm (d) Quadcopter

(e) Anymal-D (f) Unitree-Go2 (g) Unitree-H1 (h) Shadow Hand

Figure 5: Eight Isaaclab benchmark visualizations.

IsaacLab [2] serves as the experimental platform in this work. It is a robot-learning framework built upon NVIDIA Isaac Sim and is designed to support scalable research on reinforcement learning, imitation learning, and planning. By integrating the GPU-accelerated PhysX simulator and efficient rendering interfaces, IsaacLab enables parallelized simulation for a wide range of robotic systems and control scenarios.

Table 2: Summary of the IsaacLab benchmark environments used in our experiments.

Environment	**Category**	**State Dim.**	**Action Dim.**
Isaac-Ant-v0	Classic control	60	8
Isaac-Humanoid-v0	Classic control	87	21
Isaac-Lift-Cube-Franka-v0	Arm manipulation	36	8
Isaac-Repose-Cube-Shadow-Direct-v0	Dexterous manipulation	157	20
Isaac-Velocity-Flat-Anymal-D-v0	Quadruped locomotion	48	12
Isaac-Velocity-Rough-Unitree-Go2-v0	Rough-terrain locomotion	235	12
Isaac-Velocity-Rough-H1-v0	Humanoid locomotion	256	19
Isaac-Quadcopter-Direct-v0	Aerial control	12	4

The selected environments are summarized in Table 2. The classical control tasks, *Isaac-Ant-v0* and *Isaac-Humanoid-v0*, evaluate basic locomotion capability in continuous action spaces. The manipulation tasks include *Isaac-Lift-Cube-Franka-v0*, where a Franka arm must lift and move a cube to a target location, and *Isaac-Repose-Cube-Shadow-Direct-v0*, where a Shadow Dexterous Hand performs in-hand cube reorientation. For legged locomotion, we include *Isaac-Velocity-Flat-Anymal-D-v0*, *Isaac-Velocity-Rough-Unitree-Go2-v0*, and *Isaac-Velocity-Rough-H1-v0*, which require robots with different morphologies to track commanded velocities under flat or rough-terrain conditions. Finally, *Isaac-Quadcopter-Direct-v0* evaluates aerial control by requiring a quadcopter to reach and hover around a target position through thrust commands.

Overall, this benchmark suite covers a broad spectrum of control difficulties, from low-dimensional aerial stabilization to high-dimensional humanoid and dexterous-hand control. The large variation in observation size, action dimension, robot morphology, and contact complexity makes it suitable for testing both the performance and robustness of policy optimization algorithms.

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

Table 3: Hyper-parameter settings used in the experiments.

(a) Environment-specific hyper-parameters.

Task	Max Iteration	Steps per Env.	Policy/Value Size	Policy/Value LR
Ant	1000	32	$[400, 200, 100]$	0.0010
Humanoid	2000	32	$[400, 200, 100]$	0.0005
Franka Arm	1500	24	$[256, 128, 64]$	0.0001
Quadcopter	500	24	$[64, 64]$	0.0005
Shadow Hand	10000	16	$[512, 512, 256, 128]$	0.0005
Anymal-D	300	24	$[128, 128, 128]$	0.0010
Unitree-H1	3000	24	$[512, 256, 128]$	0.0010
Unitree-Go2	1500	24	$[512, 256, 128]$	0.0010

(b) Shared hyper-parameters across all environments.

Parameter	Value	Parameter	Value
Training Configuration			
Number of environments	2048	Number of learning epochs	5
Mini-batch size	4	Gradient norm clipping	1.0
RL and Optimization Parameters			
Policy/value activation	Mish	Policy ratio clipping	0.2
Time embedding dimension	32	Hidden layers in time embedding	$[256, 256]$
Discount factor	0.99	GAE coefficient λ	0.95
Compression coefficient	0.01	Flow steps	5

B.2 SAFLOW Model Architecture

The actor network, corresponding to the flow policy, is implemented as a multi-layer perceptron (MLP) with hidden layer dimensions. 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. 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 [29]. This approach, inspired by the positional encoding technique, maps each timestep $t \in \{0, 1, \ldots, T\}$to a fixed-dimensional vector $e_t \in \mathbb{R}^d$ as follows:

$$e_t^{(2i)} = \sin\left(\frac{t}{10000^{2i/d}}\right), \quad e_t^{(2i+1)} = \cos\left(\frac{t}{10000^{2i/d}}\right), \quad \text{for } i = 0, 1, \ldots, \frac{d}{2} - 1. \tag{8}$$

The resulting embedding e_t is then passed through a MLP to allow for nonlinear 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:

$$\tilde{x}_t = \text{concat}(s_t, a_t, \text{Embed}(e_t)), \tag{9}$$

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. The critic network is also implemented as an MLP with the same hidden layer structure. It receives only the current state as input and outputs a scalar value corresponding to the estimated state value.

B.3 Reproducibility Statement & Detailed Hyperparameters

Our proposed method, SAFLOW, 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. For each baseline algorithm, we referred to its official core implementation and integrated it into the same codebase. For classical PPO and TRPO, we use the official

implementation in https://github.com/isaac-sim/IsaacLab. For FPO, we use the official implementation in https://github.com/akanazawa/fpo. For GenPO, we use the official implementation in https://github.com/wadx2019/genpo. The detailed hyper-parameters are listed in Table 3.

All experiments were conducted on a workstation equipped with an Intel Core i9-12900K CPU, 128 GB of system memory, and two NVIDIA A100 GPUs, each with 80 GB of GPU memory. For the software environment, we used Isaac Sim 4.5.0 together with IsaacLab 0.36.3. The benchmark environments were built from `isaaclab_tasks` 0.10.27, `isaaclab_rl` 0.1.3, and `isaaclab_assets` 0.2.1. GPU acceleration relied on CUDA 12.4-related NVIDIA libraries, with cuDNN 9.1.0 and NCCL 2.21.5. All benchmark experiments, including training and evaluation, were performed under the same configuration to ensure a fair comparison across different methods.

C Additional Experiments

C.1 Time Efficiency Analysis.

We further report the wall-clock training time on the Ant and Humanoid environments to evaluate the computational efficiency of different algorithms. Each method is evaluated with 5 random seeds, and we report the mean and standard deviation of the elapsed time.

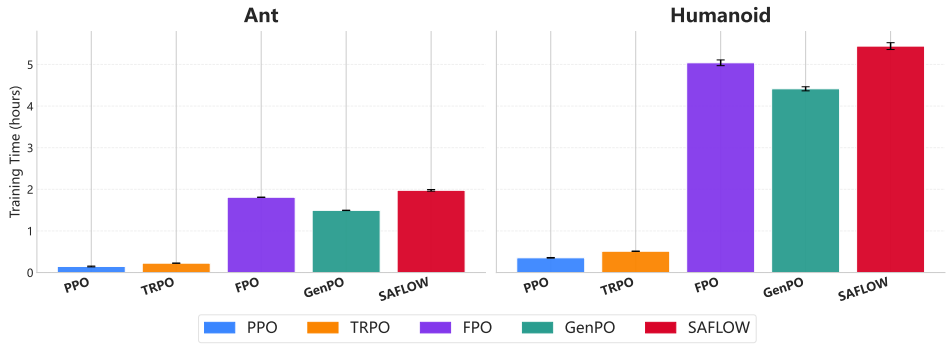


Figure 6: Comparison of training time efficiency.

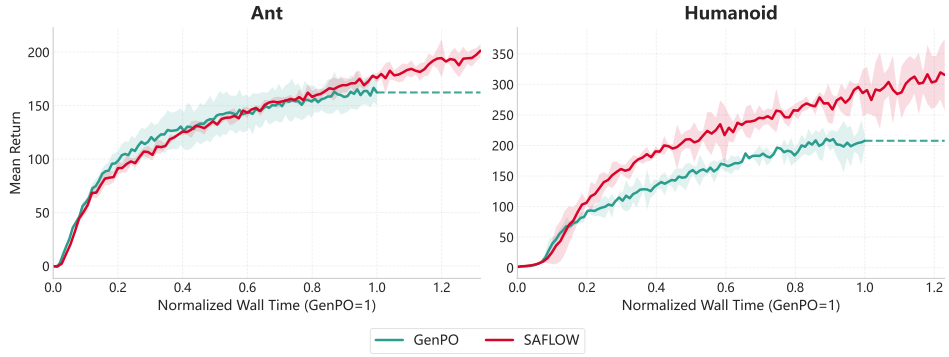


Figure 7: Curves of Performance vs wall time.

As shown in Figures 6 and 7 and Table 4, PPO and TRPO require the least wall-clock time, which is expected since they rely on relatively simple Gaussian policy parameterizations. In contrast, generative-policy methods introduce additional computation due to iterative action generation and more expressive policy modeling. Among these methods, SAFLOW requires more training time than FPO and GenPO on both Ant and Humanoid. However, considering the substantial performance improvements reported in the main results, this additional computational cost is acceptable when final policy performance is the primary concern, particularly for challenging control tasks where higher returns are desired.

Table 4: Wall-clock training time on the Ant and Humanoid environments.

Algorithm	Ant		Humanoid	
	Time (s)	Time (h)	Time (s)	Time (h)
PPO	527.33 ± 34.77	0.146 ± 0.010	1275.77 ± 28.40	0.354 ± 0.008
TRPO	809.04 ± 4.54	0.225 ± 0.001	1844.24 ± 15.61	0.512 ± 0.004
FPO	6511.85 ± 11.33	1.809 ± 0.003	18142.87 ± 248.16	5.040 ± 0.069
GenPO	5379.82 ± 18.52	1.494 ± 0.005	15893.10 ± 178.19	4.415 ± 0.049
SAFLOW	7115.62 ± 69.90	1.976 ± 0.019	19589.14 ± 296.93	5.441 ± 0.082

C.2 Controlled Comparison of Flow Policy Integration Schemes.

To isolate the effect of the integration scheme, we compare the default second-order self-adjoint update with a first-order alternating variant implemented in the same SAFLOW codebase. The second-order variant consistently achieves higher returns, indicating that the performance gain majorly comes from the self-adjoint integration structure rather than from other implementation differences.

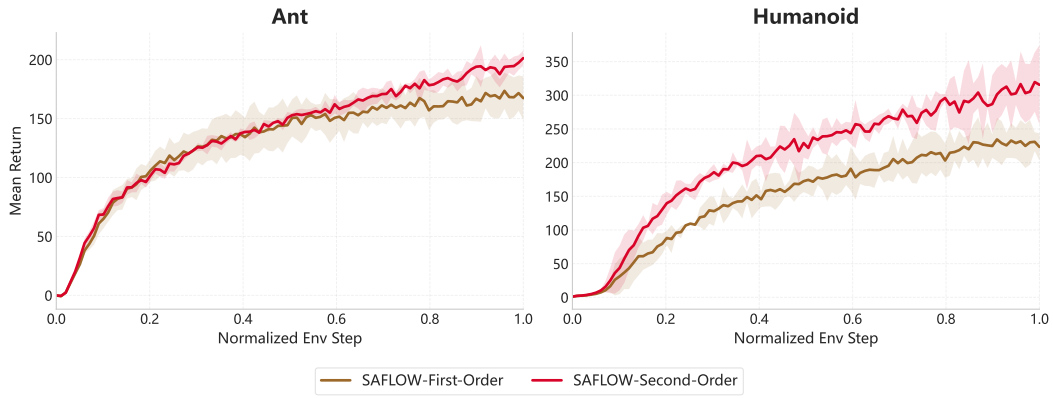


Figure 8: Controlled comparison of flow policy integration schemes.

C.3 Likelihood Stability During Training.

We recorded the standard deviation of the likelihood ratio for each batch during training. The results show that SAFLOW has a lower standard deviation, suggesting more stable and accurate likelihood-ratio estimates.

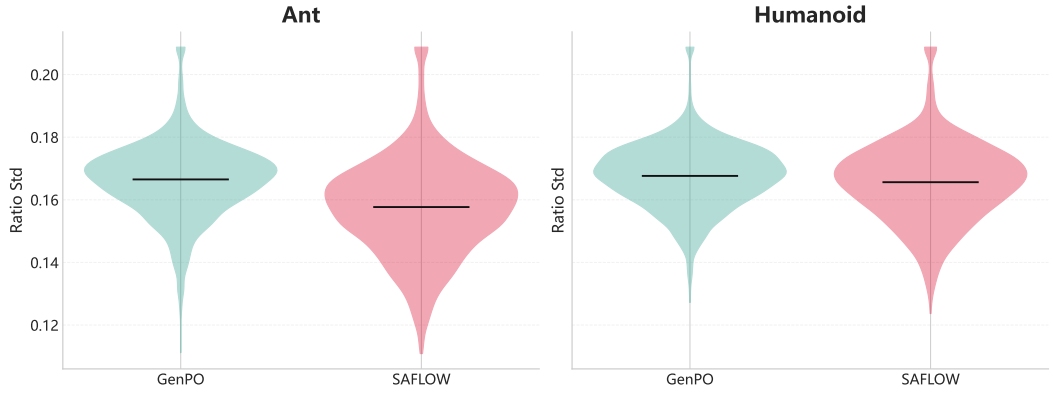


Figure 9: Likelihood stability comparison between GenPO and SAFLOW during training.

C.4 Toy Validation of Discretization and Likelihood Accuracy.

Objective. We evaluate whether SAFLOW yields higher discretization accuracy and more accurate likelihood estimation than GenPO under a controlled setting with closed-form ground truth.

Experimental protocol. We consider the 2D time-dependent system

$$\frac{dx}{dt} = \omega(t)y, \qquad \frac{dy}{dt} = \omega(t)x, \qquad \omega(t) = 1 + 0.5\sin(2\pi t),\ t \in [0,1]. \tag{10}$$

With $u = x + y$ and $v = x - y$, the system decouples into

$$\frac{du}{dt} = \omega(t)u, \qquad \frac{dv}{dt} = -\omega(t)v, \tag{11}$$

which admits an analytic solution. Therefore, both trajectory error and likelihood error can be measured against exact targets.

We sample $(x_0, y_0) \sim \mathcal{N}(0, I)$ with 10,000 samples, and sweep the number of integration steps

$$K \in \{2, 4, 8, 16, 32, 64, 128\}. \tag{12}$$

Metrics. For each K, we report:

$$\text{StateErr}(K) = \mathbb{E}\left[\|z_1^{\text{num}} - z_1^\star\|_2\right], \tag{13}$$

$$\text{InvErr}(K) = \mathbb{E}\left[\|z_0^{\text{inv}} - z_0\|_2\right], \tag{14}$$

$$\text{LLErr}(K) = \mathbb{E}\left[|\log p_{\text{num}}(z_1^\star) - \log p_\star(z_1^\star)|\right]. \tag{15}$$

We estimate empirical convergence order by

$$\text{order}(K \to 2K) = \log_2 \frac{\text{err}(K)}{\text{err}(2K)}. \tag{16}$$

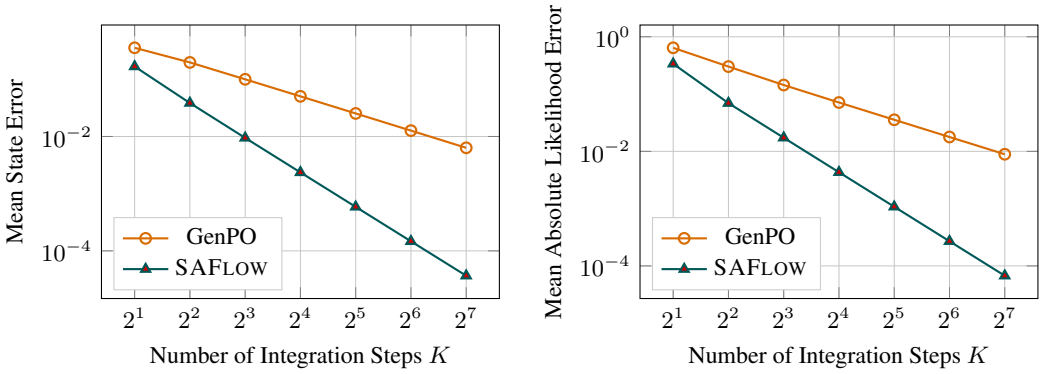


Figure 10: Log-log error curves versus integration steps K. Left: mean state error. Right: mean absolute likelihood error. SAFLOW exhibits a steeper slope, indicating higher-order convergence.

Results and analysis. Figure 10 shows that both methods exhibit approximately linear trends on log-log axes, while SAFLOW has a visibly steeper slope for both state error (left) and likelihood error (right). This indicates a faster error decay rate as K increases. Table 5 confirms the same pattern numerically across all tested K.

For concrete comparisons from Table 5, at $K = 8$, SAFLOW reduces state error from 9.999×10^{-2} to 9.456×10^{-3} (about $10.6\times$ lower) and likelihood error from 1.438×10^{-1} to 1.728×10^{-2} (about $8.3\times$ lower). At $K = 32$, the gains are larger: state error is reduced by about $43\times$ (2.529×10^{-2} vs. 5.882×10^{-4}), inverse error by about $31\times$ (4.014×10^{-2} vs. 1.289×10^{-3}), and likelihood error by about $33\times$ (3.558×10^{-2} vs. 1.079×10^{-3}).

Table 6 summarizes the estimated convergence order and matches the visual trend in Fig. 10: GenPO is close to first-order (state: 0.9920, likelihood: 1.0168), while SAFLOW is close to second-order (state: 2.0062, likelihood: 2.0030). Overall, at matched computational discretization level K, SAFLOW consistently provides higher discretization fidelity and more accurate likelihood estimation.

Table 5: Numerical errors for GenPO and SAFLOW under different numbers of integration steps K.

K	State Error		Inverse Error		Likelihood Error	
	GenPO	SAFLOW	GenPO	SAFLOW	GenPO	SAFLOW
2	0.355545	0.167358	0.626692	0.379097	0.642042	0.337652
4	0.197274	0.038445	0.327340	0.082160	0.301857	0.069460
8	0.099987	0.009455	0.160986	0.020607	0.143792	0.017278
16	0.050374	0.002355	0.080285	0.005154	0.071244	0.004316
32	0.025285	0.000588	0.040140	0.001288	0.035578	0.001078
64	0.012667	0.000147	0.020074	0.000322	0.017794	0.000269
128	0.006339	0.000036	0.010039	0.000080	0.008900	0.000067

Table 6: Empirical convergence order (mean over $K \geq 4$).

Error Order	GenPO	SAFLOW
State error	0.9920	2.0062
Likelihood error	1.0168	2.0030

C.5 Complete Learning Curves for the Scaling Experiment.

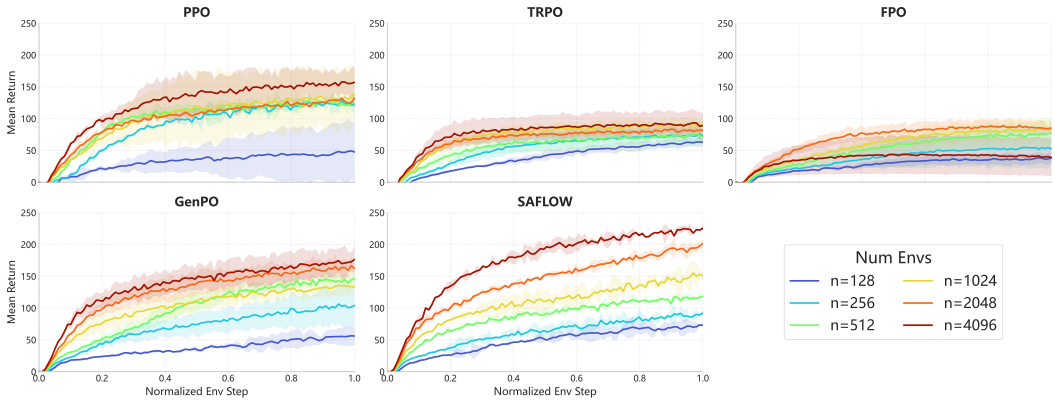


Figure 11: Full Learning curves of scaling experiment.

D Societal Impacts

Potential Positive Societal Impacts. This work improves generative policy optimization for continuous-control reinforcement learning by enabling more expressive and numerically consistent policy parameterizations. In the long term, such methods may benefit robotics and embodied AI systems that require robust control in complex environments, including assistive robotics, industrial automation, logistics, and autonomous inspection. More stable and sample-efficient policy optimization could reduce the computational cost of training RL agents, lowering energy consumption and making advanced control methods more accessible to smaller research groups.

Potential Negative Societal Impacts. As with many advances in RL and robotics, this work may also contribute to the development of autonomous systems that could be deployed in unsafe or harmful ways if not properly constrained. More capable continuous control policies may improve the performance of robots, drones, or other automated platforms in settings where failures, misuse, or insufficient human oversight could pose physical risks. In addition, stronger RL methods may increase reliance on simulation-trained policies whose real-world behavior is not fully verified. Careful evaluation, safety constraints, human oversight, and domain-specific deployment standards are therefore important when applying these methods outside controlled benchmarks.

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: the main claims in the abstract and introduction accurately reflect the paper's contributions and scope. Please refer to the Abstract and Section 1.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Limitations are discussed in Section 6.

   Guidelines:

   - The answer [N/A] 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: Assumptions and complete proofs are provided in Appendix A.

Guidelines:

- The answer [N/A] 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: Implementation details are described in Appendix B.

   Guidelines:

   - The answer [N/A] 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.

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: The training code is provided in the supplemental material. Detailed information regarding the benchmark and configuration is available in Appendix B.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.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 [No] 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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed information regarding the benchmark and configuration is available in Appendix B.

   Guidelines:

   - The answer [N/A] 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: We run 5 random seeds for each experimental setting and report the std to reflect the statistical significance of the experiments.

   Guidelines:

   - The answer [N/A] 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: Detailed information regarding the compute resources is available in Appendix B.

   Guidelines:

   - The answer [N/A] 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 adheres to the established ethical guidelines

   Guidelines:

   - The answer [N/A] 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: [Yes]

    Justification: Both potential positive societal impacts and negative societal impacts can be found in Appendix B.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] 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: All the assets used in the paper are credited and explicitly mentioned and properly respected.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: No new assets in the paper.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] 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.
    - 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 rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.