Flow Matching Reinforcement Learning via SDE Inference

Maojiang Su[*,†,‡] **Tsung-En Lin**[*,§] **Jerry Yao-Chieh Hu**[*,†,‡] **Guo Ye**[†]
Haoran Lu[†] **Shang Wu**[†,‡] **Zhaoran Wang**[†,♭] **Han Liu**[†,‡,♮]

[†] Department of Computer Science, Northwestern University, Evanston, IL 60208, USA
[‡] Center for Foundation Models and Generative AI, Northwestern University, Evanston, IL 60208, USA
[§] National Taiwan University, Taipei, Taiwan
[♭] Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208, USA
[♮] Department of Statistics and Data Science, Northwestern University, Evanston, IL 60208, USA

Abstract

We develop a unified framework for flow-based reinforcement learning (RL) grounded in diffusion–flow duality and establish its theoretical foundations. The framework enables RL for deterministic flow matching inference by introducing diffusion stochastic inference dynamics that supports exploration while admitting deterministic deployment. We show that our framework subsumes several existing flow-based RL methods as special cases and inspire effective new methods. Then we provide theoretical foundations for the frameworks by establishing the correctness and performance transfer guarantee. Specifically, we prove that the policies optimized under stochastic dynamics close to deterministic dynamics at deployment when the pretrained model is well-trained. Moreover, we show that the policy improvement achieved under training-time SDE inference transfers to deployment-time ODE inference. Experiments on synthetic and continuous-control tasks validate the theory and demonstrate the proposed algorithms' effectiveness.

1 Introduction

Flow Matching (FM) [1–3] has become a powerful class of generative models and achieves state-of-the-art performance across image, speech, video generation, and robotics [4–7]. Motivated by this success, recent work explores Reinforcement Learning (RL) fine-tuning of flow-based models and policies to improve alignment, generation quality, and task-specific performance [8–11].

Growing interest in Stochastic Differential Equation (SDE) inference during FM training is motivated by its potential for effective exploration and policy gradient optimization. Several recent empirical works illustrate this trend. Flow-GRPO [12] introduces SDE-based inference for RL of flow matching models and achieves strong gains on text-to-image generation, including compositional generation and human preference alignment. DanceGRPO [13] scales this paradigm to large-scale visual generation with improved stability across diverse prompts. MixGRPO [14] and TempFlow-GRPO [15] further investigate partial or time-localized SDE inference by combining ODE and SDE sampling, leading to accelerated training. In parallel, ReinFlow [16] and Flow-SDE [17] apply SDE inference to flow policy optimization in robotics, demonstrating substantial gains under sparse rewards.

However, despite their strong empirical performance, these methods lack a unified framework that characterizes their common structure and theoretical guarantees for transforming deterministic ODE inference into stochastic SDE inference during training. We address these gaps by introducing

[*]These authors contributed equally.

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

Table 1: Flow-based reinforcement learning methods under a unified stochastic inference perspective.

Method	SDE Inference Horizon	Surrogate	Advantage Estimator	Noise Level λ_t
Flow Matching Reinforcement Learning (FMRL)				
Flow-GRPO [12]	Full trajectory	GRPO	Group-relative	$a(1-t)/t$
DanceGRPO [13]	Full trajectory	GRPO	Group-relative	Constant
MixGRPO [14]	Local window	GRPO	Group-relative	$a(1-t)/t$
TempFlow-GRPO [15]	Time-localized	GRPO	Group-relative	$a(1-t)/t$
Flow Matching Policy Optimization (FMPO)				
ReinFlow [16]	Full trajectory	PPO	GAE	Learnable
Flow-SDE [17]	Full trajectory	PPO	GAE	$a(1-t)/t$
FMPO-CPPO (ours)	Full trajectory	ChainPPO	GAE	$a(1-t)$

a unified framework for flow matching RL based on SDE inference. The proposed framework formulates SDE inference as an inner Markov Decision Process (MDP) and applies policy gradient methods, such as PPO [18] and GRPO [19], to optimize flow policies. It subsumes existing flow-based RL methods [12, 13, 16, 17] as special cases and inspires new algorithmic instantiations. Moreover, we present a theoretical analysis of correctness and RL performance transfer within this framework. These theoretical results also apply to the aforementioned methods as special cases of our framework.

Contributions. Our contributions are threefold:

- **Flow Matching RL Via Stochastic Inference.** We present a unified framework for flow matching RL based on stochastic SDE inference. For flow matching RL fine-tuning (Section 3.1), with no external environment MDP, as in image generation, the framework applies policy gradient methods to optimize the flow matching model in the *inner* MDP (3.2) induced by stochastic inference. For flow policy optimization (Section 3.2), with an environment MDP, as in robotic control, the framework composes the inner inference and environment MDPs into a *composite* MDP (3.3) and optimizes the flow-based policy in this composite MDP using policy gradient methods. Our framework subsumes existing flow-based RL algorithms through their surrogate objectives, noise levels and advantage estimators (Section 3.3). Moreover, we introduce a new flow matching policy optimization algorithm FMPO-CPPO (Algorithm 1), an instantiation of the proposed framework.
- **Theoretical Analysis.** We provide a rigorous theoretical analysis for flow matching RL via stochastic SDE inference. First, we establish a correctness guarantee (Theorem 4.1), showing that when the pretrained flow model has small flow-matching error, stochastic SDE inference and deterministic ODE inference induce close terminal distributions. Second, we prove a performance transfer theorem (Theorem 4.2), showing that policy improvement under training-time SDE inference transfers to deployment-time ODE inference up to an error term controlled by the flow-matching error. Together, these results give *sufficient* conditions for using stochastic SDE inference during training while preserving deterministic ODE inference at deployment.
- **Empirical Validation.** We experimentally validate the theoretical results and evaluate the proposed algorithms. Specifically, we validate the correctness theorem in Section 5.1 and the performance transfer theorem in Section 5.2. The results confirm that smaller flow-matching error leads to smaller SDE-ODE mismatch and more reliable transfer from stochastic training to deterministic deployment. In addition, we evaluate FMPO-CPPO across representative continuous-control tasks in Section 5.3, where it outperforms existing flow-based policy optimization baselines.

Related Works. We defer extended related-work discussion to Section A due to page limits.

Organization. Section 2 reviews flow matching and SDE inference. Section 3 presents a unified framework for flow-based RL under stochastic inference. Section 4 establishes its theoretical properties. Section 5 presents validation experiments and empirical results. Section 6 concludes with practical insights. The appendix provides additional related work (Section A) and detailed proofs (Sections C and D).

2 Preliminaries

Flow Matching. A flow model is a continuous transformation that maps samples $X_0 \sim p_0$ from a source distribution to samples $X_1 \sim p_1$ from a target distribution. A time-dependent flow ψ :

$[0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ implements this transformation, where $\psi_t(x)$ denotes the state of x at time t. The generative process $(X_t)_{t\in[0,1]}$ follows $X_0 \sim p_0$,

$$X_t = \psi_t(X_0), \quad t \in [0,1].$$

A velocity field $u : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ defines a flow ψ via an Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\psi_t}{\mathrm{d}t} = u(\psi_t(x), t), \quad \psi_0(x) = x. \tag{2.1}$$

This ODE induces a probability path $(p_t)_{t\in[0,1]}$ for the marginal distribution $X_t = \psi_t(X_0)$.

Flow Matching [1] learns a parameterized velocity field u_θ to match the pre-designed velocity u inducing the desired probability path p_t from p_0 to p_1. Define the flow matching loss

$$\mathcal{L}_{\text{FM}}(\theta) = \mathop{\mathbb{E}}_{t, X_t\sim p_t}[\|u_\theta(X_t,t) - u(X_t,t)\|_2^2], \tag{2.2}$$

where $t \sim U[0,1]$. However, the target velocity $u(x,t)$ is intractable. To resolve this issue, Lipman et al. [1] introduce an equivalent conditional flow matching loss

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathop{\mathbb{E}}_{t, X_1, X_t}\left[\left\|u_\theta(X_t,t) - u(X_t|X_1,t)\right\|_2^2\right], \tag{2.3}$$

where $X_1 \sim p_1$ and $X_t \sim p_{t|1}(\cdot|X_1)$. For a fixed probability path, coupling, and time distribution, $\mathcal{L}_{\text{CFM}}(\theta) = \mathcal{L}_{\text{FM}}(\theta) + C$, where C is independent of θ; hence the parameter gradients and minimizers coincide. Following [1, 20], we use the affine path $X_t = \alpha_t X_1 + \sigma_t X_0$. Its ground-truth conditional velocity is $u(x|x_1,t) = \dot\alpha_t x_1 + (\dot\sigma_t/\sigma_t)(x - \alpha_t x_1)$ for $\sigma_t > 0$. For the linear path, this becomes $u(x|x_1,t) = (x_1 - x)/(1-t)$ for $t < 1$. The schedules $\alpha_t, \sigma_t : [0,1] \to [0,1]$ are smooth and monotone functions satisfying $\alpha_0 = \sigma_1 = 0$ and $\alpha_1 = \sigma_0 = 1$. After training u_θ, we sample by solving (2.1) with u replaced by u_θ; the terminal distribution approximates p_1.

Flow Matching SDE Inference. For any noise level $\lambda_t \geq 0$, we consider two stochastic processes initialized from the same distribution p_0

$$\mathrm{d}X_t = u(X_t,t)\mathrm{d}t \tag{2.4}$$

$$\mathrm{d}\widetilde{X}_t = [u(\widetilde{X}_t,t) + \frac{\lambda_t}{2}\nabla \log p_t(\widetilde{X}_t)]\mathrm{d}t + \sqrt{\lambda_t}\mathrm{d}W_t, \tag{2.5}$$

where W_t is the Wiener process. (2.4) corresponds to the standard ODE inference used in flow matching. Then both processes induce the same marginal distributions p_t for all $t \in [0,1]$. For the affine path with $X_0 \sim \mathcal{N}(0,I)$ independent of X_1, Lipman et al. [20] derive the following score-based representation of the velocity at times with $\alpha_t \neq 0$:

$$u(x,t) = \frac{\dot\alpha_t}{\alpha_t}x - \frac{\dot\sigma_t\sigma_t\alpha_t - \dot\alpha_t\sigma_t^2}{\alpha_t}\nabla \log p_t(x). \tag{2.6}$$

Substituting (2.6) into (2.5) yields the following SDE with $\widetilde{X}_0 \sim p_0$ at times where $\dot\sigma_t\sigma_t\alpha_t - \dot\alpha_t\sigma_t^2 \neq 0$; endpoint coefficients are defined by limits when those limits exist:

$$\mathrm{d}\widetilde{X}_t = [(1 - \frac{\lambda_t}{2}\frac{\alpha_t}{\dot\sigma_t\sigma_t\alpha_t - \dot\alpha_t\sigma_t^2})u(\widetilde{X}_t,t) + \frac{\lambda_t}{2}\frac{\dot\alpha_t}{\dot\sigma_t\sigma_t\alpha_t - \dot\alpha_t\sigma_t^2}\widetilde{X}_t]\mathrm{d}t + \sqrt{\lambda_t}\mathrm{d}W_t. \tag{2.7}$$

Markov Decision Processes. We consider a Markov Decision Process (MDP) with tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $P(s'|s,a)$ is the transition kernel, $R(s,a)$ is the reward function, and $\gamma \in [0,1)$ is the discount factor. A policy $\pi(a|s)$ induces a distribution over trajectories $\tau = (s_0, a_0, s_1, a_1, \ldots)$, where $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim P(\cdot|s_t,a_t)$. The objective of RL is to maximize the expected cumulative return

$$J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t\geq 0}\gamma^t R(s_t,a_t)\right]. \tag{2.8}$$

Policy Optimization. Policy gradient methods (e.g., REINFORCE [21]) optimize the policy $\pi_\theta(a|s)$ by approximating the gradient of the objective $J(\pi_\theta)$:

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t\geq 0}\gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t)\, r_t(s_t,a_t)\right], \quad \text{with} \quad r_t(s_t,a_t) \coloneqq \sum_{\tau\geq t}\gamma^{\tau-t}R(s_\tau,a_\tau),$$

where $r_t(s_t,a_t)$ is the discounted cumulative future reward. In practice, we replace $r_t(s_t,a_t)$ with an advantage estimator $A(s_t,a_t)$ to reduce the variance of gradient estimates.

3 Methodology

We use SDE-based inference for flow matching (2.7) to construct a unified reinforcement learning framework with two settings: (i) flow matching RL fine-tuning (Section 3.1) and (ii) flow policy optimization (Section 3.2). The first treats stochastic flow inference as an *inner* MDP for direct RL fine-tuning of a pretrained flow model with terminal rewards. The second composes this inner process with an *environment* MDP, yielding a *composite* MDP for flow policy optimization. Finally, our unified view recovers existing flow-based RL methods and yields new algorithms (Section 3.3).

3.1 Flow Matching RL Fine-Tuning

Given a pretrained velocity field u_θ, let r be a reward function and p^θ_τ the probability path induced by u_θ. Flow Matching RL (FMRL) maximizes expected terminal rewards,

$$\mathcal{J}_{\text{FMRL}} = \mathop{\mathbb{E}}_{X_1 \sim p_1^\theta}[r(X_1)]. \tag{3.1}$$

The SDE inference procedure in (2.7) injects randomness into the inference process, which allows us to reframe flow matching inference as an *inner* MDP. Use flow times $\tau_k = k/K$, $k = 0, \dots, K$, and step size $\Delta\tau = 1/K$. Write $x^k = x_{\tau_k}$ for the sample at inference step k. Inspired by denoising diffusion policy optimization [22, 23], we construct the following finite-horizon MDP:

$$\begin{aligned}
&s_k = (\tau_{k-1}, x^{k-1}), \quad \pi_\theta(a_k|s_k) = p^\theta_{\tau_{k-1}}(x^k|x^{k-1}),\\
&a_k = x^k, \quad P(\cdot|s_k, a_k) = \delta_{(\tau_k, a_k)},\\
&P_0 = \delta_0 \otimes \mathcal{N}(0, I), \quad R(s_k, a_k) = \begin{cases} r(x^K), & k = K,\\ 0, & 1 \le k < K. \end{cases}
\end{aligned} \tag{3.2}$$

where $k = 1, \dots, K$, the state $(1, x^K)$ is terminal, and δ_y denotes the Dirac delta distribution. For conditional generation, include the fixed condition c in the state, $s_k = (\tau_{k-1}, x^{k-1}, c)$, and condition all inference kernels on c throughout the rollout. The finite-horizon inner return uses discount 1. The inner MDP has the same RL objective (3.1):

$$\mathcal{J}(\theta) = \mathop{\mathbb{E}}_{a_k \sim \pi_\theta, s_k \sim P}[\sum_{k=1}^{K} R(s_k, a_k)] = \mathcal{J}_{\text{FMRL}}(\theta).$$

We therefore optimize $\mathcal{J}(\theta)$ using standard policy gradient methods. For example, Proximal Policy Optimization (PPO) [18] maximizes the surrogate objective

$$\mathcal{J}_{\text{PPO}}(\theta) = \mathop{\mathbb{E}}_{(a_k, s_k) \sim \pi_{\text{old}}}[\min\{r_k(\theta)\widehat{A}_k, \widetilde{r}_k(\theta)\widehat{A}_k\}],$$

where $\widehat{A}_k$ is an advantage estimate computed from the old-policy rollout and held fixed when differentiating with respect to θ, and

$$r_k(\theta) = \frac{\pi_\theta(a_k|s_k)}{\pi_{\text{old}}(a_k|s_k)}, \ \widetilde{r}_k(\theta) = \text{Clip}\left(r_k(\theta), 1-\epsilon, 1+\epsilon\right).$$

Here $p^\theta_{\tau_{k-1}}$ is the Euler–Maruyama kernel for step k. Writing $b_\theta(x, \tau)$ for the drift in (2.7) with u replaced by u_θ, a positive-noise step has

$$p^\theta_{\tau_{k-1}}(x'|x) = \mathcal{N}\big(x'; x + \Delta\tau\, b_\theta(x, \tau_{k-1}), \Delta\tau\, \lambda_{\tau_{k-1}} I\big).$$

Thus the discretized transition likelihoods and their gradients are tractable.

3.2 Flow Matching Policy Optimization

We consider an *environment* MDP $\mathcal{M}_{\text{env}} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$. At each environment step t, the agent observes $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$ according to a *flow policy* $\pi_\theta(\cdot|s_t)$. The objective of Flow Matching Policy Optimization (FMPO) is to maximize the expected return

$$\mathcal{J}_{\text{env}}(\theta) = \mathop{\mathbb{E}}_{a_t \sim \pi_\theta, s_t \sim P}[\sum_{t \ge 0} \gamma^t R(s_t, a_t)].$$

As shown in Section 3.1, we view the SDE inference procedure (2.7) of a flow model as an *inner* MDP. Following diffusion policy policy optimization [24], we combine the *inner* MDP and *environment* MDP into a *composite* MDP. This allows us to use policy gradient methods to optimize flow policy.

Use the same flow times $\tau_k = k/K$ and step size $\Delta\tau = 1/K$ as in Section 3.1. At environment step t, SDE inference (2.7) starts from $a_t^0 \sim \mathcal{N}(0, I)$ and generates a_t^k at flow time τ_k, for $k = 1, \dots, K$. We execute $a_t := a_t^K$ in the environment. t indexes environment steps and k inference steps. We define the composite MDP $\mathcal{M}_{\text{comp}}$ as:

$$\begin{aligned}
&\bar{s}_{t,k} := (s_t, \tau_{k-1}, a_t^{k-1}), \quad \overline{\pi}_\theta(\bar{a}_{t,k}|\bar{s}_{t,k}) := p^\theta_{\tau_{k-1}}(a_t^k|a_t^{k-1}, s_t), \\
&\overline{R}(\bar{s}_{t,k}, \bar{a}_{t,k}) := \begin{cases} 0, & 1 \le k < K, \\ R(s_t, a_t^K), & k = K, \end{cases} \qquad \bar{a}_{t,k} := a_t^k, \\
&\bar{s}' = \begin{cases} (s_t, \tau_k, a_t^k), & 1 \le k < K, \\ (s_{t+1}, \tau_0, a_{t+1}^0), & k = K. \end{cases}
\end{aligned} \tag{3.3}$$

Here $\bar{s}'$ is the next composite state under $\overline{P}$. The kernel $p^\theta_{\tau_{k-1}}$ is the Euler–Maruyama step from Section 3.1, conditioned on the environment state s_t. At $k = K$, draw $s_{t+1} \sim P(\cdot|s_t, a_t^K)$ and fresh noise $a_{t+1}^0 \sim \mathcal{N}(0, I)$ independently. Inner transitions use discount 1 for $k < K$, and discount γ only at the environment boundary $k = K$. Optimizing the expected return in the composite MDP is equivalent to optimizing the original environment objective:

$$\mathcal{J}_{\text{com}}(\theta) = \mathop{\mathbb{E}}_{\bar{a}_{t,k} \sim \overline{\pi}_\theta, \bar{s}_{t,k} \sim \overline{P}} \Big[\sum_{t,k} \gamma^t \overline{R}(\bar{s}_{t,k}, \bar{a}_{t,k}) \Big] = \mathcal{J}_{\text{env}}(\theta).$$

For positive-noise Euler–Maruyama steps, the composite policy $\overline{\pi}_\theta$ admits a Gaussian likelihood, so standard policy gradient methods apply.

3.3 Unified Framework And Instantiations

Our framework admits three algorithmic degrees of freedom: (i) the choice of surrogate objective $\mathcal{J}(\theta)$; (ii) the window over which stochastic differential equation (SDE) inference is applied; and (iii) the noise schedule λ_t used in SDE inference. Different choices induce a broad family of flow-based RL algorithms, encompassing both existing methods and the instantiations proposed in this work.

Flow Matching Policy Optimization With ChainPPO. For FM policy optimization in an environment MDP, we instantiate the framework in Section 3.2 with a chain-level PPO surrogate, yielding *Flow Matching Policy Optimization with ChainPPO* (FMPO-CPPO). Following Section 3.2, the flow policy generates an environment action $a_t = a_t^K$ through the stochastic inference chain $a_t^0 \to a_t^1 \to \cdots \to a_t^K$. The composite MDP (3.3) suggests treating this full inference chain as the stochastic action at environment step t. Given samples collected from an old policy $\overline{\pi}_{\text{old}}$, define

$$r_{t,k}(\theta) = \frac{\overline{\pi}_\theta(\bar{a}_{t,k}|\bar{s}_{t,k})}{\overline{\pi}_{\text{old}}(\bar{a}_{t,k}|\bar{s}_{t,k})}, \qquad \rho_t^{\text{chain}}(\theta) = \prod_{k=1}^K r_{t,k}(\theta).$$

Since each inner action $a_t^k \in \mathbb{R}^D$ is an action chunk of dimension $D = Hd_a$ (horizon H, action dimension d_a), we normalize log-ratios by D to make clipping ranges comparable across tasks:

$$\widetilde{r}_{t,k}(\theta) = r_{t,k}(\theta)^{1/D}, \qquad \widetilde{\rho}_t^{\text{chain}}(\theta) = \prod_{k=1}^K \widetilde{r}_{t,k}(\theta) = \exp\Big(\frac{1}{D} \sum_{k=1}^K \log r_{t,k}(\theta)\Big).$$

The normalized quantities are tempered ratios; in general they are not importance ratios for the original policy. FMPO-CPPO applies PPO clipping to the normalized chain-level ratio:

$$\mathcal{J}_{\text{chain}}(\theta) = \mathop{\mathbb{E}}_{\overline{\tau}_t \sim \overline{\pi}_{\text{old}}} \Big[\min \Big\{ \widetilde{\rho}_t^{\text{chain}}(\theta) \widehat{A}_t,\ \text{Clip}\left(\widetilde{\rho}_t^{\text{chain}}(\theta), 1 - \epsilon_{\text{c}}, 1 + \epsilon_{\text{c}}\right) \widehat{A}_t \Big\} \Big],$$

where $\overline{\tau}_t = \{(\bar{s}_{t,k}, \bar{a}_{t,k})\}_{k=1}^K$ and $\widehat{A}_t$ is the environment-level advantage estimate for the executed action $a_t = a_t^K$, held fixed during actor updates. Since $\widetilde{\rho}_t^{\text{chain}} = (\rho_t^{\text{chain}})^{1/D}$ is monotone in ρ_t^{chain}, clipping $\widetilde{\rho}_t^{\text{chain}}$ to $[1 - \epsilon_{\text{c}}, 1 + \epsilon_{\text{c}}]$ corresponds to clipping the chain likelihood ratio ρ_t^{chain}

to $[(1-\epsilon_c)^D, (1+\epsilon_c)^D]$. This correspondence concerns the clipping thresholds; the tempered and untempered PPO objectives generally differ. In practice, we optimize the mixed surrogate

$$\mathcal{J}_{\text{FMPO-CPPO}}(\theta) = \mathcal{J}_{\text{chain}}(\theta) + \alpha \mathcal{J}_{\text{step}}(\theta),$$

where $\alpha \geq 0$ and $\mathcal{J}_{\text{step}}$ is the standard PPO surrogate on the composite MDP (3.3):

$$\mathcal{J}_{\text{step}}(\theta) = \mathop{\mathbb{E}}_{(\bar{s}_{t,k}, \bar{a}_{t,k}) \sim \bar{\pi}_{\text{old}}} \left[\min \left\{ \widetilde{r}_{t,k}(\theta) \widehat{A}_t,\ \text{Clip}\left(\widetilde{r}_{t,k}(\theta), 1-\epsilon_s, 1+\epsilon_s\right) \widehat{A}_t \right\} \right].$$

Before clipping, both the step-wise and chain-level surrogates have policy gradients at $\theta = \theta_{\text{old}}$ proportional to that of the composite-MDP objective $\mathcal{J}_{\text{com}}$ (Section E), which equals $\mathcal{J}_{\text{env}}$ by Section 3.2; the normalization only rescales them by $1/D$, since $\nabla_\theta \widetilde{\rho}_t^{\text{chain}} = \frac{1}{D} \nabla_\theta \log \rho_t^{\text{chain}}$ at $\theta = \theta_{\text{old}}$. The step-wise term clips individual inference-transition surrogate contributions, while the chain term clips the full-chain surrogate contribution. PPO clipping does not impose a hard bound on every likelihood ratio or on the policy KL divergence. The chain-level term accounts jointly for all inference transitions producing the executed action, whereas the step-wise auxiliary term provides local surrogate contributions and improves gradient stability. For positive-noise Euler–Maruyama steps, Gaussian transition kernels make these likelihood ratios tractable; see Algorithm 1.

Unified Framework. Several recent flow-based RL methods instantiate our unified framework. Flow-GRPO [12] instantiates flow matching RL fine-tuning (FMRL) with a GRPO surrogate. DanceGRPO [13] follows the same design principles and fits FMRL. Flow-SDE [17] instantiates FMPO using SDE inference and a PPO objective. ReinFlow [16] approximately instantiates FMPO with a PPO surrogate objective. Unlike standard SDE inference in FM (2.7), ReinFlow injects stochasticity through learnable noise. Methods such as [14, 15] instantiate FMRL with restricted SDE inference windows to accelerate training. We compare these methods with our FMPO-CPPO in Table 1.

4 Theoretical Analysis

We theoretically analyze the methods in Section 3. We first establish a correctness guarantee in Section 4.1. Under a small flow-matching error, the SDE-based training dynamics and the ODE-based deployment dynamics induce close terminal distributions. We then study RL implications in Section 4.2. A one-step SDE-based RL improvement transfers to the ODE-based deployment policy when flow-matching error is small. Together, these results justify flow RL via diffusion–flow duality algorithms at both distributional and RL levels. We consider the linear path $X_t = tX_1 + (1-t)X_0$ with independent $X_0 \sim \mathcal{N}(0, I_d)$ and $X_1 \sim p_1$. For an early stopping time $0 < T < 1$, define

$$\mathcal{L}_{\text{FM},T}(\theta) := \int_0^T \mathop{\mathbb{E}}_{X_t \sim p_t^\star} [\|u_\theta(X_t, t) - u(X_t, t)\|_2^2] \, \mathrm{d}t,$$
$$\epsilon_\theta := \sqrt{\mathcal{L}_{\text{FM},T}(\theta)}.$$

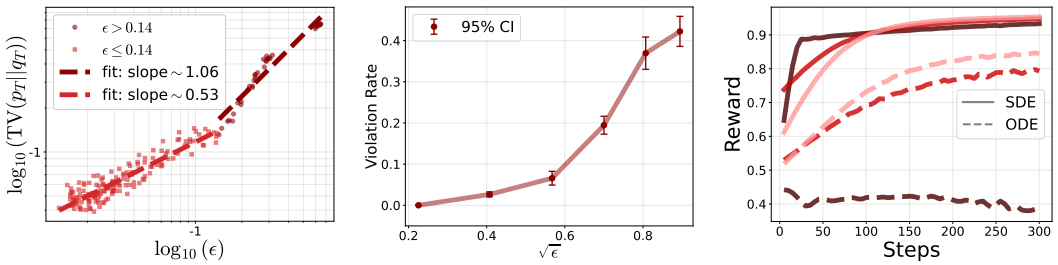
Here $p_t^\star$ is the reference path induced by the true velocity u. The unindexed ϵ in an error-bound assumption denotes an upper bound on ϵ_θ.

4.1 Correctness

For an estimated velocity field u_θ, the identity (2.6) does not always hold. Consequently, the ODE inference dynamics in (2.4) and the SDE inference dynamics in (2.5) are generally not equivalent. This ODE–SDE mismatch creates two inconsistencies in the algorithms in Section 3.

First, consider a pretrained velocity field u_{θ_1} before reinforcement learning. At this stage, the base policy follows ODE inference, while exploration during RL relies on SDE inference. Second, consider the velocity field u_{θ_2} obtained after reinforcement learning. Although u_{θ_2} is optimized under SDE dynamics during training, deployment again follows ODE-based inference for efficiency. In both cases, the transition between ODE and SDE inference is not exactly equivalent, since the parameterized velocities u_{θ_1} and u_{θ_2} differ from the true affine-path-induced velocity u.

Fortunately, when the estimated velocity u_θ stays close to the true velocity u, identity (2.6) holds approximately. The following theorem formalizes this observation.



(a) TV$(p_T||q_T)$ versus ϵ in log scale **(b) One-step improvement violation rate versus $\sqrt{\epsilon}$** **(c) SDE and ODE reward trajectories**

Figure 1: **(a) TV$(p_T\|q_T)$ versus ϵ.** Recall the definition of flow-matching error as $\epsilon^2 = \int_0^T \mathbb{E}_{X_t\sim p_t^\star}[\|u(X_t,t) - u_\theta(X_t,t)\|_2^2]\mathrm{d}t$. For small errors ($\epsilon \leq 0.14$), the fitted slope ($\approx 0.53$) matches the theoretical $O(\sqrt{\epsilon})$ rate. For larger errors ($\epsilon > 0.14$), the fitted slope (≈ 1.06) corresponds to the $O(\epsilon)$ regime, consistent with the bound in Theorem 4.1. **(b) Violation rate versus $\sqrt{\epsilon}$.** The violation rate measures the fraction of consecutive checkpoint pairs where SDE reward improves but ODE reward decreases. Pairs are binned by $\sqrt{\epsilon}$. The rate increases with $\sqrt{\epsilon}$, confirming better SDE-to-ODE transfer for small ϵ. **(c) Training-time SDE and deployment-time ODE reward trajectories.** Solid curves show rewards under training-time SDE inference; dashed curves show rewards under deployment-time ODE inference. Lighter curves indicate better-pretrained base models with smaller ϵ. ODE rewards closely track SDE rewards for well-pretrained models, but oscillate for poorly pretrained models (darkest) despite steady SDE improvement, validating the failure mode in Remark 4.1.

Theorem 4.1 (Correctness for Flow RL via Stochastic Inference, Informal)**.** Fix an early stopping time $0 < T < 1$. Let p_T and q_T be the terminal distributions of ODE inference (2.4) and SDE inference (2.7), respectively, using u_θ from the same Gaussian source. Assume p_1 has a finite second moment and the velocities are continuous in time. Suppose u is continuously differentiable and globally Lipschitz in space, uniformly over $t \in [0, T]$. Suppose u_θ is twice continuously differentiable in space, with uniformly bounded Jacobian and gradient of its divergence over x, t, θ. Assume the noise schedule λ is continuous and strictly positive on $[0, T]$. If $\epsilon_\theta \leq \epsilon$, then

$$\mathrm{TV}(p_T||q_T) \leq O(\sqrt{\epsilon})$$

as $\epsilon \to 0$, with a constant independent of θ and ϵ.

Proof. Please see Section C for the formal statement and detailed proof. □

Theorem 4.1 establishes correctness for our Flow Matching RL framework under a fine-tuning setting. Before the first transition, we require pretraining. This ensures that the base policy u_{θ_1} stays close to the true constructed velocity. The base policy then satisfies condition (C.1). For the second transition, we assume the RL algorithm performs fine-tuning without large changes to the velocity. Under this assumption, u_{θ_2} remains close to u_{θ_1} and to the true constructed velocity. The fine-tuned policy then satisfies condition (C.1).

4.2 SDE-to-ODE Performance Transfer

Section 4.1 establishes conditions for distributional consistency between SDE- and ODE-based inference dynamics. Under the same assumptions, we extend this analysis to RL. The distributional correctness guarantee implies a one-step performance improvement bound for the ODE inference policy.

Theorem 4.2 (Transfer of Improvement from SDE to ODE)**.** Let $r : \mathbb{R}^d \to \mathbb{R}$ be the terminal reward, with $|r(x)| < R < \infty$ for any $x \in \mathbb{R}^d$. Fix early stopping time $0 < T < 1$ and noise level $\lambda_t > 0$. For each parameter θ, we define SDE inference process Z_t^θ and ODE inference Y_t^θ as in Theorem C.1. Define the corresponding performances

$$J_{\mathrm{SDE}}(\theta) := \mathbb{E}[r(Z_T^\theta)], \quad J_{\mathrm{ODE}}(\theta) := \mathbb{E}[r(Y_T^\theta)].$$

Let $\{\theta_k\}_{k\geq 0}$ be the parameter sequence of an RL algorithm satisfying the following one-step SDE performance improvement condition:

$$J_{\mathrm{SDE}}(\theta_{k+1}) \geq J_{\mathrm{SDE}}(\theta_k) + \Delta_k, \quad \Delta_k > 0.$$

Assume that the hypotheses of Theorem C.1 hold for every θ_k with the same constants L, H and noise schedule. Then the ODE performance satisfies

$$J_{\mathrm{ODE}}(\theta_{k+1}) \geq J_{\mathrm{ODE}}(\theta_k) + \Delta_k - 2R(\delta_k + \delta_{k+1}),$$

where $\delta_k := C_{\mathrm{ODE}}\sqrt{\epsilon_k} + \frac{\sqrt{C_{\lambda,T}}}{2}\epsilon_k = O(\sqrt{\epsilon_k})$ bounds the total variation distance between the terminal distributions induced by the ODE and SDE dynamics at iteration k, and $\epsilon_k := \sqrt{\mathcal{L}_{\mathrm{FM},T}(\theta_k)}$ denotes the FM error for θ_k.

Proof. Please see Section D for a detailed proof. □

Remark 4.1 (Practical Implication)**.** The performance transfer guarantee requires the estimated velocity $u_\theta(x,t)$ to stay close to the ground-truth affine-path velocity $u(x,t)$. This condition holds in fine-tuning settings, where the pretrained flow model has small flow-matching error and RL updates remain moderate. In contrast, RL from scratch produces velocities far from the affine-path dynamics, creating a large ODE-SDE mismatch. Thus, our analysis identifies pretraining quality and controlled RL drift as key factors for reliable transfer from SDE-based training to ODE-based deployment.

Theorem 4.2 shows that when $\Delta_k \geq 2R(\delta_k + \delta_{k+1})$, the deployment-time ODE policy improves monotonically. Since all bound terms scale with the reward magnitude R, rescaling rewards alone does not help. Moreover, maintaining a small flow-matching error $\mathcal{L}_{\mathrm{FM},T}(\theta_k)$ during RL is difficult in practice. So controlling δ_k alone does not help either. Instead, the condition guarantees deployment-time improvement when the SDE-based RL update yields a sufficiently large improvement Δ_k. This result provides a deployment-time guarantee: despite SDE-based RL optimization, the resulting ODE inference policy is ensured to improve monotonically.

5 Experiments

In this section, Section 5.1 and Section 5.2 present numerical experiments to validate the correctness theorem (Theorem 4.1) and improvement transfer theorem (Theorem 4.2). In addition, Section 5.3 performs experiments to demonstrate the effectiveness of our method FMPO-CPPO (Algorithm 1).

5.1 Validating Correctness (Theorem 4.1)

Setup. Let $\epsilon^2 = \int_0^T \mathbb{E}_{X_t\sim p_t^\star}[\|u(X_t,t) - u_\theta(X_t,t)\|_2^2]\mathrm{d}t$ be the flow matching loss. Let p_T and q_T be the distributions induced by deployment-time ODE and training-time SDE inference, respectively (as in Theorem 4.1). We validate the bound in Theorem 4.1,

$$\mathrm{TV}(p_T||q_T) \leq O(\sqrt{\epsilon}).$$

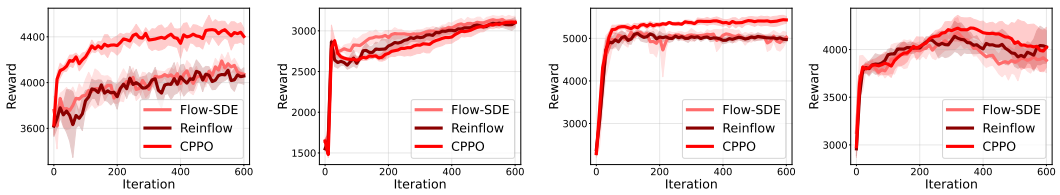
Data. We consider a synthetic setting in $\mathbb{R}^2$ with $p_0 = N(0, I_2)$ and $p_1 = N(\mu, \Sigma)$. With a linear probability path, the induced velocity field admits the closed-form expression

$$u^*(x,t) = \mu + (t\Sigma - (1-t)I)C_t^{-1}(x - t\mu),$$

where $\mu = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$, $\Sigma = \begin{bmatrix} 0.7 & 0.25 \\ 0.25 & 0.5 \end{bmatrix}$, $C_t = (1-t)^2 I + t^2\Sigma$.

Results. In Figure 1(a), we numerically validate Theorem 4.1 by training a velocity network u_θ to approximate the closed-form Gaussian-to-Gaussian velocity field u^*. We vary approximation quality through training iterations and compute the flow-matching error ϵ. We observe two-regime scaling consistent with the theoretical bound. Under the assumptions of Theorem 4.1, Section C gives

$$\mathrm{TV}(p_T||q_T) \leq C_{\mathrm{ODE}}\sqrt{\epsilon} + \frac{\sqrt{C_{\lambda,T}}}{2}\epsilon,$$



(a) Ant-v2 **(b) Hopper-v2** **(c) Humanoid-v3** **(d) Walker-v2**

Figure 2: Learning curves of evaluation reward on four OpenAI Gym MuJoCo environments: **(a) Ant-v2**, **(b) Hopper-v2**, **(c) Humanoid-v3**, and **(d) Walker-v2**. Solid lines show mean performance over 5 random seeds; shading indicates one standard deviation. FMPO-CPPO achieves higher reward than ReinFlow and Flow-SDE on Ant-v2, Humanoid-v3 and Walker-v2 and comparable performance on Hopper-v2.

where C_{ODE} and $C_{\lambda,T}$ are defined in Section C.2 and T is the early stopping time. For larger errors ($\epsilon > 0.14$), the linear ϵ term dominates, yielding $\mathrm{TV}(p_T||q_T) = O(\epsilon)$. The fitted log-log slope is approximately 1.06, consistent with this prediction. For smaller errors ($\epsilon \leq 0.14$), the $C_{\mathrm{ODE}}\sqrt{\epsilon}$ term dominates, yielding $\mathrm{TV} = O(\sqrt{\epsilon})$. The fitted slope converges to 0.53, matching the $O(\sqrt{\epsilon})$ rate predicted by Theorem 4.1. These results confirm that ODE–SDE TV distance scales as $O(\sqrt{\epsilon})$ in the fine-tuning regime where ϵ is small.

5.2 Validating Performance Transfer (Theorem 4.2)

Setup. We validate Theorem 4.2 and Remark 4.1 by fine-tuning pretrained base models spanning different flow-matching error levels. From each base model, we run flow matching RL fine-tuning via SDE inference and evaluate checkpoints under both training-time SDE and deployment-time ODE inference. Checkpoints are saved every 5 RL training steps to capture fine-grained reward transitions.

Data. We work in $\mathbb{R}^2$ with $p_0 = N(0, I_2)$ and $p_1 = N(\mu, \Sigma)$. With a linear path, the velocity is

$$u^*(x,t) = \mu + (t\Sigma - (1-t)I)C_t^{-1}(x - t\mu),$$

where $\mu = \begin{pmatrix} 2.5 \\ 2.5 \end{pmatrix}$, $\Sigma = \begin{bmatrix} 0.25 & 0.18 \\ 0.18 & 0.80 \end{bmatrix}$, $C_t = (1-t)^2 I + t^2\Sigma$. The terminal reward is bounded in $(0, 1]$ and is defined as a temperature-scaled Mahalanobis score:

$$r(x) = \exp\{-1.5(x-\mu)^\top \Sigma^{-1}(x-\mu)\}.$$

Metric. We use the violation rate to validate Theorem 4.2. For consecutive checkpoint pairs $(k, k+1)$, let $\Delta_{\mathrm{SDE}} = R_{\mathrm{SDE}}^{(k+1)} - R_{\mathrm{SDE}}^{(k)}$, $\Delta_{\mathrm{ODE}} = R_{\mathrm{ODE}}^{(k+1)} - R_{\mathrm{ODE}}^{(k)}$. The violation rate is:

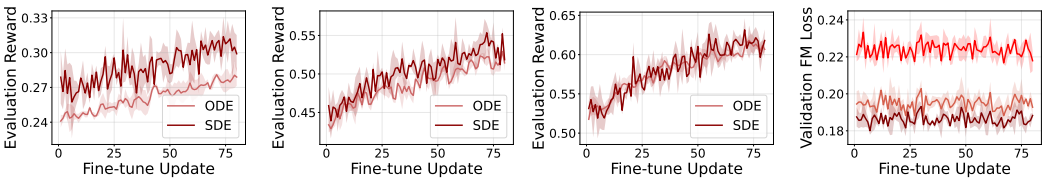
$$\text{Violation Rate} = \frac{\sum_k \mathbb{1}\{\Delta_{\mathrm{SDE}} > 0,\ \Delta_{\mathrm{ODE}} < 0\}}{\sum_k \mathbb{1}\{\Delta_{\mathrm{SDE}} > 0\}}.$$

Among pairs with SDE reward improvement (the one-step improvement condition of Theorem 4.2), the violation rate is the fraction with ODE reward decline. We report the violation rate for checkpoint pairs binned by $\sqrt{\epsilon}$.

Results. The inequality suggests that SDE-to-ODE transfer is more reliable when the flow-matching error ϵ is small, since smaller ϵ reduces ODE–SDE mismatch and the penalty term $R(\delta_k + \delta_{k+1})$. We therefore measure the ODE monotonicity violation rate conditioned on one-step SDE improvement, $\Pr(\Delta_{\mathrm{ODE}} < 0 \mid \Delta_{\mathrm{SDE}} > 0)$, and plot it against ϵ. Empirically, this violation probability increases with ϵ, supporting the insight from Theorem 4.2: SDE fine-tuning transfers more reliably to ODE for low-ϵ base models, while high-ϵ bases show weaker transfer and more frequent violations. We also observe the failure mode predicted by Remark 4.1 in Figure 1(c). For large-ϵ bases, the SDE reward improves during RL fine-tuning, but the ODE reward only oscillates around its initial value. In contrast, small-ϵ bases show effective SDE-to-ODE transfer. This confirms that the transfer guarantee requires small flow-matching error, a condition typical of fine-tuning but not training from scratch.

5.3 FMPO-CPPO Evaluation

Setup. We evaluate FMPO-CPPO, ReinFlow [16] and Flow-SDE [17] on four continuous-control locomotion benchmarks from OpenAI Gym [25]: Hopper, Walker2d, Ant, and Humanoid, ordered by increasing task difficulty. Each task is averaged over 5 random seeds. We put ablations in Section G.



(a) Weak **(b) Medium** **(c) Strong** **(d) Flow matching error**

Figure 3: **(a) Weak, (b) Medium, and (c) Strong:** ODE and SDE fine-tuning reward trajectories of the three pretrained base models. **(d) FM error:** Validation flow-matching error of these models during RL fine-tuning. The dark red, orange, and red lines represent Strong, Medium, and Weak, respectively. Curves show means over three independent seeds, and shaded bands show one standard deviation.

Results. As shown in Figure 2, FMPO-CPPO attains higher evaluation reward on Ant-v2, Walker-v2 and Humanoid-v3, while showing comparable performance on Hopper-v2. The gains are most pronounced on the higher-dimensional Humanoid-v3 and Ant-v2 tasks, where FMPO-CPPO converges to a stronger final policy and improves reward faster early in training.

5.4 Validating Correctness And Performance Transfer In The Image Generation Setting

Setup. We validate Theorem 4.1 and Theorem 4.2 on class-conditional CIFAR-10 generative fine-tuning. We use validation-set conditional flow matching loss as a substitute for the flow-matching error since the ground-truth velocity is unobtainable. We train three base models: **Weak**, **Medium**, **Strong** with validation flow-matching error of 0.225, 0.196, 0.184, respectively. We fine-tune the three base models with Flow-GRPO and evaluate ODE and SDE rewards after each update.

Results. The reward trajectories in Figure 3 exhibit the trend predicted by Theorem 4.1. The ODE-SDE performance gap decreases as pretraining quality improves. The trajectories show visible separation for the Weak model and substantial overlap for the Strong model. We compute the violation rate defined in Section 5.2 on seed-averaged trajectories. The violation rates are 0.467, 0.419, and 0.413 for Weak, Medium, and Strong, respectively, exhibiting the trend predicted by Theorem 4.2. Validation flow matching errors also fluctuate within a bounded range throughout fine-tuning (Figure 3(d)). This is consistent with the fine-tuning regime of our theories: SDE-based RL updates improve the reward without meaningfully drifting the underlying velocity field away from the pretrained one.

6 Discussion and Conclusion

We introduce a unified framework using SDE inference for flow matching RL fine-tuning (Section 3.1) and flow matching policy optimization (Section 3.2). We introduce FMPO-CPPO (Algorithm 1) as an FMPO instantiation and show that the framework subsumes several recent flow-based RL methods (Section 3.3), summarized in Table 1. Theoretically, we establish two key guarantees. First, we prove a correctness result: when the flow-matching error is small, training-time SDE and deployment-time ODE inference induce close terminal distributions (Theorem 4.1). Second, we derive a performance transfer guarantee: SDE-based policy improvements transfer to ODE-based deployment up to an error term controlled by the flow-matching error (Theorem 4.2). We numerically validate both theoretical predictions (Section 5.1 and Section 5.2). We also evaluate FMPO-CPPO, which achieves performance comparable to or better than existing baselines across representative continuous-control tasks (Section 5.3). Our analysis establishes sufficient conditions for SDE-to-ODE performance transfer, but does not prove their necessity. Future work will study RL effectiveness beyond fine-tuning, including training flow policies from scratch.

References

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

[2] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

[3] 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.

[4] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[5] Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, et al. Voicebox: Text-guided multilingual universal speech generation at scale. *Advances in neural information processing systems*, 36:14005–14034, 2023.

[6] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *CoRR*, 2024.

[7] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *CoRR*, 2024.

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

[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] Jiajun Fan, Shuaike Shen, Chaoran Cheng, Yuxin Chen, Chumeng Liang, and Ge Liu. Online reward-weighted fine-tuning of flow matching with wasserstein regularization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[11] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

[12] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

[13] Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

[14] Junzhe Li, Yutao Cui, Tao Huang, Yinping Ma, Chun Fan, Miles Yang, and Zhao Zhong. Mixgrpo: Unlocking flow-based grpo efficiency with mixed ode-sde. *arXiv preprint arXiv:2507.21802*, 2025.

[15] Xiaoxuan He, Siming Fu, Yuke Zhao, Wanli Li, Jian Yang, Dacheng Yin, Fengyun Rao, and Bo Zhang. Tempflow-grpo: When timing matters for grpo in flow models. *arXiv preprint arXiv:2508.04324*, 2025.

[16] Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *arXiv preprint arXiv:2505.22094*, 2025.

[17] Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, et al. π_{RL}: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025.

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

[19] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[20] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

[21] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[22] Ying Fan and Kangwook Lee. Optimizing ddpm sampling with shortcut fine-tuning. *arXiv preprint arXiv:2301.13362*, 2023.

[23] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

[24] 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.

[25] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[26] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[27] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[28] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

[29] Joe Benton, George Deligiannidis, and Arnaud Doucet. Error bounds for flow matching methods. *arXiv preprint arXiv:2305.16860*, 2023.

[30] Gen Li, Yuting Wei, Yuxin Chen, and Yuejie Chi. Towards faster non-asymptotic convergence for diffusion-based generative models. *arXiv preprint arXiv:2306.09251*, 2023.

[31] Maojiang Su, Jerry Yao-Chieh Hu, Sophia Pi, and Han Liu. On flow matching kl divergence. *arXiv preprint arXiv:2511.05480*, 2025.

[32] Ioannis Karatzas and Steven Shreve. *Brownian motion and stochastic calculus*. springer, 2014.

Appendix

A Related Works **14**

B Algorithms **16**

C Proof Of Theorem 4.1 **17**

C.1 Preliminaries 17

C.2 Main Proof Of Theorem 4.1 19

D Proof Of Theorem 4.2 **22**

E Connection Between ChainPPO And The Composite-MDP Objective **24**

F Additional Details Of Experiments **25**

F.1 Validating Correctness (Theorem 4.1) 25

F.2 Validating Performance Transfer (Theorem 4.2) 25

F.3 FMPO-CPPO Evaluation 26

F.4 Validating Correctness And Performance Transfer In The Image Generation Setting 27

G Ablation Of FMPO-CPPO **28**

Funding And Competing Interests Statement

This work received no external funding. The authors declare no competing interests.

Impact Statement

By the theoretical nature of this work, we do not anticipate any negative social impact.

LLM Usage Disclosure

We used large language models (LLMs) to aid and polish writing, such as improving clarity, grammar, and conciseness. We also used LLMs for retrieval and discovery, for example searching the literature to identify potential missing related work. All technical content, proofs, experiments, and results are original contributions by the authors.

A Related Works

Flow Matching And Flow Matching Policy. Flow matching [1–3] emerges as a powerful class of generative modeling techniques, achieving strong empirical performance across a wide range of domains, including image, speech, and video generation [4–6]. Recent works extend flow matching to policy learning [7], where the generative model represents a control policy or action distribution. Flow-based vision-language-action models demonstrate state-of-the-art performance on complex manipulation and control tasks [26, 27], highlighting the potential of flow-based policies.

Flow Matching Error Bounds. A line of recent work develops theoretical error bounds for fully deterministic flow matching models under different notions of distributional discrepancy. Early analyses focus on Wasserstein distances, where Albergo and Vanden-Eijnden [28] establish a 2-Wasserstein bound assuming uniform Lipschitz regularity of the velocity field, and Benton et al. [29] later relax this requirement to time-dependent regularity conditions. Beyond Wasserstein metrics, Li et al. [30] derive a total variation bound, but their result relies on the estimated score error being small. More recently, Su et al. [31] provide a non-asymptotic KL divergence bound that depends only on the regularity and smoothness of the data path and velocity fields, without imposing additional regularity assumptions on the estimated score path.

RL Fine-Tuning For Flow-Based Policies. Aligned with our main text, we consider two closely related settings: flow matching RL fine-tuning, where no external environment MDP is involved (e.g., image generation), and flow matching policy optimization, where the flow-based policy interacts with an environment MDP (e.g., robotics).

For flow matching RL fine-tuning, early work such as Fan et al. [10] adopts reward-weighted regression methods to fine-tune flow matching models with Wasserstein regularization. More recent approaches [12–15] introduce stochastic differential equation (SDE) inference during training and apply policy gradient methods with GRPO surrogate objectives. These methods are special instantiations of our flow matching RL fine-tuning framework under particular choices of surrogate objectives, inference windows, and noise schedules.

For flow matching policy optimization, which involves an external environment MDP, Park et al. [8] propose Flow Q-learning (FQL), an offline RL approach that trains a flow-based policy and distills it into a one-step policy for fine-tuning, inspired by diffusion Q-learning. Flow Policy Optimization (FPO) [9] reframes policy optimization as minimizing an advantage-weighted conditional flow matching loss. Wagenmaker et al. [11] perform reinforcement learning in the latent noise space rather than updating policy parameters. Zhang et al. [16] propose a policy gradient–based method that injects learnable noise during inference and reformulates the inference process as a Markov decision process.

While these policy gradient–based approaches demonstrate strong empirical performance, they lack a unified framework that characterizes their common structure and provides theoretical guarantees for transferring improvements from training-time stochastic inference to deployment-time deterministic inference. This work addresses this gap.

At the time of finalizing this paper, we became aware of a concurrent work [17], which proposes two approaches for flow matching policy optimization. One approach, Flow-SDE, shares the FMPO setting of SDE inference with a PPO surrogate, while Flow-Noise uses learnable noise injection within the denoising process.

B Algorithms

The algorithm summarizes rollout collection and actor updates using a supplied advantage estimator.

Algorithm 1 Flow Matching Policy Optimization with ChainPPO (FMPO-CPPO)

Require: Environment MDP $\mathcal{M}_{\text{env}} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$; pretrained velocity u_θ; denoising steps K; flow times $\tau_k = k/K$, $k = 0, \dots, K$; noise schedule λ_τ; chain-level clip ϵ_{c}; step-wise clip ϵ_{s}; auxiliary coefficient α; advantage estimator $\widehat{\mathcal{A}}$; learning rate η; rollout horizon T_{env}; rollouts per iteration N; training iterations M; PPO epochs E; action-chunk dimension $D = Hd_a$.

for training iteration $m = 1, \dots, M$ **do**
  $\theta_{\text{old}} \leftarrow \theta$, $\mathcal{D} \leftarrow \emptyset$
  for rollout $i = 1, \dots, N$ **do**
    Reset environment and observe s_0^i
    for $t = 0, \dots, T_{\text{env}} - 1$ **do**
      Sample initial noise $a_t^{0,i} \sim N(0, I)$
      for $k = 1, \dots, K$ **do**
        Sample $a_t^{k,i} \sim \overline{\pi}_{\theta_{\text{old}}}(\cdot \mid s_t^i, \tau_{k-1}, a_t^{k-1,i})$ by the Euler–Maruyama kernel evaluated at flow time τ_{k-1} with noise level $\lambda_{\tau_{k-1}}$
        Store $(s_t^i, a_t^{k-1,i}, a_t^{k,i}, k)$ in $\mathcal{D}$
      end for
      Execute $a_t^i \leftarrow a_t^{K,i}$ in the environment
      Observe $s_{t+1}^i \sim P(\cdot \mid s_t^i, a_t^i)$ and reward $R_t^i = R(s_t^i, a_t^i)$
      Store $(s_t^i, a_t^{0:K,i}, R_t^i)$ in $\mathcal{D}$
    end for
  end for
  Estimate environment-step advantages $\{\widehat{A}_t^i\} \leftarrow \widehat{\mathcal{A}}(\mathcal{D}, \{R_t^i\})$ and hold them fixed during actor updates
  for PPO epoch $e = 1, \dots, E$ **do**
    Compute normalized step-wise ratios

$$\widetilde{r}_{t,k}^i(\theta) = \left(\frac{\overline{\pi}_\theta(a_t^{k,i} \mid s_t^i, \tau_{k-1}, a_t^{k-1,i})}{\overline{\pi}_{\theta_{\text{old}}}(a_t^{k,i} \mid s_t^i, \tau_{k-1}, a_t^{k-1,i})} \right)^{1/D}.$$

    Compute normalized chain-level ratios

$$\widetilde{\rho}_t^i(\theta) = \prod_{k=1}^{K} \widetilde{r}_{t,k}^i(\theta).$$

    Form the chain-level PPO surrogate

$$\mathcal{J}_{\text{chain}}(\theta) = \widehat{\mathbb{E}}_{i,t} \left[\min \left\{ \widetilde{\rho}_t^i(\theta) \widehat{A}_t^i,\ \text{Clip}\left(\widetilde{\rho}_t^i(\theta), 1 - \epsilon_{\text{c}}, 1 + \epsilon_{\text{c}}\right) \widehat{A}_t^i \right\} \right].$$

    Form the auxiliary step-wise PPO surrogate

$$\mathcal{J}_{\text{step}}(\theta) = \widehat{\mathbb{E}}_{i,t,k} \left[\min \left\{ \widetilde{r}_{t,k}^i(\theta) \widehat{A}_t^i,\ \text{Clip}\left(\widetilde{r}_{t,k}^i(\theta), 1 - \epsilon_{\text{s}}, 1 + \epsilon_{\text{s}}\right) \widehat{A}_t^i \right\} \right].$$

    Update $\theta \leftarrow \theta + \eta \nabla_\theta \big(\mathcal{J}_{\text{chain}}(\theta) + \alpha \mathcal{J}_{\text{step}}(\theta) \big)$
  end for
end for

C Proof Of Theorem 4.1

We compare the learned ODE and SDE with the reference path generated by the true velocity. Section C.1 provides a Wasserstein stability bound, a density regularity bound, and a Wasserstein-to-TV interpolation inequality for the ODE comparison, together with a Girsanov bound for the SDE comparison. Section C.2 combines these bounds to prove Theorem 4.1.

C.1 Preliminaries

We start with the definition of total variation distance.

Definition C.1 (Total Variation Distance)**.** Let P and Q be probability measures on $\mathbb{R}^d$. Their total variation (TV) distance is

$$\mathrm{TV}(P,Q) := \sup_{A\in\mathcal{B}(\mathbb{R}^d)} |P(A)-Q(A)|.$$

If they have densities p and q with respect to Lebesgue measure, then

$$\mathrm{TV}(P,Q) = \frac{1}{2}\int_{\mathbb{R}^d} |p(x)-q(x)|\,\mathrm{d}x.$$

The SDE comparison uses the following consequence of Girsanov's theorem [32]. We include a localization argument for globally Lipschitz drifts.

Lemma C.1 (Girsanov's Theorem, [32])**.** Let Z and Z' be the unique non-explosive solutions of

$$\begin{aligned}\mathrm{d}Z_t &= b(Z_t,t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}W_t, \qquad Z_0\sim p_0,\\ \mathrm{d}Z'_t &= b'(Z'_t,t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}W_t, \qquad Z'_0\sim p_0,\end{aligned}$$

where $\sigma:[0,T]\to(0,\infty)$ is continuous. Let p_t, p'_t denote their marginal laws and $\mathbb{P}$ the path law of Z. Assume either Novikov's condition

$$\mathop{\mathbb{E}}_{\mathbb{P}}\left[\exp\left(\frac{1}{2}\int_0^T \frac{\|(b-b')(Z_t,t)\|_2^2}{\sigma^2(t)}\,\mathrm{d}t\right)\right] < \infty$$

holds, or both drifts are continuous, globally Lipschitz in space uniformly in time, and have at most linear growth uniformly in time. Then

$$\mathrm{KL}(p_T||p'_T) \le \frac{1}{2}\int_0^T \mathop{\mathbb{E}}_{Z_t\sim p_t}\left[\frac{\|(b-b')(Z_t,t)\|_2^2}{\sigma^2(t)}\right]\mathrm{d}t.$$

Proof. Under Novikov's condition, Girsanov's theorem gives the relative entropy between the path laws. The stated bound follows by applying the data processing inequality to the terminal evaluation map.

For globally Lipschitz drifts, let χ_n be a smooth function with $0\le\chi_n\le 1$, equal to 1 on the ball of radius n and supported in the ball of radius $2n$. Define $b'_n = b+\chi_n(b'-b)$, and let $p'_{n,T}$ be the terminal law under drift b'_n. Since $b'_n - b$ is bounded and σ has a positive lower bound, Novikov's condition holds for this comparison. Thus,

$$\begin{aligned}\mathrm{KL}(p_T||p'_{n,T}) &\le \frac{1}{2}\int_0^T \mathop{\mathbb{E}}_{p_t}\left[\frac{\chi_n^2(Z_t)\|(b-b')(Z_t,t)\|_2^2}{\sigma^2(t)}\right]\mathrm{d}t\\ &\le \frac{1}{2}\int_0^T \mathop{\mathbb{E}}_{p_t}\left[\frac{\|(b-b')(Z_t,t)\|_2^2}{\sigma^2(t)}\right]\mathrm{d}t.\end{aligned}$$

Couple the processes with drifts b'_n and b' using the same initial variable and Brownian motion. Their paths agree until they leave the ball of radius n, by pathwise uniqueness. Since the process with drift

b' is non-explosive,

$$\mathrm{TV}(p'_{n,T}, p'_T) \le \Pr\left(\sup_{0\le t\le T} \|Z'_t\|_2 \ge n\right) \longrightarrow 0.$$

Lower semicontinuity of relative entropy then gives the same bound for p'_T. □

For the ODE comparison, we first recall the Wasserstein stability estimate for flow matching. For $r \in \{1, 2\}$, write

$$W_r(p, q) := \inf_{\pi\in\Pi(p,q)} \left(\int \|x - y\|_2^r \, \pi(\mathrm{d}x, \mathrm{d}y)\right)^{1/r},$$

where $\Pi(p, q)$ is the set of couplings of p and q. The following is the time-rescaled stability estimate of Benton et al. [29, Theorem 1]. We give the coupling proof for completeness.

Lemma C.2 (Wasserstein Bound For Flow Matching)**.** Let $u, v \in C([0, T]; C^1(\mathbb{R}^d; \mathbb{R}^d))$, and let X_t, Y_t be their ODE flows with the same initial law of finite second moment. Assume that both flows exist on $[0, T]$, that $v(\cdot, t)$ is globally L-Lipschitz, and that

$$\int_0^T \mathop{\mathbb{E}}_{X_t\sim p_t^\star}[\|u(X_t, t) - v(X_t, t)\|_2^2] \, \mathrm{d}t \le \epsilon^2,$$

where $p_t^\star$ is the marginal law of X_t. If p_T^v is the law of Y_T, then

$$W_1(p_T^\star, p_T^v) \le W_2(p_T^\star, p_T^v) \le \sqrt{T} e^{LT} \epsilon.$$

Proof. Couple the two ODE flows using $X_0 = Y_0$. By the Lipschitz condition on v and Gronwall's inequality,

$$\|Y_T - X_T\|_2 \le \int_0^T e^{L(T-s)} \|v(X_s, s) - u(X_s, s)\|_2 \, \mathrm{d}s.$$

Cauchy–Schwarz gives

$$\begin{aligned}\mathbb{E}\,\|Y_T - X_T\|_2^2 &\le T e^{2LT} \int_0^T \mathop{\mathbb{E}}_{p_s^\star}[\|v(X_s, s) - u(X_s, s)\|_2^2] \, \mathrm{d}s \\ &\le T e^{2LT} \epsilon^2.\end{aligned}$$

The conclusion follows from the definition of W_2 and $W_1 \le W_2$. □

To convert this Wasserstein bound into a TV bound, we control the spatial variation of the terminal densities. For a density p, define

$$S(p) := \|\nabla p\|_{L^1} = \int_{\mathbb{R}^d} \|\nabla p(x)\|_2 \, \mathrm{d}x.$$

Lemma C.3 (Density Regularity Under An ODE Flow)**.** Let $v \in C([0, T]; C^2(\mathbb{R}^d; \mathbb{R}^d))$ satisfy $\|D_x v\|_{\mathrm{op}} \le L$ and $\|\nabla_x \operatorname{div} v\|_2 \le H$. Let p_t^v be the density induced by its ODE flow from $p_0 = \mathcal{N}(0, I_d)$. Then

$$S(p_T^v) \le e^{LT} \, \mathbb{E}\,\|G\|_2 + H \int_0^T e^{L(T-s)} \, \mathrm{d}s, \qquad G \sim \mathcal{N}(0, I_d).$$

Proof. The Lipschitz condition and time continuity give a global flow F_t that is C^2 in the initial state. The change-of-variables formula gives a positive C^1 density p_t^v. Let $g_t = \nabla \log p_t^v$. Differentiating

the change-of-variables formula along the flow yields

$$\frac{\mathrm{d}}{\mathrm{d}t} g_t(F_t(a)) = -D_x v(F_t(a), t)^\top g_t(F_t(a)) - \nabla_x \operatorname{div} v(F_t(a), t).$$

Since $g_0(a) = -a$, Gronwall's inequality gives

$$\|g_T(F_T(a))\|_2 \le e^{LT}\|a\|_2 + H \int_0^T e^{L(T-s)}\,\mathrm{d}s.$$

Integrating over $a \sim p_0$ and using $S(p_T^v) = \mathbb{E}_{p_T^v} \|g_T\|_2$ proves the claim. □

The next lemma is the standard smoothing argument that relates Wasserstein distance to TV for densities with integrable gradients.

Lemma C.4 (Wasserstein-To-TV Interpolation)**.** Let $p, q \in W^{1,1}(\mathbb{R}^d)$ be probability densities with finite first moments. Then

$$\mathrm{TV}(p, q) \le \sqrt{\frac{2}{\pi}} \sqrt{W_1(p, q)\big(S(p) + S(q)\big)}.$$

Proof. Let γ_h be the density of $\mathcal{N}(0, h^2 I_d)$, let $G \sim \mathcal{N}(0, I_d)$, and set $c = \sqrt{2/\pi}$. We first prove the estimates for smooth densities; convolution approximation extends them to $W^{1,1}$, since it preserves the gradient bounds and converges in both L^1 and W_1. The fundamental theorem of calculus, translation invariance, and $\mathbb{E}\,|G^\top z| = c\|z\|_2$ give

$$\|p - p * \gamma_h\|_{L^1} \le h \int_{\mathbb{R}^d} \mathbb{E}\,|G^\top \nabla p(x)|\,\mathrm{d}x = ch S(p).$$

The same bound holds for q. Moreover, for any coupling $\pi \in \Pi(p, q)$,

$$\begin{aligned} \|p * \gamma_h - q * \gamma_h\|_{L^1} &\le \int \|\gamma_h(\cdot - x) - \gamma_h(\cdot - y)\|_{L^1}\,\pi(\mathrm{d}x, \mathrm{d}y) \\ &\le \frac{c}{h} \int \|x - y\|_2\,\pi(\mathrm{d}x, \mathrm{d}y). \end{aligned}$$

Here the second inequality follows from $\|\partial_e \gamma_h\|_{L^1} = c/h$ for every unit vector e. Taking the infimum over couplings and applying the triangle inequality yields

$$\mathrm{TV}(p, q) \le \frac{c}{2}\left(h\big(S(p) + S(q)\big) + \frac{W_1(p, q)}{h}\right).$$

Optimizing over $h > 0$ proves the stated bound. □

C.2 Main Proof Of Theorem 4.1

Theorem C.1 (Formal Version Of Theorem 4.1)**.** Let $0 < T < 1$ and assume that p_1 has a finite second moment. Let $u \in C([0, T]; C^1(\mathbb{R}^d; \mathbb{R}^d))$ be the true velocity of the independent linear path, and assume that $u(\cdot, t)$ is globally Lipschitz uniformly in t. Let $u_\theta \in C([0, T]; C^2(\mathbb{R}^d; \mathbb{R}^d))$ be the estimated velocity satisfying

$$\|D_x u_\theta(x, t)\|_{\mathrm{op}} \le L, \qquad \|\nabla_x \operatorname{div} u_\theta(x, t)\|_2 \le H$$

for all $x \in \mathbb{R}^d$ and $t \in [0, T]$, where $L, H < \infty$ are uniform over the models under consideration. Let W_t be a standard Wiener process and let the noise schedule $\lambda : [0, T] \to (0, \infty)$ be continuous.

For parameter θ, we define three processes X_t, Y_t^θ, and Z_t^θ by

$$\begin{aligned} \mathrm{d}X_t &= u(X_t, t)\mathrm{d}t, \quad \mathrm{d}Y_t^\theta = u_\theta(Y_t^\theta, t)\mathrm{d}t, \\ \mathrm{d}Z_t^\theta &= \left[\left(1 + \frac{\lambda_t t}{2(1-t)}\right) u_\theta(Z_t^\theta, t) \right. \\ &\qquad \left. - \frac{\lambda_t}{2(1-t)} Z_t^\theta\right] \mathrm{d}t + \sqrt{\lambda_t}\mathrm{d}W_t. \end{aligned}$$

We assume that u_θ is close to u,

$$\int_0^T \mathop{\mathbb{E}}_{X_t \sim p_t^\star} [\|u(X_t, t) - u_\theta(X_t, t)\|_2^2]\mathrm{d}t \le \epsilon^2. \tag{C.1}$$

Let $p_t^\star$, p_t, and q_t denote the probability paths induced by X_t, Y_t^θ, and Z_t^θ, respectively. All processes share the same initial distribution $p_0 = \mathcal{N}(0, I_d)$. Then the terminal distributions satisfy

$$\mathrm{TV}(p_T || q_T) \le C_{\mathrm{ODE}}\sqrt{\epsilon} + \frac{\sqrt{C_{\lambda,T}}}{2}\epsilon = O(\sqrt{\epsilon})$$

as $\epsilon \to 0$, where C_{ODE} and $C_{\lambda,T}$ depend only on d, T, L, H and the noise schedule, and are defined in Section C.2.

Proof. We first compare the learned ODE with the reference ODE. Let ρ_T be the law of TX_1, let γ_h be the density of $\mathcal{N}(0, h^2 I_d)$, and write $c_d = \mathbb{E}\,\|G\|_2$ for $G \sim \mathcal{N}(0, I_d)$. The independent linear path gives $p_T^\star = \rho_T * \gamma_{1-T}$. Therefore,

$$S(p_T^\star) \le \|\nabla \gamma_{1-T}\|_{L^1} = \frac{c_d}{1-T}.$$

By Lemma C.3, the learned ODE density satisfies

$$S(p_T) \le e^{LT} c_d + H \int_0^T e^{L(T-s)}\,\mathrm{d}s.$$

Define

$$\begin{aligned} B_T &:= \frac{c_d}{1-T} + e^{LT} c_d + H \int_0^T e^{L(T-s)}\,\mathrm{d}s, \\ C_{\mathrm{ODE}} &:= \sqrt{\frac{2}{\pi}}\, T^{1/4} e^{LT/2} \sqrt{B_T}. \end{aligned}$$

All terminal laws have finite second moments by the linear-growth bounds on the velocities. Combining Lemmas C.2 and C.4 gives

$$\begin{aligned} \mathrm{TV}(p_T^\star, p_T) &\le \sqrt{\frac{2}{\pi}} \sqrt{W_1(p_T^\star, p_T) B_T} \\ &\le C_{\mathrm{ODE}} \sqrt{\epsilon}. \end{aligned}$$

We next compare the learned SDE with the reference path. For the true velocity u, (2.6) gives

$$\nabla \log p_t^\star(x) = \frac{t\, u(x,t) - x}{1-t},$$

with the value at $t = 0$ given by continuity. Thus, the reference ODE has a dual stochastic process

$$\mathrm{d}\widetilde{X}_t = \left[\left(1 + \frac{\lambda_t t}{2(1-t)}\right) u(\widetilde{X}_t, t) - \frac{\lambda_t}{2(1-t)} \widetilde{X}_t\right] \mathrm{d}t + \sqrt{\lambda_t}\,\mathrm{d}W_t,$$

with initial law p_0 and the same marginal laws $p_t^\star$. Indeed, its Fokker–Planck equation reduces to the reference continuity equation by (2.6). Uniqueness of the Fokker–Planck equation for globally Lipschitz drifts identifies its marginal laws with $p_t^\star$. Both SDE drifts are globally Lipschitz with linear growth, since $T < 1$ and λ is continuous and positive on $[0, T]$. Consequently, Lemma C.1 applies and gives

$$\mathrm{KL}(p_T^\star || q_T) \le \frac{1}{2} \int_0^T \frac{1}{\lambda_t} \left(1 + \frac{\lambda_t t}{2(1-t)}\right)^2 \mathop{\mathbb{E}}_{p_t^\star} \|(u - u_\theta)(X_t, t)\|_2^2 \, \mathrm{d}t.$$

Define

$$C_{\lambda,T} := \operatorname*{ess\,sup}_{t \in [0,T]} \frac{1}{\lambda_t} \left(1 + \frac{\lambda_t t}{2(1-t)}\right)^2.$$

This coefficient is finite under the noise-schedule assumption. For constant noise $\lambda_t \equiv \lambda > 0$, it equals $\lambda^{-1}(1 + \lambda T/(2(1-T)))^2$. Using (C.1) and Pinsker's inequality, we obtain

$$\begin{aligned} \mathrm{KL}(p_T^\star || q_T) &\le \frac{C_{\lambda,T}}{2} \epsilon^2, \\ \mathrm{TV}(p_T^\star, q_T) &\le \frac{\sqrt{C_{\lambda,T}}}{2} \epsilon. \end{aligned}$$

Finally, the triangle inequality yields

$$\begin{aligned} \mathrm{TV}(p_T || q_T) &\le \mathrm{TV}(p_T, p_T^\star) + \mathrm{TV}(p_T^\star, q_T) \\ &\le C_{\mathrm{ODE}} \sqrt{\epsilon} + \frac{\sqrt{C_{\lambda,T}}}{2} \epsilon = O(\sqrt{\epsilon}). \end{aligned}$$

The constants are independent of θ and ϵ when L, H and the noise schedule are fixed. This completes the proof. □

Remark C.1 (Verification Of The Regularity Conditions)**.** The reference-velocity assumption holds for Gaussian targets and for bounded targets on every fixed interval $[0, T]$ with $T < 1$. For Gaussian targets, the reference velocity is affine in x, as in Section 5.1. For bounded targets with $\|X_1\|_2 \le R_1$, define $m_t(x) = \mathbb{E}[X_1 \mid X_t = x]$ along the independent linear path. Gaussian conditioning gives

$$\begin{aligned} u(x,t) &= \frac{m_t(x) - x}{1-t}, \\ D_x m_t(x) &= \frac{t}{(1-t)^2} \mathrm{Cov}(X_1 \mid X_t = x). \end{aligned}$$

Since the conditional covariance has operator norm at most R_1^2, the reference velocity is smooth in space and satisfies

$$\|D_x u(x,t)\|_{\mathrm{op}} \le \frac{1}{1-T} + \frac{T R_1^2}{(1-T)^3}, \qquad t \le T.$$

The learned-velocity assumptions hold for each fixed finite-parameter SiLU MLP used in the synthetic experiments. To see this, write $h_\ell = \sigma(W_\ell h_{\ell-1} + b_\ell)$ and let $\beta_1 = \sup_z |\sigma'(z)| < \infty$ and $\beta_2 = \sup_z |\sigma''(z)| < \infty$. If J_ℓ, K_ℓ bound the first and second spatial derivatives of h_ℓ, respectively, the chain rule gives

$$\begin{aligned} J_\ell &\le \beta_1 \|W_\ell\|_{\mathrm{op}} J_{\ell-1}, \\ K_\ell &\le \beta_2 \|W_\ell\|_{\mathrm{op}}^2 J_{\ell-1}^2 + \beta_1 \|W_\ell\|_{\mathrm{op}} K_{\ell-1}. \end{aligned}$$

Starting from $J_0 = 1, K_0 = 0$, finite depth and finite weights give finite derivative bounds, and the linear output layer preserves them. A bound K on the second derivative of the output gives

$\|\nabla \operatorname{div} u_\theta\|_2 \le dK$. For a fixed architecture with uniformly bounded layer weights, these recursions give uniform L, H over the model family. For a single model, the bound remains valid with model-dependent L_θ, H_θ and $C_{\mathrm{ODE},\theta}$.

D Proof Of Theorem 4.2

This section provides the proof of Theorem 4.2.

Theorem D.1 (Theorem 4.2 Restated: Transfer of improvement from SDE to ODE). Let $r : \mathbb{R}^d \to \mathbb{R}$ be the terminal reward. Assume the terminal reward is bounded, i.e., $|r(x)| < R < \infty$ for any $x \in \mathbb{R}^d$. Fix early stopping time $0 < T < 1$ and noise level $\lambda_t > 0$. For each parameter θ, we define two processes:

- The SDE (training-time) inference process $Z^\theta = (Z_t^\theta)_{t\in[0,T]}$ by

$$\mathrm{d}Z_t^\theta = [(1 + \frac{\lambda_t}{2}\frac{t}{1-t})u_\theta(Z_t^\theta, t) - (\frac{\lambda_t}{2}\frac{1}{1-t})Z_t^\theta]\mathrm{d}t + \sqrt{\lambda_t}\mathrm{d}W_t, \quad Z_0^\theta \sim p_0.$$

- The ODE (deployment-time) inference process $Y^\theta = (Y_t^\theta)_{t\in[0,T]}$ by

$$\mathrm{d}Y_t = u_\theta(Y_t^\theta, t)\mathrm{d}t, \quad Y_0^\theta \sim p_0.$$

Define the corresponding performances

$$J_{\mathrm{SDE}}(\theta) := \mathbb{E}[r(Z_T^\theta)], \quad J_{\mathrm{ODE}}(\theta) := \mathbb{E}[r(Y_T^\theta)].$$

Let $\{\theta_k\}_{k\ge0}$ be a sequence of parameters produced by a reinforcement learning algorithm that satisfies the following one-step improvement condition on the SDE performance: for some $\Delta_k > 0$,

$$J_{\mathrm{SDE}}(\theta_{k+1}) \ge J_{\mathrm{SDE}}(\theta_k) + \Delta_k.$$

Assume that the hypotheses of Theorem C.1 hold for every θ_k with the same constants L, H and noise schedule. Then the ODE performance satisfies the transfer improvement bound

$$J_{\mathrm{ODE}}(\theta_{k+1}) \ge J_{\mathrm{ODE}}(\theta_k) + \Delta_k - 2R(\delta_k + \delta_{k+1}),$$

where $\delta_k := C_{\mathrm{ODE}}\sqrt{\epsilon_k} + \frac{\sqrt{C_{\lambda,T}}}{2}\epsilon_k = O(\sqrt{\epsilon_k})$ is induced by the flow-matching error $\epsilon_k := \sqrt{\mathcal{L}_{\mathrm{FM},T}(\theta_k)}$.

Proof. Let q_t^k, and p_t^k denote the probability paths induced by $Z_t^{\theta_k}$, and $Y_t^{\theta_k}$, respectively. Then from the correctness for flow RL via diffusion-flow duality (Theorem C.1), we have for any $k \ge 0$

$$\mathrm{TV}(p_T^k || q_T^k) \le \delta_k. \tag{D.1}$$

Here $\delta_k = C_{\mathrm{ODE}}\sqrt{\epsilon_k} + \frac{\sqrt{C_{\lambda,T}}}{2}\epsilon_k = O(\sqrt{\epsilon_k})$ is induced by the flow-matching error

$$\epsilon_k^2 := \int_0^T \mathop{\mathbb{E}}_{X_t\sim p_t^\star}[\|u(X_t,t) - u_{\theta_k}(X_t,t)\|_2^2]\mathrm{d}t,$$

where u is the true (constructed) velocity and p^* is the probability path induced by u.

A standard property of total variation distance is that for any two distributions P, Q on $\mathbb{R}^d$ and any bounded measurable function $f : \mathbb{R}^d \to \mathbb{R}$,

$$\left| \mathop{\mathbb{E}}_{X\sim P}[f(X)] - \mathop{\mathbb{E}}_{X\sim Q}[f(X)] \right| \le 2\|f\|_\infty \mathrm{TV}(P||Q).$$

Let f be the terminal reward function r (bounded by R). Then by (D.1), we have

$$\left| \mathop{\mathbb{E}}_{X \sim p_T^k}[r(X)] - \mathop{\mathbb{E}}_{X \sim q_T^k}[r(X)] \right| \leq 2R \cdot \mathrm{TV}(p_T^k || q_T^k) \leq 2R\delta_k, \tag{D.2}$$

$$\left| \mathop{\mathbb{E}}_{X \sim p_T^{k+1}}[r(X)] - \mathop{\mathbb{E}}_{X \sim q_T^{k+1}}[r(X)] \right| \leq 2R \cdot \mathrm{TV}(p_T^{k+1} || q_T^{k+1}) \leq 2R\delta_{k+1}. \tag{D.3}$$

Next, by the one-step improvement of the SDE performance assumption,

$$J_{\mathrm{SDE}}(\theta_{k+1}) \geq J_{\mathrm{SDE}}(\theta_k) + \Delta_k \quad \text{for some } \Delta_k \geq 0. \tag{D.4}$$

We now prove the one-step improvement bound for ODE performance by applying these inequalities:

$$\begin{aligned} J_{\mathrm{ODE}}(\theta_{k+1}) &\geq J_{\mathrm{SDE}}(\theta_{k+1}) - 2R\delta_{k+1}, && \big(\text{By (D.3)}\big) \\ &\geq J_{\mathrm{SDE}}(\theta_k) + \Delta_k - 2R\delta_{k+1}, && \big(\text{By (D.4)}\big) \\ &\geq J_{\mathrm{ODE}}(\theta_k) + \Delta_k - 2R(\delta_{k+1} + \delta_k). && \big(\text{By (D.2)}\big) \end{aligned}$$

This completes the proof. □

E Connection Between ChainPPO And The Composite-MDP Objective

We first show that treating the full inference chain as a macro-action does not change the original objective. We then analyze the policy-gradient directions induced by the chain-level and step-wise PPO surrogates before clipping. Throughout this section, J_{chain} and J_{step} refer to the corresponding unclipped surrogate objectives. They are used to compare gradient directions with the true composite-MDP objective, whereas the clipped versions are used in practice for trust-region control.

Proposition E.1 (Macro-action objective equivalence)**.** Consider the composite MDP in Section 3.2. At each environment step t, let

$$c_t := (a_t^0, a_t^1, \dots, a_t^K)$$

denote the full stochastic inference chain, where $a_t^0 \sim \mu_0 = N(0, I)$ and

$$\Pi_\theta(c_t \mid s_t) = \mu_0(a_t^0) \prod_{k=1}^{K} p^\theta_{\tau_{k-1}}(a_t^k \mid a_t^{k-1}, s_t).$$

Define the macro-action reward and transition kernel by

$$R_{\text{chain}}(s_t, c_t) := R(s_t, a_t^K), \qquad P_{\text{chain}}(s_{t+1} \mid s_t, c_t) := P(s_{t+1} \mid s_t, a_t^K).$$

Let

$$\mathcal{J}_{\text{macro}}(\theta) := \mathop{\mathbb{E}}_{\Pi_\theta, P_{\text{chain}}} \left[\sum_{t \geq 0} \gamma^t R_{\text{chain}}(s_t, c_t) \right]$$

be the expected return of this macro-action MDP. Then

$$\mathcal{J}_{\text{macro}}(\theta) = \mathcal{J}_{\text{com}}(\theta) = \mathcal{J}_{\text{env}}(\theta).$$

Proof. By definition, the macro-action c_t generates the executed environment action $a_t = a_t^K$. Thus the macro-action reward satisfies

$$R_{\text{chain}}(s_t, c_t) = R(s_t, a_t^K),$$

which is exactly the reward assigned to the terminal inner step of the composite MDP. The macro-action transition kernel also matches the environment transition after executing a_t^K:

$$P_{\text{chain}}(s_{t+1} \mid s_t, c_t) = P(s_{t+1} \mid s_t, a_t^K).$$

Therefore, marginalizing over the intermediate inference variables $a_t^0, \dots, a_t^{K-1}$ gives the same distribution over environment trajectories $(s_t, a_t^K, s_{t+1})_{t \geq 0}$ as the original flow policy. Hence the macro-action MDP and the environment MDP have the same expected return:

$$\mathcal{J}_{\text{macro}}(\theta) = \mathcal{J}_{\text{env}}(\theta).$$

The equality $\mathcal{J}_{\text{com}}(\theta) = \mathcal{J}_{\text{env}}(\theta)$ follows from the construction of the composite MDP in Section 3.2. Thus

$$\mathcal{J}_{\text{macro}}(\theta) = \mathcal{J}_{\text{com}}(\theta) = \mathcal{J}_{\text{env}}(\theta).$$

□

F Additional Details Of Experiments

F.1 Validating Correctness (Theorem 4.1)

Model And Training Details. We approximate the velocity field $u_\theta(x,t)\colon \mathbb{R}^d \times [0,T] \to \mathbb{R}^d$ with a multilayer perceptron. The input $(x,t) \in \mathbb{R}^{d+1}$ is formed by concatenating the spatial state $x \in \mathbb{R}^d$ with the scalar time $t \in [0,T]$. The network has three hidden layers of width 96 with SiLU activations $\sigma(z) = z\,\mathrm{sigmoid}(z)$, i.e. the architecture is

$$(d{+}1) \xrightarrow{\text{Linear+SiLU}} 96 \xrightarrow{\text{Linear+SiLU}} 96 \xrightarrow{\text{Linear+SiLU}} 96 \xrightarrow{\text{Linear}} d,$$

so that the output layer is linear and matches the dimension of the velocity field.

We train u_θ by marginal flow-matching regression, using the closed-form target below, along the linear interpolation path $X_t = (1-t)X_0 + tX_1$ with independent coupling $X_0 \sim \mathcal{N}(0, I_d)$ and $X_1 \sim \mathcal{N}(\mu, \Sigma)$. At each iteration we draw a fresh mini-batch of size 512 by sampling $t \sim \mathcal{U}(0,T)$ with $T = 0.99$, independent X_0, X_1 from the marginals above, and forming X_t on-the-fly; the empirical loss is the per-coordinate mean squared error between $u_\theta(X_t,t)$ and the target velocity. Because (X_0, X_1) is jointly Gaussian, the Bayes-optimal velocity

$$u^\star(x,t) = \mathbb{E}[X_1 - X_0 \mid X_t = x] = \mu + \big(t\Sigma + (t-1)I\big)\big((1-t)^2 I + t^2\Sigma\big)^{-1}(x - t\mu)$$

is available in closed form; we use it as the regression target, which has the same minimizer and the same expected gradient as the standard CFM loss but lower variance, allowing us to control the training error ϵ precisely. Optimization uses Adam with learning rate 10^{-3}, default momentum parameters $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon_{\text{Adam}} = 10^{-8}$, and no learning-rate schedule. The number of optimizer steps is varied over 20 values evenly spaced between 100 and 4000 to sweep out a range of approximation errors ϵ, and each configuration is repeated over 10 independent seeds (controlling network initialization, data sampling, and integrator noise) to produce the error bars in Figure 1(a).

F.2 Validating Performance Transfer (Theorem 4.2)

Model And Training Details. The base policy $u_\theta(x,t)\colon \mathbb{R}^d \times [0,T] \to \mathbb{R}^d$ is parameterized by a multilayer perceptron that takes the concatenated input $(x,t) \in \mathbb{R}^{d+1}$ and returns the velocity field. The network has three hidden layers of width 128 with SiLU activations $\sigma(z) = z\,\mathrm{sigmoid}(z)$ and a linear readout, i.e.

$$(d{+}1) \xrightarrow{\text{Linear+SiLU}} 128 \xrightarrow{\text{Linear+SiLU}} 128 \xrightarrow{\text{Linear+SiLU}} 128 \xrightarrow{\text{Linear}} d,$$

Both pretraining and fine-tuning phases use the time horizon $T = 0.99$. We pretrain u_θ by marginal flow-matching regression along the linear interpolation path $X_t = (1-t)X_0 + tX_1$ with independent coupling $X_0 \sim \mathcal{N}(0, I_d)$ and $X_1 \sim \mathcal{N}(\mu_{\text{tgt}}, \Sigma_{\text{tgt}})$, where $\mu_{\text{tgt}} = (2.5, 2.5)^\top$ and $\Sigma_{\text{tgt}} = \Sigma_{\text{base}}/\tau$ with $\Sigma_{\text{base}} = \left(\begin{smallmatrix} 0.25 & 0.18 \\ 0.18 & 0.80 \end{smallmatrix}\right)$ and reward temperature $\tau = 3.0$. At each iteration we draw a fresh mini-batch of 2048 pairs by sampling $t \sim \mathcal{U}(0,T)$ and independent X_0, X_1 from the marginals above, form X_t on-the-fly, and minimize

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,\,X_t}\big\|u_\theta(X_t,t) - u^\star(X_t,t)\big\|_2^2,$$

Pretraining uses Adam with learning rate 2×10^{-3}, batch size 2048, default momentum $(\beta_1, \beta_2) = (0.9, 0.999)$, no schedule, and is run for 900 steps; checkpoints are logged every 45 steps to expose a family of pretrained velocities spanning a range of ϵ values, from which we select three starts (worst / middle / best) for the trajectory figure.

The reward is a temperature-scaled, log-clamped Mahalanobis kernel,

$$r(x) = \exp\Big(\max\big\{-0.5\,\tau\,(x-\mu_{\text{tgt}})^\top \Sigma_{\text{base}}^{-1}(x-\mu_{\text{tgt}}),\, L_{\min}\big\}\Big) \in (0,1],$$

with $\tau = 3.0$ and lower log-clamp $L_{\min} = -40$. With this choice, the implied target distribution is exactly $\mathcal{N}(\mu_{\text{tgt}}, \Sigma_{\text{base}}/\tau)$, aligning the pretraining and fine-tuning objectives.

During RL fine-tuning, $\lambda = 1$, the number of Euler–Maruyama steps is 100. Optimization uses Adam with learning rate 5×10^{-5}, batch size 256, gradient-norm clipping at 1.0, and is run for 300 steps; checkpoints are taken every 5 steps to give the fine-grained reward trajectories shown in Fig. 1(c). At each checkpoint we evaluate two test-time samplers from the same $X_0 \sim \mathcal{N}(0, I_d)$ (common random numbers across configurations to stabilize differences): (i) the deterministic ODE $\mathrm{d}Y_t = u_\theta(Y_t, t)\,\mathrm{d}t$, and (ii) the same training-time SDE with $\lambda = 1$. Both are integrated with 400 Euler / Euler–Maruyama steps on 20000 initial samples, yielding the reported J_{ODE} and J_{SDE}. The flow-matching error $\epsilon = \sqrt{\int_0^T \mathbb{E}\,\|u_\theta - u^\star\|^2\,\mathrm{d}t}$ is estimated on 20000 fresh (t, X_t) samples and is used only as a diagnostic.

F.3 FMPO-CPPO Evaluation

We use the same settings as ReinFlow. Below are the implementation details of ReinFlow.

ReinFlow Model And Training Details. The ReinFlow policy is a conditional rectified-flow velocity field $u_\theta(x_t, t, o)$, where $x_t \in \mathbb{R}^{Hd_a}$ is an action chunk, $t \in [0, 1)$ is the flow time, and o is the normalized observation history. It uses action chunking horizon $H = 4$ and one observation frame. The flow network embeds t with a sinusoidal embedding of dimension 16, followed by a two-layer time MLP. For the Gym locomotion tasks, the policy trunk is a residual MLP with three hidden layers of width 512 and ReLU activations, with a linear output of dimension Hd_a:

$$[x_t, \phi(t), o] \to 512 \to 512 \to 512 \to Hd_a.$$

The critic is a separate residual MLP over observations with three hidden layers of width 256 and Mish activations.

The policy is first pretrained by rectified-flow matching on offline action chunks. Given a dataset action chunk x_1 and Gaussian noise $x_0 \sim \mathcal{N}(0, I)$, it samples $t \sim \mathcal{U}(0, 1)$ and forms

$$x_t = (1 - t)x_0 + tx_1, \qquad v = x_1 - x_0.$$

The pretraining objective is the mean squared error

$$\mathcal{L}_{\text{RF}}(\theta) = \mathbb{E}\left[\|u_\theta(x_t, t, o) - (x_1 - x_0)\|_2^2\right].$$

Pretraining uses AdamW with learning rate 10^{-3}, weight decay 10^{-6}, batch size 128, cosine warmup scheduling with minimum learning rate 10^{-4}, and EMA decay 0.995.

For online fine-tuning, ReinFlow uses PPO on the action chunks generated by Euler integration of the learned velocity field with $K = 4$ denoising steps. It runs 40 parallel environments, collects 500 rollout steps per iteration, uses discount $\gamma = 0.99$ and GAE parameter $\lambda = 0.95$, and updates for 5 PPO epochs. The PPO clipping coefficient is $\epsilon = 0.01$, the value loss coefficient is 0.5. Actor and critic are optimized with AdamW; the default Gym actor learning rate is 4.5×10^{-5} for Hopper, Ant, and Humanoid, and 4×10^{-4} for Walker2d, while the critic learning rate is 6.5×10^{-4} except Walker2d, which uses 4×10^{-3}. Both use cosine warmup schedules and critic weight decay 10^{-5}.

Flow-SDE Training Details: For Flow-SDE, all settings are the same as ReinFlow except the constant coefficient a of noise level λ_t. Following ReinFlow, the PPO ratio is computed from the chain log-likelihood divided by $(K + 1)Hd_a$. For each task, we search a from $\{0.1, 0.2, \ldots, 0.9\}$ and select the best a for each task. As a result, a is 0.5,0.9,0.4,0.5 for Hopper, Walker2d, Ant, and Humanoid respectively.

FMPO-CPPO Training Details: For FMPO-CPPO, all settings are the same as ReinFlow unless otherwise specified. We additionally set the auxiliary weight α to 0.25 and the chained PPO clip coefficient to twice the PPO clip coefficient used by ReinFlow. The likelihood ratios are normalized by the action-chunk dimension $D = Hd_a$ as in Section 3.3, so $\epsilon_{\text{c}} = 0.02$ and $\epsilon_{\text{s}} = 0.01$ apply to $\widetilde{\rho}_t^{\text{chain}}$ and $\widetilde{r}_{t,k}$; the entropy bonus is normalized by the same D. We also search a following the same procedure as Flow-SDE. The resulting a is 0.8,0.6,0.3,0.8 for Hopper, Walker2d, Ant, and Humanoid respectively.

Latency: The table below is the training time per iteration for each algorithm. The unit is seconds. Overall, the training times of the three algorithms are similar. We use a single NVIDIA A100 GPU

Table 2: Training time per iteration in seconds.

Environment	Flow-SDE	ReinFlow	FMPO-CPPO
Hopper	9.58 ± 0.35	12.28 ± 1.81	9.74 ± 0.41
Walker2d	14.95 ± 1.50	13.99 ± 2.49	15.12 ± 1.47
Ant	16.40 ± 1.96	14.19 ± 0.73	16.46 ± 2.01
Humanoid	17.24 ± 1.33	19.65 ± 0.59	17.42 ± 1.29

with 40 GB of memory and a 10-core CPU for all training runs.

F.4 Validating Correctness And Performance Transfer In The Image Generation Setting

Data. We work on CIFAR-10, the standard 10-class natural-image benchmark consisting of 60000 RGB images of resolution 32×32 partitioned into 50000 training and 10000 test images, with 6000 images per class. We use the official train split as the joint training pool for flow-matching pretraining and reserve a fixed 10% slice as a validation set; the official test split is held out and never touched during pretraining or RL fine-tuning. Pixels are rescaled to $[-1, 1]$ via $x \mapsto 2x - 1$ and kept in this range throughout, so that the data distribution $p_{\text{data}}(\cdot \mid y)$ on which we regress the flow-matching velocity sits on $\mathbb{R}^{3\times32\times32}$ with empirical support in $[-1, 1]^{3\times32\times32}$; the noise distribution is the matched isotropic Gaussian $X_0 \sim \mathcal{N}(0, I_{3\times32\times32})$. The class label $y \in \{0, \dots, 9\}$ enters the model as conditioning, and during RL we visit the conditioning classes in a fresh uniformly random permutation at each update, so each of the 10 classes is visited exactly once per RL update. The reward model is a class-conditional reward $r(x \mid y) = \phi(x)_y$ given by the y-th softmax probability of a frozen CIFAR-10-pretrained ResNet-18 classifier $\phi\colon \mathbb{R}^{3\times32\times32} \to \Delta^9$. For monitoring the validation flow-matching loss during fine-tuning we use a *deterministic class-balanced subset* of the validation slice—50 images per class for a total of 500 images, drawn once with a fixed seed and reused across all RL updates, all selections, and all three base models—to make the FM-loss curves in Figure 3(d) comparable across runs; for each call we average the conditional FM loss over a fresh batch of independent (t, X_0) noise draws per image so that the only sources of variation across updates are the policy parameters themselves.

Model And Training Details. We approximate the class-conditional velocity field $u_\theta(x, t, y)\colon \mathbb{R}^{3\times32\times32} \times [0, T] \times \{0, \dots, 9\} \to \mathbb{R}^{3\times32\times32}$ with a small UNet operating directly on CIFAR-10 images. The scalar time is mapped through a sinusoidal positional embedding of dimension 192 refined by a two-layer MLP $\text{Linear} \to \text{SiLU} \to \text{Linear}$; the label y is one-hot encoded and projected to the same 192-dimensional conditioning vector by a single linear map; the two are summed to form the per-step conditioning $c \in \mathbb{R}^{192}$. The image backbone is a two-level residual UNet with base width 96:

$$
\begin{aligned}
3\times32\times32 &\xrightarrow{\text{Conv}_{3\times3}} 96\times32\times32 \xrightarrow{\text{ResBlock}_c} 96\times32\times32 \xrightarrow{\text{Down}} 96\times16\times16 \\
&\xrightarrow{\text{ResBlock}_c} 192\times16\times16 \xrightarrow{\text{Down}} 192\times8\times8 \xrightarrow{\text{ResBlock}_c} 192\times8\times8 \\
&\xrightarrow{\text{Up}} 192\times16\times16 \xrightarrow{\text{ResBlock}_c} 192\times16\times16 \xrightarrow{\text{Up}} 96\times32\times32 \\
&\xrightarrow{\text{ResBlock}_c} 96\times32\times32 \xrightarrow{\text{GN-SiLU-Conv}_{3\times3}} 3\times32\times32,
\end{aligned}
$$

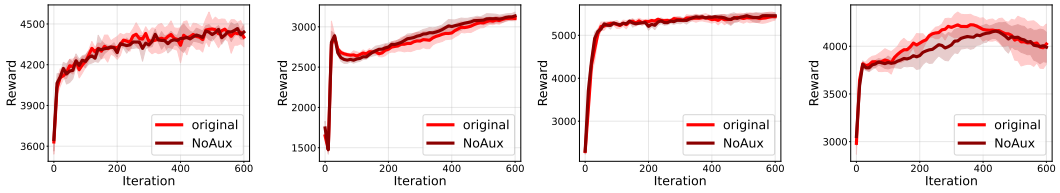
with skip concatenations between matching down/up stages. Each ResBlock_c is a pre-norm residual block $\text{GN}_8 \to \text{SiLU} \to \text{Conv}_{3\times3} \to (+\ \text{Linear}_{c\to\text{ch}}) \to \text{GN}_8 \to \text{SiLU} \to \text{Conv}_{3\times3}$ with a 1×1-conv skip when input and output widths differ; downsampling is a stride-2 4×4 convolution and upsampling is the corresponding transposed convolution. SiLU denotes $\sigma(z) = z\,\text{sigmoid}(z)$.

We pretrain u_θ by minimizing the conditional flow-matching objective (2.3) along the linear interpolation path $X_t = (1 - t)X_0 + tX_1$ with independent coupling $X_0 \sim \mathcal{N}(0, I)$ and $X_1 \sim p_{\text{data}}(\cdot \mid y)$. Each iteration draws a fresh mini-batch of size 192 by sampling $t \sim \mathcal{U}(0, 1)$, an independent noise X_0, and an image X_1 with its class label, and regresses $u_\theta(X_t, t, y)$ onto the target velocity $X_1 - X_0$ in per-coordinate mean squared error. Optimization uses AdamW with learning rate 2×10^{-4}, weight

decay 10^{-4}, default momentum parameters $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon_{\text{Adam}} = 10^{-8}$, gradient-norm clip 1.0, and a cosine annealing schedule over the full training horizon. To produce three pretrained base models spanning a range of flow-matching error, we save checkpoints periodically and select *Weak*/*Medium*/*Strong* with validation FM losses $\mathcal{L}_{\text{FM}}^{\text{val}} = 0.225,\ 0.196,\ 0.184$ by varying the number of training epochs (early-stage, mid-training, and a 30-epoch fully converged run, respectively). All three share the identical architecture above.

RL fine-tuning is then run independently from each base model. One RL update consists of 10 inner policy-gradient steps, one per CIFAR class drawn in a fresh uniformly random permutation of $\{0, \ldots, 9\}$. At each inner step we snapshot a frozen copy $u_{\theta_{\text{old}}}$ of the current policy, draw a group of $G = 32$ SDE rollouts conditioned on a single class y from $X_0 \sim \mathcal{N}(0, I)$, evaluate the terminal reward $r(X_T \mid y) = \phi(X_T)_y$, group-normalize the rewards into advantages, and take a single PPO-clipped update with clip ratio 0.2, advantage clip 3.0, and gradient-norm clip 1.0. SDE rollouts during training use 10 Euler–Maruyama steps with the FLOW-GRPO noise schedule $\sigma_t = a\sqrt{(1-t)/t}$, $a = 0.7$, in the noise-to-data orientation; deployment-time ODE evaluation uses 20 Euler steps. We add a per-step velocity-KL penalty against the frozen pretrained policy with weight $\beta_{\text{KL}} = 0.05$. Optimization uses AdamW with learning rate 4×10^{-6}, no weight decay, default momentum parameters $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon_{\text{Adam}} = 10^{-8}$, and a single policy epoch per inner step. Each base model is fine-tuned for 80 RL updates and the entire procedure is repeated over 3 independent seeds to produce the mean and standard-deviation bands reported in the figures.

G Ablation Of FMPO-CPPO



(a) Ant-v2 **(b) Hopper-v2** **(c) Humanoid-v3** **(d) Walker-v2**

Figure 4: Ablation of the auxiliary step-wise PPO term in FMPO-CPPO on four continuous-control tasks.

We run FMPO-CPPO with the same settings and hyperparameters without the auxiliary term, leaving $\mathcal{J}_{\text{chain}}(\theta)$ as the only optimization objective. As shown in Figure 4, the performance is similar on Ant, Hopper, and Humanoid, while the original FMPO-CPPO performs better on Walker. With only $\mathcal{J}_{\text{chain}}(\theta)$ objective, FMPO-CPPO without the auxiliary term still beats ReinFlow and Flow-SDE with large margins on Ant and Humanoid, while performing similarly on Walker and Hopper.

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: All main claims are stated in the abstract, introduction, and theorem statements.

   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: We discuss limitations 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: All assumptions are stated in the main text, and complete proofs are provided in the appendix.

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: We include numerical studies that validate the theory and test the proposed flow RL method in Section 5.

   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: We show the details of experiments in the paper and we submit the code in the supplemental material.

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: Model and experimental settings are described in Sections 5 and F.

   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: The continuous-control curves report means and one standard deviation over five random seeds (Figure 2); the CIFAR-10 curves use three independent seeds (Figure 3).

   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: We have mentioned all information about compute resources in Section 5 and Section F.

   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: We reviewed the NeurIPS Code of Ethics and believe the research conforms to it.

   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: The paper includes an ethics statement and discusses the likely scope of the work.

    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: The paper does not release high-risk datasets, pre-trained models, or generative systems that require special safeguards. The contribution is primarily theoretical.

    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 external assets used in this work are credited and used in accordance with their licenses.

    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: The paper does not release a new dataset, model checkpoint, or software asset as part of the submission.

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: The paper does not involve crowdsourcing experiments or research with human subjects.

    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: The paper does not involve crowdsourcing or research with human subjects.

    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: We describe the usage of LLMs in the appendix.

    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.