Doob-Based Generative Policy Optimization

Maojiang Su[†,♮] **Qijie Zhu**[‡] **Yu Xiao** [†] **Shuyang Yu**[†]
Minshuo Chen[§] **Zhaoran Wang**[§,†] **Han Liu**[†,‡,♮]

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

https://maojiangsu.github.io/DoobPO/

Abstract

Diffusion and flow models provide expressive policy classes for robotic and sequential decision-making systems. However, their action likelihoods are generally intractable, limiting the direct use of widely adopted policy optimization methods such as PPO and GRPO. We propose Doob Policy Optimization (DoobPO), a framework that leverages Doob's h-transform for likelihood-free reinforcement learning fine-tuning of generative policies. DoobPO separates policy improvement from generative policy learning. It first learns a lightweight ratio model that represents the desired policy update, and then distills the resulting ratio-tilted target policy into the generative policy via conditional drift matching. This formulation applies uniformly to both diffusion and flow policies and supports a broad family of policy improvement rules, including PPO and GRPO. We prove that the practical conditional drift matching objective is gradient-equivalent to the ideal Doob drift matching objective. We further show that the induced diffusion process solves an endpoint-constrained stochastic optimal control problem. This establishes that it achieves the minimum-energy modification required to realize the desired policy tilt. Across MuJoCo continuous-control benchmarks, DoobPO achieves strong performance against competitive diffusion- and flow-policy baselines, showing a principled and practical route to RL fine-tuning for generative policies.

1 Introduction

Generative policies based on diffusion models [1–3] and flow matching models [4, 5] have emerged as a powerful paradigm for policy learning. Diffusion policies generate actions through iterative denoising, while flow policies generate actions by integrating a learned velocity field, often providing a deterministic and more efficient alternative. Due to their flexibility and accuracy in modeling high-dimensional action distributions, generative policies have achieved strong empirical gains over deterministic or unimodal policies in challenging continuous-control settings [6–10].

A natural way to further improve generative policies is through reinforcement learning fine-tuning. However, a central challenge is that the exact likelihoods of generative policies are generally intractable, since each action is produced only after simulating an entire denoising or flow trajectory. This prevents the direct use of many stable and widely adopted policy optimization methods, such as PPO [11] and GRPO [12]. Several recent works have attempted to bypass this likelihood challenge [13–18]. However, these methods often suffer from biased estimation, limited data efficiency, substantial computational overhead, or restriction to a particular generative policy class or policy update rule. Consequently, the full potential of RL for diffusion and flow policies remains underexplored.

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

To address this problem, we propose **Doob Policy Optimization (DoobPO)**, a likelihood-free framework that separates policy improvement from generative policy learning. The first stage represents the desired policy improvement through a target policy ratio. The second stage distills the resulting ratio-tilted target into the generative policy. This decomposition avoids evaluating the likelihood of the generative policy itself. The policy-improvement stage is agnostic to the underlying generative model and accommodates a broad family of updates, including regularized Policy Mirror Descent [19], Soft Actor-Critic [20], PPO [11], and GRPO [12]. The generative stage uses Doob's h-transform to characterize the ratio-tilted target process and learns this process through Doob's Drift Matching. The same construction applies to both diffusion and flow policies.

Contributions. Our contributions are three-fold:

- **Method.** We propose **Doob Policy Optimization (DoobPO)**, a unified framework for RL fine-tuning of generative policies without evaluating exact action likelihoods (Section 3). DoobPO follows a two-stage design: it first obtains the target policy ratio and then transfers the resulting policy improvement into the generative policy through a tractable conditional drift matching objective. The framework supports both diffusion and flow policies (Sections 3.1 and 3.2). We summarize the procedures in Algorithms 1 and 2. Instantiating DoobPO with different policy updates yields **DoobPMD**, **DoobPPO**, and **DoobGRPO**. We further show that several existing methods arise as special cases under particular choices of the policy update and generative policy class (Section 3.3). We summarize these instantiations in Table 1.
- **Theory.** We characterize the ratio-tilted target policy through Doob's h-transform (Lemma 3.1). This characterization leads to Doob's Drift Matching as a principled way to realize the desired policy update. For tractability, we introduce a conditional drift matching objective and prove that its gradient is proportional to that of the ideal Doob's Drift Matching objective for both diffusion and flow policies (Theorems 3.1 and 3.2). From the Stochastic Optimal Control perspective (Section 3.4), we further show that the Doob-transformed diffusion solves the corresponding endpoint-constrained control problem (Proposition 3.1). Hence, DoobPO learns the minimum-energy modification of the pretrained diffusion process that realizes the terminal tilt (Remark 3.2).
- **Experiments.** We evaluate DoobPO on MuJoCo continuous-control benchmarks with both diffusion and flow policies (Section 4). For diffusion policies, DoobPO achieves the strongest overall performance against competitive diffusion-policy RL baselines, with leading results on most tasks (Section 4.2). For flow policies, DoobGRPO achieves the best performance on 9 out of 10 tasks under the same interaction budget and delivers large gains on challenging locomotion environments (Section 4.3). These results establish DoobPO as a strong and practical framework for RL fine-tuning across both diffusion and flow policies.

Notation. We summarize notation and time conventions in Section A.

Related Work. We defer an extended discussion of related work to Section B due to page limits.

2 Preliminaries

Reinforcement Learning. We formulate the task by defining a Markov Decision Process (MDP) [21] with a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces. We use $o \in \mathcal{S}$ to denote a state and $a \in \mathcal{A}$ to denote an action. The transition kernel is denoted by $P(o'|o, a)$, the reward function by $R(o, a)$, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi(a|o)$ induces a distribution over trajectories $\tau = (o_0, a_0, o_1, a_1, \ldots)$, where $a_t \sim \pi(\cdot|o_t)$ and $o_{t+1} \sim P(\cdot|o_t, a_t)$. The standard objective of RL is to maximize

$$J(\theta) = \mathop{\mathbb{E}}_{\tau \sim d_{\pi_\theta}(\tau)} \Big[\sum_{t \geq 0} \gamma^t R(o_t, a_t) \Big], \tag{2.1}$$

where $d_{\pi_\theta}(\tau)$ is the distribution of trajectories generated by the policy π_θ and the transition kernel. Policy gradient methods (e.g., REINFORCE [22]) update θ by ascending an estimate of $\nabla_\theta J(\theta)$,

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau \sim d_{\pi_\theta}(\tau)} \Big[\sum_{t \geq 0} \gamma^t \nabla_\theta \log \pi_\theta(a_t|o_t) \sum_{k \geq t} \gamma^{k-t} R(o_k, a_k) \Big], \tag{2.2}$$

where $\nabla_\theta \log \pi_\theta(a_t|o_t)$ is the score function of the policy. To reduce variance, we often replace the cumulative future reward $\sum_{k \geq t} \gamma^{k-t} R(o_k, a_k)$ with an advantage estimate $A(o_t, a_t)$.

In practice, directly optimizing (2.2) may be unstable due to overly large policy updates. Trust-region style policy gradient methods [23, 11, 12] instead constrain the policy change between successive

Table 1: Doob Policy Optimization

Method	Generative policy	Policy update method	Ratio
DPMD [15]	Diffusion	Policy Mirror Descent (KL-proximal)	Analytic
SDAC [15]	Diffusion	Soft Actor Critic	Analytic
DoobPMD (ours)	Flow	Policy Mirror Descent (KL-proximal)	Analytic
DoobPPO (ours)	Diffusion / Flow	Proximal Policy Optimization	Learned
DoobGRPO (ours)	Diffusion / Flow	Group Relative Policy Optimization	Learned

iterates. A popular example is Proximal Policy Optimization (PPO) [11], which maximizes $J_{\mathrm{PPO}}(\theta)$

$$\mathop{\mathbb{E}}_{\tau\sim d_{\pi_{\mathrm{old}}}(\tau)}\Big[\sum_{t\geq 0}\min\Big(r_\theta(o_t,a_t)\widehat{A}_{\mathrm{old}}(o_t,a_t),\mathrm{clip}\big(r_\theta(o_t,a_t),1-\epsilon,1+\epsilon\big)\widehat{A}_{\mathrm{old}}(o_t,a_t)\Big)\Big], \tag{2.3}$$

where π_{old} is the behavior policy, $r_\theta(o,a) := \pi_\theta(a|o)/\pi_{\mathrm{old}}(a|o)$ is the probability ratio, $\epsilon > 0$ is the clipping parameter, and $\widehat{A}_{\mathrm{old}}(o,a)$ is an advantage estimator. The clipped surrogate upweights actions that are estimated to be better than baseline and limits the policy update away from π_{old}.

Diffusion Models and Score Matching. Let p_0 denote the data distribution. Diffusion models define a forward noising process $(Y_t)_{t\in[0,T]}$ that gradually transforms data into noise through the Stochastic Differential Equation (SDE) [2]

$$\mathrm{d}Y_t = f(Y_t,t)\mathrm{d}t + \sigma_t \mathrm{d}W_t, \quad Y_0\sim p_0, \quad t\in[0,T], \tag{2.4}$$

where f and σ_t are pre-designed drift and diffusion coefficients, and W_t is a standard Brownian motion. We denote by p_t the marginal distribution of Y_t. The generative process is the reverse-time denoising process $(X_t)_{t\in[0,T]}$, evolving from $t=T$ to $t=0$. It transports noise back to data, with $X_T\sim p_T$ and $X_0\sim p_0$. Its time-t marginal is p_t, and the process is governed by the reverse-time SDE

$$\mathrm{d}X_t = \big[f(X_t,t) - \sigma_t^2\nabla_x\log p_t(X_t)\big]\mathrm{d}t + \sigma_t\mathrm{d}\overline{W}_t, \quad X_T\sim p_T, \quad t: T\to 0. \tag{2.5}$$

Here, $\overline{W}_t$ denotes a Wiener process in the reverse-time direction.

Directly using (2.5) is difficult in practice because the noise distribution p_T and the true score $\nabla_x\log p_t$ are not known explicitly. Therefore, (2.5) is implemented through two approximations. First, the noise distribution p_T is approximated by a standard Gaussian distribution. Second, the ground-truth score function $s(x,t) := \nabla_x\log p_t(x)$ is replaced by a parameterized estimator $s_\phi(x,t)$. The score estimator is learned by minimizing the score matching loss

$$\mathcal{L}_{\mathrm{SM}}(\phi) = \mathop{\mathbb{E}}_{t,Y_t\sim p_t}\big[\|s(Y_t,t) - s_\phi(Y_t,t)\|_2^2\big]. \tag{2.6}$$

Flow Matching. Following the notation in [24], let p_0 denote the noise distribution and p_T the data distribution (opposite to the diffusion convention). Flow matching constructs a deterministic transport from p_0 to p_T via the Ordinary Differential Equation (ODE)

$$\mathrm{d}X_t = u(X_t,t)\mathrm{d}t, \tag{2.7}$$

where $u(x,t)$ is the velocity field and p_t denotes the marginal probability density function of X_t. Given a reference velocity $u(x,t)$ that induces the desired path $\{p_t\}_{t\in[0,T]}$, flow matching learns a parameterized velocity $u_\phi(x,t)$ by minimizing the flow matching loss

$$\mathcal{L}_{\mathrm{FM}}(\phi) = \mathop{\mathbb{E}}_{t,X_t\sim p_t}\Big[\big\|u(X_t,t) - u_\phi(X_t,t)\big\|_2^2\Big]. \tag{2.8}$$

After training, new samples are generated by simulating the ODE (2.7) from $t=0$ to $t=T$, replacing the unknown velocity $u(x,t)$ with its learned estimator $u_\phi(x,t)$.

Diffusion Policy and Flow Policy. A diffusion policy [7] is a policy induced by a diffusion model. Let $s_\phi(x,t)$ be a parameterized score network, and let $\{p_t^\phi\}_{t\in[0,T]}$ denote the marginal distributions obtained by simulating the reverse-time SDE (2.5) with the score $\nabla_x\log p_t(x)$ replaced by $s_\phi(x,t)$. The diffusion policy π_ϕ is defined as the distribution over actions induced at the terminal time of the denoising process, $\pi_\phi(\cdot|o) = p_0^\phi(\cdot|o)$. A flow policy [9, 10] is a policy induced by a flow matching model. Let $u_\phi(x,t)$ be a parameterized velocity field, and let $\{p_t^\phi\}_{t\in[0,T]}$ denote the marginals obtained by simulating the ODE (2.7) with $u(x,t)$ replaced by $u_\phi(x,t)$. The flow policy π_ϕ is defined as the action distribution at the terminal time of the flow, i.e., $\pi_\phi(\cdot|o) = p_T^\phi(\cdot|o)$.

For both diffusion and flow policies, evaluating the likelihood $\pi_\phi(a|o)$ is expensive. As a result, the probability ratio $\pi_\phi(a|o)/\pi_{\text{old}}(a|o)$ in trust-region policy gradient methods (e.g., PPO, GRPO) is intractable. This limits applying these methods to improve diffusion and flow policies. DoobPO overcomes this obstacle and enables stable policy gradient style optimization for generative policies.

3 Doob-Based Generative Policy Optimization

We introduce **Doob Policy Optimization (DoobPO)**, a general framework for RL fine-tuning of diffusion policies (Section 3.1) and flow policies (Section 3.2) without evaluating exact action likelihoods. DoobPO supports a broad family of trust-region policy gradient methods, while Section 3.3 shows how both existing and proposed algorithms arise under the same framework.

3.1 Diffusion Policy Optimization with DoobPO

In this section, we introduce DoobPO for diffusion policies. We start from a base diffusion model with score network $s_\phi(x,t)$, which induces a diffusion policy π_ϕ. Our goal is to optimize the policy objective (2.1) over parameters θ, starting from the pretrained initialization $\theta=\phi$.

Ratio Learning. We first learn a probability ratio that captures the policy improvement signal induced by trust-region policy-gradient updates, including regularized Policy Mirror Descent [19], Soft Actor-Critic [20], PPO [11], and GRPO [12]. Specifically, we introduce a lightweight ratio network $r_\beta(o,a)$ to approximate the intractable likelihood ratio $r_\theta(o,a)=\pi_\theta(a|o)/\pi_{\text{old}}(a|o)$. We train r_β on trajectories collected under the behavior policy π_{old} by optimizing a clipped reinforcement learning objective J_{RL}. For example, under PPO, we use $J_{\text{RL}}=J_{\text{PPO}}$ following (2.3), with r_θ replaced by r_β. Intuitively, $r_\beta(o,a)>1$ upweights an action relative to π_{old}, while $r_\beta(o,a)<1$ downweights it. The clipping term enforces a conservative trust-region-style update. In practice, we parameterize the ratio model as $r_\beta(o,a)=\exp(\widetilde{r}_\beta(o,a))$ to ensure $r_\beta(o,a)>0$.

Doob's Drift Matching. We now distill the policy improvement information encoded by the learned ratio r_β into the score model. We consider a diffusion policy, where the denoising SDE (2.5) is conditioned on the environment state o, and the score estimator takes the form $s_\phi(x,t,o)$. This score model induces the behavior policy $\pi_{\text{old}}(\cdot|o)$ as the terminal distribution of the denoising process.

Given the learned ratio $r_\beta(o,a)$, we would like to update the policy toward the ratio-tilted distribution $\pi(\cdot|o)\propto r_\beta(o,\cdot)\pi_{\text{old}}(\cdot|o)$. We realize this tilt by modifying the denoising dynamics with an additional drift term. We characterize this modification via Doob's h-transform.

Lemma 3.1 (Doob's h-Transform)**.** Fix a state $o\in\mathcal{S}$. Let $(X_t)_{t\in[0,T]}$ follow the reverse-time denoising SDE (2.5) with $\nabla_x \log p_t(x|o)$ replaced by its estimator $s_\phi(x,t,o)$, inducing conditional marginals $\{p_t^\phi(\cdot|o)\}_{t\in[0,T]}$. Let $h_0(\cdot,o):\mathcal{A}\to\mathbb{R}_+$ be a positive integrable function. Let $h(\cdot,o,\cdot):\mathcal{A}\times[0,T]\to\mathbb{R}_+$ be the associated Doob h-function, denoted by $h_t(x,o)$, defined as

$$h_t(x,o)=\mathbb{E}\left[h_0(X_0,o)|X_t=x,o\right], \tag{3.1}$$

where the conditional expectation is taken under the original (untilted) denoising process induced by s_ϕ. The Doob h-transformed denoising process starts from

$$X_T^h\sim p_T^h(\cdot|o),\qquad p_T^h(x|o):=\frac{p_T^\phi(x|o)h_T(x,o)}{\mathbb{E}_{X_0\sim p_0^\phi(\cdot|o)}[h_0(X_0,o)]},$$

and evolves backward from T to 0 according to

$$\mathrm{d}X_t^h=\Big[f(X_t^h,t)-\sigma_t^2 s_\phi(X_t^h,t,o)-\sigma_t^2\nabla_x\log h_t(X_t^h,o)\Big]\mathrm{d}t+\sigma_t\mathrm{d}W_t. \tag{3.2}$$

Then its conditional marginals satisfy

$$p_t^h(x|o)=\frac{p_t^\phi(x|o)h_t(x,o)}{\mathbb{E}_{X_0\sim p_0^\phi(\cdot|o)}[h_0(X_0,o)]},\quad t\in[0,T]. \tag{3.3}$$

Particularly, taking $h_0(x,o)=r_\beta(o,x)$ yields $p_0^h(x|o)\propto p_0^\phi(x|o)r_\beta(o,x)$. With $p_0^\phi(\cdot|o)=\pi_{\text{old}}(\cdot|o)$, the time-0 policy induced by the tilted denoising process satisfies $\pi(a|o)\propto r_\beta(o,a)\pi_{\text{old}}(a|o)$.

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

Remark 3.1 (Initialization)**.** In Lemma 3.1, the Doob-transformed denoising process (3.2) is initialized from $p_T^h(\cdot|o)$, which does not necessarily coincide with the original terminal law $p_T^\phi(\cdot|o)$.

Under the idealized fully-noised setting, where X_T is independent of X_0 conditional on o, we have

$$h_T(x, o) = \mathbb{E}[h_0(X_0, o)|X_T = x, o] = \mathbb{E}_{X_0 \sim p_0^\phi(\cdot|o)}[h_0(X_0, o)].$$

Therefore, $p_T^h(\cdot|o) = p_T^\phi(\cdot|o)$ under this assumption. In practice, we initialize (3.2) from the same standard Gaussian prior as the reference process.

We then learn a parameterized conditional score model $s_\theta(x, t, o)$ to match the ratio-induced drift $\nabla_x \log h_t(x, o)$ via the Doob's Drift Matching objective

$$\mathcal{L}_{\mathrm{DM}}(\theta) = \mathop{\mathbb{E}}_{t,o,X_t \sim p_t^h(\cdot|o)} \Big[\big\| \nabla_x \log h_t(X_t, o) - s_\theta(X_t, t, o) \big\|_2^2 \Big], \tag{3.4}$$

where θ denotes the learnable parameters and t samples from uniform distribution $U[0, T]$.

Although the Doob's Drift Matching (DM) objective is clean and direct, it requires access to both $\nabla_x \log h_t(x, o)$ and samples from the tilted distribution $p_t^h(\cdot|o)$, which are generally intractable. To address this issue, we introduce a conditional objective that attains the same optimum as $\mathcal{L}_{\mathrm{DM}}$. Specifically, we define the Conditional Drift Matching (CDM) objective

$$\mathcal{L}_{\mathrm{CDM}}(\theta) = \mathop{\mathbb{E}}_{t,o,X_t \sim p_t^\phi, X_0 \sim p_{0|t}^\phi} \Big[r_\beta(o, X_0) \big\| \nabla_x \log p_{0|t}^\phi(X_0|X_t, o) - s_\theta(X_t, t, o) \big\|_2^2 \Big], \tag{3.5}$$

where $p_{0|t}^\phi(\cdot|X_t, o)$ denotes the diffusion conditional distribution. More importantly, for a fixed state o, we show that the objectives in (3.4) and (3.5) share the same gradient direction.

Theorem 3.1 (Gradient Equivalence for Diffusion)**.** Fix $o \in \mathcal{S}$, define normalizing constant $Z_\beta(o) := \mathbb{E}_{X_0 \sim p_0^\phi(\cdot|o)} \big[r_\beta(o, X_0) \big]$ and

$$\ell_{\mathrm{DM}}(\theta, o) := \mathop{\mathbb{E}}_{t, X_t \sim p_t^h(\cdot|o)} \big[\| \nabla_x \log h_t(X_t, o) - s_\theta(X_t, t, o) \|_2^2 \big] ,$$

$$\ell_{\mathrm{CDM}}(\theta, o) := \mathop{\mathbb{E}}_{t, X_t \sim p_t^\phi(\cdot|o), X_0 \sim p_{0|t}^\phi(\cdot|X_t, o)} \Big[r_\beta(o, X_0) \| \nabla_x \log p_{0|t}^\phi(X_0|X_t, o) - s_\theta(X_t, t, o) \|_2^2 \Big] .$$

Then their gradients satisfy $\nabla_\theta \ell_{\mathrm{CDM}}(\theta, o) = Z_\beta(o) \cdot \nabla_\theta \ell_{\mathrm{DM}}(\theta, o)$.

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

Theorem 3.1 is stated for each fixed observation o. After averaging over observations, we obtain

$$\nabla_\theta \mathcal{L}_{\mathrm{CDM}}(\theta) = \nabla_\theta \mathop{\mathbb{E}}_{o} \left[Z_\beta(o) \ell_{\mathrm{DM}}(\theta, o) \right] .$$

Therefore, $\mathcal{L}_{\mathrm{CDM}}$ has the same gradient field as the reweighted DM objective $\mathbb{E}_o[Z_\beta(o)\ell_{\mathrm{DM}}(\theta, o)]$. If the model class $\{s_\theta : \theta \in \Theta\}$ is realizable and the minimum of $\mathcal{L}_{\mathrm{CDM}}$ is attained, this reweighting does not change the global minimizer. Indeed, there exists $\theta^\star \in \Theta$ such that $s_{\theta^\star}(x, t, o) = \nabla_x \log h_t(x, o)$ for every $x \in \mathcal{A}$, $t \in [0, T]$, and $o \in \mathcal{S}$. Hence $\ell_{\mathrm{DM}}(\theta^\star, o) = 0$ for every $o \in \mathcal{S}$, so $\theta^\star$ also minimizes both the original DM objective and the reweighted DM objective, which implies $\theta^\star$ also minimizes $\mathcal{L}_{\mathrm{CDM}}$.

For variance-preserving DDPMs, the forward noising process (2.4) has the closed-form representation $X_t = \sqrt{\bar{\alpha}_t} X_0 + \sqrt{1 - \bar{\alpha}_t} Z, \quad Z \sim \mathcal{N}(0, I)$, where $\bar{\alpha}_t = \exp\Big(- \int_0^t \sigma_s^2 \mathrm{d}s \Big)$ is the cumulative signal retention coefficient. Moreover, the inference process follows the backward SDE (2.5) by replacing the drift term by $-\sigma_t^2 x/2 - \sigma_t^2 s_\phi(x, t)$. Therefore, following the Bayes formula $p_{0|t} = p_{t|0} p_0 / p_t$, the CDM objective takes the form

$$\begin{aligned} \mathcal{L}_{\mathrm{CDM}}(\theta) &= \mathop{\mathbb{E}}_{t,o,X_t,X_0} [r_\beta(o, X_0) \| \nabla \log p_{t|0}^\phi(X_t|X_0, o) - (s_\phi + s_\theta)(X_t, t, o) \|_2^2] \\ &\approx \mathop{\mathbb{E}}_{t,o,X_t,X_0} [r_\beta(o, X_0) \| - \frac{X_t - \sqrt{\bar{\alpha}_t} X_0}{1 - \bar{\alpha}_t} - (s_\phi + s_\theta)(X_t, t, o) \|_2^2], \end{aligned} \tag{3.6}$$

where $X_t \sim p_t$, $t \sim U[0, T]$, and X_0 is sampled from the conditional distribution $p_{0|t}(\cdot|X_t)$. Let the score model s_ϕ be the pre-trained base model that fixes the noise schedule and the corresponding regression target. The approximation in the second line replaces the conditional score $\nabla \log p_{t|0}^\phi(X_t|X_0, o)$ by the Gaussian regression target induced by the forward noising kernel. The approximation is reliable when s_ϕ is well-trained under the chosen diffusion construction and the trust-region update keeps π_{old} close to the base behavior.

Drift Parametrization. The conditional drift matching loss (3.6) requires evaluating the base score network s_ϕ. In the advantage-tilted denoising process (3.2), both the base score s_ϕ and the advantage-induced drift $\nabla_x \log h_t$ appear with the same coefficient σ_t^2. We exploit this structure and adopt a simplified parametrization. Specifically, we parametrize the drift model g_θ to match the sum $\nabla_x \log h_t(x) + s_\phi(x,t)$. Then the conditional drift matching loss (3.6) reduces to

$$\mathcal{L}_{CDM}(\theta) = \mathop{\mathbb{E}}_{t,o,X_t,X_0}\left[r_\beta(o,X_0)\| - \frac{X_t - \sqrt{\bar\alpha_t}X_0}{1-\bar\alpha_t} - g_\theta(X_t,t,o)\|_2^2\right], \tag{3.7}$$

which no longer requires evaluating the base score network s_ϕ during training. This parametrization also simplifies inference. Once we obtain the trained score network g_θ, sampling from the target distribution p_0^h only requires simulating the following SDE:

$$\mathrm{d}X_t^h = [f(X_t^h,t) - \sigma_t^2 g_\theta(X_t^h,t,o)]\mathrm{d}t + \sigma_t \mathrm{d}W_t. \tag{3.8}$$

We summarize the training procedure in Algorithm 1.

3.2 Flow Policy Optimization with DoobPO

In this section, we instantiate DoobPO for flow policies. We start from a pre-trained flow matching model $u_\phi(x,t,o)$, which induces a flow policy $\pi_\phi(\cdot|o)$ with ODE (2.7). Our goal is to solve (2.1). We reuse the same ratio learning stage as in Section 3.1.

Doob's Drift Matching. We then distill the policy improvement signal encoded by r_β into the velocity model. For flow matching, the policy is given by the terminal distribution at time T. We consider the joint distribution $p_{t,T}^\phi$ induced by the independent-coupling affine interpolation $X_t = \alpha_t X_0 + \beta_t X_T$, where $X_0 \sim \mathcal{N}(0,I)$, $X_T \sim \pi_{\text{old}}(\cdot|o)$, and $X_0 \perp X_T$. We assume the pretrained flow matching model learns the marginal velocity field of this interpolation. We define the tilted flow path by terminally reweighting the base joint law: $p_{t,T}^\beta(x,a|o) := r_\beta(o,a)p_{t,T}^\phi(x,a|o)/Z_\beta(o)$, where $Z_\beta(o) := \mathbb{E}_{X_T\sim p_T^\phi(\cdot|o)}[r_\beta(o,X_T)]$. This gives the tilted terminal policy $\pi_\beta(a|o) = p_T^\beta(a|o) \propto r_\beta(o,a)\pi_{\text{old}}(a|o)$. Let $u^\beta(x,t,o)$ denote the velocity inducing the marginals $p_t^\beta(\cdot|o)$ and terminal $\pi_\beta(\cdot|o)$. We define the drift matching objective for flow policies as

$$\mathcal{L}_{\text{DM}}(\theta) = \mathop{\mathbb{E}}_{t,o,X_t\sim p_t^\beta(\cdot|o)}\left[\left\|u^\beta(X_t,t,o) - u_\theta(X_t,t,o)\right\|_2^2\right]. \tag{3.9}$$

This objective is not implementable since it requires samples from p_t^β and access to u^β. Therefore, we rewrite (3.9) as a conditional drift matching objective that only relies on the behavior process,

$$\begin{aligned}\mathcal{L}_{\text{CDM}}(\theta) &= \mathop{\mathbb{E}}_{t,o,X_t\sim p_t^\phi, X_T\sim p_{T|t}^\phi}\left[r_\beta(o,X_T)\left\|u^\beta(X_t|X_T,t,o) - u_\theta(X_t,t,o)\right\|_2^2\right] \\ &\approx \mathop{\mathbb{E}}_{t,o,X_0\sim p_0^\phi, X_T\sim p_T^\phi, X_t\sim p_{t|T,0}^\phi}\left[r_\beta(o,X_T)\left\|\dot\alpha_t X_0 + \dot\beta_t X_T - u_\theta(X_t,t,o)\right\|_2^2\right].\end{aligned} \tag{3.10}$$

The approximation is reliable when u_ϕ is well-trained under the affine path $X_t = \alpha_t X_0 + \beta_t X_T$, independent coupling and the trust-region update keeps π_{old} close to the base behavior. We summarize the training procedure in Algorithm 2. Finally, we show $\mathcal{L}_{\text{DM}}$ and $\mathcal{L}_{\text{CDM}}$ are equivalent for learning.

Theorem 3.2 (Gradient Equivalence for Flow)**.** Let $\mathcal{L}_{\text{DM}}$ and $\mathcal{L}_{\text{CDM}}$ be defined in (3.9) and (3.10), respectively. Fix $o \in \mathcal{S}$, define normalizing constant $Z_\beta(o) := \mathbb{E}_{X_T\sim p_T^\phi(\cdot|o)}\left[r_\beta(o,X_T)\right]$ and

$$\ell_{\text{DM}}(\theta,o) := \mathop{\mathbb{E}}_{t,X_t\sim p_t^\beta(\cdot|o)}\left[\left\|u^\beta(X_t,t,o) - u_\theta(X_t,t,o)\right\|_2^2\right].$$

$$\ell_{\text{CDM}}(\theta,o) := \mathop{\mathbb{E}}_{t,X_t\sim p_t^\phi, X_T\sim p_{T|t}^\phi}\left[r_\beta(o,X_T)\left\|u^\beta(X_t|X_T,t,o) - u_\theta(X_t,t,o)\right\|_2^2\right].$$

Then their gradients satisfy $\nabla_\theta \ell_{\text{CDM}}(\theta,o) = Z_\beta(o)\cdot\nabla_\theta \ell_{\text{DM}}(\theta,o)$.

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

Ratio Regularization. In policy gradient methods, the probability ratio $r_\theta(o,a) = \pi_\theta(a|o)/\pi_{\text{old}}(a|o)$ is induced by two normalized policies and therefore satisfies $\mathbb{E}_{a\sim\pi_{\text{old}}(\cdot|o)}[r_\theta(o,a)] = 1$ for each state o. In our setting, $r_\beta(o,a)$ is learned as an auxiliary model, so this normalization constraint is not automatically satisfied. We therefore add a ratio regularizer that penalizes violations of $\mathbb{E}_{a\sim\pi_{\text{old}}(\cdot|o)}[r_\beta(o,a)] = 1$ in Algorithms 1 and 2, which stabilizes ratio learning.

3.3 Unified Framework and Instantiations

DoobPO is a unified framework with two design choices. The first is the generative policy class. The second is the policy update method that specifies the target policy improvement. Different choices along these two dimensions induce a broad family of diffusion/flow matching RL algorithms.

Under this view, several prior methods arise as special cases of DoobPO. In particular, they correspond to policy updates that admit closed-form solutions. In these cases, the ratio-learning stage becomes unnecessary, and DoobPO reduces to a single reweighted drift matching step. For example, DPMD [15] corresponds to a policy update with an analytic target ratio. Under a KL-proximal policy mirror descent update [19], the updated policy satisfies $\pi(a|o) \propto \pi_{\text{old}}(a|o) \exp\Big(\widehat{A}_{\text{old}}(o,a)/\lambda\Big)$. Therefore, the corresponding probability ratio satisfies $r(o,a) \propto \exp\Big(\widehat{A}_{\text{old}}(o,a)/\lambda\Big)$.

For more general trust-region policy-gradient methods such as PPO [11] and GRPO [12], the target ratio does not admit a closed form. DoobPO then learns a lightweight ratio model and distills the resulting policy improvement into the generative policy through drift matching. In this work, we derive DoobPPO and DoobGRPO for both diffusion policies and flow policies by instantiating the RL update with PPO and GRPO. Hence, the same framework covers both prior closed-form methods and our new learned-ratio instantiations. We summarize representative examples in Table 1.

3.4 Optimality from the Perspective of Stochastic Optimal Control

Section 3.1 shows that Doob's Drift Matching recovers the ratio-tilted action distribution through Doob's h-transform. In this section, we establish a stronger optimality property from the perspective of endpoint-constrained stochastic optimal control (SOC). Specifically, among all controlled reverse diffusions with prescribed endpoint marginals p_T^h and p_0^h, the minimizer of conditional drift matching induces the minimum-energy drift modification of the pretrained reverse diffusion process. Here, minimality is measured by the quadratic control energy. We now formulate this result.

Proposition 3.1 (Conditional Drift Matching Solves the SOC Problem)**.** Let h_t be defined in (3.1). Fix a state $o \in \mathcal{S}$ and let $\mathbb{P}^\phi(\cdot|o)$ be the reference path measure induced by the reverse-time SDE

$$\mathrm{d}X_t = \big[f(X_t,t) - \sigma_t^2 s_\phi(X_t,t,o)\big]\mathrm{d}t + \sigma_t \mathrm{d}W_t,$$

where $s_\phi(X_t,t,o)$ is the base model. Consider the conditional drift matching objective

$$\mathcal{L}_{\text{CDM}}(\theta) = \mathbb{E}_{t,X_t\sim p_t^\phi, X_0\sim p_{0|t}^\phi}\left[h_0(X_0,o)\left\|\nabla_x \log p_{0|t}^\phi(X_0|X_t,o) - s_\theta(X_t,t,o)\right\|_2^2\right].$$

If the model class $\{s_\theta : \theta \in \Theta\}$ is realizable and the minimum of L_{CDM} is attained, then any minimizer $s_\theta^\star$ induces the optimal control $u_t^\star(x,o) := -s_\theta^\star(x,t,o)$ that solves the SOC problem

$$\inf_u \frac{1}{2}\mathbb{E}_{\mathbb{Q}^u(\cdot|o)}\int_0^T \|\sigma_t u_t(Y_t,o)\|_2^2 \mathrm{d}t, \tag{3.11}$$

$$\text{s.t. } \mathrm{d}Y_t = \big[f(Y_t,t) - \sigma_t^2 s_\phi(Y_t,t,o) + \sigma_t^2 u_t(Y_t,o)\big]\mathrm{d}t + \sigma_t \mathrm{d}W_t,\ Y_T \sim p_T^h,\ Y_0 \sim p_0^h, \tag{3.12}$$

where $\mathbb{Q}^u(\cdot|o)$ is the path measure induced by the controlled reverse-time SDE in (3.12), initialized at $t = T$ with $Y_T \sim p_T^h$, and p_0^h, p_T^h are the endpoint distributions given by (3.3).

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

Remark 3.2 (Minimum-Energy Modification and Practical Implication)**.** Given a target terminal tilt, the SOC problem selects, among all controlled reverse diffusions that realize the tilted endpoint marginals, the one with minimum control energy. The RL methods under our framework are special cases of this principle: we choose $h_0(a,o) \propto r_\beta(o,a)$, so that $p_0^h(a|o) \propto r_\beta(o,a)p_0^\phi(a|o)$. Proposition 3.1 shows that Conditional Drift Matching does not only recover the reward-guided terminal policy; it learns the minimum-energy drift modification that realizes this policy. In practice, under the drift parametrization g_θ and the practical loss (3.7), the population target satisfies $g_\theta^\star = s_\phi - u_t^\star$, where $u_t^\star(x,o)$ is the optimal SOC control in Proposition 3.1. Thus Conditional Drift Matching learns the closest feasible modification of the pretrained score model in the SOC sense. This provides a theoretical explanation for the stability and efficiency of DoobPO.

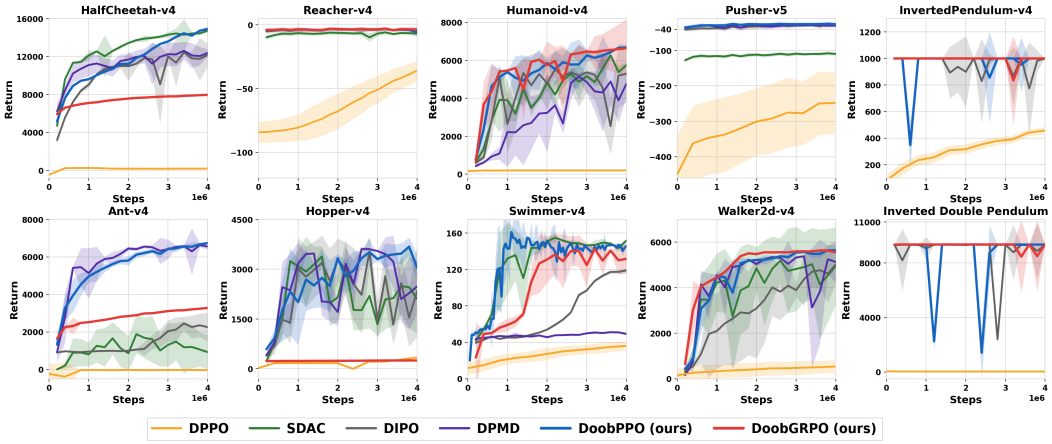


Figure 1: Learning curves on 10 MuJoCo continuous-control tasks, evaluated every 200k environment steps. Curves show mean undiscounted return over 3 training seeds, with shaded regions indicating one standard deviation. We compare DoobPO against strong diffusion-policy baselines DPPO, SDAC, DIPO, and DPMD.

4 Experimental Studies

This section presents the experimental results. We evaluate DoobPO on OpenAI Gym MuJoCo continuous-control benchmarks and on Robomimic manipulation tasks. We describe the experimental setup in Section 4.1. We then report the results of RL fine-tuning diffusion and flow policies with DoobPO in Section 4.2 and Section 4.3, and the Robomimic results in Section 4.4.

4.1 Experimental Setup

Environments: OpenAI Gym MuJoCo. We evaluate our method on 10 OpenAI Gym MuJoCo continuous-control tasks covering locomotion, manipulation, and stabilization. All diffusion-policy methods are trained from scratch for 4M environment steps, while all flow-policy methods are trained from scratch for 5M environment steps, under the same interaction budget within each setting.

Evaluation Metric. We use episode return as the evaluation metric, where higher return indicates better control performance. During training, the policy is evaluated periodically, and at each evaluation point we report the mean return and its standard deviation over 3 random seeds.

4.2 Diffusion Policy Comparison

Baselines. We compare DoobPO against recent diffusion-policy RL baselines, including DIPO [25], DPPO [16], DPMD [15], DACER [26], and SDAC [15], as well as PPO [11], TRPO [23], and SAC [20]. Please see Section B for details.

Diffusion Policy Results. Table 2 and Figure 1 report the diffusion-policy results. Overall, DoobPO achieves the highest mean returns among diffusion-policy RL methods on 8 out of 10 MuJoCo tasks, including ties. DoobPPO obtains the best results on HalfCheetah, Humanoid, and Swimmer, while DoobGRPO performs best on Reacher and Pusher. Both variants achieve competitive performance on the remaining tasks, while DACER obtains the highest returns on Ant and Hopper.

Compared with DIPO, DoobPPO improves performance by 22.6% on HalfCheetah, 20.9% on Humanoid, 175.2% on Ant, and 12.2% on Walker2d. It also improves over DPMD by 18.3%, 30.0%, 6.0%, and 4.9% on the same tasks, respectively. Against SDAC, DoobPPO achieves gains of 8.2% on Humanoid, 8.8% on Hopper, and 8.9% on Walker2d. The strong performance of DoobGRPO on Reacher and Pusher further demonstrates that the framework is not limited to a single PPO-style instantiation. Together, these results establish DoobPO as a competitive and flexible framework for diffusion-policy RL.

4.3 Flow Policy Comparison

Baselines. We compare DoobPO against three representative baselines for RL fine-tuning of flow policies: FPO [18], ReinFlow [17], and Flow-SDE [27]. Please see Section B for more details.

Table 2: Diffusion Policy Performance on MuJoCo environments (mean ± std over 3 random seeds). The best results are in bold and the second-best results are underlined.

	Method	HalfCheetah	Reacher	Humanoid	Pusher	InvertedPendulum
Classic Model-Free RL	PPO	1696 ± 223	-3.63 ± 1.54	492 ± 113	-25.28 ± 2.73	**1000 ± 0**
	TRPO	2941 ± 417	-3.83 ± 1.89	440 ± 82	-31.84 ± 6.32	**1000 ± 0**
	SAC	14422 ± 20	-3.03 ± 1.21	5866 ± 16	-23.96 ± 3.47	**1000 ± 0**
Diffusion Policy RL	DIPO	12169 ± 1172	-3.09 ± 1.36	5523 ± 980	-27.85 ± 2.08	**1000 ± 0**
	DPPO	248 ± 28	-35.83 ± 7.74	195 ± 17	-248.11 ± 86.32	456 ± 25
	DPMD	12612 ± 932	-2.97 ± 1.34	5138 ± 191	-28.22 ± 6.06	**1000 ± 0**
	DACER	14037 ± 370	-2.973 ± 0.004	6460 ± 494	-24.59 ± 0.37	**1000 ± 0**
	SDAC	14731 ± 166	-5.83 ± 1.36	6173 ± 1592	-108 ± 5.23	**1000 ± 0**
	DoobPPO (ours)	**14921 ± 39**	-2.96 ± 0.02	**6679 ± 63**	-23.67 ± 3.16	**1000 ± 0**
	DoobGRPO (ours)	7800 ± 271	**-2.95 ± 1.35**	6634 ± 1517	**-23.62 ± 4.35**	**1000 ± 0**
	Method	Ant	Hopper	Swimmer	Walker2d	Inverted2Pendulum
Classic Model-Free RL	PPO	1973 ± 67	1280 ± 197	121 ± 1.7	1242 ± 693	**9360 ± 0**
	TRPO	3294 ± 76	3626 ± 9	42 ± 1.7	2765 ± 689	9188 ± 420
	SAC	6534 ± 123	3533 ± 19	139 ± 1.1	**6882 ± 60**	**9360 ± 0**
Diffusion Policy RL	DIPO	2448 ± 361	3535 ± 456	119 ± 1.8	5028 ± 849	9351 ± 0.5
	DPPO	-29.9 ± 1.7	338 ± 5.1	36 ± 5	535 ± 278	32.7 ± 3.8
	DPMD	6358 ± 276	3608 ± 73	51.1 ± 1.4	5381 ± 26	9355 ± 3.7
	DACER	**7137 ± 137**	**3774 ± 106**	145.1 ± 5.4	5384 ± 1174	4096 ± 4778
	SDAC	1880 ± 1025	3384 ± 303	154 ± 5.1	5181 ± 986	9353 ± 0.3
	DoobPPO (ours)	6738 ± 53	3681 ± 269	**161 ± 0.7**	5642 ± 13	9356 ± 3
	DoobGRPO (ours)	3278 ± 40	251 ± 1.4	142 ± 2.4	5636 ± 10	**9360 ± 0**

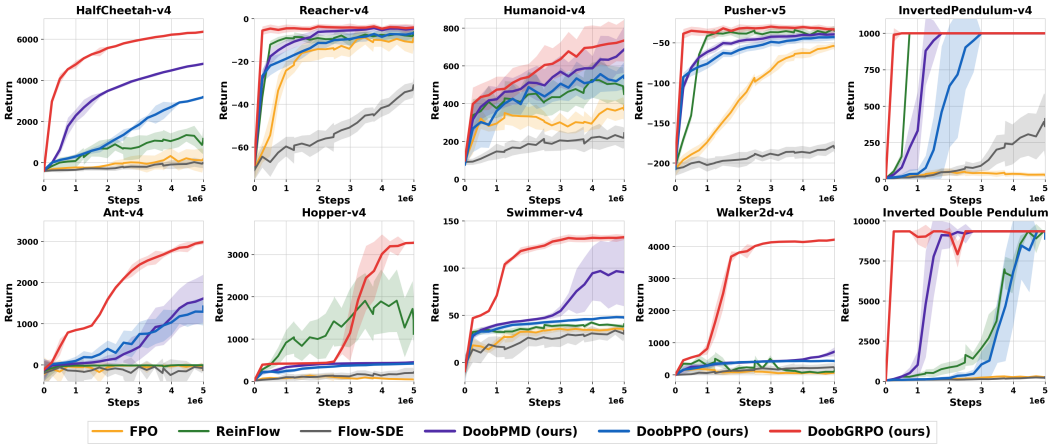


Figure 2: Learning curves on 10 MuJoCo continuous-control tasks, evaluated every 250k environment steps. Curves show mean undiscounted return over 3 training seeds, with shaded regions indicating one standard deviation. We compare DoobPO variants against flow-policy baselines FPO, ReinFlow, and Flow-SDE.

Flow Policy Results. Table 3 and Figure 2 report the flow-policy results. Overall, DoobGRPO achieves the strongest performance across the MuJoCo benchmark under the same 5M-step training budget, obtaining the best results on 9 out of 10 tasks (including ties). Compared with prior flow-policy RL baselines, DoobGRPO gives large gains on key locomotion tasks. For example, it improves over the strongest baseline ReinFlow by **373.8%** on `HalfCheetah`, **71.7%** on `Hopper`, **213.7%** on `Swimmer`, and **738.0%** on `Walker2d`. On the manipulation tasks, DoobGRPO also achieves the best performance on `Reacher` and `Pusher`, improving over ReinFlow by **42.9%** and **9.0%**. DoobPMD serves as a strong second-best on several tasks, including `HalfCheetah`, `Reacher`, and `Humanoid`. This further demonstrates the generality of the DoobPO framework across different policy-gradient methods. These results show that DoobPO provides a useful RL framework for flow policies.

4.4 Robomimic Manipulation

Setup. To evaluate DoobPO beyond MuJoCo continuous control, we test it on the `Can` and `Square` manipulation tasks from Robomimic [28]. Both tasks use low-dimensional state observations, sparse binary success rewards, and an action-chunk horizon of 4. The maximum episode lengths are 300 primitive steps for `Can` and 400 for `Square`. We follow the official Robomimic evaluation protocol with 50 parallel environments. An episode is successful if the binary task-completion reward is achieved at least once, and the success rate is computed over all completed evaluation episodes. We compare DoobPPO with DPPO [16]. Both methods are initialized from the same pretrained diffusion

Table 3: Flow Policy Performance on MuJoCo environments (mean ± std over 3 random seeds).

	Method	HalfCheetah	Reacher	Humanoid	Pusher	InvertedPendulum
Flow Policy RL	FPO	327 ± 399	-8.79 ± 3.20	390 ± 110	-53.77 ± 3.69	52 ± 24
	ReinFlow	1344 ± 310	-6.30 ± 2.83	525 ± 92	-32.85 ± 3.81	**1000 ± 0**
	Flow-SDE	27 ± 219	-31.08 ± 3.53	244 ± 60	-178.42 ± 6.41	413 ± 238
	DoobPMD (ours)	4810 ± 53	-4.35 ± 1.40	688 ± 123	-39.10 ± 4.08	**1000 ± 0**
	DoobPPO (ours)	3190 ± 60	-6.73 ± 2.16	556 ± 94	-42.81 ± 3.75	**1000 ± 0**
	DoobGRPO (ours)	**6368 ± 51**	**-3.60 ± 1.37**	**735 ± 111**	**-29.90 ± 3.74**	**1000 ± 0**
	Method	Ant	Hopper	Swimmer	Walker2d	Inverted2Pendulum
Flow Policy RL	FPO	-13 ± 13	137 ± 55	35.8 ± 2.51	186 ± 140	314 ± 84
	ReinFlow	-5 ± 12	1911 ± 740	42.3 ± 1.77	503 ± 213	**9358 ± 1**
	Flow-SDE	-45 ± 41	210 ± 100	33.9 ± 4.44	243 ± 128	268 ± 68
	DoobPMD (ours)	1614 ± 583	452 ± 4	96.9 ± 32.13	726 ± 144	9348 ± 0
	DoobPPO (ours)	1417 ± 203	423 ± 10	48.0 ± 1.30	440 ± 16	9350 ± 1
	DoobGRPO (ours)	**2986 ± 56**	**3282 ± 7**	**132.7 ± 2.71**	**4215 ± 13**	9350 ± 1

behavior-cloning policy and trained for 151 iterations on `Can` and 200 iterations on `Square`, and results are averaged over 3 random seeds. The DoobPPO hyperparameters are listed in Table 6.

Results. DoobPPO matches or slightly improves upon DPPO on both tasks, reaching success rates of 99.46% versus 99.23% on `Can` and 99.78% versus 99.59% on `Square`. Since both tasks are close to saturation, the main takeaway is not a large performance gap but that DoobPO extends to sparse-reward manipulation settings.

4.5 Implementation Details and Ablation Studies

Additional implementation details are provided in Section G, including computational resources, training hyperparameters, and other reproducibility details. We also include ablation studies in Section H to examine the effect of the ratio network size and the number of sampling steps.

5 Discussion and Conclusion

In this work, we introduce **Doob Policy Optimization (DoobPO)**, a unified framework for RL fine-tuning of diffusion and flow policies. The framework supports a broad family of policy updates and avoids exact action-likelihood evaluation by separating policy improvement from generative policy learning. Theoretically, we justify DoobPO from two directions. First, we characterize the ratio-tilted diffusion process through Doob's h-transform (Lemma 3.1). We then show that, for each fixed state, the gradient of the conditional drift matching objective is proportional to that of the ideal objective for both diffusion and flow policies (Theorems 3.1 and 3.2). Second, under the stated assumptions, the population minimizer of conditional drift matching induces a diffusion process that solves the corresponding endpoint-constrained stochastic optimal control problem (Proposition 3.1). The resulting process is the minimum-energy modification of the reference diffusion that realizes the prescribed tilted endpoint marginals. This provides a control-theoretic interpretation of how the desired policy update is transferred into generative dynamics. Experimentally, we evaluate DoobPO on MuJoCo benchmarks with both diffusion and flow policies and observe superior overall performance against strong baselines (Sections 4.2 and 4.3).

More broadly, DoobPO provides a common interface between policy optimization and generative model training. Policy updates with analytic ratios and updates represented by learned ratio models are accommodated within the same conditional drift matching framework (Section 3.3). This separates the choice of the policy update rule from the choice of the generative policy class. Nevertheless, our study mainly focuses on continuous generation, and the empirical evaluation is limited to MuJoCo continuous-control benchmarks and Robomimic manipulation tasks. The theoretical results characterize population objectives under the stated assumptions. An important direction for future work is to quantify how ratio estimation error and approximate drift matching affect the realized policy update. We also plan to extend the framework to discrete generation and evaluate its applicability to more complex policy learning settings.

References

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

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

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

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

[5] 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.

[6] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[7] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[8] Xiaoyu Huang, Yufeng Chi, Ruofeng Wang, Zhongyu Li, Xue Bin Peng, Sophia Shao, Borivoje Nikolic, and Koushil Sreenath. Diffuseloco: Real-time legged locomotion control with diffusion from offline datasets. *arXiv preprint arXiv:2404.19264*, 2024.

[9] 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.

[10] 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.

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

[12] 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.

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

[14] Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

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

[16] 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.

[17] 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.

[18] 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.

[19] Wenhao Zhan, Shicong Cen, Baihe Huang, Yuxin Chen, Jason D Lee, and Yuejie Chi. Policy mirror descent for regularized reinforcement learning: A generalized framework with linear convergence. *SIAM Journal on Optimization*, 33(2):1061–1091, 2023.

[20] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[21] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

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

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

[24] 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.

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

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

[27] Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, et al. πrl: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025.

[28] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[29] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

[30] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021.

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

[32] Lili Chen, Shikhar Bahl, and Deepak Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *Conference on Robot Learning*, pages 2012–2029. PMLR, 2023.

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

[34] Toan Nguyen, Kien Do, Duc Kieu, and Thin Nguyen. h-edit: Effective and flexible diffusion-based editing via doob's h-transform. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 28490–28501, 2025.

[35] Qijie Zhu, Zeqi Ye, Han Liu, Zhaoran Wang, and Minshuo Chen. Training-free adaptation of diffusion models via doob's h-transform. *arXiv preprint arXiv:2602.16198*, 2026.

[36] Guan-Horng Liu, Jaemoo Choi, Yongxin Chen, Benjamin K Miller, and Ricky TQ Chen. Adjoint schrödinger bridge sampler. *Advances in Neural Information Processing Systems*, 38: 15673–15708, 2025.

Appendix

A Notation **14**

B Related Works **14**

C Proof of Lemma 3.1 **15**

D Proof of Theorem 3.1 **16**

E Proof of Theorem 3.2 **18**

F Proof of Proposition 3.1 **21**

G Implementation Details **23**

G.1 Policy Evaluation 23

G.2 Unbiased Ratio Regularizer 23

G.3 Ratio Regularizer in the Off-Policy Setting 23

G.4 Computation and Memory Cost 24

G.5 Hyperparameters 24

H Ablation Studies **25**

H.1 Diffusion Policy. 25

H.2 Flow Policy. 26

I Algorithms **27**

I.1 Diffusion Policy Optimization with DoobPO 27

I.2 Flow Policy Optimization with DoobPO 28

Impact Statement

This work advances the methodological and theoretical understanding of diffusion and flow policy optimization and presents no foreseeable negative social impacts.

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 exhausting literature to identify potential missing related work. All technical content, proofs, experiments, and results are original contributions by the authors.

Funding and Competing Interests

The authors received no funding for this work and declare no competing financial interests.

A Notation

We use $\mathbb{E}[\cdot]$ and $\mathbb{P}(\cdot)$ to denote expectation and probability. Unless otherwise specified, any expectation involving a random time t is taken with respect to the uniform distribution on $[0, T]$, i.e., $t \sim \mathrm{Unif}([0, T])$. For a time horizon $T > 0$, let $\Omega = C([0, T]; \mathbb{R}^d)$ be the trajectory space. For any path measure $\mathbb{Q}$ on Ω, we denote its time-t marginal by q_t and its conditional distribution of X_t given $X_r = x_r$ by $q_{t|r}(\cdot|x_r)$. For diffusion policies, $t = 0$ denotes the action endpoint and $t = T$ denotes the noise endpoint, and the reverse denoising process is simulated from T to 0. For flow policies, the convention is reversed: $t = 0$ denotes the noise endpoint and $t = T$ denotes the action endpoint. We write $p_t^{\phi}(\cdot|o)$ for the marginal distributions induced by the pretrained score model s_ϕ or velocity model u_ϕ, and $p_t^h(\cdot|o)$ for the marginal distributions induced by the Doob-tilted process introduced in Section 3. The learned ratio model is denoted by $r_\beta(o, a)$, and in the RL setting the terminal tilt is chosen proportional to this learned ratio. In Section 3.4, $\mathbb{Q}^u(\cdot|o)$ denotes the controlled path measure induced by the control $u_t(\cdot, o)$, and $q_t^u(\cdot|o)$ denotes its time-t marginal distribution. When no ambiguity arises, we use the same lowercase notation for a probability distribution and its density. Unless otherwise specified, the state distribution used in expectations is assumed to have full support on $\mathcal{S}$. We write $\mathbb{R}_+ := (0, \infty)$. We say the model class $\{s_\theta : \theta \in \Theta\}$ is realizable for the Doob drift if there exists $\theta^\star \in \Theta$ such that $s_{\theta^\star}(x, t, o) = \nabla_x \log h_t(x, o)$ for every $x \in \mathcal{A}$, $t \in [0, T]$, and $o \in \mathcal{S}$.

B Related Works

Diffusion Model and Diffusion Policy. Diffusion models [29, 3] are probabilistic generative models that progressively transform noise into samples from a target distribution. They are closely connected to score-based generative modeling [1, 30], which offers a complementary view based on learning the gradient field of the data distribution and unifies diffusion-based generation under the stochastic differential equation framework. Owing to their strong expressiveness for complex, high-dimensional, and multimodal distributions, diffusion models have become increasingly attractive for control and decision-making. In particular, diffusion policies parameterize action distributions with diffusion models and generate actions through iterative denoising, yielding richer policy classes than standard unimodal Gaussian actors. Prior work has shown strong empirical performance of diffusion policies in visuomotor control and robotic manipulation [7], as well as in goal-conditioned imitation learning and language- or play-conditioned skill learning [31, 32].

More recently, diffusion policies have been extended to online reinforcement learning. Methods such as DPPO and related diffusion-policy RL approaches show that diffusion-based policy classes can be optimized effectively beyond pure imitation learning [16, 15]. Our work builds on this line of research, but focuses on RL fine-tuning of diffusion policies in the likelihood-intractable setting.

We compare against five recent diffusion-policy RL baselines: DIPO [25], DPPO [16], DPMD [15], DACER [26], and SDAC [15]. These baselines cover both off-policy and on-policy diffusion RL methods: DIPO, DPMD, DACER, and SDAC are off-policy, whereas DPPO is on-policy. DIPO adopts a two-stage strategy that first updates state-action particles using value gradients and then distills

them into a diffusion policy. DPPO formulates policy optimization over a diffusion-augmented MDP and applies Proximal Policy Optimization to the resulting process. DPMD studies efficient online RL for diffusion policies through a policy-mirror-descent-style update with an analytic target ratio. DACER directly backpropagates value gradients through the reverse diffusion process and introduces entropy regularization to improve exploration. SDAC extends diffusion-policy optimization to the soft-actor-critic regime with an analytic-ratio-based update.

Our work generalizes the analytic-ratio ideas underlying DPMD and SDAC into a unified likelihood-free framework, extending them from diffusion policies to flow matching policies and to a broader class of policy optimization methods, including PPO [11] and GRPO [12].

Flow Matching Model and Flow Policy. Flow matching [4, 5] provides an alternative to diffusion models by learning a deterministic ODE transport between a noise prior and the data distribution. Compared to diffusion models, flow matching avoids simulating stochastic dynamics during training and offers faster sampling through fewer integration steps. These properties have made flow matching models increasingly popular in decision-making applications. π_0 [9] and its successor $\pi^*_{0.5}$ [10] adopt flow matching as the generative backbone of vision-language-action models for general robot manipulation, demonstrating that flow policies can scale to diverse real-world tasks. However, the likelihood of a flow policy is as intractable as that of a diffusion policy, since it requires solving an ODE and computing the associated log-determinant of the Jacobian. This intractability poses the same challenge for online RL fine-tuning: standard policy gradient methods that rely on likelihood ratios cannot be directly applied. Flow Q-Learning [33] is an early attempt that sidesteps this issue by using Q-function guidance to steer the flow sampling process, analogous to Q-guided diffusion methods. For policy gradient approaches, several workarounds have been proposed. FPO [18] is a PPO-style method that replaces explicit policy ratios with an implicit surrogate derived from the flow matching training loss, avoiding likelihood computation entirely. ReinFlow [17] converts a deterministic flow policy into a stochastic short-horizon Markov process via learned noise injection, enabling tractable stepwise likelihood computation for policy gradient updates. Flow-SDE [27] instead replaces the deterministic flow ODE with an analytically constructed stochastic differential equation, yielding explicit transition probabilities suitable for policy gradient estimation. All these methods require some form of likelihood evaluation or surrogate at the denoising steps; in contrast, our approach bypasses this requirement entirely by learning a lightweight ratio network that directly estimates the probability ratio needed for policy optimization.

C Proof of Lemma 3.1

Lemma C.1 (Lemma 3.1 Restated: Doob's h-Transform)**.** Fix a state $o \in \mathcal{S}$. Let $(X_t)_{t\in[0,T]}$ follow the reverse-time denoising SDE (2.5) with $\nabla_x \log p_t(x|o)$ replaced by its estimator $s_\phi(x,t,o)$, inducing conditional marginals $\{p^\phi_t(\cdot|o)\}_{t\in[0,T]}$. Let $h_0(\cdot,o) : \mathcal{A} \to \mathbb{R}_+$ be a positive integrable function. Let $h(\cdot,o,\cdot) : \mathcal{A}\times[0,T] \to \mathbb{R}_+$ be the associated Doob h-function, denoted by $h_t(x,o)$, defined as

$$h_t(x,o) = \mathbb{E}\left[h_0(X_0,o)|X_t = x, o\right],$$

where the conditional expectation is taken under the original (untilted) denoising process induced by s_ϕ. The Doob h-transformed denoising process starts from

$$X^h_T \sim p^h_T(\cdot|o), \qquad p^h_T(x|o) := \frac{p^\phi_T(x|o)h_T(x,o)}{\mathbb{E}_{X_0\sim p^\phi_0(\cdot|o)}[h_0(X_0,o)]},$$

and evolves backward from T to 0 according to

$$\mathrm{d}X^h_t = \Big[f(X^h_t,t) - \sigma_t^2 s_\phi(X^h_t,t,o) - \sigma_t^2\nabla_x \log h_t(X^h_t,o)\Big]\mathrm{d}t + \sigma_t \mathrm{d}W_t.$$

Then its conditional marginals satisfy

$$p_t^h(x|o) = \frac{p_t^\phi(x|o)h_t(x,o)}{\mathbb{E}_{X_0\sim p_0^\phi(\cdot|o)}[h_0(X_0,o)]}, \quad t\in[0,T].$$

In particular, taking $h_0(x,o) = r_\beta(o,x)$ yields

$$p_0^h(x|o) \propto p_0^\phi(x|o)r_\beta(o,x).$$

Since $p_0^\phi(\cdot|o) = \pi_{\text{old}}(\cdot|o)$, the time-0 policy induced by the tilted denoising process satisfies

$$\pi(a|o) \propto r_\beta(o,a)\pi_{\text{old}}(a|o).$$

Proof. Conditioning on the fixed state o, the original denoising process is a reverse-time Markov process with marginals $\{p_t^\phi(\cdot|o)\}_{t\in[0,T]}$ and time-0 tilting function $h_0(\cdot,o)$. By the continuous-time Doob h-transform for reverse-time diffusion processes [34, Proposition 2], the transformed process evolves backward from T to 0 according to

$$\mathrm{d}X_t^h = \Big[f(X_t^h,t) - \sigma_t^2 s_\phi(X_t^h,t,o) - \sigma_t^2\nabla_x \log h_t(X_t^h,o)\Big]\mathrm{d}t + \sigma_t \mathrm{d}W_t.$$

which is exactly (3.2). It remains to identify its marginals. By the reverse-time Doob bridge marginal identity [34, Proposition 1], also see [35, Section 3.2], if the transformed process starts from

$$p_T^h(x|o) = \frac{p_T^\phi(x|o)h_T(x,o)}{\mathbb{E}_{X_0\sim p_0^\phi(\cdot|o)}[h_0(X_0,o)]},$$

then, for every $t\in[0,T]$,

$$p_t^h(x|o) = \frac{p_t^\phi(x|o)h_t(x,o)}{\mathbb{E}_{X_0\sim p_0^\phi(\cdot|o)}[h_0(X_0,o)]}.$$

This proves (3.3). Taking $t=0$ gives

$$p_0^h(a|o) = \frac{p_0^\phi(a|o)h_0(a,o)}{\mathbb{E}_{X_0\sim p_0^\phi(\cdot|o)}[h_0(X_0,o)]}.$$

In particular, with $h_0(a,o) = r_\beta(o,a)$,

$$p_0^h(a|o) \propto p_0^\phi(a|o)r_\beta(o,a).$$

Since $p_0^\phi(\cdot|o) = \pi_{\text{old}}(\cdot|o)$, the time-0 policy induced by the tilted denoising process satisfies

$$\pi(a|o) \propto r_\beta(o,a)\pi_{\text{old}}(a|o).$$

□

D Proof of Theorem 3.1

Theorem D.1 (Theorem 3.1 Restated: Gradient Equivalence for Diffusion)**.** Let $\mathcal{L}_{\text{DM}}$ and $\mathcal{L}_{\text{CDM}}$ be defined in (3.4) and (3.5), respectively. Fix state $o\in\mathcal{S}$, define normalizing constant $Z_\beta(o) := \mathbb{E}_{X_0\sim p_0^\phi(\cdot|o)}\left[r_\beta(o,X_0)\right]$and

$$\ell_{\text{DM}}(\theta,o) := \mathop{\mathbb{E}}_{t,X_t\sim p_t^h(\cdot|o)}\left[\|\nabla_x \log h_t(X_t,o) - s_\theta(X_t,t,o)\|_2^2\right],$$

$$\ell_{\mathrm{CDM}}(\theta, o) := \mathop{\mathbb{E}}_{t, X_t \sim p_t^\phi(\cdot|o), X_0 \sim p_{0|t}^\phi(\cdot|X_t, o)} \left[r_\beta(o, X_0) \|\nabla_x \log p_{0|t}^\phi(X_0|X_t, o) - s_\theta(X_t, t, o)\|_2^2 \right].$$

Then their gradients satisfy

$$\nabla_\theta \ell_{\mathrm{CDM}}(\theta, o) = Z_\beta(o) \cdot \nabla_\theta \ell_{\mathrm{DM}}(\theta, o).$$

Proof. For notational simplicity, we first present the proof in the unconditional setting. That is, the conditioning variable o is fixed and omitted from the notation. The conditional statement follows by applying the same argument pointwise for each $o \in \mathcal{S}$, with $p_t(\cdot)$, $p_t^h(\cdot)$, $p_{0|t}(\cdot|\cdot)$, $h_t(\cdot)$, and $r_\beta(\cdot)$ replaced by $p_t^\phi(\cdot|o)$, $p_t^h(\cdot|o)$, $p_{0|t}^\phi(\cdot|\cdot, o)$, $h_t(\cdot, o)$, and $r_\beta(o, \cdot)$, respectively. We now expand the unconditional drift matching loss.

$$\begin{aligned}
\ell_{\mathrm{DM}}(\theta) &= \mathop{\mathbb{E}}_{t, X_t \sim p_t^h} [\|\nabla \log h_t(X_t) - s_\theta(X_t, t)\|_2^2] && \text{(D.1)} \\
&= \frac{1}{T} \int_0^T \int_{\mathbb{R}^d} \left[\|\nabla_x \log h_t(x)\|_2^2 + \|s_\theta(x,t)\|_2^2 - 2 s_\theta(x,t)^\top \nabla_x \log h_t(x) \right] p_t^h(x) \mathrm{d}x \mathrm{d}t \\
&= \frac{1}{T} \int_0^T \int_{\mathbb{R}^d} \left[\|s_\theta(x,t)\|_2^2 - 2 s_\theta(x,t)^\top \nabla_x \log h_t(x) \right] p_t^h(x) \mathrm{d}x \mathrm{d}t + C_1, && \text{(D.2)}
\end{aligned}$$

where $C_1 := \frac{1}{T} \int_0^T \int_{\mathbb{R}^d} \|\nabla \log h_t(x)\|_2^2 p_t^h(x) \mathrm{d}x \mathrm{d}t$ is independent to θ. For the first term,

$$\begin{aligned}
&\frac{1}{T} \int_0^T \int_{\mathbb{R}^d} \|s_\theta(x,t)\|_2^2 p_t^h(x) \mathrm{d}x \mathrm{d}t \\
&= \frac{1}{Z_\beta(o) T} \int_0^T \int_{\mathbb{R}^d} \|s_\theta(x,t)\|_2^2 \cdot h_t(x) p_t(x) \mathrm{d}x \mathrm{d}t && \left(\text{By } p_t^h(x) \propto p_t(x) h_t(x)\right) \\
&= \frac{1}{Z_\beta(o)} \mathop{\mathbb{E}}_{t, X_t \sim p_t} [\|s_\theta(X_t, t)\|_2^2 h(X_t, t)] \\
&= \frac{1}{Z_\beta(o)} \mathop{\mathbb{E}}_{t, X_t \sim p_t} [\|s_\theta(X_t, t)\|_2^2 \mathop{\mathbb{E}}_{X_0 \sim p_{0|t}} [r_\beta(X_0)|X_t]] \quad \left(\text{By } h(X_t, t) = \mathbb{E}[h(X_0, 0)|X_t] \text{ follows from (3.1)}\right) \\
&= \frac{1}{Z_\beta(o)} \mathop{\mathbb{E}}_{t, X_t \sim p_t, X_0 \sim p_{0|t}} [\|s_\theta(X_t, t)\|_2^2 \cdot r_\beta(X_0)], && \text{(D.3)}
\end{aligned}$$

where $Z_\beta(o) = \int_{\mathbb{R}^d} p_t(x) h_t(x) \mathrm{d}x = \mathbb{E}_{X_t \sim p_t}[h(X_t, t)] = \mathbb{E}_{X_0 \sim p_0}[r_\beta(X_0)]$ is the normalizing constant and the last equation follows from (3.1) and tower property. Next we focus on the second term. We show that

$$\begin{aligned}
&\frac{1}{T} \int_0^T \int_{\mathbb{R}^d} s_\theta(x,t)^\top \nabla_x \log h_t(x) p_t^h(x) \mathrm{d}x \mathrm{d}t \\
&= \frac{1}{T} \int_0^T \int_{\mathbb{R}^d} s_\theta(x,t)^\top \frac{\nabla_x h_t(x)}{h_t(x)} p_t^h(x) \mathrm{d}x \mathrm{d}t && \left(\text{By } \nabla_x \log h_t(x) = \nabla_x h_t(x) / h_t(x)\right) \\
&= \frac{1}{Z_\beta(o) T} \int_0^T \int_{\mathbb{R}^d} s_\theta(x,t)^\top \frac{\nabla_x h_t(x)}{h_t(x)} p_t(x) h_t(x) \mathrm{d}x \mathrm{d}t && \left(\text{By } p_t^h(x) \propto p_t(x) h_t(x)\right) \\
&= \frac{1}{Z_\beta(o) T} \int_0^T \int_{\mathbb{R}^d} s_\theta(x,t)^\top \nabla_x h_t(x) p_t(x) \mathrm{d}x \mathrm{d}t.
\end{aligned}$$

Next, we consider conditioning by recalling $h_t(x) = \int_{\mathbb{R}^d} r_\beta(z) p_{0|t}(z|x) \mathrm{d}z$ follows from (3.1).

$$\begin{aligned}
&\frac{1}{T} \int_0^T \int_{\mathbb{R}^d} s_\theta(x,t)^\top \nabla_x \log h_t(x) p_t^h(x) \mathrm{d}x \mathrm{d}t \\
&= \frac{1}{Z_\beta(o) T} \int_0^T \int_{\mathbb{R}^d} s_\theta(x,t)^\top \left(\int_{\mathbb{R}^d} r_\beta(z) \nabla_x p_{0|t}(z|x) \mathrm{d}z \right) p_t(x) \mathrm{d}x \mathrm{d}t
\end{aligned}$$

$$= \frac{1}{Z_\beta(o)T} \int_0^T \int_{\mathbb{R}^d} s_\theta(x,t)^\top \left(\int_{\mathbb{R}^d} r_\beta(z) \frac{\nabla_x p_{0|t}(z|x)}{p_{0|t}(z|x)} \cdot p_{0|t}(z|x) \mathrm{d}z \right) p_t(x) \mathrm{d}x \mathrm{d}t$$
$$= \frac{1}{Z_\beta(o)T} \int_0^T \int_{\mathbb{R}^d} s_\theta(x,t)^\top \int_{\mathbb{R}^d} r_\beta(z) \nabla_x \log p_{0|t}(z|x) \cdot p_{0|t}(z|x) p_t(x) \mathrm{d}z \mathrm{d}x \mathrm{d}t$$
$$= \frac{1}{Z_\beta(o)} \mathop{\mathbb{E}}_{t, X_t \sim p_t, X_0 \sim p_{0|t}} [r_\beta(X_0) \cdot s_\theta(X_t,t)^\top \nabla_{X_t} \log p_{0|t}(X_0|X_t)]. \tag{D.4}$$

Substituting (D.3) and (D.4) back to (D.1), we show that

$$\ell_{\mathrm{DM}}(\theta) = \frac{1}{Z_\beta} \mathop{\mathbb{E}}_{t, X_t \sim p_t, X_0 \sim p_{0|t}} [r_\beta(X_0) \cdot \left(\|s_\theta(X_t,t)\|_2^2 - 2 s_\theta(X_t,t)^\top \nabla_{X_t} \log p_{0|t}(X_0|X_t) \right)] + C_1.$$

By the definition of Conditional Drift Matching loss, we have

$$\begin{aligned}
&\ell_{\mathrm{CDM}}(\theta) \\
&= \mathop{\mathbb{E}}_{t, X_t \sim p_t, X_0 \sim p_{0|t}} [r_\beta(X_0) \cdot \|\nabla_{X_t} \log p_{0|t}(X_0|X_t) - s_\theta(X_t,t)\|_2^2] \qquad \text{(By definition (3.5))} \\
&= \mathop{\mathbb{E}}_{t, X_t \sim p_t, X_0 \sim p_{0|t}} [r_\beta(X_0) \cdot \left(\|s_\theta(X_t,t)\|_2^2 - 2 s_\theta(X_t,t)^\top \nabla_{X_t} \log p_{0|t}(X_0|X_t) \right)] + C_2,
\end{aligned}$$

where $C_2 := \mathbb{E}_{t, X_t \sim p_t, X_0 \sim p_{0|t}} [\|\nabla_{X_t} \log p_{0|t}(X_0|X_t)\|_2^2 \cdot r_\beta(X_0)]$ is independent of θ. Therefore,

$$\ell_{\mathrm{CDM}}(\theta) = Z_\beta(o)(\ell_{\mathrm{DM}}(\theta) - C_1) + C_2.$$

Taking gradients with respect to θ gives

$$\nabla_\theta \ell_{\mathrm{CDM}}(\theta) = Z_\beta(o) \nabla_\theta \ell_{\mathrm{DM}}(\theta),$$

since C_1 and C_2 are independent of θ.

□

E Proof of Theorem 3.2

Theorem E.1 (Theorem 3.2 Restated: Gradient Equivalence for Flow)**.** Let $\mathcal{L}_{\mathrm{DM}}$ and $\mathcal{L}_{\mathrm{CDM}}$ be defined in (3.9) and (3.10), respectively. Fix $o \in \mathcal{S}$, define normalizing constant $Z_\beta(o) := \mathbb{E}_{X_T \sim p_T^\phi(\cdot|o)} \left[r_\beta(o, X_T) \right]$ and

$$\ell_{\mathrm{DM}}(\theta, o) := \mathop{\mathbb{E}}_{t, X_t \sim p_t^\beta(\cdot|o)} \left[\left\| u^\beta(X_t, t, o) - u_\theta(X_t, t, o) \right\|_2^2 \right].$$
$$\ell_{\mathrm{CDM}}(\theta, o) := \mathop{\mathbb{E}}_{t, X_t \sim p_t^\phi, X_T \sim p_{T|t}^\phi} \left[r_\beta(o, X_T) \left\| u^\beta(X_t|X_T, t, o) - u_\theta(X_t, t, o) \right\|_2^2 \right].$$

Then their gradients satisfy

$$\nabla_\theta \ell_{\mathrm{CDM}}(\theta, o) = Z_\beta(o) \cdot \nabla_\theta \ell_{\mathrm{DM}}(\theta, o).$$

Proof. For each $t \in [0, T]$, define $\bar{r}_t(x, o)$

$$\bar{r}_t(x, o) := \mathop{\mathbb{E}}_{X_T \sim p_{T|t}^\phi(\cdot|x,o)} \left[r_\beta(o, X_T) \right].$$

By the ratio-tilted construction of the flow path, the marginal $p_t^\beta(\cdot|o)$ satisfies

$$Z_\beta(o) p_t^\beta(x|o) = p_t^\phi(x|o) \bar{r}_t(x, o). \tag{E.1}$$

Consequently, for any fixed $t \in [0, T]$ and any integrable test function ψ, we have

$$\begin{aligned}
&Z_\beta(o) \mathop{\mathbb{E}}_{X_t \sim p_t^\beta(\cdot|o)} \big[\psi(X_t)\big] \\
&= Z_\beta(o) \int \psi(x) p_t^\beta(x|o) \mathrm{d}x \\
&= \int \psi(x) p_t^\phi(x|o) \bar{r}_t(x, o) \mathrm{d}x \\
&= \int \psi(x) p_t^\phi(x|o) \left[\int r_\beta(o, a) p_{T|t}^\phi(a|x, o) \mathrm{d}a \right] dx \\
&= \iint r_\beta(o, a) \psi(x) p_{T|t}^\phi(a|x, o) p_t^\phi(x|o) \mathrm{d}a \mathrm{d}x \\
&= \mathop{\mathbb{E}}_{\substack{X_t \sim p_t^\phi(\cdot|o), \\ X_T \sim p_{T|t}^\phi(\cdot|X_t, o)}} \big[r_\beta(o, X_T) \psi(X_t)\big].
\end{aligned} \tag{E.2}$$

To derive the velocity reweighting identity, we use the marginalization identity for the tilted flow:

$$u^\beta(x, t, o) = \mathop{\mathbb{E}}_{X_T \sim p_{T|t}^\beta(\cdot|x,o)} \big[u^\beta(x|X_T, t, o)\big].$$

Multiplying both sides by $p_t^\beta(x|o)$, we obtain

$$\begin{aligned}
p_t^\beta(x|o) u^\beta(x, t, o) &= p_t^\beta(x|o) \int u^\beta(x|a, t, o) p_{T|t}^\beta(a|x, o) \mathrm{d}a \\
&= \int u^\beta(x|a, t, o) p_t^\beta(x|o) p_{T|t}^\beta(a|x, o) \mathrm{d}a.
\end{aligned}$$

Using the ratio-tilted joint relation

$$p_t^\beta(x|o) p_{T|t}^\beta(a|x, o) = \frac{r_\beta(o, a)}{Z_\beta(o)} p_t^\phi(x|o) p_{T|t}^\phi(a|x, o),$$

we get

$$\begin{aligned}
p_t^\beta(x|o) u^\beta(x, t, o) &= \frac{1}{Z_\beta(o)} p_t^\phi(x|o) \int r_\beta(o, a) u^\beta(x|a, t, o) p_{T|t}^\phi(a|x, o) \mathrm{d}a \\
&= \frac{1}{Z_\beta(o)} p_t^\phi(x|o) \mathop{\mathbb{E}}_{X_T \sim p_{T|t}^\phi(\cdot|x,o)} \big[r_\beta(o, X_T) u^\beta(x|X_T, t, o)\big].
\end{aligned}$$

Equivalently,

$$Z_\beta(o) p_t^\beta(x|o) u^\beta(x, t, o) = p_t^\phi(x|o) \mathop{\mathbb{E}}_{X_T \sim p_{T|t}^\phi(\cdot|x,o)} \big[r_\beta(o, X_T) u^\beta(x|X_T, t, o)\big]. \tag{E.3}$$

Now we convert (E.3) into a test-function identity. For any integrable test function h, multiply both sides of (E.3) by $h(x, t, o)^\top$ and integrate over x. This gives

$$\begin{aligned}
&Z_\beta(o) \mathop{\mathbb{E}}_{X_t \sim p_t^\beta(\cdot|o)} \big[h(X_t, t, o)^\top u^\beta(X_t, t, o)\big] \\
&= Z_\beta(o) \int h(x, t, o)^\top u^\beta(x, t, o) p_t^\beta(x|o) \mathrm{d}x
\end{aligned}$$

$$
\begin{aligned}
&= \int h(x,t,o)^\top p_t^\phi(x|o) \mathop{\mathbb{E}}_{X_T \sim p_{T|t}^\phi(\cdot|x,o)} \big[r_\beta(o, X_T) u^\beta(x|X_T,t,o) \big] \mathrm{d}x \\
&= \iint r_\beta(o,a) h(x,t,o)^\top u^\beta(x|a,t,o) p_{T|t}^\phi(a|x,o) p_t^\phi(x|o) \mathrm{d}a \mathrm{d}x \\
&= \mathop{\mathbb{E}}_{\substack{X_t \sim p_t^\phi(\cdot|o), \\ X_T \sim p_{T|t}^\phi(\cdot|X_t,o)}} \big[r_\beta(o, X_T) h(X_t,t,o)^\top u^\beta(X_t|X_T,t,o) \big]. \qquad \text{(E.4)}
\end{aligned}
$$

We now prove the gradient identity directly. Under the usual regularity conditions allowing differentiation under the expectation,

$$
\nabla_\theta \ell_{\mathrm{DM}}(\theta, o) = 2 \mathop{\mathbb{E}}_{t,\, X_t \sim p_t^\beta(\cdot|o)} \Big[\big(\nabla_\theta u_\theta(X_t,t,o) \big)^\top \big(u_\theta(X_t,t,o) - u^\beta(X_t,t,o) \big) \Big].
$$

Multiplying both sides by $Z_\beta(o)$, and applying (E.2) componentwise to

$$
\psi(x) = \big(\nabla_\theta u_\theta(x,t,o) \big)^\top u_\theta(x,t,o),
$$

together with (E.4) with $h = \nabla_\theta u_\theta$, gives

$$
\begin{aligned}
&Z_\beta(o) \nabla_\theta \ell_{\mathrm{DM}}(\theta, o) \\
&= 2 \mathop{\mathbb{E}}_{\substack{t \sim U[0,T],\, X_t \sim p_t^\phi(\cdot|o), \\ X_T \sim p_{T|t}^\phi(\cdot|X_t,o)}} \Big[r_\beta(o, X_T) \big(\nabla_\theta u_\theta(X_t,t,o) \big)^\top \big(u_\theta(X_t,t,o) - u^\beta(X_t|X_T,t,o) \big) \Big]. \qquad \text{(E.5)}
\end{aligned}
$$

On the other hand, differentiating $\ell_{\mathrm{CDM}}(\theta, o)$ directly gives

$$
\begin{aligned}
&\nabla_\theta \ell_{\mathrm{CDM}}(\theta, o) \\
&= 2 \mathop{\mathbb{E}}_{\substack{t \sim U[0,T],\, X_t \sim p_t^\phi(\cdot|o), \\ X_T \sim p_{T|t}^\phi(\cdot|X_t,o)}} \Big[r_\beta(o, X_T) \big(\nabla_\theta u_\theta(X_t,t,o) \big)^\top \big(u_\theta(X_t,t,o) - u^\beta(X_t|X_T,t,o) \big) \Big]. \qquad \text{(E.6)}
\end{aligned}
$$

Comparing (E.5) and (E.6), we obtain

$$
\nabla_\theta \ell_{\mathrm{CDM}}(\theta, o) = Z_\beta(o) \nabla_\theta \ell_{\mathrm{DM}}(\theta, o).
$$

□

F Proof of Proposition 3.1

Proposition F.1 (Proposition 3.1 Restated: Conditional Drift Matching Solves the SOC Problem)**.** Let h_t be defined in (3.1). Fix a state $o \in \mathcal{S}$ and let $\mathbb{P}^\phi(\cdot|o)$ be the reference path measure induced by the reverse-time SDE

$$\mathrm{d}X_t = \big[f(X_t,t) - \sigma_t^2 s_\phi(X_t,t,o)\big]\mathrm{d}t + \sigma_t \mathrm{d}W_t,$$

where $s_\phi(X_t,t,o)$ is the base model. Consider the conditional drift matching objective

$$\mathcal{L}_{\mathrm{CDM}}(\theta) = \mathbb{E}_{t,X_t\sim p_t^\phi, X_0\sim p_{0|t}^\phi}\left[h_0(X_0,o)\left\|\nabla_x \log p_{0|t}^\phi(X_0|X_t,o) - s_\theta(X_t,t,o)\right\|_2^2\right].$$

If the model class $\{s_\theta : \theta \in \Theta\}$ is realizable and the minimum of L_{CDM} is attained, then any minimizer $s_\theta^\star$ induces the optimal control $u_t^\star(x,o) := -s_\theta^\star(x,t,o)$ that solves the SOC problem

$$\inf_u \frac{1}{2}\mathbb{E}_{\mathbb{Q}^u(\cdot|o)}\int_0^T \|\sigma_t u_t(Y_t,o)\|_2^2 \mathrm{d}t, \tag{F.1}$$

$$\text{s.t. } \mathrm{d}Y_t = \big[f(Y_t,t) - \sigma_t^2 s_\phi(Y_t,t,o) + \sigma_t^2 u_t(Y_t,o)\big]\mathrm{d}t + \sigma_t \mathrm{d}W_t,\ Y_T \sim p_T^h,\ Y_0 \sim p_0^h, \tag{F.2}$$

where $\mathbb{Q}^u(\cdot|o)$ is the path measure induced by the controlled reverse-time SDE in (F.2), initialized at $t = T$ with $Y_T \sim p_T^h$, and p_0^h, p_T^h are the endpoint distributions given by (3.3).

Proof. Fix $o \in \mathcal{S}$. For notational simplicity, the unconditional version is presented in this proof. The conditional case follows identically by conditioning on o throughout and replacing $p_t(\cdot), h_t(\cdot)$ with $p_t(\cdot|o), h_t(\cdot|o)$, respectively.

By Theorem 3.1, the drift matching objective has the same population minimizer as

$$\mathcal{L}_{\mathrm{DM}}(\theta) = \mathop{\mathbb{E}}_{t,X_t\sim p_t^h}\left[\|\nabla_x \log h_t(X_t) - s_\theta(X_t,t)\|_2^2\right].$$

Therefore, by realizability and attainment of the minimum, any minimizer satisfies

$$s_\theta^\star(x,t) = \nabla_x \log h_t(x).$$

It remains to identify the corresponding SOC optimizer. We use the SOC theory in [36]. Their formulation is written in forward time. Our denoising process is written in reverse time. Let

$$\tau := T - t, \qquad Z_\tau := Y_{T-\tau}.$$

Then the controlled process (F.2) is written in forward time as

$$\mathrm{d}Z_\tau = \big[-f(Z_\tau, T-\tau) + \sigma_{T-\tau}^2 s_\phi(Z_\tau, T-\tau) - \sigma_{T-\tau}^2 u_{T-\tau}(Z_\tau)\big]\mathrm{d}\tau + \sigma_{T-\tau}\mathrm{d}W_\tau,$$

where $Z_0 \sim p_T^h,\ Z_T \sim p_0^h$. For notation simplicity, we rewrite the process as

$$\mathrm{d}Z_\tau = [\bar{f}_\tau(Z_\tau) + \bar{\sigma}_\tau \bar{u}_\tau(Z_\tau)]\mathrm{d}\tau + \bar{\sigma}_\tau \mathrm{d}W_\tau, \quad Z_0 \sim p_T^h, \quad Z_T \sim p_0^h.$$

where $\bar{\sigma}_\tau := \sigma_{T-\tau}, \bar{f}_\tau(z) := -f(z, T-\tau) + \sigma_{T-\tau}^2 s_\phi(z, T-\tau), \bar{u}_\tau(z) := -\sigma_{T-\tau} u_{T-\tau}(z)$.

The associated SOC objective becomes

$$\inf_{\bar{u}} \frac{1}{2}\mathbb{E}\int_0^T \|\bar{u}_\tau(Z_\tau)\|_2^2 \mathrm{d}\tau,$$

since

$$\int_0^T \frac{1}{2}\|\sigma_t u_t(Y_t)\|_2^2 \mathrm{d}t = \int_0^T \frac{1}{2}\|\bar{u}_\tau(Z_\tau)\|_2^2 \mathrm{d}\tau.$$

According to [36], the optimal control satisfies

$$\bar{u}_\tau^\star(z) = \bar{\sigma}_\tau \nabla_z \log \Phi_\tau(z),$$

where the Schrödinger potentials satisfy

$$\Phi_\tau(z) = \int \bar{p}^\phi_{T|\tau}(a|z)\Phi_T(a)\mathrm{d}a, \qquad \Phi_0(z)\widehat{\Phi}_0(z) = p_T^h(z), \tag{F.3}$$

$$\widehat{\Phi}_\tau(z) = \int \bar{p}^\phi_{\tau|0}(z|a)\widehat{\Phi}_0(a)\mathrm{d}a, \qquad \Phi_T(z)\widehat{\Phi}_T(z) = p_0^h(z). \tag{F.4}$$

Here $\bar{p}^\phi_\tau$ denotes the marginal density of the forward-time process $Z_\tau = X_{T-\tau}$, and $\bar{p}^\phi_{b|a}$ denotes the conditional density. Define

$$\begin{aligned} A &:= \mathop{\mathbb{E}}_{X_0 \sim p_0^\phi}[h_0(X_0)], \\ \Phi_\tau(z) &:= h_{T-\tau}(z), \\ \widehat{\Phi}_\tau(z) &:= \frac{p^\phi_{T-\tau}(z)}{A}. \end{aligned}$$

We now verify that these potentials satisfy the optimality equations. First, for (F.3),

$$\begin{aligned} \Phi_\tau(z) &= h_{T-\tau}(z) \\ &= \mathbb{E}[h_0(X_0)|X_{T-\tau} = z] && \text{(By the definition of } h_t \text{ following Lemma 3.1)} \\ &= \int \bar{p}^\phi_{T|\tau}(a|z)h_0(a)\mathrm{d}a && \text{(Rewrite the expectation as integral.)} \\ &= \int \bar{p}^\phi_{T|\tau}(a|z)\Phi_T(a)\mathrm{d}a. && \text{(By the definition of } \Phi_\tau = h_{T-\tau}\text{)} \end{aligned}$$

Second, for (F.4), we have

$$\begin{aligned} \widehat{\Phi}_\tau(z) &= \frac{p^\phi_{T-\tau}(z)}{A} \\ &= \int \bar{p}^\phi_{\tau|0}(z|z_0)\frac{p_T^\phi(z_0)}{A}\mathrm{d}z_0 && \text{(By the law of total probability)} \\ &= \int \bar{p}^\phi_{\tau|0}(z|z_0)\widehat{\Phi}_0(z_0)\mathrm{d}z_0. && \text{(By the definition of } \widehat{\Phi}_\tau\text{)} \end{aligned}$$

Exploiting (3.3), we now verify the boundary conditions. At $\tau = 0$,

$$\Phi_0(z)\widehat{\Phi}_0(z) = h_T(z)\frac{p_T^\phi(z)}{A} = p_T^h(z).$$

At $\tau = T$,

$$\Phi_T(a)\widehat{\Phi}_T(a) = h_0(a)\frac{p_0^\phi(a)}{A} = p_0^h(a).$$

Thus $\Phi_\tau(z) = h_{T-\tau}(z)$ satisfies the Schrödinger optimality equations.

Therefore,

$$\begin{aligned}\overline{u}_\tau^\star(z) &= \overline{\sigma}_\tau \nabla_z \log \Phi_\tau(z) \\ &= \sigma_{T-\tau} \nabla_z \log h_{T-\tau}(z).\end{aligned}$$

Since $\overline{u}_\tau(z) = -\sigma_{T-\tau} u_{T-\tau}(z)$, the optimal control in the original reverse-time convention is

$$u_t^\star(x) = -\nabla_x \log h_t(x).$$

Combining this with $s_\theta^\star(x,t) = \nabla_x \log h_t(x)$ gives

$$u_\theta^\star(x,t) = -\nabla_x \log h_t(x) = -s_\theta^\star(x,t).$$

Hence any minimizer of L_{CDM} induces the optimal control for the SOC problem. □

G Implementation Details

G.1 Policy Evaluation

Policy Evaluation for DoobPPO. The policy evaluation of DoobPPO learns a state-value function by minimizing the regression loss

$$\mathcal{L}^V(\psi) = \mathbb{E}_{o_t \sim \mathcal{D}}\Big[\big(V_\psi(o_t) - \widehat{R}_t\big)^2\Big], \tag{G.1}$$

where $\widehat{R}_t$ is the empirical return or bootstrap target computed from rollout data. Following the standard PPO-style evaluation, we estimate the advantage with generalized advantage estimation,

$$A_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}, \qquad \delta_t = r_t + \gamma V_\psi(o_{t+1}) - V_\psi(o_t). \tag{G.2}$$

In practice, the resulting advantage estimator is used as the policy improvement signal for training the ratio network in the first stage.

Policy Evaluation for DoobGRPO. The policy evaluation of DoobGRPO computes group-relative advantages from rollout returns. For each state, we sample a group of candidate actions or trajectories under the current diffusion policy, evaluate their returns, and normalize them within the group. The resulting relative advantage is given by

$$A_i^{\mathrm{GRPO}} = \frac{R_i - \mathrm{mean}(\{R_j\}_{j=1}^G)}{\mathrm{std}(\{R_j\}_{j=1}^G) + \epsilon}, \tag{G.3}$$

where R_i is the return of the i-th sample and G is the group size. In practice, these normalized relative advantages are directly used as the policy improvement signal for the first-stage ratio learning objective.

G.2 Unbiased Ratio Regularizer

The ratio regularizer $\mathbb{E}_{o\sim\mathcal{D}}\left[\left(\mathbb{E}_{a'\sim\pi_{\mathrm{old}}(\cdot|o)}[r_\beta(o,a')] - 1\right)^2\right]$ contains a quadratic moment term, whose naive Monte Carlo estimate is biased. To obtain an unbiased estimator, we use double sampling. For each sampled state o, two independent action batches are drawn from $\pi_{\mathrm{old}}(\cdot|o)$, yielding two conditionally independent estimates $\widehat{g}_1(o)$ and $\widehat{g}_2(o)$ of $g(o) := \mathbb{E}_{a'\sim\pi_{\mathrm{old}}(\cdot|o)}[r_\beta(o,a')] - 1$. We then replace $g(o)^2$ with $\widehat{g}_1(o)\widehat{g}_2(o)$, which satisfies $\mathbb{E}[\widehat{g}_1(o)\widehat{g}_2(o)|o] = g(o)^2$.

G.3 Ratio Regularizer in the Off-Policy Setting

In the on-policy variant, the ratio centering regularizer enforces $\mathbb{E}_{a'\sim\pi_{\mathrm{old}}(\cdot|o)}[r_\beta(o,a')] \approx 1$ by sampling actions from the frozen rollout policy, which corresponds exactly to the normalization

constraint of the density ratio. In the off-policy variant with a replay buffer, there is no single well-defined behavior policy associated with each stored transition, since the buffer aggregates experience from many past policy versions. We therefore estimate the centering constraint by sampling a' from the current actor at the same state o. This ensures that the learned ratio remains well-calibrated within the support of the evolving policy, effectively regularizing r_β against degenerate solutions regardless of the age of the replay sample. Empirically, this regularizer is applied with a small coefficient ($\lambda = 0.01$) and we observe that it is sufficient to stabilize ratio learning throughout training.

G.4 Computation and Memory Cost

We report the GPU memory allocation and total training time in Table 4. All measurements are obtained on a desktop workstation with an Intel Xeon Gold 6242R CPU, 376 GB memory, and an NVIDIA A6000 GPU. Although DoobPPO and DoobGRPO are not the most lightweight methods in terms of raw memory or wall-clock time, their computational overhead remains moderate and is comparable to several strong diffusion-policy baselines. In particular, DoobPPO requires 1578 MB GPU memory and 25.28 minutes, while DoobGRPO requires 1795 MB and 28.12 minutes, both remaining in the same practical regime as DPMD, DACER, and SDAC. Considering that our methods achieve state-of-the-art or near-state-of-the-art performance on most benchmark tasks, these results indicate a favorable performance–efficiency trade-off. Overall, DoobPO delivers stronger empirical performance without incurring prohibitive computational or memory cost.

Table 4: GPU memory allocation and total compute time of 200K iterations and 1 million environment interactions.

Algorithm	GPU Memory (MB)	Training Time (min)
DIPO	5212	15.96
DPPO	1087	12.71
DPMD	1175	25.29
DACER	1312	29.12
SDAC	1206	23.52
DoobPPO	1578	25.28
DoobGRPO	1795	28.12

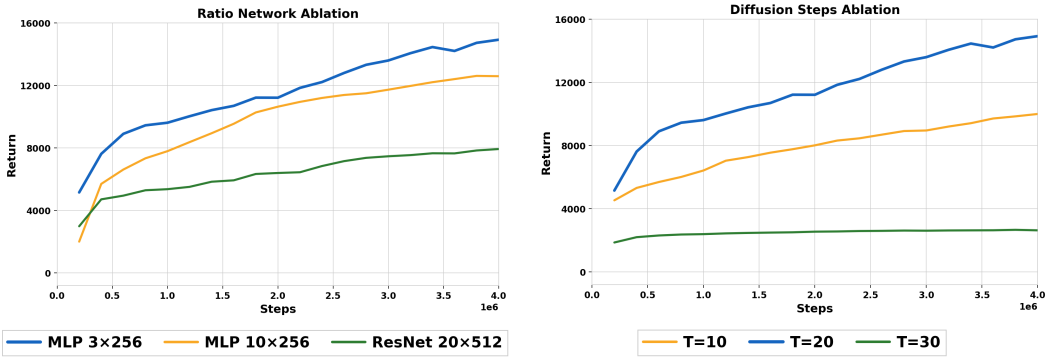
G.5 Hyperparameters

Table 5: Hyperparameters

Name	Value
Critic learning rate	3e-4
Policy learning rate	3e-4, linear annealing to 3e-5
Diffusion steps	20
Diffusion noise schedules	Linear
Policy network hidden layers	3
Policy network hidden neurons	256
Policy network activation	Mish
Ratio network hidden layers	3
Ratio network activation	Mish
Replay buffer size	1 million

Table 6: DoobPPO hyperparameters on Robomimic (`Can` and `Square`).

Name	Value
Discount factor γ	0.999
Actor learning rate	2×10^{-5}
Ratio-network learning rate	1×10^{-4}
Reward scale	2.5
PPO clipping coefficient	0.2
Maximum ratio weight	2
Ratio regularization coefficient	0.03
Action-chunk horizon	4
Training iterations	151 (`Can`), 200 (`Square`)
Parallel evaluation environments	50



(a) Effect of ratio network architecture.

(b) Effect of the number of diffusion steps.

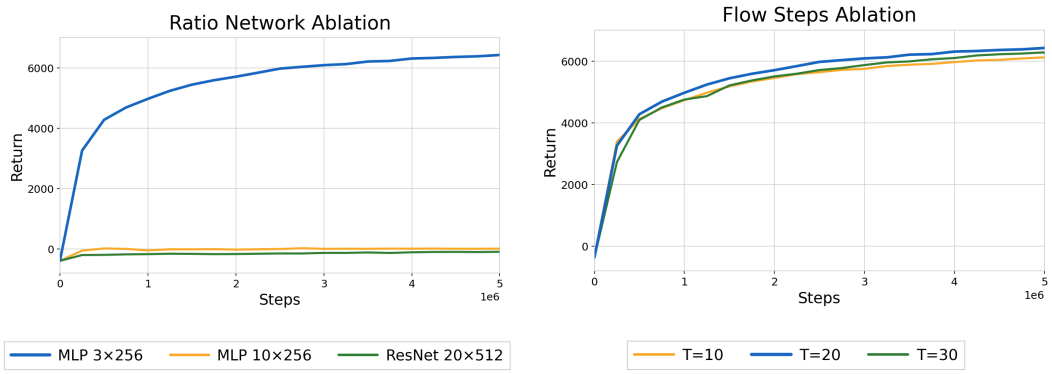
Figure 3: Ablation study of DoobPPO on OpenAI Gym MuJoCo `HalfCheetah-v4`. **(a)** Effect of the ratio network architecture on policy performance. **(b)** Different diffusion steps lead to significantly different performance, with moderate steps performing best.

H Ablation Studies

H.1 Diffusion Policy.

Ablation Study on Ratio Network Size. We study the sensitivity of DoobPPO to the ratio-network architecture on `HalfCheetah-v4`. We compare three variants: an MLP ratio network with 3 layers of width 256, a deeper MLP with 10 layers of width 256, and a larger ResNet-style architecture with 20 layers of width 512. As shown in Figure 3, the simple 3×256 MLP achieves the best performance, consistently outperforming both the deeper MLP and the larger ResNet-style model. This suggests that increasing the capacity of the ratio network does not necessarily improve performance, and may even make optimization less effective. Based on this result, we use the 3×256 MLP ratio network in all diffusion-policy experiments.

Ablation Study on Diffusion Steps. We also study the sensitivity of DoobPPO to the number of diffusion steps on `HalfCheetah-v4`. We compare three settings, namely $T = 10$, $T = 20$, and $T = 30$. The results show that $T = 20$ achieves the best overall performance, while $T = 10$ remains competitive but consistently underperforms $T = 20$, and $T = 30$ leads to a substantial degradation. These results suggest that DoobPO benefits from a moderately sized diffusion process rather than a very shallow or overly long denoising chain. Therefore, we use 20 diffusion steps in all diffusion-policy experiments.



(a) Effect of ratio network architecture.

(b) Effect of the number of flow steps.

Figure 4: Ablation study of DoobGRPO flow policy on OpenAI Gym MuJoCo `HalfCheetah-v4`. **(a)** Only the lightweight MLP 3×256 ratio network learns effectively, while deeper and larger architectures fail to train. **(b)** Performance is robust to the number of flow steps, with T=20 achieving the best result and T=10, T=30 remaining competitive.

H.2 Flow Policy.

Ablation Study on Ratio Network Size. We study the sensitivity of DoobGRPO to the ratio-network architecture on `HalfCheetah-v4` with flow policies. We compare the same three variants as in the diffusion-policy setting: an MLP ratio network with 3 layers of width 256, a deeper MLP with 10 layers of width 256, and a ResNet-style architecture with 20 blocks of width 512. As shown in Figure 4(a), the simple 3×256 MLP achieves the best performance with a peak return of 6433, while both the deeper MLP (peak return 23) and the larger ResNet (peak return -94) fail to learn meaningful policies. This is consistent with the diffusion-policy ablation, confirming that a lightweight ratio network is critical for stable training across both generative-model backbones.

Ablation Study on Flow Steps. We also study the sensitivity of DoobGRPO to the number of flow integration steps on `HalfCheetah-v4`. We compare $T = 10$, $T = 20$, and $T = 30$. As shown in Figure 4(b), the default setting $T = 20$ achieves the best peak performance (6433), while $T = 10$ (6120) and $T = 30$ (6282) remain competitive but slightly underperform $T = 20$. These results suggest that DoobPO is relatively robust to the choice of flow steps, with $T = 20$ providing the best trade-off between sampling quality and computational cost. We use 20 flow steps in all flow-policy experiments.

I Algorithms

I.1 Diffusion Policy Optimization with DoobPO

Algorithm 1 Diffusion Policy Optimization with DoobPO

Require: Environment MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$; pre-trained score model g_ϕ; reinforcement learning objective J_{RL}; advantage estimator $\widehat{A}$; ratio network r_β; ratio regularization coefficient λ_{reg}; iteration K

Initialize: drift model $\theta_1 = \phi$; ratio net β_1.

for $k = 1, 2, \ldots, K$ **do**

Add rollout samples to replay buffer $\mathcal{D}$ from $\mathcal{M}$ using current policy π^k induced by g_{θ_k}

Estimate advantage of current policy $A^{\pi^k} \leftarrow \widehat{A}(\mathcal{D})$

Ratio learning update

Update β by optimizing the regularized reinforcement learning objective

$$\beta_{k+1} \leftarrow \operatorname*{argmax}_{\beta} \Big\{ J_{\mathrm{RL}}(\beta) - \lambda_{\mathrm{reg}} \mathop{\mathbb{E}}_{o\sim\mathcal{D}} \Big[\Big(\mathop{\mathbb{E}}_{a\sim\pi^k(\cdot|o)} [r_\beta(o,a)] - 1 \Big)^2 \Big] \Big\}.$$

Doob's drift matching update

For each $(o, a) \in \mathcal{D}$, sample $t \sim U[0, T]$ and generate noised action x_t

$$x_t \leftarrow \sqrt{\overline{\alpha}_t}\, a + \sqrt{1 - \overline{\alpha}_t}\, z, \quad z \sim \mathcal{N}(0, I)$$

Update θ by minimizing the conditional drift matching objective

$$\theta_{k+1} \leftarrow \operatorname*{argmin}_{\theta} \mathop{\mathbb{E}}_{(o,a,x_t,t)\sim\mathcal{D}} \Big[\Big\| r_{\beta_{k+1}}(o,a) \Big\| - \frac{x_t - \sqrt{\overline{\alpha}_t}\, a}{1 - \overline{\alpha}_t} - g_\theta(x_t, t, o) \Big\|_2^2 \Big]$$

end for

Output: Policy π^{K+1} with learned drift $g_{\theta_{K+1}}$

I.2 Flow Policy Optimization with DoobPO

Algorithm 2 Flow Policy Optimization with DoobPO

Require: Environment MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$; pre-trained velocity model u_ϕ; reinforcement learning objective J_{RL}; advantage estimator $\widehat{A}$; ratio network r_β; ratio regularization coefficient λ_{reg}; iteration K

Initialize: velocity model $\theta_1 = \phi$; ratio net β_1.

for $k = 1, 2, \ldots, K$ **do**

Add rollout samples to replay buffer $\mathcal{D}$ from $\mathcal{M}$ using current policy π^k induced by u_{θ_k}

Estimate advantage of current policy $A^{\pi^k} \leftarrow \widehat{A}(\mathcal{D})$

Ratio learning update

Update β by optimizing the regularized reinforcement learning objective

$$\beta_{k+1} \leftarrow \operatorname*{argmax}_{\beta} \Big\{ J_{\mathrm{RL}}(\beta) - \lambda_{\mathrm{reg}} \mathop{\mathbb{E}}_{o\sim\mathcal{D}} \Big[\Big(\mathop{\mathbb{E}}_{a\sim\pi^k(\cdot|o)} [r_\beta(o,a)] - 1 \Big)^2 \Big] \Big\}.$$

Drift matching update

For each $(o, a) \in \mathcal{D}$, sample $t \sim U[0, T]$ and generate intermediate action x_t by

$$x_t \leftarrow \alpha_t z + \beta_t a, \quad z \sim N(0, I)$$

Update θ by minimizing the conditional drift matching objective

$$\theta_{k+1} \leftarrow \operatorname*{argmin}_{\theta} \mathop{\mathbb{E}}_{(o,a,x_t,t)\sim\mathcal{D}} \Big[r_{\beta_{k+1}}(o,a) \, \| \dot{\alpha}_t z + \dot{\beta}_t a - u_\theta(x_t, t, o) \|_2^2 \Big]$$

end for

Output: Policy π^{K+1} with learned velocity $u_{\theta_{K+1}}$

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 5. We discuss the method now only support continuous generation. We may extend this to discrete setting.

   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 fully disclose all the information needed.

   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: The paper describes the model setting in Section 4 and Section G.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We run the experiments with different seeds and report the mean and std.

   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 G.

   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.

    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: [Yes]

    Justification: All new assets introduced in the paper are well documented.

    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.