Learning To Sample From Diffusion Models Via Inverse Reinforcement Learning

Constant Bourdrez
École Normale Supérieure, Université PSL,
CNRS, Inria, DI ENS, 75005 Paris
constant.bourdrez@ens.psl.eu

Alexandre Vérine
École Normale Supérieure, Université PSL,
CNRS, Inria, DI ENS, 75005 Paris

Olivier Cappé
École Normale Supérieure, Université PSL,
CNRS, Inria, DI ENS, 75005 Paris

Abstract

Diffusion models generate samples through an iterative denoising process guided by a pretrained neural network. Once the denoiser is fixed, the sampling algorithm itself (noise schedules, guidance scales, stochasticity profiles) still requires careful tuning, a process typically carried out through costly empirical grid search. In this work, we introduce an inverse reinforcement learning framework for learning sampling strategies without retraining the denoiser. We formulate the diffusion sampling procedure as a discrete-time finite-horizon Markov Decision Process whose actions correspond to optional modifications of the sampling dynamics. To optimize action scheduling, we avoid defining an explicit reward function and instead directly match the target behavior expected from the sampler using policy gradient techniques. We provide experimental evidence that this approach matches fine-tuned samplers and comes at a modest cost compared to grid search: on ImageNet-64, a single training run replaces exhaustive search at up to $18\times$ lower cost, with only 16% overhead at inference.

1 Introduction

Diffusion models and related generative processes based on differential equations have become a standard tool for modeling high-dimensional data, with applications ranging from image synthesis to scientific and biological modeling. Sample generation proceeds by simulating a reverse-time stochastic or deterministic process that gradually transforms a simple Gaussian distribution into a complex data distribution. This evolution is governed by a vector field implemented by a large neural network, commonly referred to as the denoiser.

While the training of diffusion models has been extensively studied, the sampling procedure itself remains a critical bottleneck. A growing body of work proposes heuristic modifications of the sampling dynamics, including restart sampling [44], guidance mechanisms [18, 8], noise schedule tuning [28], and stochasticity injection [22]. Although often effective, these approaches are typically designed in an ad hoc manner and rely on manually tuned hyperparameters, a process that is rarely acknowledged but can consume as many resources as training the denoiser itself. For instance, the EDM framework [22], one of the strongest image generation baselines, required extensive grid search over its stochasticity profile: a search over 3 hyperparameters on ImageNet-64 alone would require up to 2869 EFLOPs, exceeding the 1645 EFLOPs needed to train the denoiser. More generally, evaluating a single hyperparameter configuration requires generating tens of thousands of samples to compute evaluation metrics, and the cost grows exponentially with the number of hyperparameters and

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

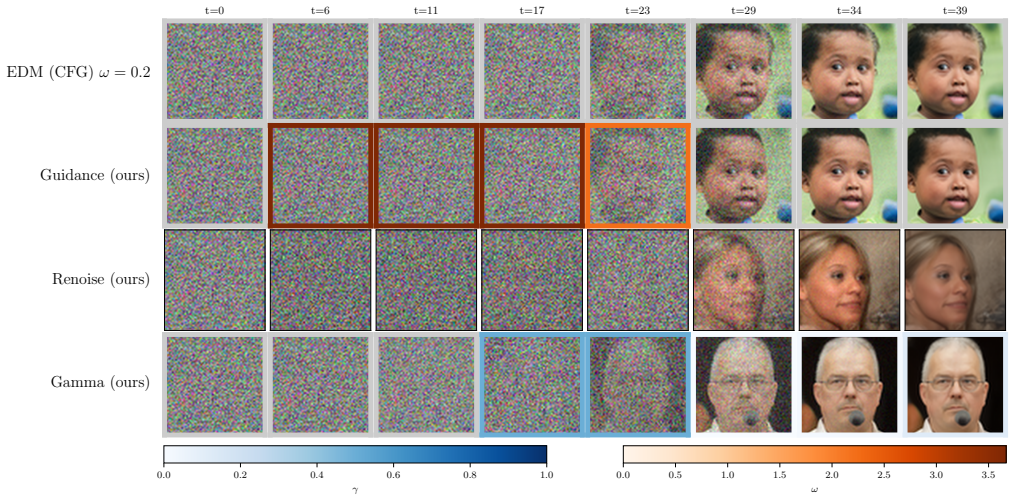


Figure 1: **Learned sampling policies adapt trajectories at inference time.** We compare trajectories from a standard guided diffusion sampler with trajectories produced by policies that learn guidance scales, renoising decisions, or stochasticity injection. All trajectories use the same pretrained denoiser; only the inference-time control policy changes. Frame colors indicate the selected action along the trajectory, showing that the learned sampler intervenes selectively rather than applying a fixed global schedule. These state-dependent controls adapt the sampler without denoiser retraining and reduce the need for manual hyperparameter tuning.

Table 1: Computational cost. *Top*: training, *Middle*: hyperparameter (HP) tuning and our method *Bottom*: inference for one sample. Wall-clock time and memory are reported in Appendix B.3.

	CIFAR-10	FFHQ	ImageNet-64	ImageNet-512[1]
Training Denoiser (EFLOPs)				
EDM [22] / EDM2 [23]	25.7	50.9	1645.0	579.2
Sampling Optimization (EFLOPs)				
Grid search, 2 HP, 5^2	1.9	8.4	140.1	7.0
Grid search, 2 HP, 10^2	7.5	33.5	560.4	28.1
Grid search, 3 HP, 5^3	9.4	41.9	700.5	35.2
Grid search, 3 HP, 8^3	38.3	171.6	2869.2	144.1
Ours	10.2	40.3	160.6	9.7×10^{-3}
Inference cost (TFLOPs, one sample)				
Denoiser size (resolution)	62M (32^2)	66M (64^2)	296M (64^2)	125M (512^2)
Denoiser only	1.5	6.7	8.6	5.6
Denoiser + policy	1.9	9.4	9.9	6.5
Overhead	+26%	+40%	+16%	+15%

model complexity. In this work, we propose to learn sampling-time hyperparameters automatically by framing diffusion sampling as a sequential decision-making problem.

Because sampling is a trajectory optimization problem without an obvious per-step reward, inverse reinforcement learning (IRL) provides a natural tool for learning sampling policies from target state distributions [46, 30, 17]. However, existing IRL methods typically assume expert trajectories or actions, whereas diffusion sampling provides only data marginals across noise levels. RL and control approaches for diffusion models usually optimize explicit rewards or fine-tune the denoiser [5, 11, 10, 15, 41, 3, 9, 35]. Learning the sampler itself through state-only IRL, while keeping the denoiser fixed, has therefore remained largely unexplored.

[1]ImageNet-512 uses the latent EDM2-XS model [23], whose compact latent-space architecture makes it much cheaper than the pixel-space EDM models; its sampling costs include the VAE decode.

We frame diffusion sampling as an IRL problem adapted to continuous states and absorbing terminal dynamics. Rather than tuning hyperparameters by exhaustive search, we learn a policy that matches an expert occupancy measure specified only through data marginals at selected noise levels *i.e* no expert trajectories, no reward model, no denoiser retraining. Our method achieves results matching or exceeding the best hand-crafted sampling strategies at a fraction of the cost: a single training run replaces grid search, reducing the optimization budget by up to $18\times$ on ImageNet-64 as shown in Table 1, with only 16% overhead at inference. Our contributions are threefold: (i) we formulate diffusion sampling as an MDP for adapting samplers at inference time without retraining the denoiser, (ii) we derive a policy-gradient objective that directly matches state occupancy measures through f-divergences, and (iii) we validate the framework using guidance, stochasticity injection, and renoising policies on a representative set of state-of-the-art pretrained diffusion models. Figure 1 illustrates the type of state-dependent interventions learned by our policies. Starting from the same pretrained denoiser and standard reverse process, the learned sampler can decide when to strengthen guidance, when to inject stochasticity, or when to revisit earlier noise levels. This perspective is important: the denoiser is not modified, but the trajectory followed through the denoising process is. The resulting policies therefore act as learned inference-time controls rather than as new generative models.

2 Sample Generation with Diffusion Models

Diffusion models generate data in $\mathbb{R}^d$ by transporting samples from a Gaussian distribution to a target data distribution P_0, via a reverse-time stochastic or deterministic process [37, 38]. The reverse dynamics rely on a neural network F (the *denoiser*) trained across noise levels to estimate the score of the noised data distribution. In practice, F is trained to predict the added noise [19] or the denoised sample [8], and may be conditioned on auxiliary information c such as class labels or text prompts.

While diffusion models are commonly described in continuous time, practical implementations rely on a finite discretization. We can see the generation process as a sequence of states $s_t = (x_t, \sigma_t)$, where x_t is the generated sample at time index t and σ_t is the corresponding noise level defined by the noise schedule $\{\Sigma_N, \dots, \Sigma_0\}$ where $\Sigma_0 = 0 < \Sigma_1 < \cdots < \Sigma_N$. This sequence of noise levels defines a sequence of marginal distributions $\{P_i\}_{i=0}^N$ obtained by convolving the data distribution with Gaussian noise of variance Σ_i^2. We assume that Σ_0 is small enough for P_0 to approximate the target data distribution, and that Σ_N is large enough for P_N to approximate pure Gaussian noise. The sequence starts at $t = 0$ from $s_0 = (x_0, \Sigma_N)$ with $x_0 \sim \mathcal{N}(0, \Sigma_N^2 I_d)$ and ends at $t = T$ with $s_T = (x_T, \Sigma_0)$, where x_T is expected to follow the target distribution P_0. Each step of the reverse process maps (x_t, σ_t) to (x_{t+1}, σ_{t+1}) according to

$$(x_{t+1}, \sigma_{t+1}) = H(x_t, \sigma_t, \dots), \tag{1}$$

where H denotes a generic update operator that uses the denoiser F, evaluated at the current state (x_t, σ_t) and possibly additional inputs such as conditioning information c. We treat $\sigma_t = \Sigma_0$ as an absorbing condition, i.e., once reached, the process remains at this noise level for all subsequent steps. The operator H can represent a variety of samplers: it is stochastic for DDPM [19] and Score-SDE [39], and deterministic for probability-flow ODE (PF-ODE) based samplers, e.g., first-order Euler's method [38], second-order Heun's method [20, 22], or higher-order solvers such as DPM-Solver [29]. Notably, H depends on hyperparameters that influence the sampling dynamics. In this work, we focus on three popular techniques to modify the sampling process: stochasticity injection, classifier-free guidance, and restart sampling.

Stochasticity Injection: The EDM approach of Karras et al. [22], which serves as a strong baseline for image generation, employs an hyperparameter γ_{EDM} to introduce controlled stochasticity. Using H_{Heun} to denote the deterministic Heun update operator, the modified update rule relies on an amplified noise level $\hat{\sigma}_t \coloneqq \sigma_t\,(1 + \gamma_{\text{EDM}})$ and a noise perturbation $z \sim \mathcal{N}(0, I)$:

$$H_{\text{EDM}}(x_t, \sigma_t) = H_{\text{Heun}}(x_t + \sqrt{\hat{\sigma}_t^2 - \sigma_t^2}\, z, \hat{\sigma}_t). \tag{2}$$

For some datasets such as CIFAR-10 [26] or FFHQ [21], $\gamma_{\text{EDM}} = 0$ yields optimal results, while for more complex datasets such as ImageNet [7], non-zero values can improve sample quality, requiring the joint tuning of four parameters that define the schedule of stochasticity injection.

Classifier-Free Guidance: Classifier-free guidance [18] is one of the most popular techniques to enhance sample quality in conditional diffusion models. For any H, where the model F has been

trained to be conditioned on c or unconditioned ($c = \emptyset$), the update rule is kept identical to H but the denoiser is modified as

$$F_{\mathrm{CFG}}(x_t, \sigma_t, c) = (1 + \omega)F(x_t, \sigma_t, c) - \omega F(x_t, \sigma_t, \emptyset). \tag{3}$$

Depending on the guidance scale ω, this technique can significantly improve sample fidelity, at the cost of diversity [18, 34]. However, there is no consensus on the optimal value of ω, nor on whether it should remain fixed or vary across the sampling trajectory.

Restart Sampling: Another approach consists in restarting the sampling process to escape poor local generations [44]. When the sample reaches Σ_i, it is sent back K times to a higher noise level Σ_j ($j > i$) by adding Gaussian noise $z \sim \mathcal{N}(0, I)$, then resampled with a PF-ODE operator H_{ODE}: denoting by k_t the number of restarts up to time t,

$$H(x_t, \sigma_t) = \begin{cases} H_{\mathrm{ODE}}(x_t + \sqrt{\Sigma_j^2 - \Sigma_i^2}\, z,\, \Sigma_j) & \text{if } k_t < K,\ \sigma_t = \Sigma_i, \\ H_{\mathrm{ODE}}(x_t, \sigma_t) & \text{otherwise.} \end{cases}$$

The variant we consider (*Renoise*) allows renoising to one of the $M{=}4$ previous noise levels upon reaching any restart level, with a fixed number of function evaluations (NFE) budget instead of a fixed restart count.

3 State Distribution Matching with Inverse Reinforcement Learning

In this section, we review inverse reinforcement learning (IRL) tools that motivate our formulation in Section 4. Our goal is to cast the diffusion sampling algorithm from Section 2 as a decision-making problem, in order to select sampler hyperparameters (e.g., guidance scales, injected noise, or renoise rules). Crucially, we do *not* assume access to an expert policy or expert actions. Instead, we only observe samples from the data distribution and their noisy counterparts across noise levels, which naturally aligns with IRL settings based on matching state distributions.

MDP formulation We formalize diffusion sampling as a finite-horizon Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \rho, T)$. We consider discrete-time policies with a fixed horizon T, initial states sampled from distribution ρ, and transition dynamics defined by the kernel $\mathcal{P}$. The state space $\mathcal{S}$ is continuous, and the action space $\mathcal{A}$ is discrete. Let $\pi_\theta(a_t \mid s_t)$ be a parametrized policy over actions conditioned on states, and let $\boldsymbol{s}_{1:T} = (s_1, \ldots, s_T)$ denote a trajectory generated by this policy. In Section 4, we will specify how each component of the MDP is instantiated for diffusion samplers.

State occupancy measures In our setting, the expert policy is unknown and we do not observe expert actions or full trajectories; the only supervision comes from expert states (data samples and their noised versions). This state-only regime motivates comparing policies through their *state occupancy measures*, i.e., the marginal distribution over states visited along trajectories.

Definition 3.1 (Occupancy Measure)**.** The policy π_θ induces a state occupancy measure μ_θ defined as

$$\forall \Phi \in \mathcal{B}(\mathcal{S}), \quad \int \Phi(s)\, \mu_\theta(ds) = \mathbb{E}_{\boldsymbol{s}_{1:T} \sim p_\theta}\left[\frac{1}{T}\sum_{t=1}^{T} \Phi(s_t)\right], \tag{4}$$

where $\mathcal{B}(\mathcal{S})$ is the set of bounded measurable functions on $\mathcal{S}$, and p_θ is the trajectory distribution induced by π_θ.

When $\mathcal{S}$ is discrete, μ_θ admits the classical interpretation of an expected visitation frequency: $\mu_\theta(s) = \mathbb{E}_{\boldsymbol{s}_{1:T} \sim p_\theta}\left[\frac{1}{T}\sum_{t=1}^{T} \mathbb{1}_{\{s_t = s\}}\right]$. We will not assume $\mathcal{S}$ is discrete in what follows; the discrete case is only meant as intuition. Similarly, we define the expert occupancy measure μ_E as the state occupancy measure induced by the (unknown) expert policy under the same environment dynamics.

State-marginal matching via f-divergences State-marginal matching IRL seeks a policy whose state occupancy measure μ_θ matches the expert occupancy measure μ_E. Concretely, one minimizes a discrepancy between these probability measures where several approaches focus on the Kullback–Leibler (KL) divergence, typically via maximum-entropy RL and reward learning [46, 30]. More generally, one can use the broader class of f-divergences [6], which provides a unified way to quantify distribution mismatch.

Definition 3.2 (f-divergence)**.** The f-divergence between two probability measures Q and P, induced by a convex function f satisfying $f(1) = 0$, is

$$D_f(Q \parallel P) \coloneqq \mathbb{E}_P\left[f\left(\frac{dQ}{dP}\right)\right]. \tag{5}$$

This family includes KL, Reverse-KL (rKL), and total variation distance. The choice of f matters: different divergences induce different trade-offs e.g., mode-covering (KL) vs. mode-seeking behavior (rKL), which can change the qualitative properties of the learned policy [31, 42].

In this work, we consider objectives of the form $\min_{\pi_\theta} \; D_f(\mu_E \parallel \mu_\theta)$ subject to the MDP dynamics induced by the sampler.

Prior work and our approach Classical IRL approaches first learn a reward under maximum-entropy RL [46, 30]. GAIL [17] minimizes a state–action occupancy divergence via adversarial training, and Ghasemipour et al. [13] extend this to general f-divergences via convex duality. Both require expert actions, which are unavailable in our setting. f-PG [1] focuses on terminal state matching, which yields high-variance gradients over long trajectories. A related IRL approach is proposed by Yoon et al. [45], who learn an energy-based reward to guide sampling. Our approach (i) directly minimizes a state-marginal f-divergence without reward learning, (ii) targets intermediate noise levels rather than only the terminal state, and (iii) uses PPO-style clipping to stabilize training.

4 Improving Diffusion with Inverse Reinforcement Learning

In this section, we present our theoretical contributions and introduce a general framework for controlling generative sequential processes. We formalize diffusion sampling as a finite-horizon MDP (Section 4.1), derive a policy gradient for directly minimizing f-divergences between state occupancy measures (Section 4.2), and describe the practical optimization procedure.

4.1 The Markov Decision Process Model for Diffusion Sampling

To apply our inverse reinforcement learning algorithm, we frame each previously defined component as an element of diffusion sampling. The state space $\mathcal{S}$ is continuous, and each state is defined as

$$s_t = (x_t, \sigma_t),$$

where $\sigma_t \in \{\Sigma_0, \ldots, \Sigma_N\}$ denotes the noise level at time step t and $x_t \in \mathbb{R}^d$ is the current sample. The action space $\mathcal{A}$ is discrete and encodes control decisions applied to the sampling dynamics. For instance, actions can correspond to selecting different discretized guidance scales, noise injection levels, or whether to renoise and how much at each timestep. The initial distribution ρ corresponds to the marginal distribution at the highest noise level, the Gaussian distribution $\mathcal{N}(0, \Sigma_N^2 I_d)$. Given a state-action pair (s_t, a_t), the transition kernel $\mathcal{P}$ is induced by the discretized diffusion dynamics together with the selected control action. The policy does not explicitly depend on time but the available actions may depend on the current timestep t. We will treat $\sigma_t = \Sigma_0$ as an absorbing condition, so that once the process reaches this noise level, it remains there for all subsequent timesteps. In practice, this can be implemented by defining H such that $s_{t+1} = s_t$ whenever $\sigma_t = \Sigma_0$. In this setup, the average time required to reach the terminal noise level may be incorporated in the performance metric of a given policy.

As the state space is continuous, we rely on Definition 3.1. Exploiting the structure of the state space, we further decompose the occupancy measure as $\mu_\theta(s) = w_\theta(\sigma) \times p_\theta(x|\sigma)$, where $w_\theta(\sigma)$ denotes the weight assigned by the policy π_θ to noise level σ, and $p_\theta(x|\sigma)$ is the marginal distribution of x at noise level σ under the policy. For samplers such as noise injection or guidance, w_θ is uniform over noise levels, since the process always progresses from Σ_N to Σ_0 in T steps. For samplers with renoising, w_θ reflects the distribution over noise levels induced by the policy, which may spend more time at certain noise levels depending on the restart strategy.

Expert occupancy measure In our framework, the expert is specified through samples from the target distribution at selected noise levels, without access to expert trajectories, actions, or transition dynamics. The expert occupancy measure $\mu_E(s) = w_E(\sigma) \times p_E(x|\sigma)$ has weights $w_E(\sigma)$ that are a design choice: we set them uniformly over $\{\Sigma_N, \ldots, \Sigma_1\}$ and concentrate the remaining weight on

Σ_0, so that the absorbing terminal states provide a dense learning signal and $w_E(\Sigma_0)$ acts as a proxy for the mean time the policy should spend at the terminal noise level.

On the leverage of w_E By the data processing inequality, $\mathcal{D}_f(\mu_E\|\mu_\theta) \geq \mathcal{D}_f(w_E\|w_\theta)$, so w_E directly shapes the optimization. Additionally, for KL and rKL the objective decomposes into a schedule term $\mathcal{D}_f(w_E\|w_\theta)$ and a weighted sum of per-noise-level divergences $\mathcal{D}_f(p_E(\cdot|\sigma)\|p_\theta(\cdot|\sigma))$:

$$\mathcal{D}_f(\mu_E\|\mu_\theta) = \mathcal{D}_f(w_E\|w_\theta) + \textstyle\sum_\sigma w_E(\sigma)\, \mathcal{D}_f(p_E(\cdot \mid \sigma)\|p_\theta(\cdot \mid \sigma))\,. \tag{6}$$

4.2 Learning Policies by Minimizing f-Divergences

We now describe how to optimize sampling policies within the MDP framework introduced above. We directly optimize the policy to match the expert occupancy measure by optimizing the following:

$$\min_\theta \mathcal{L}_f(\theta) = \min_\theta\ \mathcal{D}_f(\mu_E \,\|\, \mu_\theta)\,. \tag{7}$$

Policy gradient optimization We show in Appendix A.1 that the gradient of the f-divergence objective with respect to the policy parameters, denoted $\nabla_\theta \mathcal{L}_f(\theta)$, can be expressed in the form of a policy gradient.

Theorem 4.1. *Let π_θ be a policy parameterized by θ and let $\mathcal{L}_f(\theta) = \mathcal{D}_f(\mu_E\|\mu_\theta)$ be the f-divergence between the expert occupancy measure μ_E and the occupancy measure μ_θ induced by the policy π_θ. Then, the gradient of $\mathcal{L}_f(\theta)$ with respect to θ is given by*

$$\nabla_\theta \mathcal{L}_f(\theta) = \mathbb{E}_{\boldsymbol{s}_{1:T}\sim p_\theta}\left[\frac{1}{T}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'\geq t} h_f\left(\frac{\mu_E(s_{t'})}{\mu_\theta(s_{t'})}\right)\right], \tag{8}$$

where $h_f(x) = f(x) - xf'(x)$ for all x and f is the generator function of the f-divergence (see Appendix A for details).

This result enables the use of standard policy gradient methods to optimize the sampling policy using Monte Carlo estimates obtained from rollouts of the current policy. While this result seems simple and similar to other policy gradient theorems, the main novelty lies in the direct optimization of the policy instead of a reward function.

Density ratio estimation Computing the learning signal $h_f(\mu_E(s_t)/\mu_\theta(s_t))$ requires estimating the density ratio μ_E/μ_θ at each visited state. We do so by training a discriminator $D_\phi : \mathcal{S} \to [0,1]$ to distinguish between expert states (data samples and their noised versions) and policy states (samples generated by the current policy) [40, 14]. The discriminator is retrained jointly with the policy at each iteration; full details and the training procedure are given in Appendix B.2.

Sample efficiency and practical considerations Generating trajectories is computationally expensive, as each rollout requires simulating a full diffusion sampling trajectory. To improve sample efficiency, we reuse trajectories over multiple policy updates and add importance sampling corrections to the policy gradient. While this modification is theoretically sound, we found it to be unstable since it leads to high-variance gradient estimates. We attempted to mitigate this issue by subtracting a baseline, a standard technique for reducing variance in policy gradient methods. However, in our setting, there is no clear choice of baseline; we use the global batch mean $\bar{A}$ of the cumulative learning signals as a variance-reducing baseline, following standard PPO practice.

To further reduce variance, we adopt a Proximal Policy Optimization (PPO)-style clipped objective [36], which constrains the policy updates at each iteration as well as reward normalization. Under the same assumptions as Theorem 4.1, the PPO-style clipped gradient for minimizing an f-divergence is given by

$$\nabla_\theta \mathcal{L}_f^{\text{PPO}}(\theta) = \mathbb{E}_{p_{\theta_0}}\left[\sum_{t=1}^{T} \nabla_\theta \max\Big(r_t(\theta)\,\hat{A}_t,\ \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\,\hat{A}_t\Big)\right], \tag{9}$$

with the probability ratio r_t, the cumulative learning signal A_t, and its version $\hat{A}_t$ recentered by the global batch mean $\bar{A}$:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_0}(a_t|s_t)}, \qquad A_t = \sum_{t'\geq t} h_f\left(\frac{\mu_E(s_{t'})}{\mu_\theta(s_{t'})}\right), \qquad \hat{A}_t = \frac{1}{T}\big(A_t - \bar{A}\big).$$

Table 2: Results across all strategies. Bold indicates the best value within each dataset–strategy block; standard deviations over repeated runs are reported in Appendix B.4. As expected, no method dominates on all metrics: the f-divergence sets the precision–recall trade-off.

Strategy	Method	FID ↓	Prec ↑	Rec ↑	NFE	Strategy	Method	FID ↓	Prec ↑	Rec ↑	NFE
CIFAR-10 32×32						**FFHQ** 64×64					
	EDM+CFG	2.07	80.6	70.9	35		EDM+CFG	3.11	90.7	87.8	79
CFG	KL	**2.02**	79.8	**71.9**	35	CFG	KL	**3.03**	90.6	**88.6**	79
	rKL	2.38	**81.2**	70.6	35		rKL	3.04	**90.8**	88.3	79
	EDM	**1.98**	78.7	72.9	35		EDM	2.56	90.1	**89.4**	79
γ_{EDM}	KL	2.01	78.8	**73.1**	35	γ_{EDM}	KL	2.53	90.4	88.2	79
	rKL	2.04	**79.4**	72.3	35		rKL	**2.49**	**90.9**	87.9	79
	EDM	3.42	76.7	**72.5**	17		EDM	**2.60**	90.5	**89.0**	59
Renoise	KL	**3.18**	77.3	**72.5**	17.7	Renoise	KL	3.04	**90.9**	88.1	87.4
	rKL	3.21	**78.1**	71.6	30.5		rKL	3.07	**90.9**	87.9	68.2
ImageNet 64×64						**ImageNet** 512×512					
	EDM	**2.12**	85.9	91.7	511		EDM2+CFG	**3.80**	42.3	**77.6**	35
γ_{EDM}	KL	2.14	85.7	**92.0**	511	CFG	KL	3.85	42.3	76.6	35
	rKL	2.24	**86.3**	91.1	511		rKL	**3.80**	42.3	**77.6**	35
	EDM	3.01	**85.8**	91.8	350		EDM2	**8.42**	41.0	84.3	35
Renoise	KL	2.96	85.4	**92.3**	51.8	γ_{EDM}	KL	8.50	41.0	**84.4**	35
	rKL	**2.92**	85.4	92.1	52.1		rKL	8.50	**41.6**	84.1	35

Finally, following common practice, we approximate $h_f(\mu_E(s_t)/\mu_\theta(s_t))$ by $h_f(\mu_E(s_t)/\mu_{\theta_0}(s_t))$ to avoid recomputing the density ratio at every policy update. We provide a detailed Algorithm 1 in Appendix B.1.

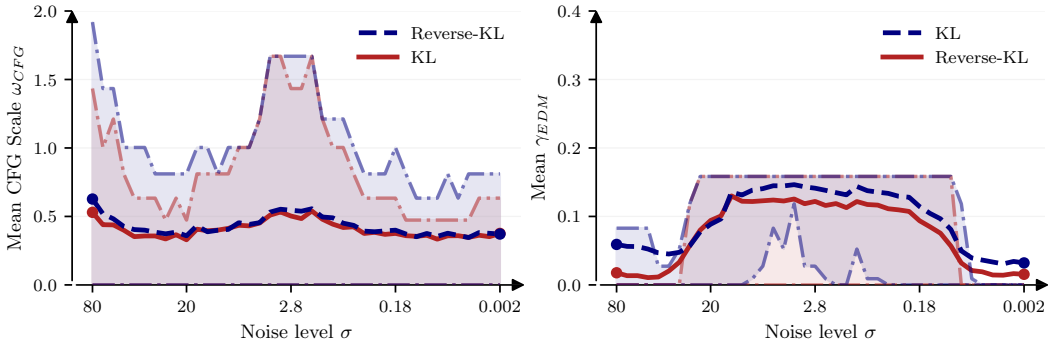
5 Experiments

We evaluate whether state-occupancy matching can replace manual sampler tuning once the denoiser is fixed. The experiments are organized around three sampling-time interventions: adaptive classifier-free guidance, stochasticity injection, and renoising. The main results in Table 2 support three claims: learned policies can match or improve strong hand-tuned samplers, the best intervention is dataset-dependent, and the choice of f-divergence induces a stable precision–recall trade-off. The ablations then isolate the design choices that make these policies effective.

5.1 Experimental Setup

Experiments are conducted on CIFAR-10 (32×32), FFHQ (64×64), and ImageNet (64×64) using pretrained EDM models [22], and on ImageNet (512×512) using the latent EDM2-XS model [23], which runs in the latent space of a pretrained VAE. The expert conditional distribution $p_E(x|\sigma)$ is the empirical dataset distribution at noise level $\Sigma_0 = 0$ with additional Gaussian noise at level σ. All policy generations are evaluated using FID [16] with 50k samples, Precision and Recall [27] using the improved topological method of Kim et al. [25], and the number of function evaluations (NFE) when relevant.

Computational cost Table 1 compares training and inference costs of our method against systematic hyperparameter tuning via grid search. As the number of hyperparameters or the denoiser complexity grows, grid search, which follows the approach of Karras et al. [22], becomes prohibitively expensive by requiring to pay the computational cost of FID evaluation for every hyperparameter configuration. In contrasts, with our approach, the overall cost comes from training the policy only once. At inference time, the policy adds between 15% and 40% overhead in FLOPs, less than one extra denoiser evaluation, and this overhead shrinks as the denoiser grows; quality and NFE are reported in Table 2.



(a) Adaptive guidance on FFHQ

(b) Adaptive stochasticity on FFHQ

Figure 2: Learned $\gamma_{\text{EDM}}(x, \sigma)$ and $\omega(x, \sigma)$ profiles on FFHQ for different f-divergences. At each noise level, the solid curve is the average value selected by the policy across generated samples, and the shaded band shows the interquartile range (25th–75th percentiles).

5.2 Results

We evaluate three strategies, all learned with the same state-occupancy matching objective and without modifying the denoiser. Table 2 reports FID, Precision, Recall, and NFE. Baselines are chosen per strategy: best constant-guidance CFG from a grid search, the EDM stochasticity hyperparameters from Karras et al. [22], and EDM with reduced NFE for comparison to the adaptive renoising policies.

Analysis 1: adaptive guidance improves the quality–diversity trade-off We learn an adaptive guidance schedule $\omega(x, \sigma)$ on CIFAR-10 and FFHQ using conditional EDM models. The action space consists of a discrete set of guidance scales; the baseline is the best constant guidance value found by grid search (ω=0.2 for CIFAR-10, ω=0.1 for FFHQ). Adaptive policies improve the fixed-guidance baseline on the main quality–diversity trade-off. KL achieves better Recall (diversity), while rKL improves Precision (fidelity). The learned profile in Figure 2a reveals that the policy applies high guidance to a small fraction of samples at each timestep, concentrating this effect at medium-to-low noise levels corresponding to the speciation phase of sampling [4], where topological structures in the images emerge.

Analysis 2: stochasticity injection is hard to hand tune We learn a state-dependent stochasticity profile $\gamma_{\text{EDM}}(x, \sigma)$ on CIFAR-10, FFHQ, and ImageNet. The action space consists of a discrete set of noise injection levels; the baseline is EDM with the stochasticity hyperparameters proposed by Karras et al. [22]. The learned policy matches or improves this baseline on most metrics, while retaining the same denoiser and number of function evaluations. On FFHQ (Figure 2b), the learned profile recovers the qualitative EDM pattern: little stochasticity at early noise levels and almost none near the end, with KL injecting more noise overall to increase diversity. The CIFAR-10 profiles are more state-dependent (Appendix B.8, Figure B.8), illustrating why exhaustive search over the four EDM hyperparameters is hard to replace manually.

Analysis 3: renoising trades quality against computation We learn an adaptive renoising schedule on CIFAR-10, FFHQ, and ImageNet. At each step, the policy chooses to continue denoising or revert to one of M=4 previous noise levels; the baseline is EDM run with a reduced NFE budget chosen to compare against the compute regime reached by adaptive renoising. Renoising improves FID on CIFAR-10 and ImageNet, and substantially reduces NFE on ImageNet relative to the restart-style EDM baseline (Table 2); on FFHQ, however, the deterministic EDM sampler remains stronger in FID, leaving little room for adaptive renoising. KL-based policies reach the terminal state faster (lower NFE), while rKL allocates more computation to high-noise regions, resulting in higher NFE without additional FID gains.

Analysis 4: divergence choice controls precision and recall Across the three interventions, the choice of divergence provides a direct and interpretable mechanism for controlling the precision–recall trade-off [31, 42]. KL is mode-covering: it encourages the policy to visit diverse regions of the data distribution, improving Recall at the cost of Precision. rKL is mode-seeking: it concentrates probability mass on high-density regions, improving Precision at the cost of Recall. This behavior is

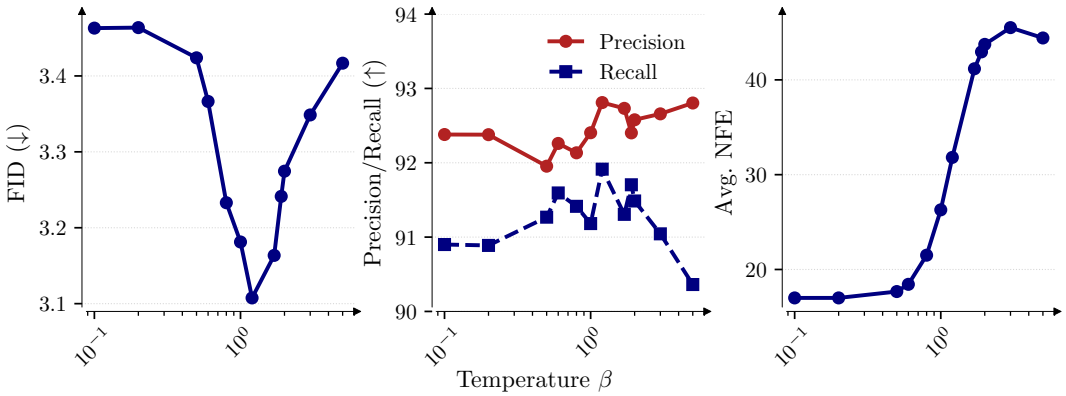


Figure 3: Effect of temperature β on Precision, Recall, and NFE for learned renoise policies on CIFAR-10. FID increases for $\beta < 1$ highlighting the importance of stochasticity, while higher β increases NFE without significant Precision/Recall changes.

visible across guidance, stochasticity injection, and renoising, and is consistent with the occupancy-measure decomposition of Section 4.2.

5.3 Ablation Studies

The ablations address the main implementation questions raised by the learned-policy formulation: whether stochastic policies are necessary, whether the method is sensitive to the selected divergence, whether state conditioning matters, and how much architecture/action-space design contributes to performance.

Temperature The policy entropy can be adjusted at inference time via a temperature parameter β by sampling action a_t from $\pi_\theta(a_t \mid s_t)^{1/\beta}/Z_\beta$. As shown in Figure 3, higher β increases NFE as the policy more frequently revisits higher-noise states, but Precision and Recall remain stable across a wide range, confirming robustness to entropy changes. Lower β (more deterministic) degrades FID, suggesting that deterministic schedules commonly used in practice may be inherently sub-optimal. A similar behavior is observed for γ_{EDM} policies (Appendix B.9).

Choice of f-divergence We evaluate additional f-divergences beyond KL and rKL for adaptive CFG on CIFAR-10 (Appendix B.7, Table B.10). All choices yield competitive FID scores, confirming that the framework is robust to this hyperparameter. However, symmetric divergences (TV, Hellinger, JS) do not exhibit the same strong mode-covering or mode-seeking tendency as KL or rKL, and produce less interpretable guidance profiles. KL and rKL therefore remain the recommended choices when a controlled precision–recall trade-off is desired.

Table 3: State conditioning ablation for adaptive CFG on CIFAR-10: policies conditioned on the noise level σ, on the sampler state (x, σ), or on (x, σ, c) with the class label. Bold: best within each divergence block.

Divergence	Policy	FID ↓	Prec ↑	Rec ↑
KL	$\pi(\sigma)$	2.26	80.3	71.1
	$\pi(x, \sigma)$	**2.02**	79.8	**71.9**
	$\pi(x, \sigma, c)$	2.12	**81.1**	70.6
rKL	$\pi(\sigma)$	2.25	80.3	**71.1**
	$\pi(x, \sigma)$	2.38	**81.2**	70.6
	$\pi(x, \sigma, c)$	**2.13**	81.1	70.4
EDM + CFG	—	2.07	80.6	70.9

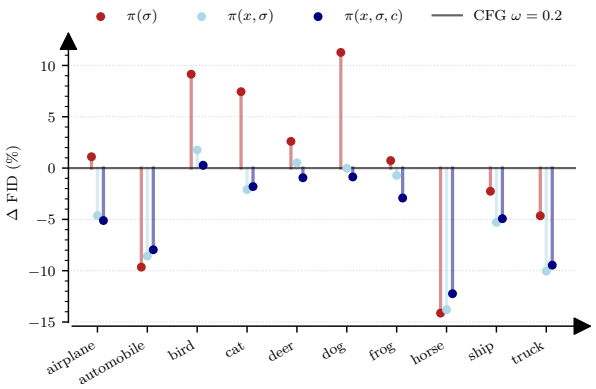


Figure 4: Per-class FID for CFG policies on CIFAR-10: conditioning on x improves the policy.

State conditioning We compare three conditioning variants for the adaptive CFG policy on CIFAR-10: $\pi(\sigma)$ (noise level only), $\pi(x, \sigma)$ (state-dependent), and $\pi(x, \sigma, c)$ (state-dependent with explicit class label c). Table 3 reports results for both KL and rKL divergences. Adding x to the conditioning brings consistent FID improvements over $\pi(\sigma)$, confirming that per-sample adaptivity is essential and that noise-level alone is insufficient to determine the optimal guidance scale. Crucially, $\pi(x, \sigma)$ achieves performance close to $\pi(x, \sigma, c)$, indicating that the policy implicitly recovers class-discriminative information from the image content x alone without requiring explicit class supervision. Figure 4 further shows that this gain is not driven by a single class: the state-dependent policy improves over the noise-level-only policy across most classes while remaining close to explicit class conditioning. This suggests that the state representation carries enough structure for the policy to identify the modality of each sample and adapt guidance accordingly. Appendix B.6 provides a qualitative trajectory comparison illustrating how state-dependent guidance changes the sampling path.

Action space and policy backbone Action space design and backbone choice both significantly impact performance (Appendix B.11, B.12, Tables B.11, B.12). Each strategy admits an optimal action space cardinality beyond which performance saturates, confirming that expressivity matters but should be kept moderate. Using a pretrained ADM encoder [8] as a frozen feature extractor is crucial: removing it causes a large FID drop ($2.05 \rightarrow 3.04$), as rich perceptual features greatly facilitate policy and density-ratio training.

Table 4: Combining γ_{EDM} and CFG (NFE $=$ 35). *Composition*: independently trained policies applied jointly, without retraining; *Multi-head*: one policy predicting both actions.

Strategy	Method	FID ↓	Prec ↑	Rec ↑
CIFAR-10 32×32				
Baseline	EDM	1.98	78.7	72.9
	EDM+CFG	2.07	80.6	70.9
Composition	KL	**1.75**	79.0	73.2
	rKL	1.79	78.9	**73.3**
Multi-head	KL	2.03	80.2	70.5
	rKL	4.81	**82.2**	65.5
ImageNet 512×512				
Baseline	EDM2	8.42	41.0	**84.3**
	EDM2+CFG	**3.80**	42.3	77.6
Composition	rKL	3.83	**42.4**	76.8

Combining policies and training multi-head policies We combine guidance and stochasticity injection, either by applying independently trained policies jointly at inference, or by training a single two-head policy (Table 4). Renoising is excluded, as it modifies the trajectory and conflicts with the other actions. Without retraining, composition reduces FID on CIFAR-10 from 1.98 to 1.75, and matches the tuned EDM2+CFG baseline on ImageNet-512 (3.83 vs. 3.80). The joint KL policy improves over EDM+CFG (2.03 vs. 2.07) but not over composition, while rKL collapses to high-Precision, low-Recall samples.

6 Conclusion

We introduced an inference-time framework for adapting diffusion sampling by optimizing sampling policies via inverse reinforcement learning, without retraining the denoiser or learning explicit rewards. Experiments on CIFAR-10, FFHQ, and ImageNet indicate improved cost–quality trade-offs across multiple sampling strategies. From a practical standpoint, the framework replaces costly hyperparameter grid searches with a single training run. On ImageNet-64, this reduces the computational budget from up to 2869 EFLOPs (3-hyperparameter grid search, 8^3 trials) to 161 EFLOPs, while adding only 16% overhead at inference time. This makes systematic sampling optimization feasible even for large-scale models where per-trial evaluation costs are prohibitive.

Discussion and limitations. Our experiments suggest that learned samplers are most useful when the policy is state-dependent and that its action is difficult to summarize by a small hand-tuned deterministic schedule. The improvement (over fixed settings) is clear for adaptive guidance and stochasticity injection, while results obtained for the renoising strategy are more contrasted as the benefit depends on the dataset. The current framework also depends on discrete action-space design and can be sensitive to policy initialization. Finally, the policy is trained on the same data as the denoiser; whether it transfers when the two differ remains an open question. Future work should therefore study more robust initialization, continuous or adaptive action spaces, and extensions to stronger pretrained models and other generative processes.

AI use statement

A generative AI assistant was used to support literature searches, discuss mathematical expressions and experimental results, and assist with manuscript drafting, editing, and LaTeXpreparation. The authors take responsibility for all claims, citations, derivations, reported results, and the final content of the paper.

Acknowledgments and Disclosure of Funding

This work was granted access to the HPC resources of IDRIS under the allocations 2025-AD011016159R1, 2025-A0181016159 and 2025-AD011017138 made by GENCI. This project was partly funded by the French government's Agence Nationale de la Recherche under the "France 2030" program (ANR-23-IACL-0008, PRAIRIE-PSAI and ANR-22-PESN-0014).

References

[1] Siddhant Agarwal, Peter Stone, Ishan Durugkar, and Amy Zhang. f -Policy Gradients: A General Framework for Goal Conditioned RL using f -Divergences. In *Advances in Neural Information Processing Systems*, 2023. NeurIPS 2023.

[2] Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator Rejection Sampling. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1GkToR5tm.

[3] Iskander Azangulov, Peter Potaptchik, Qinyu Li, Eddie Aamari, George Deligiannidis, and Judith Rousseau. Adaptive diffusion guidance via stochastic optimal control, 2025. URL https://arxiv.org/abs/2505.19367.

[4] Giulio Biroli, Tony Bonnaire, Valentin de Bortoli, and Marc Mézard. Dynamical Regimes of Diffusion Models. *Nature Communications*, 15:9957, 2024. URL https://www.nature.com/articles/s41467-024-54281-3.

[5] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=YCWjhGrJFD.

[6] Imre Csiszár. Information-type measures of difference of probability distributions and indirect observations. *Studia Scientiarum Mathematicarum Hungarica*, 2:299–318, 1967.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848. URL https://ieeexplore.ieee.org/document/5206848. ISSN: 1063-6919.

[8] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

[9] Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control, 2025. URL https://arxiv.org/abs/2409.08861.

[10] Ying Fan and Kangwook Lee. Optimizing DDPM sampling with shortcut fine-tuning. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 9623–9639, 2023. URL https://arxiv.org/abs/2301.13362.

[11] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Reinforcement learning for fine-tuning text-to-image diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[12] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models, 2016. URL https://arxiv.org/abs/1611.03852. Workshop on Adversarial Training, NIPS 2016.

[13] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A Divergence Minimization Perspective on Imitation Learning Methods. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1259–1277. PMLR, 2020. URL https://proceedings.mlr.press/v100/ghasemipour20a.html.

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. In *27th Conference on Neural Information Processing Systems (NeurIPS 2014)*, June 2014. URL http://arxiv.org/abs/1406.2661. arXiv: 1406.2661.

[15] Xiefan Guo, Miaomiao Cui, Liefeng Bo, and Di Huang. ShortFT: Diffusion model alignment via shortcut-based fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 678–687, 2025.

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html.

[17] Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/cc7e2b878868cbae992d1fb743995d8f-Abstract.html.

[18] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

[20] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta Go Fast When Generating Data with Score-Based Models, May 2021. URL http://arxiv.org/abs/2105.14080. arXiv:2105.14080 [cs, math, stat].

[21] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Karras_A_Style-Based_Generator_Architecture_for_Generative_Adversarial_Networks_CVPR_2019_paper.html.

[22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/a98846e9d9cc01cfb87eb694d946ce6b-Abstract-Conference.html.

[23] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and Improving the Training Dynamics of Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. URL https://arxiv.org/abs/2312.02696.

[24] Dongjun Kim, Yeongmin Kim, Se Jung Kwon, Wanmo Kang, and Il-Chul Moon. Refining Generative Process with Discriminator Guidance in Score-based Diffusion Models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 16567–16598. PMLR, July 2023. URL https://proceedings.mlr.press/v202/kim23i.html.

[25] Pum Jun Kim, Yoojin Jang, Jisu Kim, and Jaejun Yoo. TopP&R: Robust Support Estimation Approach for Evaluating Fidelity and Diversity in Generative Models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/185969291540b3cd86e70c51e8af5d08-Abstract-Conference.html.

[26] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[27] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved Precision and Recall Metric for Assessing Generative Models. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.*, October 2019. arXiv: 1904.06991.

[28] Holden Lee, Jianfeng Lu, and Yixin Tan. Convergence for score-based generative modeling with polynomial complexity, May 2023. URL http://arxiv.org/abs/2206.06227. arXiv:2206.06227 [cs, math, stat].

[29] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/260a14acce2a89dad36adc8eefe7c59e-Abstract-Conference.html.

[30] Tianwei Ni, Harshit Sikchi, Yufei Wang, Tejus Gupta, Lisa Lee, and Benjamin Eysenbach. f-IRL: Inverse Reinforcement Learning via State Marginal Matching. In *Proceedings of the Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 529–551. PMLR, 2021. URL https://proceedings.mlr.press/v155/ni21a.html.

[31] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/cedebb6e872f539bef8c3f919874e9d7-Abstract.html.

[32] Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[33] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing Training of Generative Adversarial Networks through Regularization. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/7bccfde7714a1ebadf06c5f4cea752c1-Abstract.html.

[34] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. In *Advances in Neural Information Processing Systems*, volume 35, pages 36479–36494, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/ec795aeadae0b7d230fa35cbaf04c041-Abstract-Conference.html.

[35] Riccardo De Santi, Marin Vlastelica, Ya-Ping Hsieh, Zebang Shen, Niao He, and Andreas Krause. Provable maximum entropy manifold exploration via diffusion models, 2025. URL https://arxiv.org/abs/2506.15385.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. URL http://arxiv.org/abs/1707.06347. arXiv:1707.06347 [cs].

[37] Jiaming Song and Stefano Ermon. Bridging the Gap Between f-GANs and Wasserstein GANs. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9078–9087. PMLR, November 2020. URL https://proceedings.mlr.press/v119/song20a.html. ISSN: 2640-3498.

[38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

[39] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

[40] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density Ratio Estimation: A Comprehensive Review. In *Proceedings of Workshop on Information-Based Induction Sciences*, volume 1703 of *RIMS Kôkyûroku*, pages 10–31, 2010. URL http://hdl.handle.net/2433/170027.

[41] Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control, 2024. URL https://arxiv.org/abs/2402.15194.

[42] Alexandre Verine. *Quality and Diversity in Generative Models through the lens of f-divergences*. PhD thesis, Université PSL, January 2024. URL https://theses.fr/279916922.

[43] Alexandre Verine, Ahmed Mehdi Inane, Florian Le Bronnec, Benjamin Negrevergne, and Yann Chevaleyre. Improving Discriminator Guidance in Diffusion Models. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, Lecture Notes in Computer Science, 2025. URL https://arxiv.org/abs/2503.16117.

[44] Yilun Xu, Mingyang Deng, Xiang Cheng, Yonglong Tian, Ziming Liu, and Tommi Jaakkola. Restart Sampling for Improving Generative Processes. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f2543511e5f4d4764857f9ad833a977d-Abstract-Conference.html.

[45] Sangwoong Yoon, Himchan Hwang, Dohyun Kwon, Yung-Kyun Noh, and Frank C. Park. Maximum Entropy Inverse Reinforcement Learning of Diffusion Models with Energy-Based Models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2bed6c14cd5ea97a9bc1e6094941bde7-Abstract-Conference.html.

[46] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 23, pages 1433–1438, 2008. URL https://aaai.org/papers/01433-AAAI08-227-maximum-entropy-inverse-reinforcement-learning/.

Appendix

A Proofs

A.1 Gradient formula proof

Proof. Let's consider our objective function:

$$\mathcal{L}_f(\theta) = \mathcal{D}_f(\mu_E \| \mu_\theta)$$

We can compute the gradient of $\mathcal{L}_f(\theta)$ with respect to θ using the chain rule and the properties of the f-divergence:

$$\begin{aligned}
\nabla_\theta \mathcal{L}_f(\theta) &= \nabla_\theta \mathcal{D}_f(\mu_E \| \mu_\theta) \\
&= \nabla_\theta \int_{\mathcal{S}} f\left(\frac{\mu_E(s)}{\mu_\theta(s)}\right) \mu_\theta(s)\, ds \\
&= \int_{\mathcal{S}} \nabla_\theta \mu_\theta(s)\, f\left(\frac{\mu_E(s)}{\mu_\theta(s)}\right) - \mu_\theta(s) \frac{\nabla_\theta \mu_\theta(s)}{\mu_\theta(s)^2} \mu_E(s) \dot{f}\left(\frac{\mu_E(s)}{\mu_\theta(s)}\right) ds \\
&= \int_{\mathcal{S}} \nabla_\theta \mu_\theta(s)\, h_f\left(\frac{\mu_E(s)}{\mu_\theta(s)}\right) ds \\
&= \int_{\mathcal{S}} \nabla_\theta \log \mu_\theta(s) h_f\left(\frac{\mu_E(s)}{\mu_\theta(s)}\right) \mu_\theta(s) ds
\end{aligned}$$

with $h_f(x) = f(x) - x\dot{f}(x)$.

Now, using the property of the state occupancy measure, we can write:

$$\begin{aligned}
\nabla_\theta \mathcal{L}_f(\theta) &= \mathbb{E}_{\boldsymbol{s}_{1:T} \sim p_\theta}\left[\frac{1}{T} \sum_{t'=1}^{T} \nabla_\theta \log \mu_\theta(s_{t'}) h_f\left(\frac{\mu_E(s_{t'})}{\mu_\theta(s_{t'})}\right)\right] \\
&= \mathbb{E}_{\boldsymbol{s}_{1:T} \sim p_\theta}\left[\frac{1}{T} \sum_{t'=1}^{T} \left(\sum_{t=1}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)\right) h_f\left(\frac{\mu_E(s_{t'})}{\mu_\theta(s_{t'})}\right)\right] \\
&= \mathbb{E}_{\boldsymbol{s}_{1:T} \sim p_\theta}\left[\frac{1}{T} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t' \geq t} h_f\left(\frac{\mu_E(s_{t'})}{\mu_\theta(s_{t'})}\right)\right]
\end{aligned}$$

by reindexing the sums. □

A.2 Adding importance sampling

Proof. Adding importance sampling to the gradient formula derived in Theorem 4.1 gives:

$$\nabla_\theta \mathcal{L}_f(\theta) = \mathbb{E}_{\boldsymbol{s}_{1:T} \sim p_{\theta_0}}\left[\frac{p_\theta(\boldsymbol{s}_{1:T})}{p_{\theta_0}(\boldsymbol{s}_{1:T})} \frac{1}{T} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t' \geq t} h_f\left(\frac{\mu_E(s_{t'})}{\mu_\theta(s_{t'})}\right)\right]$$

where θ_0 is the parameter of the policy used to sample the trajectories. □

B Experimental Details

We parameterize the sampling policy and the state density estimator using a shared neural architecture, implemented as a dual-head network based on an EDM-style Encoder–UNet. The resulting model enables joint representation learning while keeping the policy and density objectives decoupled.

Backbone and Preconditioning Given a noisy state x and noise level σ, we apply the standard EDM preconditioning and map σ to the corresponding normalized sigma variable τ. The network is conditioned on τ and, when applicable, on class labels for conditional generation. For non-stationary policies, the discrete trajectory step index is additionally provided as an explicit conditioning variable. We also use a pretrained ADM encoder as a frozen feature extractor. The input image is first processed by the ADM encoder, and the resulting latent representation is used as input to the policy and density heads. All ADM parameters are kept fixed throughout training.

Policy Head The policy head outputs logits over a discrete action space of size A, where each action corresponds to a sampling-time control operation (e.g., continuing denoising or restarting to a previous noise level). The policy head is implemented as convolutional layers with output channels equal to A, sharing the same architectural design and conditioning mechanisms as the

density head. For stationary policies, the policy is conditioned only on the current state and noise level. For non-stationary policies, the trajectory step index is included as an additional conditioning signal. During training and inference, actions are sampled from the resulting categorical distribution.

Density Head The density head estimates a scalar quantity associated with the state distribution, used for density-ratio-based training objectives. It is implemented as convolutional layers with a single scalar output. The density head shares the same backbone structure and conditioning as the policy head but has independent parameters in the output layers. The output layers of the density head are initialized using Xavier initialization to stabilize early training.

B.1 Training Details

Initialization The initialization of both the policy and density networks plays an important role in practice. Since the learning problem is non-convex, we do not guarantee convergence to an unique policy, and different initializations may lead to different local optima. We initialize the policy network using a task-specific heuristic corresponding to a reasonable sampling strategy (for instance favoring continuation of denoising at high noise levels and allowing restarts at later stages). This initialization provides a stable starting point and significantly improves training stability compared to random initialization.

The density network is initialized using expert trajectories and rollouts generated by the initialized policy. These expert samples are used to initialize the density-ratio estimator before joint training with the policy, which helps stabilize the early stages of optimization.

Optimization For both the policy and density networks, we apply an exponential moving average (EMA) to the network parameters during training [32, 23]. EMA is used for evaluation and inference, as it consistently yields more stable behavior and improved performance. The density-ratio estimate is further stabilized by a 120-step warm-up of the density network before any policy update, and by predicting log-ratios rather than probabilities, which avoids sigmoid saturation. Finally, when training the density estimator, we apply label smoothing at high noise levels. At large noise scales, the input states contain limited information, and hard labels can lead to unstable gradients. Label smoothing in this regime improves robustness and reduces variance during training.

We provide details of the complete training procedure in Table B.5.

Table B.5: Parameters used for training the sampling policies across different datasets and strategies. Compute is reported in normalized GPU hours (V100-equivalent: one H100 hour counts as 4 normalized hours).

Parameters	Renoise			Gammas				Guidance		
	CIFAR-10	FFHQ 64×64	ImageNet 64×64	CIFAR-10	FFHQ 64×64	ImageNet 64×64	ImageNet 512×512†	CIFAR-10	FFHQ 64×64	ImageNet 512×512†
Cond. model	×	×	✓	×	×	✓	✓	✓	✓	✓
LR	1e-4	1e-5	1e-5	1e-4	1e-5	1e-5	1e-5	1e-4	1e-5	1e-5
# traj.	1024	1536	2048	1024	1536	2048	512	1024	1536	256
# DRE Init.	100	200	200	100	200	200	400	100	200	400
Action dim.	4	4	4	10	10	20	10	20	20	20
Res.	32^2	64^2	64^2	32^2	64^2	64^2	512^2	32^2	64^2	512^2
# params.	5.8M	5.8M	6.8M	5.8M	5.8M	6.8M	0.37M‡	5.8M	5.8M	0.37M‡
# Noise levels	9	20	20	18	40	256	18	18	40	18
Norm. GPU hrs	30	48	320	60	80	640	348	100	150	200
GPU	V100	V100	H100	V100	V100	H100	MI250	V100	V100	A100

† Latent diffusion on a frozen EDM2-XS backbone (125M params); policy/density heads are trained on top of the frozen, shared EDM2 encoder rather than an independently trained encoder. ‡ Trainable parameters of the policy/density heads only.

B.2 Density Ratio Estimation

Discriminator-based density ratio estimation Estimating density ratios in continuous spaces is a classical problem, with approaches ranging from kernel density estimation to direct ratio fitting and probabilistic modeling [40]. In this work, we adopt a classifier-based approach, which has been shown to scale effectively to high-dimensional settings [17, 12, 37, 33, 2] and has successfully been applied to diffusion models [24, 43].

We train a binary classifier $D_\varphi(x, \sigma)$ to discriminate between states sampled from the expert and from the policy, conditioning on the noise level σ. Since the expert provides only marginal distributions at each noise level, we sample σ uniformly and draw states independently from $p_E(\cdot \mid \sigma)$ and $p_\theta(\cdot \mid \sigma)$. The classifier is trained by minimizing the binary cross-entropy loss

$$\mathcal{L}_D(\varphi) = \mathbb{E}_{\sigma \sim \mathbb{U}[\Sigma_0, \ldots, \Sigma_N]} \Big[- \mathbb{E}_{x \sim p_E(\cdot \mid \sigma)} \big[\log D_\varphi(x, \sigma) \big] - \mathbb{E}_{x \sim p_\theta(\cdot \mid \sigma)} \big[\log(1 - D_\varphi(x, \sigma)) \big] \Big].$$

Assuming an optimal discriminator, the density ratio is

$$\frac{p_E(x \mid \sigma)}{p_\theta(x \mid \sigma)} = \frac{D_\varphi(x, \sigma)}{1 - D_\varphi(x, \sigma)} = \exp\Big(\tilde{D}_\varphi(x, \sigma) \Big), \tag{10}$$

where $\tilde{D}_\varphi(x, \sigma)$ denotes the logit output of the classifier.

Algorithm 1 Policy Optimization via State-Marginal f-Divergence Minimization

Require: Pretrained denoiser F, policy π_θ, discriminator D_φ, replay buffer B, number of epochs n_{epoch}, batch size N_{batch}, expert state samples $\mathcal{S}_E$, clipping parameter ϵ, learning rate η, and policy update frequency K.

Initialize policy π_θ and discriminator D_φ.
for $k = 1$ **to** n_{epoch} **do**
  if $k \bmod K = 0$ **then**
    Generate trajectories

$$\boldsymbol{s}_{1:T} = \{(s_t, a_t)\}_{t=1}^{T}$$

    using policy π_θ and denoiser F.
    Train discriminator D_φ to distinguish expert states $\mathcal{S}_E$ from policy-generated states $\{s_t\}$.
    Estimate density ratios using

$$\frac{\mu_E(x,\sigma)}{\mu_\theta(x,\sigma)} \;\approx\; \frac{D_\varphi(x,\sigma)}{1 - D_\varphi(x,\sigma)} \times \frac{w_E(\sigma)}{w_\theta(\sigma)}.$$

    Compute advantage estimates

$$A_t \;=\; \frac{1}{T}\sum_{t'=t}^{T} h_f\left(\frac{\mu_E(s_{t'})}{\mu_\theta(s_{t'})}\right), \quad h_f(x) = f(x) - xf'(x),$$

    and store (s_t, a_t, A_t) in buffer B.
    Save current policy parameters: $\theta_0 \leftarrow \theta$.
  end if
  for each minibatch $\{(s_i, a_i, A_i)\}_{i=1}^{N_{\text{batch}}}$ sampled from B **do**
    Compute PPO-style objective:

$$\hat{\mathcal{L}}_f(\theta) = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \max\big(r_i(\theta)\,(A_i - \bar{A}),\, \text{clip}\,(r_i(\theta),\, 1-\epsilon,\, 1+\epsilon)\;(A_i - \bar{A})\big)\,,$$

    where

$$r_i(\theta) = \frac{\pi_\theta(a_i \mid s_i)}{\pi_{\theta_0}(a_i \mid s_i)}, \qquad \bar{A} = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} A_i.$$

    Update policy parameters:

$$\theta \leftarrow \theta - \eta \nabla_\theta \hat{\mathcal{L}}_f(\theta).$$

  end for
end for

Ensure: Trained sampling policy π_θ.

Computing the occupancy measure ratio Because expert trajectories are unavailable, the expert is specified only through state marginals at each noise level. As a result, the discriminator estimates a density ratio under a uniform distribution over noise levels rather than under the policy-induced state occupancies. In addition, while the policy accumulates occupancy mass at the terminal noise level, the expert does not provide temporal occupancy information.

We compute the full occupancy measure ratio by using importance sampling corrections to account for the mismatch between sampling and target proportions.

$$\frac{\mu_E(x,\sigma)}{\mu_\theta(x,\sigma)} = \frac{w_E(\sigma)}{w_\theta(\sigma)} \times \frac{p_E(x|\sigma)}{p_\theta(x|\sigma)}. \tag{11}$$

As previously discussed, the term w_E is a design choice while w_θ is induced by the policy and is estimated empirically from the simulated trajectories. In practice w_E matters little when the number of noise levels matches the number of sampling steps, since the weighting is then nearly uniform; we observed an effect only in low-dimensional renoising experiments, where a larger terminal weight shortens trajectories.

B.3 Wall-Clock Time and Memory

Table 1 reports the inference overhead of the policy in FLOPs. Table B.6 complements it with measured wall-clock time and peak GPU memory for a single network evaluation (batch size 1, V100 32GB, averaged over 50 runs after 10 warm-up runs). The policy adds roughly 21–25 ms per evaluation, an approximately constant cost that becomes relatively smaller as the denoiser grows ($+103\%$ on CIFAR-10 vs. $+57\%$ on ImageNet-64), and a negligible memory footprint (at most 6 MB).

Table B.6: Wall-clock time and peak GPU memory of a single forward pass, with and without the policy (batch size 1, V100 32GB, mean over 50 runs after 10 warm-up runs).

	Wall-clock time (ms)			Peak memory (MB)		
Dataset	Denoiser	+ Policy	Overhead	Denoiser	+ Policy	Overhead
CIFAR-10 32×32	20.80	42.24	+21.44 (+103%)	474	480	+6
FFHQ 64×64	26.82	51.52	+24.70 (+92%)	589	589	+0
ImageNet 64×64	43.89	68.85	+24.96 (+57%)	1565	1565	+0

B.4 Results with Variability

The main paper reports point estimates in Table 2 to keep the comparison readable. For guidance and stochasticity injection on CIFAR-10 and FFHQ, learned policies were trained and evaluated over five random seeds; Tables B.7 and B.8 provide mean$\pm$standard deviation values when available; remaining entries are single point estimates.

Table B.7: Results across all strategies with mean±standard deviation when repeated runs are available. Variability is reported for repeated CIFAR-10 and FFHQ learned-policy evaluations; ImageNet results are single-run evaluations due to the cost of 50k-sample metrics at that scale.

Strategy	Method	FID ↓	Prec ↑	Rec ↑	NFE
CIFAR-10 32×32					
CFG	EDM+CFG	2.07	80.6	70.9	35
	KL	$\mathbf{2.02} \pm 0.15$	79.8 ± 0.9	$\mathbf{71.9} \pm 1.1$	35
	rKL	2.38 ± 0.14	$\mathbf{81.2} \pm 0.9$	70.6 ± 0.6	35
γ_{EDM}	EDM	**1.98**	78.7	72.9	35
	KL	2.01 ± 0.03	78.8 ± 0.2	$\mathbf{73.1} \pm 0.3$	35
	rKL	2.04 ± 0.02	$\mathbf{79.4} \pm 0.3$	72.3 ± 0.4	35
Renoise	EDM	3.42	76.7	**72.5**	17
	KL	**3.18**	77.3	**72.5**	17.7
	rKL	3.21	**78.1**	71.6	30.5
FFHQ 64×64					
CFG	EDM+CFG	3.11	90.7	87.8	79
	KL	$\mathbf{3.03} \pm 0.02$	90.6 ± 0.2	$\mathbf{88.6} \pm 0.1$	79
	rKL	3.04 ± 0.01	$\mathbf{90.8} \pm 0.1$	88.3 ± 0.1	79
γ_{EDM}	EDM	2.56	90.1	**89.4**	79
	KL	2.53 ± 0.01	90.4 ± 0.1	88.2 ± 0.2	79
	rKL	$\mathbf{2.49} \pm 0.02$	$\mathbf{90.9} \pm 0.1$	87.9 ± 0.1	79
Renoise	EDM	**2.60**	90.5	**89.0**	59
	KL	3.04	**90.9**	88.1	87.4
	rKL	3.07	**90.9**	87.9	68.2
ImageNet 64×64					
γ_{EDM}	EDM	**2.12**	85.9	91.7	511
	KL	2.14	85.7	**92.0**	511
	rKL	2.24	**86.3**	91.1	511
Renoise	EDM	3.01	**85.8**	91.8	350
	KL	2.96	85.4	**92.3**	51.8
	rKL	**2.92**	85.4	92.1	52.1

The repeated runs support the qualitative conclusions of the main table. On CIFAR-10, the standard deviations of FID are small relative to the gaps between the main trends: adaptive CFG with KL remains competitive with fixed guidance while improving Recall, and rKL consistently shifts the trade-off toward Precision. For stochasticity injection, the gap to deterministic EDM is within a few hundredths of FID, but KL and rKL still produce the expected Recall/Precision shifts. For FFHQ, the reported variability is very small for CFG and γ_{EDM}, suggesting that the observed gains are not driven by evaluation noise.

Table B.8: State conditioning ablation for adaptive CFG on CIFAR-10 with mean±standard deviation over repeated runs.

Divergence	Policy	FID ↓	Prec ↑	Rec ↑
KL	$\pi(\sigma)$	2.26 ± 0.01	80.3 ± 0.1	71.1 ± 0.2
	$\pi(x, \sigma)$	$\mathbf{2.02} \pm 0.15$	79.8 ± 0.9	$\mathbf{71.9} \pm 1.1$
	$\pi(x, \sigma, c)$	2.12 ± 0.05	$\mathbf{81.1} \pm 0.1$	70.6 ± 0.1
rKL	$\pi(\sigma)$	2.25 ± 0.01	80.3 ± 0.1	$\mathbf{71.1} \pm 0.2$
	$\pi(x, \sigma)$	2.38 ± 0.14	$\mathbf{81.2} \pm 0.9$	70.6 ± 0.6
	$\pi(x, \sigma, c)$	$\mathbf{2.13} \pm 0.01$	81.1 ± 0.3	70.4 ± 0.2
EDM + CFG	—	2.07	80.6	70.9

Table B.8 confirms that the benefit of state conditioning is robust. For KL, conditioning on (x, σ) improves FID over the σ-only policy and increases Recall, while the standard deviation remains modest. For rKL, explicit class conditioning gives the best FID, but $\pi(x, \sigma)$ still achieves the highest Precision and remains close to $\pi(x, \sigma, c)$, supporting the claim that the noisy image state already carries substantial class-discriminative information.

B.5 Learned CFG Guidance Profiles

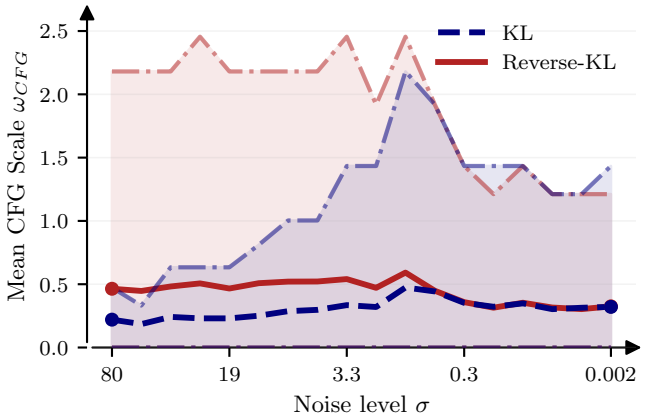


Figure B.5: Learned guidance profiles $\omega(x, \sigma)$ on CIFAR-10 for KL and rKL (mean and $[0.25, 0.75]$ quantiles). KL applies higher guidance to a larger fraction of samples, improving Recall; rKL concentrates guidance on fewer samples, improving Precision.

The CIFAR-10 profile mirrors the behavior observed on FFHQ (Figure 2a): the policy increases the proportion of highly guided samples at medium-to-low noise levels, coinciding with the speciation phase of sampling [4]. Around medium noise levels, guidance strength exhibits higher mean and variance, indicating a greater need for state-dependent control during the phase when topological structures emerge in the images.

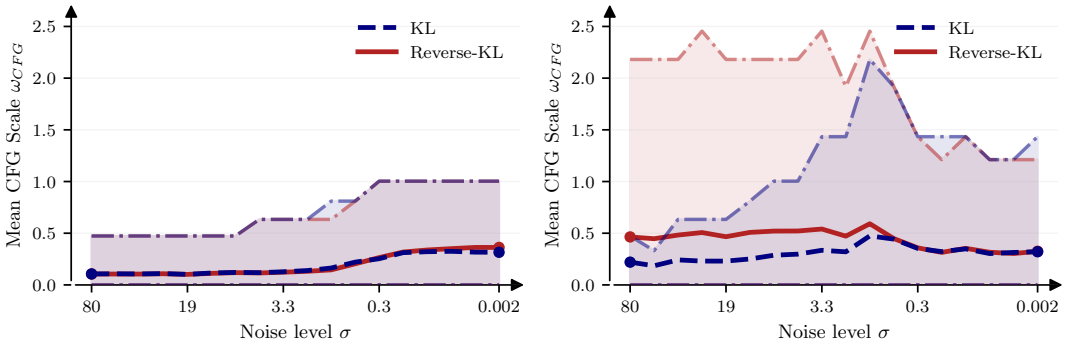
B.6 Is it necessary to have x-dependent policies?

In this appendix, we analyze whether conditioning the sampling policy on the image state x is necessary when learning adaptive classifier-free guidance (CFG). In our framework, this corresponds to comparing policies of the form $\pi_\theta(\omega \mid x, \sigma)$ against simpler policies that depend only on the noise level, $\pi_\theta(\omega \mid \sigma)$.

A noise-level–dependent policy learns a single guidance scale per noise level, which can be interpreted as a global trade-off between fidelity and diversity at each stage of the sampling process. While this is sufficient to recover standard hand-tuned guidance schedules, it cannot adapt guidance strength across samples. In contrast, an x-dependent policy can modulate the guidance scale based on the current state of the sample, allowing different images to receive different amounts of guidance at the same noise level.

Empirically, we observe that x-dependent policies consistently outperform σ-only policies in terms of sample quality. As reported in Table 3, conditioning on (x, σ) improves FID, Precision, and Recall across datasets and divergences. Qualitatively, the learned policies exhibit sparse behavior: at a given noise level, only a subset of samples receives strong guidance, while the majority remain weakly guided. This behavior cannot be captured by a single scalar guidance value shared across all samples.

This observation aligns with recent analyses of CFG, which suggest that excessive guidance can lead to mode collapse and loss of diversity, while insufficient guidance degrades fidelity. An x-dependent policy enables the sampler to selectively apply strong guidance where it is most beneficial, while preserving diversity elsewhere. Figure B.6 illustrates this difference at the policy-profile level: the x-dependent policy exhibits larger dispersion across samples, whereas the σ-only policy is forced to apply a single global schedule. In practice, this adaptivity appears crucial for achieving favorable precision–recall trade-offs without manual tuning.



(a) Noise-level-only policy $\pi(\omega \mid \sigma)$ (b) State-dependent policy $\pi(\omega \mid x, \sigma)$

Figure B.6: Comparison of x-dependent and noise-level–dependent policies for adaptive CFG on CIFAR-10 using KL and Reverse-KL divergence minimization. The x-dependent policy achieves better FID, Precision, and Recall by selectively applying guidance based on the current sample state. (a) depends only on the noise level σ; (b) uses both x and σ.

Figure B.7 provides a complementary qualitative view by comparing trajectories initialized from the same noise. The fixed-guidance sampler follows a single hand-tuned schedule, while the learned state-dependent policy applies stronger guidance only at selected medium-to-low noise levels. This changes the path through the sampler state space and can steer samples toward different image content, illustrating why per-sample adaptivity can affect quality even when the same denoiser and number of solver steps are used.

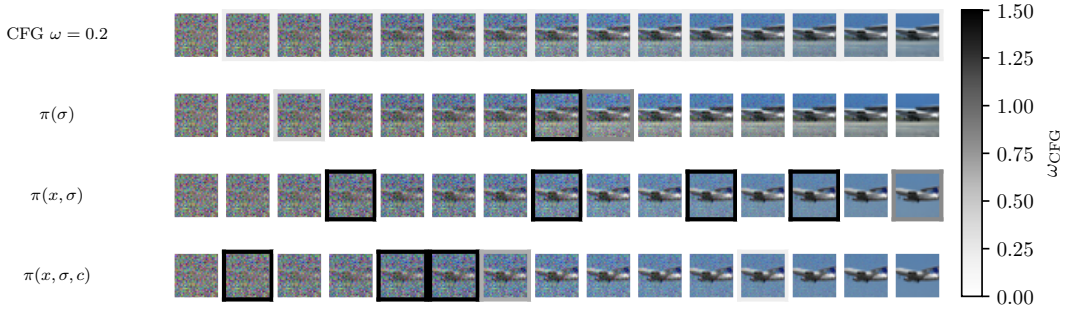


Figure B.7: Example trajectory starting from the same initial noise x for fixed CFG ($\omega = 0.2$) and learned KL-based adaptive CFG on CIFAR-10. State-dependent policies apply high guidance at medium-to-low noise levels, steering the trajectory toward different image content.

For the Gamma strategy, the comparison in Table B.9 is more mixed: conditioning on x does not consistently improve over a noise-level-only policy. This suggests that state-dependent control is especially important for guidance, while stochasticity injection can sometimes be captured by a simpler noise-level schedule.

Table B.9: Comparison of x-dependent and noise-level-dependent policies for adaptive stochasticity injection on CIFAR-10 using KL and Reverse-KL divergence minimization. Unlike adaptive CFG, x-dependence does not consistently improve this strategy, suggesting that the value of state-dependent control depends on the intervention being learned.

Method	FID ↓	Prec ↑	Rec ↑	x-dependent
KL	1.99	78.5	**73.2**	×
	2.03	78.6	72.7	✓
	2.06	79.0	72.4	✓
rKL	2.03	**79.7**	71.9	×
	2.03	78.6	72.4	✓
	2.06	79.2	72.5	✓
EDM	**1.98**	78.7	72.9	×

B.7 Impact of the Choice of f-Divergence for CFG

Table B.10 reports CFG results on CIFAR-10 (32×32) for additional f-divergences. All choices yield FID scores on par with KL and rKL, confirming that the framework is not sensitive to this choice. However, their qualitative behavior differs: Total Variation, Hellinger, and Jensen–Shannon are symmetric divergences and do not exhibit the same tendency as KL or rKL to concentrate guidance aggressively in specific regions of the noise space. The χ^2 divergence, sometimes favored for its smoothness in optimization, did not produce any noticeable improvement in learning curves compared to KL.

Table B.10: Adaptive CFG on CIFAR-10 (32×32) for different f-divergences. KL and rKL from the main paper are shown for reference (separated by a rule).

f-divergence	FID ↓	Prec ↑	Rec ↑
Total Variation	2.19	80.7	70.6
Hellinger	2.20	80.7	70.7
Jensen–Shannon	2.21	80.6	70.7
χ^2	2.21	80.6	70.8
KL	**2.05**	80.7	**71.4**
rKL	2.53	**82.4**	69.8

B.8 Learned γ_{EDM} Profiles

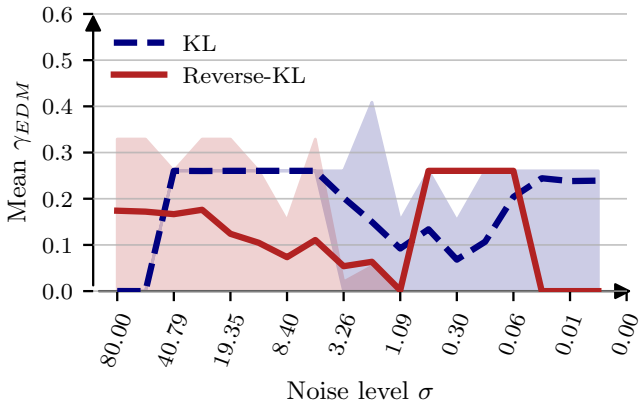


Figure B.8: Learned $\gamma_{\text{EDM}}(x, \sigma)$ profiles on CIFAR-10 for KL and rKL divergences (mean and $[0.25, 0.75]$ quantiles). The profiles lack simple structure, confirming that optimal stochasticity injection is highly state-dependent.

On CIFAR-10, rKL adds less noise with low variance at low-noise regions to increase precision, while KL does the opposite, adding more noise with higher variance to increase diversity. This behavior is consistent with the mode-seeking/mode-covering interpretation of the divergences.

B.9 Temperature

We experimented with adjusting the temperature of the policy during evaluation to study its effect on the trade-off between sample quality and diversity. Lowering the temperature sharpens the action distribution, leading to more deterministic behavior, while increasing it encourages exploration by flattening the policy over noise-injection actions. As discussed in Section 5.2, temperature therefore provides an inference-time mechanism to control the exploration–exploitation balance of the learned policy.

Figure B.9 shows that increasing temperature leads to higher Recall but also higher FID. For the Gamma strategy, this behavior can be attributed to more frequent stochasticity injection at higher noise levels, which increases sample diversity but also perturbs trajectories that would otherwise converge to high-fidelity modes. As a result, diversity gains come at the cost of reduced sample sharpness.

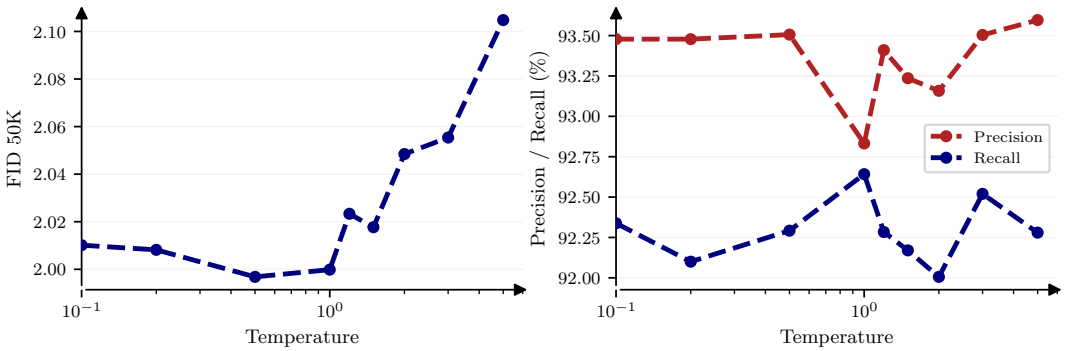


Figure B.9: Effect of temperature scaling on the learned Gamma sampling policy for CIFAR-10 using KL divergence minimization.

Notably, we observe that Precision and Recall are jointly optimized near the temperature used during training ($\beta = 1$). This suggests that the learned policy is already calibrated to balance exploration and exploitation under the training objective, and that deviating significantly from this temperature primarily trades off fidelity for diversity rather than uncovering strictly better operating points.

B.10 Controlling the Distribution of Generated Samples

We also considered settings where the output distribution of a pretrained model is biased and must be corrected at inference time. We focused on renoising strategies. On CIFAR-10, although the dataset is balanced, pretrained EDM models overproduce animal classes relative to vehicle classes. We defined a target class distribution that increases the mass of vehicle classes while preserving normalization. Figure B.10 shows that the learned renoise policy shifts the generated class distribution toward the target. While the match is not exact, the bias is substantially reduced, indicating that the policy captures the intended correction direction.

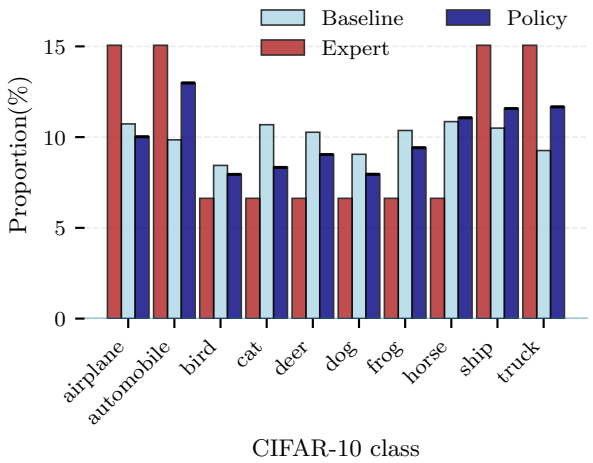


Figure B.10: Class distribution of samples generated by a pretrained EDM model on CIFAR-10, before and after applying a learned renoise policy to correct class imbalance. The target distribution increases the mass of vehicle classes (automobile, truck) relative to animal classes. The learned policy shifts the generated distribution closer to the target, reducing bias while maintaining sample quality.

A key modeling choice is whether the control policy depends explicitly on the trajectory step of the induced MDP. We distinguish between non-stationary policies $\pi_\theta(x, \sigma, t)$, which depend on the step index, and stationary policies $\pi_\theta(x, \sigma)$, which do not. Although the optimal policy for a finite sampling horizon is generally non-stationary, we find that stationary policies are significantly easier to train and often perform better in practice. Across all strategies, stationary policies consistently match or outperform non-stationary ones, yielding more stable optimization and better trade-offs between distributional control and sample quality at comparable NFE.

B.11 Dimension of the Action Space

The action space $\mathcal{A}$ consists of a discrete set of candidate values for the controlled hyperparameter (e.g., guidance scales or noise injection levels). Table B.11 reports the effect of the cardinality $|\mathcal{A}|$ on the stochasticity injection experiment on CIFAR-10.

A small action space ($|\mathcal{A}| = 5$) is insufficient to capture the range of useful noise injection levels and yields noticeably worse FID. Performance saturates quickly: $|\mathcal{A}| = 20$ already matches $|\mathcal{A}| = 100$ across all metrics, indicating that a moderate number of actions is sufficient for the policy to express the necessary behavior.

Table B.11: Effect of action space cardinality $|\mathcal{A}|$ on the stochasticity injection strategy on CIFAR-10.

card(A)	FID ↓	Prec ↑	Rec ↑
5	2.20	79.8	72.4
20	2.06	78.8	72.5
100	2.07	78.7	72.8

B.12 Role of the ADM Encoder in the Backbone

Our policy network uses a pretrained ADM encoder as a frozen feature extractor to condition the policy on the current sample x. Table B.12 ablates this design choice on the CFG experiment on CIFAR-10.

Removing the ADM encoder leads to a significant drop in performance: FID degrades from 2.05 to 3.04, Recall drops by nearly 4 points, while Precision increases, suggesting the policy collapses toward high-fidelity but less diverse modes. This confirms that rich, pretrained visual features are essential for the policy to accurately distinguish sample quality across different states and noise levels. Using the ADM encoder as a frozen backbone is therefore a key component of the architecture.

Table B.12: Impact of the ADM encoder on the CFG experiment on CIFAR-10. ✓: ADM encoder used; ✗: policy conditioned on raw x only.

ADM	FID ↓	Prec ↑	Rec ↑
✗	3.04	82.6	67.5
✓	2.05	80.7	71.4

C Broader Impact

This work proposes a framework for learning sampling policies for diffusion models via inverse reinforcement learning, with the goal of improving image quality and controllability without retraining the underlying generative model. Our method introduces a learned policy and a discriminator trained jointly with it, which adds a modest computational overhead at training time. We believe this cost is justified by the elimination of manual hyperparameter search, and we have documented it empirically in Table 1. We do not foresee direct societal harms specific to this work beyond those already associated with diffusion-based image generation more broadly.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the three contributions: (1) an IRL framework for learning diffusion sampling policies via state occupancy matching, (2) a policy gradient theorem enabling practical optimization, and (3) empirical validation on CIFAR-10, FFHQ, and ImageNet across three sampling strategies. All claims are supported by experiments in Section 5.2.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations are discussed in the Conclusion: sensitivity to initialization, dependence on the design of the discrete action space, and the restriction to discrete actions (continuous action spaces are left for future work).

   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: Theorem 4.1 (policy gradient for f-divergence minimization) is stated with explicit assumptions and proved in full in Appendix A.1. All intermediate results are numbered and cross-referenced.

   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: Full training details (architecture, optimizer, hyperparameters, action spaces, training schedules) are provided in Appendix B. All experiments use publicly available pretrained EDM models and standard datasets.

   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: Code is provided as an anonymized zip file in the supplemental material. All datasets (CIFAR-10, FFHQ, ImageNet) and pretrained models (EDM) are publicly available.

   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: Full experimental details are provided in Appendix B, including policy architectures, optimizers, learning rates, action spaces, discriminator training, and evaluation protocols for each strategy and dataset.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: The main table reports point estimates for readability, while Appendix B.4 reports mean$\pm$std values when repeated CIFAR-10 and FFHQ evaluations are available (Tables B.7 and B.8). ImageNet results are single-run evaluations due to the high computational cost of 50k-sample metrics at that scale.

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: Table 1 reports training and inference costs in EFLOPs for all strategies and datasets. GPU type, memory, and wall-clock times are reported in Appendix B.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The paper uses publicly available datasets and pretrained models. No human subjects are involved. The work does not introduce new risks beyond those already present in existing generative models.

   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: [N/A]

Justification: This is foundational research on improving sampling efficiency and quality of diffusion models. It does not introduce new application domains or deployment-ready systems. Improving generative model quality could in principle facilitate misuse (e.g., deepfakes), but this concern applies broadly to all work on generative models and is not specific to this contribution.

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 new pretrained models or datasets. It builds on existing publicly available EDM checkpoints and standard benchmarks (CIFAR-10, FFHQ, ImageNet).

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: The EDM pretrained models [22] are used under their MIT license. CIFAR-10, FFHQ, and ImageNet are standard benchmarks cited appropriately. The ADM encoder [8] is used under its MIT license.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

Answer: [N/A]

Justification: The paper does not introduce new datasets or model checkpoints. The method (code) will be released upon acceptance.

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 or 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 human subjects; no IRB approval is required.

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: LLMs were used solely for writing and reformulation assistance. They were not involved in the core methodology, experimental design, or scientific analysis, and therefore declaration is not required per NeurIPS guidelines.

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.