Optimizing DDPM Sampling with Shortcut Fine-Tuning

Ying Fan [1] **Kangwook Lee** [1]

Abstract

In this study, we propose *Shortcut Fine-Tuning (SFT)*, a new approach for addressing the challenge of fast sampling of pretrained Denoising Diffusion Probabilistic Models (DDPMs). SFT advocates for the fine-tuning of DDPM samplers through the direct minimization of Integral Probability Metrics (IPM), instead of learning the backward diffusion process. This enables samplers to discover an alternative and more efficient sampling shortcut, deviating from the backward diffusion process. Inspired by a control perspective, we propose a new algorithm **SFT- PG**: **S**hortcut **F**ine-**T**uning with **P**olicy **G**radient, and prove that under certain assumptions, gradient descent of diffusion models with respect to IPM is equivalent to performing policy gradient. To our best knowledge, this is the first attempt to utilize reinforcement learning (RL) methods to train diffusion models. Through empirical evaluation, we demonstrate that our fine-tuning method can further enhance existing fast DDPM samplers, resulting in sample quality comparable to or even surpassing that of the full-step model across various datasets.

1. Introduction

Denoising diffusion probabilistic models (DDPMs) (Ho et al., 2020) are parameterized stochastic Markov chains with Gaussian noises, which are learned by gradually adding noises to the data as the forward process, computing the posterior as a backward process, and then training the DDPM to match the backward process. Advances in DDPM (Nichol and Dhariwal, 2021; Dhariwal and Nichol, 2021) have shown the potential to rival GANs (Goodfellow et al., 2014) in generative tasks. However, one major drawback of DDPM is that a large number of steps T is needed. As a result, there is a line of work focusing on sampling fewer $T' \ll T$ steps to obtain comparable sample quality: Most works are dedicated to better approximating the backward process as stochastic differential equations (SDEs) with fewer steps, generally via better noise estimation or computing better sub-sampling schedules (Kong and Ping, 2021; San-Roman et al., 2021; Lam et al., 2021; Watson et al., 2021a; Jolicoeur-Martineau et al., 2021; Bao et al., 2021; 2022). Other works aim at approximating the backward process with fewer steps via more complicated non-gaussian noise distributions (Xiao et al., 2021).[1]

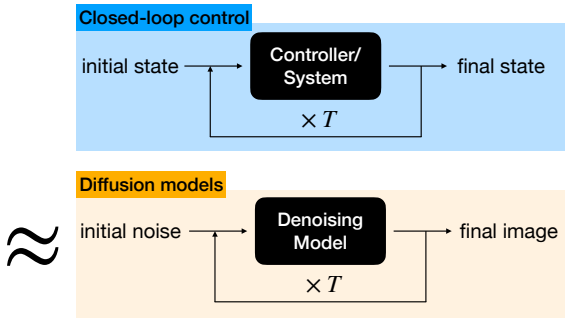


Figure 1. Image denoising is similar to a closed-loop control system: finding paths from pure noise to natural images.

To our best knowledge, existing fast samplers of DDPM stick to imitating the computed backward process with fewer steps. If we treat data generation as a control task (see Fig. 1), the backward process can be viewed as a demonstration to generate data from noise (which might not be optimal in terms of number of steps), and the training dataset could be an environment that provides feedback on how good the generated distribution is. From this view, imitating the backward process could be viewed as imitation learning (Hussein et al., 2017) or behavior cloning (Torabi et al., 2018). Naturally, one may wonder if we can do better than pure imitation, since learning via imitation is generally useful but rarely optimal, and we can explore alternative paths for optimal solutions during online optimization.

Motivated by the above observation, we study the following

[1]UW Madison. Correspondence to: Ying Fan, Kangwook Lee <yfan87@wisc.edu, kangwook.lee@wisc.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]There is another line of work focusing on fast sampling of DDIM (Song et al., 2020a) with deterministic Markov sampling chains, which we will discuss in Section 5.

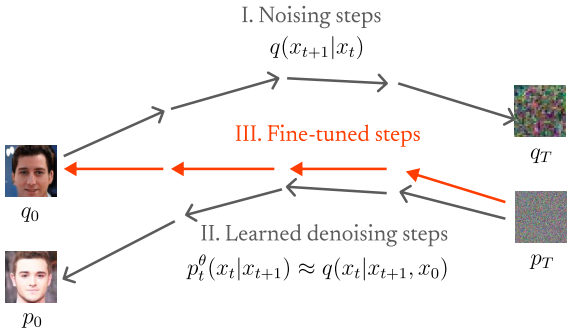


Figure 2. A visual illustration of the key idea of Shortcut Fine-Tuning (SFT). DDPMs aim at learning the backward diffusion model, but this approach is limited to a small number of steps. We propose the idea of *not* following the backward process and exploring other unexplored paths that can lead to improved data generation. To this end, we directly minimize an IPM and develop a policy gradient-like optimization algorithm. Our experimental results show that one can significantly improve data generation quality by fine-tuning a pretrained DDPM model with SFT. We also provide a visualization of the difference between steps in II and III when T is small in Appendix A.

underexplored question:

*Can we improve DDPM sampling by **not** following the backward process?*

In this work, we show that this is indeed possible. We fine-tune pretrained DDPM samplers by directly minimizing an integral probability metric (IPM) and show that finetuned DDPM samplers have significantly better generation qualities when the number of sampling steps is small. In this way, we can still enjoy diffusion models' multistep capabilities with no need to change the noise distribution, and improve the performance with fewer sampling steps.

More concretely, we first show that performing gradient descent of the DDPM sampler w.r.t. the IPM is equivalent to stochastic policy gradient, which echoes the aforementioned RL view but with a changing reward from the optimal critic function given by IPM. In addition, we present a surrogate function that can provide insights for monotonic improvements. Finally, we present a fine-tuning algorithm with alternative updates between the critic and the generator.

We summarize our main contributions as follows:

- (Section 4.1) We propose a novel algorithm to fine-tune DDPM samplers with direct IPM minimization, and we show that performing gradient descent of diffusion models w.r.t. IPM is equivalent to policy gradient. To our best knowledge, this is the first work to apply reinforcement learning methods to diffusion models.
- (Section 4.2) We present a surrogate function of IPM in theory, which provides insights on conditions for monotonic improvement and algorithm design.
- (Section 4.3.2) We propose a regularization for the critic based on the baseline function, which shows benefits for the policy gradient training.
- (Section 6) Empirically, we show that our fine-tuning can improve DDPM sampling performance in two cases: when T itself is small, and when T is large but using a fast sampler where $T' \ll T$. In both cases, our fine-tuning achieves comparable or even higher sample quality than the DDPM with 1000 steps using 10 sampling steps.

2. Background

2.1. Denoising Diffusion Probabilistic Models (DDPM)

Here we consider denoising probabilistic diffusion models (DDPM) as stochastic Markov chains with Gaussian noises (Ho et al., 2020). Consider data distribution $x_0 \sim q_0, x_0 \in \mathbb{R}^n$.

Define the forward noising process: for $t \in [0, .., T-1]$,

$$q(x_{t+1}|x_t) := \mathcal{N}(\sqrt{1-\beta_{t+1}}x_t, \beta_{t+1}I), \tag{1}$$

where $x_1, .., x_T$ are variables of the same dimensionality as x_0, $\beta_{1:T}$ is the variance schedule.

We can compute the posterior as a backward process:

$$q(x_t|x_{t+1}, x_0) = \mathcal{N}(\tilde{\mu}_{t+1}(x_{t+1}, x_0), \tilde{\beta}_{t+1}I), \tag{2}$$

where $\tilde{\mu}_{t+1}(x_{t+1}, x_0) = \frac{\sqrt{\bar{\alpha}_t}\beta_t}{1-\bar{\alpha}_{t+1}}x_0 + \frac{\sqrt{\alpha_{t+1}}(1-\bar{\alpha}_t)}{1-\bar{\alpha}_{t+1}}x_{t+1}$, $\alpha_{t+1} = 1-\beta_{t+1}$, $\bar{\alpha}_{t+1} = \prod_{s=1}^{t+1}\alpha_s$.

We define a DDPM sampler parameterized by θ, which generates data starting from some pure noise $x_T \sim p_T$:

$$\begin{aligned} x_T &\sim p_T = \mathcal{N}(0, I), \\ x_t &\sim p_t^\theta(x_t|x_{t+1}), \\ p_t^\theta(x_t|x_{t+1}) &:= \mathcal{N}\big(\mu_{t+1}^\theta(x_{t+1}), \Sigma_{t+1}\big), \end{aligned} \tag{3}$$

where Σ_{t+1} is generally chosen as $\beta_{t+1}I$ or $\tilde{\beta}_{t+1}I$. [2]

Define

$$p_{x_{0:T}}^\theta := p_T(x_T)\prod_{t=0}^{T-1} p_t^\theta(x_t|x_{t+1}), \tag{4}$$

and we have the marginal distribution $p_0^\theta(x_0) = \int p_{x_{0:T}}^\theta(x_{0:T})dx_{1:T}$.

[2] In this work we consider a DDPM sampler with a fixed variance schedule $\beta_{1:T}$ as in Ho et al. (2020), but it could also be learned as in Nichol and Dhariwal (2021).

The sampler is trained by minimizing the sum of KL divergences for each step:

$$J = \mathbb{E}_q \left[\sum_{t=0}^{T-1} D_{KL}(q(x_t|x_{t+1}, x_0), p_t^\theta(x_t|x_{t+1})) \right]. \quad (5)$$

Optimizing the above loss can be viewed as matching the conditional generator $p_t^\theta(x_t|x_{t+1})$ with the backward process $q(x_t|x_{t+1}, x_0)$ for each step. Song et al. (2020b) show that J is equivalent to score-matching loss when formulating the forward and backward process as a discrete version of stochastic differential equations.

2.2. Integral Probability Metrics (IPM)

Given $\mathcal{A}$ as a set of parameters s.t. for each $\alpha \in \mathcal{A}$, it defines a critic $f_\alpha : \mathbb{R}^n \to \mathbb{R}$. Given a critic f_α and two distributions p_0^θ and q_0, we define

$$g(p_0^\theta, f_\alpha, q_0) := \mathop{\mathbb{E}}_{x_0 \sim p_0^\theta}[f_\alpha(x_0)] - \mathop{\mathbb{E}}_{x_0 \sim q_0}[f_\alpha(x_0)]. \quad (6)$$

Let

$$\Phi(p_0^\theta, q_0) := \sup_{\alpha \in \mathcal{A}} g(p_0^\theta, f_\alpha, q_0). \quad (7)$$

If $\mathcal{A}$ satisfies that $\forall \alpha \in \mathcal{A}$, $\exists \alpha' \in \mathcal{A}$, s.t. $f_{\alpha'} = -f_\alpha$, then $\Phi(p_\theta, q)$ is a pseudo metric over the probability space of $\mathbb{R}^n$, making it so-called integral probability metrics (IPM).

In this paper, we consider $\mathcal{A}$ that makes $\Phi(p_0^\theta, q_0)$ an IPM. For example, when $\mathcal{A} = \{\alpha : ||f_\alpha||_L \leqslant 1\}$, $\Phi(p_0^\theta, q_0)$ is the Wasserstein-1 distance; when $\mathcal{A} = \{\alpha : ||f_\alpha||_\infty \leqslant 1\}$, $\Phi(p_0^\theta, q_0)$ is the total variation distance; it also includes maximum mean discrepancy (MMD) when $\mathcal{A}$ defines all functions in Reproducing Kernel Hilbert Space (RKHS).

3. Motivation

3.1. Issues with Existing DDPM Samplers

Here we review the existing issues with DDPM samplers 1) when T is not large enough, and 2) when sub-sampling with the number of steps $T' \ll T$, which inspires us to design our fine-tuning algorithm.

Case 1. Issues caused by training DDPM with a small T (Fig 2). Given a score-matching loss J, the upper bound on Wasserstein-2 distance is given by Kwon et al. (2022):

$$W_2(p_0^\theta, q_0) \leqslant \mathcal{O}(\sqrt{J}) + I(T) W_2(p_T, q_T), \quad (8)$$

where $I(T)$ is non-exploding and $W_2(p_T, q_T)$ decays exponentially with T when $T \to \infty$. From the inequality above, one sufficient condition for the score-matching loss J to be viewed as optimizing the Wasserstein distance is when T is large enough such that $I(T)W_2(p_T, q_T) \to 0$.

Now we consider the case when T is small and $p_T \not\approx q_T$.[3]. The upper bound in Eq. (8) can be high since $W_2(p_T, q_T)$ is not neglectable. As shown in Fig 2, pure imitation $p_t^\theta(x_t|x_{t+1}) \approx q(x_t|x_{t+1}, x_0)$ would not lead the model exactly to q_0 when p_T and q_T are not close enough.

Case 2. Issues caused by a smaller number of sub-sampling steps ($T' \ll T$) (Fig 8 in Appendix B). We consider DDPM sub-sampling and other fast sampling techniques, where T is large enough s.t. $p_T \approx q_T$, but we try to sample with fewer sampling steps (T'). It is generally done by choosing τ to be an increasing sub-sequence of T' steps in $[0, T]$ starting from 0. Many works have been dedicated to finding a subsequence and variance schedule to make the sub-sampling steps match the full-step backward process as much as possible (Kong and Ping, 2021; Bao et al., 2021; 2022). However, this would inevitably cause downgraded sample quality if each step is Gaussian: as discussed in Salimans and Ho (2021) and Xiao et al. (2021), a multi-step Gaussian sampler cannot be distilled into a one-step Gaussian sampler without loss of fidelity.

3.2. Problem Formulation

In both cases mentioned above, there might exist paths other than imitating the backward process that can reach the data distribution with fewer Gaussian steps. Thus one may expect to overcome these issues by minimizing the IPM.

Here we present the formulation of our problem setting. We assume that there is a target data distribution q_0. Given a set of critic parameters $\mathcal{A}$ s.t. $\Phi(p_0^\theta, q_0) = \sup_{\alpha \in \mathcal{A}} g(p_0^\theta, f_\alpha, q_0)$ is an IPM, and given a DDPM sampler with T steps parameterized by θ, our goal is to solve:

$$\min_\theta \Phi(p_0^\theta, q_0). \quad (9)$$

3.3. Pathwise Derivative Estimation for Shortcut Fine-Tuning: Properties and Potential Issues

One straightforward approach is to optimize $\Phi(p_0^\theta, q_0)$ using pathwise derivative estimation (Rezende et al., 2014) like GAN training, which we denote as **SFT** (shortcut fine-tuning). We can recursively define the stochastic mappings:

$$h_{\theta,T}(x_T) := x_T, \quad (10)$$

$$h_{\theta,t}(x_t) := \mu_\theta(h_{\theta,t+1}(x_{t+1})) + \epsilon_{t+1}, \quad (11)$$

$$x_0 = h_{\theta,0}(x_T) \quad (12)$$

where $x_T \sim \mathcal{N}(0, I), \epsilon_{t+1} \sim \mathcal{N}(0, \Sigma_{t+1}), t = 0, ..., T-1$.

[3] Recall that during the diffusion process, we need small Gaussian noise for each step set the sampling chain to also be conditional Gaussian (Ho et al., 2020). As a result, a small T means q_T is not close to pure Gaussian, and thus $p_T \not\approx q_T$.

Then we can write the objective function as:

$$\Phi(p_0^\theta, q_0) = \sup_{\alpha \in \mathcal{A}} \mathop{\mathbb{E}}_{x_T, \epsilon_{1:T}} [f_\alpha(h_{\theta,0}(x_T))] - \mathop{\mathbb{E}}_{x_0 \sim q_0} [f_\alpha(x_0)] \tag{13}$$

Assume that $\exists \alpha \in \mathcal{A}$, s.t. $g(p_0^\theta, \alpha, q_0) = \Phi(p_0^\theta, q_0)$. Let $\alpha^*(p_0^\theta, q_0) \in \{\alpha : g(p_0^\theta, \alpha, q_0) = \Phi(p_0^\theta, q_0)\}$. When f_α is 1-Lipschitz, we can compute the gradient which is similar to WGAN (Arjovsky et al., 2017):

$$\nabla_\theta \Phi(p_0^\theta, q_0) = \mathop{\mathbb{E}}_{x_T, \epsilon_{1:T}} \left[\nabla_\theta f_{\alpha^*(p_0^\theta, q_0)}(h_{\theta,0}(x_T)) \right]. \tag{14}$$

Implicit requirements on the family of critics $\mathcal{A}$: gradient regularization. In Eq. (14), we can observe that the critic f_{α^*} needs to provide meaningful gradients (w.r.t. the input) for the generator. If the gradient of the critic happens to be 0 at some generated data points, even if the critic's value could still make sense, the critic would provide no signal for the generator on these points[4]. Thus GANs trained with IPMs generally need to choose $\mathcal{A}$ such that the gradient of the critic is regularized: For example, Lipschitz constraints like weight clipping (Arjovsky et al., 2017) and gradient penalty (Gulrajani et al., 2017) for WGAN, and gradient regularizers for MMD GAN (Arbel et al., 2018).

Potential issues. Besides the implicit requirements on the critic, there might also be issues when computing Eq. (14) in practice. Differentiating a composite function with T steps can cause problems similar to RNNs: Gradient vanishing may result in long-distance dependency being lost; Gradient explosion may occur; Memory usage is high.

4. Method: Shortcut Fine-Tuning with Policy Gradient (SFT-PG)

We note that Eq. (14) is not the only way to estimate the gradient w.r.t. IPM. In this section, we show that performing gradient descent of $\Phi(p_0^\theta, q_0)$ can be equivalent to policy gradient (Section 4.1), provide analysis towards monotonic improvement (Section 4.2) and then present the algorithm design (Section 4.3).

4.1. Policy Gradient Equivalence

By modeling the conditional probability through the trajectory, we provide an alternative way for gradient estimation which is equivalent to policy gradient, without differentiating through the composite functions.

Theorem 4.1. ***(Policy gradient equivalence)*** *Assume that both $p^\theta_{x_{0:T}}(x_{0:T}) f_{\alpha^*(p_0^\theta, q_0)}(x_0)$ and $\nabla_\theta p^\theta_{x_{0:T}}(x_{0:T}) f_{\alpha^*(p_0^\theta, q_0)}(x_0)$ are continuous functions w.r.t. θ and $x_{0:T}$. Then*

$$\nabla_\theta \Phi(p_0^\theta, q_0) = \mathop{\mathbb{E}}_{p^\theta_{x_{0:T}}} \left[f_{\alpha^*(p_0^\theta, q_0)}(x_0) \nabla_\theta \log \sum_{t=0}^{T-1} p_t^\theta(x_t | x_{t+1}) \right]. \tag{15}$$

Proof.

$$\begin{aligned}
&\nabla_\theta \Phi(p_0^\theta, q_0) \\
&= \nabla_\theta \int p_0^\theta(x_0) f_{\alpha^*(p_0^\theta, q_0)}(x_0) dx_0 \\
&\quad + \nabla_\theta \alpha^*(p_0^\theta, q_0) \nabla_{\alpha^*(p_0^\theta, q_0)} \int p_0^\theta(x_0) f_{\alpha^*(p_0^\theta, q_0)}(x_0) dx_0,
\end{aligned} \tag{16}$$

where $\nabla_{\alpha^*(p_0^\theta, q_0)} \int p_0^\theta(x_0) f_{\alpha^*(p_0^\theta, q_0)}(x_0) dx_0$ is 0 from the envelope theorem. Then we have

$$\begin{aligned}
&\nabla_\theta \int p_0^\theta(x_0) f_{\alpha^*(p_0^\theta, q_0)}(x_0) dx_0 \\
&= \nabla_\theta \int \left(\int p^\theta_{x_{0:T}}(x_{0:T}) dx_{1:T} \right) f_{\alpha^*(p_0^\theta, q_0)}(x_0) dx_0, \\
&= \nabla_\theta \int p^\theta_{x_{0:T}}(x_{0:T}) f_{\alpha^*(p_0^\theta, q_0)}(x_0) dx_{0:T} \\
&= \int p^\theta_{x_{0:T}}(x_{0:T}) f_{\alpha^*(p_0^\theta, q_0)}(x_0) \nabla_\theta \log p^\theta_{x_{0:T}}(x_{0:T}) dx_{0:T} \\
&= \mathop{\mathbb{E}}_{p^\theta_{x_{0:T}}} \left[f_{\alpha^*(p_0^\theta, q_0)}(x_0) \sum_{t=0}^{T-1} \nabla_\theta \log p_t^\theta(x_t | x_{t+1}) \right],
\end{aligned} \tag{17}$$

where the second last equality is from the continuous assumptions to exchange integral and derivative and the log derivative trick. The proof is then complete. □

MDP construction for policy gradient equivalence. Here we explain why Eq. (15) could be viewed as policy gradient. We can construct an MDP with a finite horizon T: Treat $p_t^\theta(x_t | x_{t+1})$ as a policy, and assume that transition is an identical mapping such that the action is to choose the next state. Consider reward as $f_{\alpha^*(p_0^\theta, q_0)}(x_0)$ at the final step, and as 0 at any other steps. Then Eq. (15) is equivalent to performing policy gradient (Williams, 1992).

Comparing Eq. (14) and Eq. (15):

- Eq. (14) uses the gradient of the critic, while Eq. (15) only uses the value of the critic. This indicates that for policy gradient, weaker conditions are required for critics to provide meaningful guidance for the generator, which means more choices of $\mathcal{A}$ can be applied here.
- We compute the sum of gradients for each step in Eq. (15), which does not suffer from exploding or vanishing gradients. Also, we do not need to track gradients of the generated sequence during T steps.

[4] For example, MMD with very narrow kernels can produce such critic functions, where each data point defines the center of the corresponding kernel which yields gradient 0.

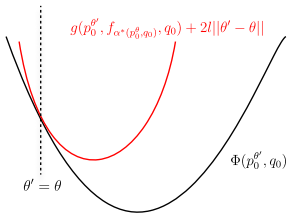


Figure 3. Illustration of the surrogate function given a fixed critic (red), and the actual objective $\Phi(p_0^{\theta'}, q_0)$ (dark). The horizontal axis represents the variable θ'. Starting from θ, a descent in the surrogate function is a sufficient condition for a descent in $\Phi(p_0^{\theta'}, q_0)$.

- However, stochastic policy gradient methods usually suffer from higher variance (Mohamed et al., 2020). Thanks to similar techniques in RL, we can reduce the variance via a baseline trick, which will be discussed in Section 4.3.1.

In conclusion, Eq. (15) is comparable to Eq. (14) in expectation, with potential benefits like numerical stability, memory efficiency, and a wider range of the critic family $\mathcal{A}$. It could suffer from higher variance but the baseline trick can help. We denote such kind of method as **SFT-PG** (shortcut fine-tuning with policy gradient).

Empirical comparison. We conduct experiments on some toy datasets (Fig 4), where we show the performance of Eq. (15) with the baseline trick is at least comparable to Eq. (14) at convergence when they use the same gradient penalty (GP) for critic regularization. We further observe SFT-PG with a newly proposed baseline regularization (B) enjoys a noticeably better performance compared to SFT with GP. The regularization methods will be introduced in Section 4.3.2. Experimental details are in Section 6.2.2.

4.2. Towards Monotonic Improvement

The gradient update discussed in Eq. (15) only supports one step of gradient update, given a fixed critic $f_{\alpha^*(p_0^\theta,q_0)}$ that is optimal to the current θ. Questions remain: When is our update guaranteed to get improvement? Can we do more than one update to get a potential descent? We answer the questions by providing a surrogate function of the IPM.

Theorem 4.2. ***(The surrogate function of IPM)***

Assume that $g(p_0^\theta, f_\alpha, q_0)$ is Lipschitz w.r.t. θ, given q_0 and $\alpha \in \mathcal{A}$. Given a fixed critic $f_{\alpha^*(p_0^\theta,q_0)}$, there exists $l \geqslant 0$ such that $\Phi(p_0^{\theta'}, q_0)$ is upper bounded by the surrogate function below:

$$\Phi(p_0^{\theta'}, q_0) \leqslant g(p_0^{\theta'}, f_{\alpha^*(p_0^\theta,q_0)}, q_0) + 2l||\theta' - \theta||. \quad (18)$$

Proof of Theorem 4.2 can be found in Appendix C. Here we provide an illustration of Theorem 4.2 in Fig 3. Given a critic that is optimal w.r.t. θ, $\Phi(p_0^{\theta'}, q_0)$ is unknown if $\theta \neq \theta'$. But if we can get a descent of the surrogate function, we are also guaranteed to get a descent of $\Phi(p_0^{\theta'}, q_0)$, which facilitates more potential updates even if $\theta' \neq \theta$. Moreover, using the Lagrange multiplier, we can convert minimizing the surrogate function to a constrained optimization problem to optimize $g(p_0^{\theta'}, f_{\alpha^*(p_0^\theta,q_0)}, q_0)$ with the constraint that $||\theta' - \theta|| \leqslant \delta$ for some $\delta > 0$. Following this idea, one simple trick is to perform $n_{\text{generator}}$ steps of gradient updates with a small learning rate, and clip the gradient norm with threshold γ. We present the empirical effect of such simple modification in Section 6.2.3, Table 2.

Discussion. One may notice that Theorem 4.2 is similar in spirit to Theorem 1 in TRPO (Schulman et al., 2015a), which provides a surrogate function for a fixed but unknown reward function. In our case, the reward function $f_{\alpha^*(p_0^\theta,q_0)}$ is known for the current θ but changing: It is dependent on the current θ so it remains unknown for $\theta' \neq \theta$. The proof techniques are also different, but they both estimate an unknown part of the objective function.

4.3. Algorithm Design

In the previous sections, we only consider the case where we have an optimal critic function given θ. In the training, we adopt similar techniques in WGAN (Arjovsky et al., 2017) to perform alternative training of the critic and generator in order to approximate the optimal critic. Consider the objective function below:

$$\min_\theta \max_{\alpha \in \mathcal{A}} g(p_0^\theta, f_\alpha, q_0). \quad (19)$$

Now we discuss techniques to reduce the variance of the gradient estimation and regularize the critic, and then give an overview of our algorithm.

4.3.1. Baseline Function for Variance Reduction

Given a critic α, we can adopt a technique widely used in policy gradient to reduce the variance of the gradient estimation in Eq. (15). Similar to Schulman et al. (2015b), we can subtract a baseline function $V_{t+1}^\omega(x_{t+1})$ from the cumulative reward $f_\alpha(x_0)$, without changing the expectation:

$$\begin{aligned}
&\nabla_\theta g(p_0^\theta, f_\alpha, q_0) \\
&= \mathop{\mathbb{E}}_{p_{x_{0:T}}^\theta}\left[f_\alpha(x_0)\sum_{t=0}^{T-1}\nabla_\theta \log p_t^\theta(x_t|x_{t+1})\right] \\
&= \mathop{\mathbb{E}}_{p_{x_{0:T}}^\theta}\left[\sum_{t=0}^{T-1}(f_\alpha(x_0) - V_{t+1}^\omega(x_{t+1}))\nabla_\theta \log p_t^\theta(x_t|x_{t+1})\right],
\end{aligned} \quad (20)$$

where the optimal choice of $V^{\omega}_{t+1}(x_{t+1})$ to minimize the variance would be $V_{t+1}(x_{t+1}, \alpha) := \mathop{\mathbb{E}}_{p^{\theta}_{x_{0:T}}}[f_\alpha(x_0)|x_{t+1}]$. Detailed derivation of Eq (20) can be found in Appendix D. Thus, given a critic α and a generator θ, we can train a value function V^{ω}_{t+1} by minimizing the objective below:

$$R_B(\alpha, \omega, \theta) = \mathop{\mathbb{E}}_{p^{\theta}_{x_{0:T}}}\left[\sum_{t=0}^{T-1}(V^{\omega}_{t+1}(x_{t+1}) - V_{t+1}(x_{t+1}, \alpha))^2\right]. \tag{21}$$

4.3.2. Choices of $\mathcal{A}$: Regularizing the Critic

Here we discuss different choices of $\mathcal{A}$, which indicates different regularization methods for the critic.

Lipschitz regularization. If we choose $\mathcal{A}$ to include parameters of all 1-Lipschitz functions, we can adopt regularization as WGAN-GP (Gulrajani et al., 2017):

$$R_{GP}(\alpha, \theta) = \mathop{\mathbb{E}}_{\hat{x_0}}\left[(||\nabla_{x_0} f_\alpha(x_0)|| - 1)^2\right], \tag{22}$$

where $\hat{x_0}$ is sampled uniformly on the line segment between $x_0' \sim p_0^{\theta}$ and $x_0'' \sim q_0$. f_α can be trained to maximize $g(p_0^{\theta}, f_\alpha, q_0) - \eta R_{GP}(\alpha, \omega, \theta)$, $\eta > 0$ is the regularization coefficient.

Reusing baseline for critic regularization. As discussed in Section 4.1, since we only use the critic value during updates, now we can afford a potentially wider range of critic family $\mathcal{A}$. Some regularization on f_α is still needed; Otherwise its value can explode. Also, regularization is shown to be beneficial for local convergence (Mescheder et al., 2018). So we consider regularization that can be weaker than gradient constraints, such that the critic is more sensitive to the changes of the generator, which could be favorable when updating the critic for a fixed number of training steps.

We found an interesting fact that the loss $R_B(\alpha, \omega, \theta)$ can be *reused* to regularize the value of f_α instead of the gradient, which implicitly defines a set $\mathcal{A}$ that shows empirical benefits in practice.

Define

$$L(\alpha, \omega, \theta) := g(p_0^{\theta}, f_\alpha, q_0) - \lambda R_B(\alpha, \omega, \theta). \tag{23}$$

Given θ, our critic α and baseline ω can be trained together to maximize $L(\alpha, \omega, \theta)$.

We provide an explanation of such kind of implicit regularization. During the update, we can view V^{ω}_{t+1} as an approximation of the expected value of f_α from the previous step. The regularization provides a trade-off between maximizing $g(p_0^{\theta}, f_\alpha, q_0)$ and minimizing changes in the expected value of f_α, preventing drastic changes in the critic and stabilizing the training. Intuitively, it helps local convergence when both the critic and generator are already near-optimal: there is an extra cost for the critic value to diverge away from the optimal value. As a byproduct, it also makes the baseline function easier to fit since the regularization loss is reused.

Empirical comparison: baseline regularization and gradient penalty. We present a comparison of gradient penalty (GP) and baseline regularization (B) for policy gradient training (SFT-PG) in Section 6.2.2, Fig 4 on toy datasets, which shows in policy gradient training, the baseline function performs comparably well or even better than gradient penalty.

4.3.3. Putting Together: Algorithm Overview

Now we are ready to present our algorithm. Our critic α and baseline ω are trained to maximize $L(\alpha, \omega, \theta) = g(p_0^{\theta}, f_\alpha, q_0) - \lambda R_B(\alpha, \omega, \theta)$, and the generator is trained to minimize $g(p_0^{\theta}, f_\alpha, q_0)$ via Eq. (20). To save memory usage, we use a buffer $\mathcal{B}$ that contains $\{x_{t+1}, x_t, x_0, t\}$ generated from the current generator without tracking the gradient, and randomly sample a batch from the buffer to compute Eq. (20) and then perform backpropagation. The maximization and minimization steps are performed alternatively. See details in Alg 1.

Algorithm 1 Shortcut Fine-Tuning with Policy Gradient and Baseline Regularization: SFT-PG (B)

Input: n_{critic}, $n_{\text{generator}}$, batch size m, critic parameters α, baseline function parameter ω , pretrained generator θ, regularization hyperparameter λ

while θ not converged **do**
  Initialize trajectory buffer $\mathcal{B}$ as $\varnothing$
  for i = 0,...,n_{critic} **do**
    Obtain m i.i.d. samples from $p^{\theta}_{x_{0:T}}$
    Add all $\{x_{t+1}, x_t, x_0, t\}$ to $\mathcal{B}$, $t = 0, ..., T-1$
    Obtain m i.i.d. samples from q_0
    Update α and ω via maximizing Eq. (23)
  end for
  for j = 0,...,$n_{\text{generator}}$ **do**
    Obtain m samples of $\{x_{t+1}, x_t, x_0, t\}$ from $\mathcal{B}$
    Update θ via policy gradient according to Eq. (20)
  end for
end while

5. Related Works

GAN and RL. There are works using ideas from RL to train GANs (Yu et al., 2017; Wang et al., 2017; Sarmad et al., 2019; Bai et al., 2019). The most relevant work is SeqGAN (Yu et al., 2017), which uses policy gradient to train the generator network. There are several main differences between their settings and ours. First, different GAN objec-

tives are used: SeqGAN uses the JS divergence while we use IPM. In SeqGAN, the next token is dependent on tokens generated from all previous steps, while in diffusion models the next image is only dependent on the model output from one previous step; Also, the critic takes the whole generated sequence as input in SeqGAN, while we only care about the final output. Besides, in our work, rewards are mathematically derived from performing gradient descent w.r.t. IPM, while in SeqGAN, rewards are designed manually. In conclusion, different from SeqGAN, we propose a new policy gradient algorithm to optimize the IPM objective, with a novel analysis of monotonic improvement conditions and a new regularization method for the critic.

Diffusion and GAN. There are other works combining diffusion and GAN training: Xiao et al. (2021) consider multi-modal noise distributions generated by GAN to enable fast sampling; Zheng et al. (2022) considers a truncated forward process by replacing the last steps in the forward process with an autoencoder to generate noise, and start with the learned autoencoder as the first step of denoising and then continue to generate data from the diffusion model; Diffusion GAN (Wang et al., 2022) perturbs the data with an adjustable number of steps, and minimizes JS divergence for all intermediate steps by training a multi-step generator with a time-dependent discriminator. To our best knowledge, there is no existing work using GAN-style training to fine-tune a pretrained DDPM sampler.

Fast samplers of DDIM and more. There is another line of work on fast sampling of DDIM (Song et al., 2020a), for example, knowledge distillation (Luhman and Luhman, 2021; Salimans and Ho, 2021) and solving ordinary differential equations (ODEs) with fewer steps (Liu et al., 2022; Lu et al., 2022). Samples generated by DDIM are generally less diverse than DDPM (Song et al., 2020a). Also, fast sampling is generally easier for DDIM samplers (with deterministic Markov chains) than DDPM samplers, since it is possible to combine multiple deterministic steps into one step without loss of fidelity, but not for combining multiple Gaussian steps as one (Salimans and Ho, 2021). Fine-tuning DDIM samplers with deterministic policy gradient for fast sampling also seems possible, but deterministic policies may suffer from suboptimality, especially in high-dimensional action space (Silver et al., 2014), though it might require fewer samples. Also, it becomes less necessary since distillation is already possible for DDIM.

Moreover, there is also some recent work that uses sample quality metrics to enable fast sampling. Instead of fine-tuning pretrained models, Watson et al. (2021b) propose to optimize the hyperparameters of the sampling schedule for a family of non-Markovian samplers by differentiating through KID (Bińkowski et al., 2018), which is calculated by pretrained inception features. It is followed by a contemporary work that fine-tunes pretrained DDIM models using MMD calculated by pretrained features (Aiello et al., 2023), which is similar to the method discussed in Section 3.3 but with a fixed critic and a deterministic sampling chain. Generally speaking, adversarially trained critics can provide stronger signals than fixed ones and are more helpful for training (Li et al., 2017). As a result, besides the potential issues discussed in Section 3.3, such training may also suffer from sub-optimal results when p_0^θ is not close enough to q_0 at initialization, and is highly dependent on the choice of the pretrained feature.

6. Experiments

In this section, we aim to answer the following questions:

- (Section 6.2.1) Does the proposed algorithm SFT-PG (B) work in practice?
- (Section 6.2.2) How does SFT-PG (Eq. (15)) work compared to SFT (Eq. (14)) with the same regularization (GP), and how does baseline regularization (B) compared to gradient penalty (GP) in SFT-PG?
- (Section 6.2.3) Do more generator steps with gradient clipping work as discussed in Section 4.2?
- (Section 6.3) Does the proposed fine-tuning SFT-PG (B) improve existing fast samplers of DDPM on benchmark datasets?

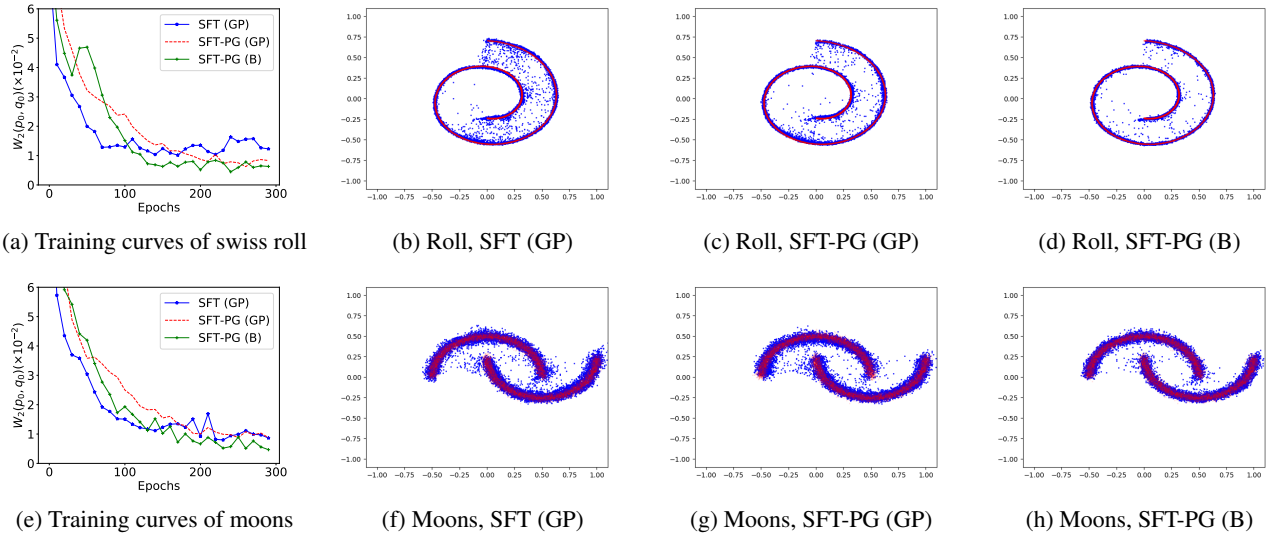
Code is available at `https://github.com/UW-Madison-Lee-Lab/SFT-PG`.

6.1. Setup

Here we provide the setup of our training algorithm on different datasets. Model architectures and training details can be found in Appendix F.

Toy datasets. The toy datasets we use are swiss roll and two moons (Pedregosa et al., 2011). We use $\lambda = 0.1$, $n_{\text{critic}} = 5, n_{\text{generator}} = 1$ with no gradient clipping. For evaluation, we use the Wasserstein-2 distance on 10K samples from p_0 and q_0 respectively, calculated by POT (Flamary et al., 2021).

Image datasets. We use MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky et al., 2009) and CelebA (Liu et al., 2015). For hyperparameters, we choose $\lambda = 1.0$, $n_{\text{critic}} = 5, n_{\text{generator}} = 10$, $\gamma = 0.1$, except when testing different choices of $n_{\text{generator}}$ and γ in MNIST, where we use $n_{\text{generator}} = 5$ and varying γ. For evaluation, we use FID (Heusel et al., 2017) measured by 50K samples generated from p_0^θ and q_0 respectively.



(a) Training curves of swiss roll (b) Roll, SFT (GP) (c) Roll, SFT-PG (GP) (d) Roll, SFT-PG (B)

(e) Training curves of moons (f) Moons, SFT (GP) (g) Moons, SFT-PG (GP) (h) Moons, SFT-PG (B)

Figure 4. Training curves (4a, 4e) and 10K randomly generated samples from SFT (GP) (4b, 4f), SFT-PG (GP) (4c, 4g), and SFT-PG (B) (4d, 4h) at convergence. In the visualizations, red dots indicate the ground truth data, and blue dots indicate generated data. We can observe that SFT-PG (B) produces noticeably better distributions, which is the result of utilizing a wider range of critics.

Method	$\mathbf{W_2}(\mathbf{p_0^\theta}, \mathbf{q_0})$ $(\times 10^{-2})$ $(\downarrow)$
$T = 10$, DDPM	8.29
$T = 100$, DDPM	2.36
$T = 1000$, DDPM	1.78
$T = 10$, SFT-PG (B)	**0.64**

Table 1. Comparison of DDPM models and our fine-tuned model on the swiss roll dataset.

6.2. Proof-of-concept Results

In this section, we fine-tune pretrained DDPMs with $T = 10$, and present the effect of the proposed algorithm SFT-PG with baseline regularization on toy datasets. We present the results of different gradient estimations discussed in Section 4.1, different critic regularization methods discussed in Section 4.3.2, and the training technique with more generator steps discussed Section 4.2.

6.2.1. Improvement from Fine-Tuning

On the swiss roll dataset, we first train a DDPM with $T = 10$ till convergence, and then use it as initialization of our fine-tuning. As in Table 1, our fine-tuned sampler with 10 steps can get better Wasserstein distance not only compared to the DDPM with $T = 10$, but can even outperform DDPM with $T = 1000$, which is reasonable since we directly optimize the IPM objective. [5] The training curve and the data visualization can be found in Fig 4a and Fig 4d.

6.2.2. Effect of Different Gradient Estimations and Regularizations

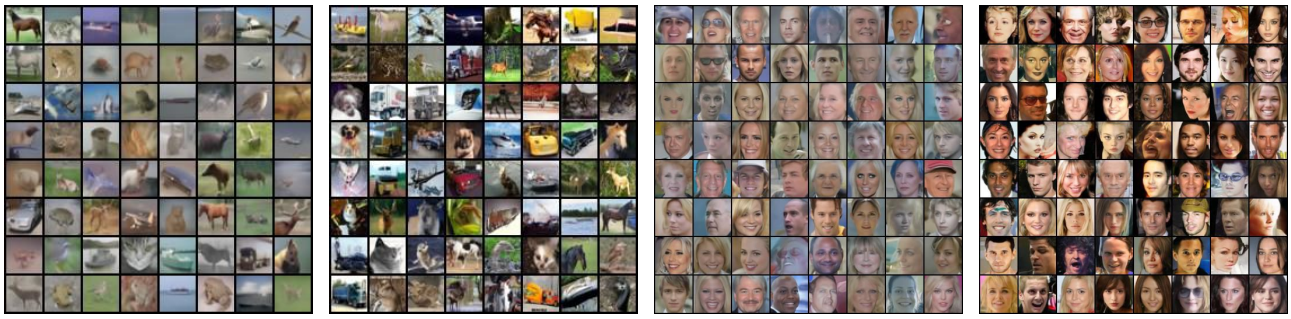
On the toy datasets, we compare gradient estimation SFT-PG and SFT, both with gradient penalty (GP). [6] We also compare them to our proposed algorithm SFT-PG (B). All methods are initialized with pretrained DDPM, $T = 10$, then trained till convergence. As shown in Fig 4, we can observe that all methods converge and the training curves are almost comparable, while SFT-PG (B) enjoys a slightly better final performance.

6.2.3. Effect of Gradient Clipping with More Generator Steps

In Section 4.2, we discussed that performing more generator steps with the same fixed critic and clipping the gradient norm can improve the training of our algorithm. Here we present the effect of $n_{\text{generator}} = 1$ or 5 with different gradient clipping thresholds γ on MNIST, initialized with a pretrained DDPM with $T = 10$, FID=7.34. From Table 2, we find that a small γ with more steps can improve the final performance, but could hurt the performance if too small. Randomly generated samples from the model with the best FID are in Fig 6. We also conducted similar experiments on the toy datasets, but we find no significant difference on the

[5] Besides, our algorithm also works when training from scratch with a final performance comparable to fine-tuning, but it will take longer time to train.

[6] For gradient penalty coefficient, we tested different choices in $[0.001, 10]$ and pick the best choice 0.001. We also tried spectral normalization for Lipschitz constraints, but we found that its performance is worse than gradient penalty on these datasets.



(a) CIFAR10, Initialization (b) CIFAR10, SFT-PG (B) (c) CelebA, Initialization (d) CelebA, SFT-PG (B)

Figure 5. Randomly generated images before and after fine-tuning, on CIFAR10 (32×32) and CelebA (64×64), $T' = 10$. The initialization is from pretrained models with $T = 1000$ and sub-sampling schedules with $T' = 10$ calculated from FastDPM (Kong and Ping, 2021).

final results, which is expected since the task is too simple.

Method	**FID** (↓)
1 step	1.35
5 steps, $\gamma = 10$	0.83
5 steps, $\gamma = 1.0$	**0.82**
5 steps, $\gamma = 0.1$	0.89
5 steps, $\gamma = 0.001$	1.46

Table 2. Effect of $n_{\text{generator}}$ and γ.

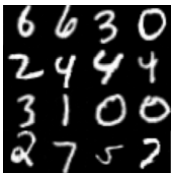


Figure 6. Generated samples.

6.3. Benchmark Results

To compare with existing fast samplers of DDPM, we take pretrained DDPMs with $T = 1000$ and fine-tune them with sampling steps $T' = 10$ on image benchmark datasets CIFAR-10 and CelebA.

Our baselines include various fast DDPM samplers with Gaussian noises: naive DDPM sub-sampling, FastDPM (Kong and Ping, 2021), and recently advanced samplers like Analytic DPM (Bao et al., 2021) and SN-DPM (Bao et al., 2022). For fine-tuning, we use the fixed variance and sub-sampling schedules computed by FastDPM with $T' = 10$ and only train the mean prediction model. From Table 3, we can observe that the performance of fine-tuning with $T' = 10$ is comparable to the pretrained model with $T = 1000$, outperforming the existing fast DDPM samplers. Randomly generated images before and after fine-tuning are in Fig 5.

We also present a comparison with DDIM sampling methods on CIFAR 10 benchmark in Appendix E, where our method is comparable to progressive distillation with $T' = 8$.

6.4. Discussions and Limitations

In our experiments, we only train μ_t^θ given a pretrained DDPM. It is also possible to learn the variance via fine-tuning with the same objective, and we leave it as future work. Although we do not need to track the gradients during all sampling steps, we still need to run T' inference steps to collect the sequence, which is inevitably slower than GAN.

Method	**CIFAR-10** (32×32)	**CelebA** (64×64)
DDPM	34.76	36.69
FastDPM	29.43	28.98
Analytic-DPM	22.94	28.99
SN-DDPM	16.33	20.60
SFT-PG (B)	**2.28**	**2.01**

Table 3. FID (↓) on CIFAR-10 and CelebA, $T' = 10$ for all methods. Our fine-tuning produces comparable results with the full-step pretrained models (FID = 3.03 for CIFAR-10, and FID = 3.26 for CelebA, $T = 1000$).

7. Conclusion

In this work, we fine-tune DDPM samplers to minimize the IPMs via policy gradient. We show performing gradient descent of stochastic Markov chains w.r.t. IPM is equivalent to policy gradient, and present a surrogate function of the IPM which sheds light on monotonic improvement conditions. Our fine-tuning improves the existing fast samplers of DDPM, achieving comparable or even higher sample quality than the full-step model on various datasets.

Acknowledgements

Support for this research was provided by the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation, and NSF Award DMS-2023239.

References

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, 2014.

Zhifeng Kong and Wei Ping. On fast sampling of diffusion probabilistic models. *arXiv preprint arXiv:2106.00132*, 2021.

Robin San-Roman, Eliya Nachmani, and Lior Wolf. Noise estimation for generative diffusion models. *arXiv preprint arXiv:2104.02600*, 2021.

Max WY Lam, Jun Wang, Rongjie Huang, Dan Su, and Dong Yu. Bilateral denoising diffusion models. *arXiv preprint arXiv:2108.11514*, 2021.

Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. Learning to efficiently sample from diffusion probabilistic models. *arXiv preprint arXiv:2106.03802*, 2021a.

Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021.

Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In *International Conference on Learning Representations*, 2021.

Fan Bao, Chongxuan Li, Jiacheng Sun, Jun Zhu, and Bo Zhang. Estimating the optimal covariance with imperfect mean in diffusion probabilistic models. In *International Conference on Machine Learning*, pages 1555–1584. PMLR, 2022.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. In *International Conference on Learning Representations*, 2021.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020a.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4950–4957, 2018.

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*, 2020b.

Dohyun Kwon, Ying Fan, and Kangwook Lee. Score-based generative modeling secretly minimizes the wasserstein distance. In *Advances in Neural Information Processing Systems*, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2021.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Michael Arbel, Danica J Sutherland, Mikołaj Bińkowski, and Arthur Gretton. On gradient regularizers for mmd gans. *Advances in neural information processing systems*, 31, 2018.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *J. Mach. Learn. Res.*, 21(132):1–62, 2020.

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

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

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 515–524, 2017.

Muhammad Sarmad, Hyunjoo Jenny Lee, and Young Min Kim. Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5898–5907, 2019.

Xueying Bai, Jian Guan, and Hongning Wang. A model-based reinforcement learning with adversarial training for online recommendation. *Advances in Neural Information Processing Systems*, 32, 2019.

Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders. *arXiv preprint arXiv:2202.09671*, 2022.

Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-gan: Training gans with diffusion. *arXiv preprint arXiv:2206.02262*, 2022.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

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. *arXiv preprint arXiv:2206.00927*, 2022.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality. In *International Conference on Learning Representations*, 2021b.

Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

Emanuele Aiello, Diego Valsesia, and Enrico Magli. Fast inference in denoising diffusion models via mmd finetuning. *arXiv preprint arXiv:2301.07969*, 2023.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. *Advances in neural information processing systems*, 30, 2017.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

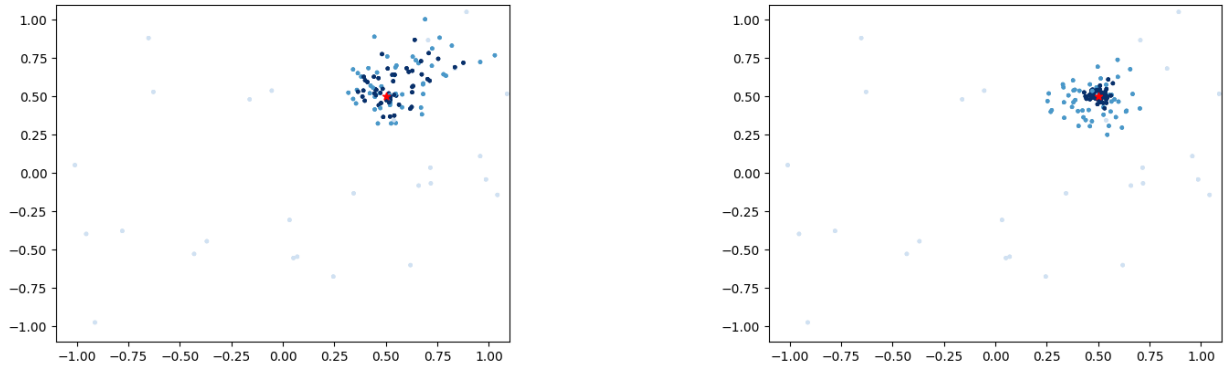
Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

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. *Advances in neural information processing systems*, 30, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A. Visualization: Effect of Shortcut Fine-Tuning



(a) Sampling steps before fine-tuning from DDPM

(b) Sampling steps after fine-tuning

Figure 7. Visualization of the sampling path before (7a) and after short-cut fine-tuning (7b).

We provide visualizations of the complete sampling chain before and after fine-tuning in Fig 7. We generate 50 data points using the same random seed for DDPM and our fine-tuned model, trained on the same Gaussian cluster centered at the red spot $(0.5, 0.5)$ with a standard deviation of 0.01 in each dimension, $T = 2$. The whole sampling path is visualized where different steps are marked with different intensities of the color: data points with the darkest color are finally generated. As shown in Fig 7, our fine-tuning does find a "shortcut" path to the final distribution.

B. Illustration of Sub-sampling with $T' \ll T$ in DDPM

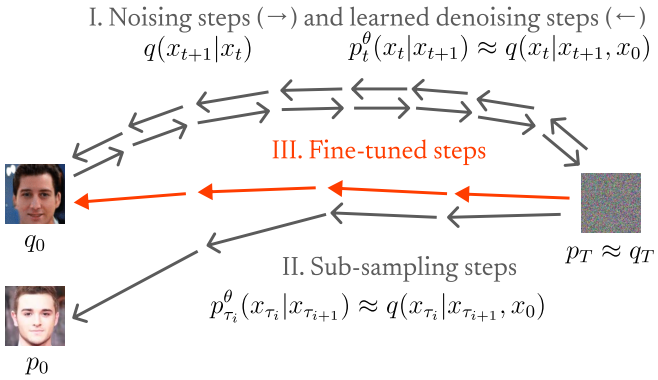


Figure 8. When T is large but we sub-sample with $T' \ll T$ cannot approximate the backward process accurately when each step is Gaussian as discussed in Case 2, Section 3.1. In this case, shortcut fine-tuning can also solve the issue by directly minimizing the IPM as an objective function.

C. Towards Monotonic Improvement

Here we present detailed proof of Theorem 4.2. For simplicity, we denote p_0^θ as p_θ, q_0 as q, and $z \in \mathbb{R}^d$ to replace x_0 as a variable in our sample space.

Recall the generated distribution: p_θ. Given target distribution q, the objective function is:

$$\min_\theta \max_{\alpha \in \mathcal{A}} g(p_\theta, f_\alpha, q), \tag{24}$$

where $g(p_\theta, f_\alpha, q) = \int (p_\theta(z) - q(z)) f_\alpha(z) dz$.

Recall $\Phi(p_\theta, q) = \max_{\alpha \in \mathcal{A}} \int (p_\theta(z) - q(z)) f_\alpha(z) dz = \int (p_\theta(z) - q(z)) f_{\alpha^*(p_\theta, q)}(z) dz$.

Assume that $g(p_\theta, f_\alpha, q)$ is Lipschitz w.r.t. θ, given q and $\alpha \in \mathcal{A}$. Our goal is to show that there exists $l \geqslant 0$ s.t.:

$$\Phi(p_{\theta'}, q) \leqslant g(p_{\theta'}, f_{\alpha^*(p_\theta, q)}, q) + 2l||\theta - \theta'||, \tag{25}$$

where the equality is achieved when $\theta = \theta'$.

If the above inequality holds, $L_\theta(\theta') = g(p_{\theta'}, f_{\alpha^*(p_\theta, q)}, q) + 2l||\theta - \theta'||$ can be a surrogate function of $\Phi(p_{\theta'}, q)$: $\Phi(p_{\theta'}, q) - \Phi(p_\theta, q) \leqslant L_\theta(\theta') - L_\theta(\theta)$, $L_\theta(\theta) = \Phi(p_\theta, q)$, which means θ' that can improve $L_\theta(\theta')$ is also guaranteed to get improvement on $\Phi(p_{\theta'}, q)$.

Proof. Consider

$$\begin{aligned}
&\Phi(p_{\theta'}, q) - \Phi(p_\theta, q) \\
=& \int (p_{\theta'}(z) - q(z)) f_{\alpha^*(p_{\theta'}, q)}(z) dz - \int (p_\theta(z) - q(z)) f_{\alpha^*(p_\theta, q)}(z) dz \\
=& \int (p_{\theta'}(z) - q(z)) f_{\alpha^*(p_{\theta'}, q)}(z) dz - \int (p_{\theta'}(z) - q(z)) f_{\alpha^*(p_\theta, q)}(z) dz \\
&+ \int (p_{\theta'}(z) - q(z)) f_{\alpha^*(p_\theta, q)}(z) dz - \int (p_\theta(z) - q(z)) f_{\alpha^*(p_\theta, q)}(z) dz \\
=& \int (p_{\theta'}(z) - q(z))(f_{\alpha^*(p_{\theta'}, q)}(z) - f_{\alpha^*(p_\theta, q)}(z)) dz + \int (p_{\theta'}(z) - p_\theta(z)) f_{\alpha^*(p_\theta, q)}(z) dz \\
=& \int (q(z) - p_{\theta'}(z))(f_{\alpha^*(p_\theta, q)}(z) - f_{\alpha^*(p_{\theta'}, q)}(z)) dz + \int (p_{\theta'}(z) - p_\theta(z)) f_{\alpha^*(p_\theta, q)}(z) dz.
\end{aligned} \tag{26}$$

We have

$$\begin{aligned}
&\int (q(z) - p_{\theta'}(z))(f_{\alpha^*(p_\theta, q)}(z) - f_{\alpha^*(p_{\theta'}, q)}(z)) dz \\
=& \int (p_\theta(z) - p_{\theta'}(z))(f_{\alpha^*(p_\theta, q)}(z) - f_{\alpha^*(p_{\theta'}, q)}(z)) dz - \int (p_\theta(z) - q(z))(f_{\alpha^*(p_\theta, q)}(z) - f_{\alpha^*(p_{\theta'}, q)}(z)) dz \\
\leqslant& \int (p_\theta(z) - p_{\theta'}(z))(f_{\alpha^*(p_\theta, q)}(z) - f_{\alpha^*(p_{\theta'}, q)}(z)) dz,
\end{aligned} \tag{27}$$

where the last inequality comes from the definition: $\alpha^*(p_\theta, q) = \arg\max_{\alpha \in \mathcal{A}} \int (p_\theta(z) - q(z)) f_\alpha(z)$.

So

$$\begin{aligned}
&\Phi(p_{\theta'}, q) - \Phi(p_\theta, q) \\
=& \int (p_{\theta'}(z) - p_\theta(z)) f_{\alpha^*(p_\theta, q)}(z) dz + \int (q(z) - p_{\theta'}(z))(f_{\alpha^*(p_\theta, q)}(z) - f_{\alpha^*(p_{\theta'}, q)}(z)) dz \\
\leqslant& g(p_{\theta'}, f_{\alpha^*(p_\theta, q)}, q) - g(p_\theta, f_{\alpha^*(p_\theta, q)}, q) + \int (p_\theta(z) - p_{\theta'}(z))(f_{\alpha^*(p_\theta, q)}(z) - f_{\alpha^*(p_{\theta'}, q)}(z)) dz \\
\leqslant& g(p_{\theta'}, f_{\alpha^*(p_\theta, q)}, q) - g(p_\theta, f_{\alpha^*(p_\theta, q)}, q) + 2l||\theta - \theta'||,
\end{aligned} \tag{28}$$

where the last inequality comes from the Lipschitz assumption of $g(p_\theta, f_{\alpha(p_\theta, q)}, q)$ given $\alpha^*(p_\theta, q)$ and $\alpha^*(p_{\theta'}, q)$. Recall that $\Phi(p_\theta, q) = g(p_\theta, f_{\alpha^*(p_\theta, q)}, q)$, the proof is then complete. □

Consider the optimization objective: minimize$_{\theta'} L_\theta(\theta')$. Using the Lagrange multiplier, we can convert the problem to a constrained optimization problem:

$$\begin{aligned} &\underset{\theta'}{\text{minimize}} \quad g(p_{\theta'}, f_{\alpha^*(p_\theta,q)}, q) \\ &\text{s.t.} \quad ||\theta' - \theta|| \leqslant \delta \end{aligned} \tag{29}$$

where $\delta > 0$. The constraint is a convex set and the projection to the set is easy to compute via norm regularization, as we discussed in Section 4.2. Intuitively, it means that as long as we only optimize in the neighborhood of the current generator θ', we can treat $g(p_{\theta'}, f_{\alpha^*(p_\theta,q)}, q)$ as an approximation of $\Phi(p_{\theta'}, q)$ during gradient updates.

D. Baseline Function for Variance Reduction

Here we present the derivation of Eq (20), which is very similar to Schulman et al. (2015b).

To show

$$\mathop{\mathbb{E}}_{p^\theta_{x_{0:T}}}\left[f_\alpha(x_0)\sum_{t=0}^{T-1}\nabla_\theta \log p_t^\theta(x_t|x_{t+1})\right] = \mathop{\mathbb{E}}_{p^\theta_{x_{0:T}}}\left[\sum_{t=0}^{T-1}(f_\alpha(x_0) - V^\omega_{t+1}(x_{t+1}))\nabla_\theta \log p_t^\theta(x_t|x_{t+1})\right], \tag{30}$$

we only need to show

$$\mathop{\mathbb{E}}_{p^\theta_{x_{0:T}}}\left[V^\omega_{t+1}(x_{t+1})\nabla_\theta \log p_t^\theta(x_t|x_{t+1})\right] = 0. \tag{31}$$

Note that

$$\begin{aligned} &\mathop{\mathbb{E}}_{p^\theta_{x_{0:T}}}\left[V^\omega_{t+1}(x_{t+1})\nabla_\theta \log p_t^\theta(x_t|x_{t+1})\right] \\ =& \mathop{\mathbb{E}}_{p^\theta_{x_{t+1:T}}}\left[\mathop{\mathbb{E}}_{p^\theta_{x_{0:t}}}\left[V^\omega_{t+1}(x_{t+1})\nabla_\theta \log p_t^\theta(x_t|x_{t+1})|x_{t+1:T}\right]\right] \\ =& \mathop{\mathbb{E}}_{p^\theta_{x_{t+1:T}}}\left[\mathop{\mathbb{E}}_{p^\theta_{x_t}}\left[V^\omega_{t+1}(x_{t+1})\nabla_\theta \log p_t^\theta(x_t|x_{t+1})|x_{t+1:T}\right]\right], \end{aligned} \tag{32}$$

where $\mathop{\mathbb{E}}_{p^\theta_{x_t}}\left[V^\omega_{t+1}(x_{t+1})\nabla_\theta \log p_t^\theta(x_t|x_{t+1})|x_{t+1:T}\right] = 0$ when $p_t^\theta(x_t|x_{t+1})$ and $\nabla_\theta p_t^\theta(x_t|x_{t+1})$ are continuous:

$$\begin{aligned} &\mathop{\mathbb{E}}_{p^\theta_{x_t}}\left[V^\omega_{t+1}(x_{t+1})\nabla_\theta \log p_t^\theta(x_t|x_{t+1})|x_{t+1:T}\right] \\ =& V^\omega_{t+1}(x_{t+1})\int p^\theta_{x_t}(x_t)\nabla_\theta \log p_t^\theta(x_t|x_{t+1})dx_t \\ =& V^\omega_{t+1}(x_{t+1})\int p^\theta_{x_t}(x_t)\nabla_\theta \log p_t^\theta(x_t|x_{t+1})dx_t \\ =& V^\omega_{t+1}(x_{t+1})\int \nabla_\theta p_t^\theta(x_t|x_{t+1})dx_t \\ =& V^\omega_{t+1}(x_{t+1})\nabla_\theta \int p_t^\theta(x_t|x_{t+1})dx_t \\ =& 0. \end{aligned} \tag{33}$$

E. Comparison with DDIM Sampling

We present a comparison with DDIM sampling methods on CIFAR 10 benchmark as below. Methods marked with * require additional model training, and NFE is the number of sampling steps (number of score function evaluations). All methods are based on the same pretrained DDPM model with $T = 1000$.

Method (DDPM, stochastic)	NFE	FID	Method (DDIM, deterministic)	NFE	FID
DDPM	10	34.76	DDIM	10	17.33
SN-DDPM	10	16.33	DPM-solver	10	4.70
SFT-PG*	10	2.28			
SFT-PG*	8	2.64	Progressive distillation*$^+$	8	2.57

Table 4. Comparison with DDIM sampling methods which is deterministic given the initial noise.

We can observe that SFT-PG with NFE=10 produces the best FID, and SFT-PG with NFE=8 is comparable to progressive distillation with the same NFE. Our method is orthogonal to other fast sampling methods like distillation. We also note that our fine-tuning is more computationally efficient than progressive distillation: For example, for CIFAR10, progressive distillation takes about a day using 8 TPUv4 chips, while our method takes about 6h using 4 RTX 2080Ti, and the original DDPM training takes 10.6h using TPU v3.8. Besides, since we use a fixed small learning rate during training (1e-6), it is also possible to further accelerate our training by choosing appropriate learning rate schedules.

F. Experimental Details

Here we provide more details for our fine-tuning settings for reproducibility.

F.1. Experiments on Toy Datasets

Training sets. For 2D toy datasets, each training set contains 10K samples.

Model architecture. The generator we adopt is a 4-layer MLP with 128 hidden units and soft-plus activations. The critic and the baseline function we use are 3-layer MLPs with 128 hidden units and ReLU activations.

Training details. For optimizers, we use Adam (Kingma and Ba, 2014) with $\text{lr} = 5 \times 10^{-5}$ for the generator, and $\text{lr} = 1 \times 10^{-3}$ for both the critic and baseline functions. Pretraining for DDPM is conducted for 2000 epochs for $T = 10, 100, 1000$ respectively. Both pretraining and fine-tuning use batch size 64 and we train 300 epochs for fine-tuning.

F.2. Experiments on Image Datasets

Training sets. We use 60K training samples from MNIST, 50K training samples from CIFAR-10, and 162K samples from CelebA.

Model architecture. For model architecture, we use U-Net as the generative model as Ho et al. (2020). For the critic, we adopt 3 convolutional layers with kernel size = 4, stride = 2, padding = 1 for downsampling, followed by 1 final convolutional layer with kernel size = 4, stride = 1, padding = 0, and then take the average of the final output. The numbers of output channels are 256,512,1024,1 for each layer, with Leaky ReLU (slope = 0.2) as activation. For the baseline function, we use a 4-layer MLP with timestep embeddings. The numbers of hidden units are 1024, 1024, 256, and the output dimension is 1.

Training details. For MNIST, we train a DDPM with $T = 10$ steps for 100 epochs to convergence as a pretrained model. For CIFAR-10 and CelebA, we use the pretrained model in Ho et al. (2020) and Song et al. (2020a) respectively with $T = 1000$, and use the sampling schedules calculated by FastDPM (Kong and Ping, 2021) with VAR approximation and DDPM sampling schedule as initialization for our fine-tuning. We found that rescaling the pixel values to [0,1] is a default choice in FastDPM, but it hurts the training if we put the rescaled images directly into the critic, so we remove the rescaling part during our fine-tuning. For optimizers, we use Adam with $\text{lr} = 1 \times 10^{-6}$ for the generator, and $\text{lr} = 1 \times 10^{-4}$ for both the critic and baseline functions. We found that smaller learning rates help the stability of training, which is compliant with the theoretical result in Section 4.2. For MNIST and CIFAR-10, we train 100 epochs with batch size = 128. For CelebA we trained 100 epochs with batch size = 64.

More generated samples. We present generated samples from the initialized FastDPM and our fine-tuned model respectively using the same random seed to show the effect of our fine-tuning in Fig 9 and Fig 10. We notice that some of

the images generated by our fine-tuned model are similar to images at initialization but with much richer colors and more details, and there are also some cases that the images after fine-tuning look very different than that from initialization.

Figure 9. Images generated from FastDPM as initialization (on the top) and from the fine-tuned model (on the bottom), generated using the same seed, trained on CIFAR-10.

Figure 10. Images generated from FastDPM as initialization (on the top) and from the fine-tuned model (on the bottom), generated using the same seed, trained on CelebA.