GENIE: Higher-Order Denoising Diffusion Solvers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Denoising diffusion models (DDMs) have emerged as a powerful class of generative models. A forward diffusion process slowly perturbs the data, while a deep model learns to gradually denoise. Synthesis amounts to solving a differential equation (DE) defined by the learnt model. Solving the DE requires slow iterative solvers for high-quality generation. In this work, we propose *Higher-Order Denoising Diffusion Solvers* (GENIE): Based on truncated Taylor methods, we derive a novel higher-order solver that significantly accelerates synthesis. Our solver relies on higher-order gradients of the perturbed data distribution, that is, higher-order score functions. In practice, only Jacobian-vector products (JVPs) are required and we propose to extract them from the first-order score network via automatic differentiation. We then distill the JVPs into a separate neural network that allows us to efficiently compute the necessary higher-order terms for our novel sampler during synthesis. We only need to train a small additional head on top of the first-order score network. We validate GENIE on multiple image generation benchmarks and demonstrate that GENIE outperforms all existing solvers. Unlike recent methods that fundamentally alter the generation process in DDMs, our GENIE solves the true generative DE and still enables applications such as encoding and guided sampling.

1 Introduction

Denoising diffusion models (DDMs) offer both state-of-the-art synthesis quality and sample diversity in combination with a robust and scalable learning objective. DDMs have been used for image [1–5] and video [6, 7] synthesis, super-resolution [8, 9], deblurring [10, 11], image editing and inpainting [5, 12–14], text-to-image synthesis [15–17], conditional and semantic image generation [18–22], image-to-image translation [14, 23, 24] and for inverse problems in medical imaging [25–31]. They also enable high-quality speech synthesis [32–37], 3D shape generation [38–42], molecular modeling [43–46], maximum likelihood training [47–50], and more [51–56]. In DDMs, a diffusion process gradually perturbs the data towards random noise, while a deep neural network learns to denoise. Formally, the problem reduces to learning the *score function*, i.e., the gradient of the log-density of the perturbed data. The (approximate) inverse of the forward diffusion can be described by an ordinary or a stochastic differential equation (ODE or SDE, respectively), defined by the learned score function, and can therefore be used for generation when starting from random noise [47, 57].

A crucial drawback of DDMs is that the generative ODE or SDE is typically difficult to solve, due to the complex score function. Therefore, efficient and tailored samplers are required for fast synthesis. In this work, building on the generative ODE [47, 57, 58], we rigorously derive a novel second-order ODE solver using *truncated Taylor methods* [59]. These higher-order methods require higher-order gradients of the ODE—in our case this includes higher-order gradients of the

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

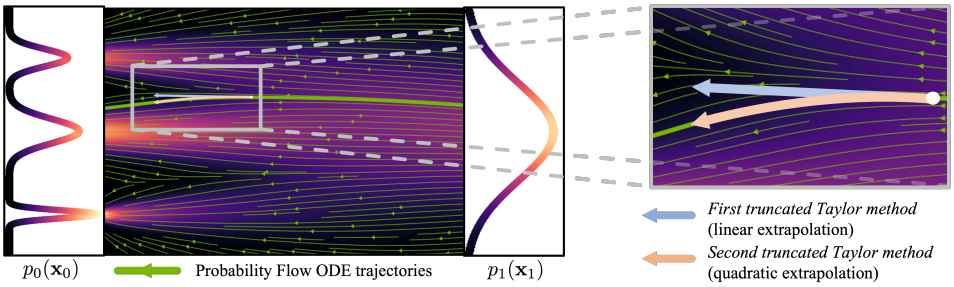


Figure 1: Our novel *Higher-Order Denoising Diffusion Solver* (GENIE) relies on the second *truncated Taylor method* (TTM) to simulate a (re-parametrized) Probability Flow ODE for sampling from denoising diffusion models. The second TTM captures the local curvature of the ODE's gradient field and enables more accurate extrapolation and larger step sizes than the first TTM (Euler's method), which previous methods such as DDIM [58] utilize.

log-density of the perturbed data, i.e., higher-order score functions. Because such higher-order scores are usually not available, existing works typically use simple first-order solvers or samplers with low accuracy [1, 57, 58, 60], higher-order methods that rely on suboptimal finite difference or other approximations [61–63], or alternative approaches [64–66] for accelerated sampling. Here, we fundamentally avoid such approximations and directly model the higher-order gradient terms: Importantly, our novel *Higher-Order Denoising Diffusion Solver (GENIE)* relies on Jacobian-vector products (JVPs) involving second-order scores. We propose to calculate these JVPs by automatic differentiation of the regular learnt first-order scores. For computational efficiency, we then distill the entire higher-order gradient of the ODE, including the JVPs, into a separate neural network. In practice, we only need to add a small head to the first-order score network to predict the components of the higher-order ODE gradient. By directly modeling the JVPs we avoid explicitly forming high-dimensional higher-order scores. Intuitively, the higher-order terms in GENIE capture the local curvature of the ODE and enable larger steps when iteratively solving the generative ODE (Fig. 1).

Experimentally, we validate GENIE on multiple image modeling benchmarks and achieve state-of-the-art performance in solving the generative ODE of DDMs with few synthesis steps. In contrast to recent methods that fundamentally modify the generation process of DDMs by training conditional GANs [67] or by distilling the full sampling trajectory [68, 69], GENIE solves the true generative ODE. Therefore, we also show that we can still encode images in the DDM's latent space, as required for instance for image interpolation, and use techniques such as guided sampling [4, 57, 70].

We make the following contributions: **(i)** We introduce GENIE, a novel second-order ODE solver for fast DDM sampling. **(ii)** We propose to extract the required higher-order terms from the first-order score model by automatic differentiation. In contrast to existing works, we explicitly work with higher-order scores without finite difference approximations. To the best of our knowledge, GENIE is the first method that *explicitly* uses higher-order scores for generative modeling with DDMs. **(iii)** We propose to directly model the necessary JVPs and distill them into a small neural network. **(iv)** We outperform all existing solvers and samplers for the generative differential equations of DDMs.

2 Background

We consider continuous-time DDMs [1, 57, 71] whose forward process can be described by

$$p_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \alpha_t \mathbf{x}_0, \sigma_t^2 \boldsymbol{I}), \tag{1}$$

where $\mathbf{x}_0 \sim p_0(\mathbf{x}_0)$ is drawn from the empirical data distribution and $\mathbf{x}_t$ refers to diffused data samples at time $t \in [0, 1]$ along the diffusion process. The functions α_t and σ_t are generally chosen such that the *logarithmic signal-to-noise ratio* [48] $\log \frac{\alpha_t^2}{\sigma_t^2}$ decreases monotonically with t and the data diffuses towards random noise, i.e., $p_1(\mathbf{x}_1) \approx \mathcal{N}(\mathbf{x}_1; \mathbf{0}, \boldsymbol{I})$. We use *variance-preserving* [57] diffusion processes for which $\sigma_t^2 = 1 - \alpha_t^2$ (however, all methods introduced in this work are applicable to more general DDMs). The diffusion process can then be expressed by the (variance-preserving) SDE

$$d\mathbf{x}_t = -\tfrac{1}{2}\beta_t \mathbf{x}_t \, dt + \sqrt{\beta_t} \, d\mathbf{w}_t, \tag{2}$$

where $\beta_t = -\frac{d}{dt} \log \alpha_t^2$, $\mathbf{x}_0 \sim p_0(\mathbf{x}_0)$ and $\mathbf{w}_t$ is a standard Wiener process. A corresponding reverse diffusion process that effectively inverts the forward diffusion is given by [57, 72, 73]

$$d\mathbf{x}_t = -\tfrac{1}{2}\beta_t \left[\mathbf{x}_t + 2\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] dt + \sqrt{\beta_t} \, d\mathbf{w}_t, \tag{3}$$

and this reverse-time generative SDE is marginally equivalent to the generative ODE [47, 57]

$$d\mathbf{x}_t = -\tfrac{1}{2}\beta_t \left[\mathbf{x}_t + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] dt, \tag{4}$$

where $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is the *score function*. Eq. (4) is referred to as the *Probability Flow* ODE [57], an instance of continuous Normalizing flows [74, 75]. To generate samples from the DDM, one can sample $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{x}_1; \mathbf{0}, \boldsymbol{I})$ and numerically simulate either the Probability Flow ODE or the generative SDE, replacing the unknown score function by a learned score model $\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$.

The DDIM solver [58] has been particularly popular to simulate DDMs due to its speed and simplicity. It has been shown that DDIM is *Euler's method* applied to an ODE based on a re-parameterization of the Probability Flow ODE [58, 69]: Defining $\gamma_t = \sqrt{\frac{1-\alpha_t^2}{\alpha_t^2}}$ and $\bar{\mathbf{x}}_t = \mathbf{x}_t\sqrt{1+\gamma_t^2}$, we have

$$\frac{d\bar{\mathbf{x}}_t}{d\gamma_t} = \sqrt{1+\gamma_t^2}\frac{d\mathbf{x}_t}{dt}\frac{dt}{d\gamma_t} + \mathbf{x}_t\frac{\gamma_t}{\sqrt{1+\gamma_t^2}} = -\frac{\gamma_t}{\sqrt{1+\gamma_t^2}}\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t), \tag{5}$$

where we inserted Eq. (4) for $\frac{d\mathbf{x}_t}{dt}$ and used $\beta(t)\frac{dt}{d\gamma_t} = \frac{2\gamma_t}{\gamma_t^2+1}$. Letting $\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) := -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\sigma_t}$ denote a parameterization of the score model, the approximate generative DDIM ODE is then given by

$$d\bar{\mathbf{x}}_t = \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right) d\gamma_t, \tag{6}$$

where we used $\sigma_t = \sqrt{1-\alpha_t^2} = \frac{\gamma_t}{\sqrt{\gamma_t^2+1}}$ (see App. A for a more detailed derivation of Eq. (6)). The model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ can be learned by minimizing the score matching objective [1, 76]

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}[t_{\text{cutoff}},1],\mathbf{x}_0\sim p(\mathbf{x}_0),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[g(t)\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\|_2^2\right], \quad \mathbf{x}_t = \alpha_t\mathbf{x}_0 + \sigma_t\boldsymbol{\epsilon}, \tag{7}$$

for small $0 < t_{\text{cutoff}} \ll 1$. As is standard practice, we set $g(t) = 1$. Other weighting functions $g(t)$ are possible; for example, setting $g(t) = \frac{\beta_t}{2\sigma_t^2}$ recovers maximum likelihood learning [47–50].

3 Higher-Order Denoising Diffusion Solver

As discussed in Sec. 2, the so-known DDIM solver [58] is simply Euler's method applied to the DDIM ODE (cf. Eq. (6)). In this work, we apply a higher-order method to the DDIM ODE, building on the *truncated Taylor method* (TTM) [59]. The p-th TTM is simply the p-th order *Taylor polynomial* applied to an ODE. For example, for the general $\frac{d\mathbf{y}}{dt} = \boldsymbol{f}(\mathbf{y}, t)$, the p-th TTM reads as

$$\mathbf{y}_{t_{n+1}} = \mathbf{y}_{t_n} + h_n\frac{d\mathbf{y}}{dt}|_{(\mathbf{y}_{t_n},t_n)} + \cdots + \frac{1}{p!}h_n^p\frac{d^p\mathbf{y}}{dt^p}|_{(\mathbf{y}_{t_n},t_n)}, \tag{8}$$

where $h_n = t_{n+1} - t_n$. Note that the first TTM is simply Euler's method. Applying the second TTM to the DDIM ODE results in the following scheme

$$\bar{\mathbf{x}}_{t_{n+1}} = \bar{\mathbf{x}}_{t_n} + h_n\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) + \frac{1}{2}h_n^2\frac{d\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{d\gamma_t}|_{(\mathbf{x}_{t_n},t_n)}, \tag{9}$$

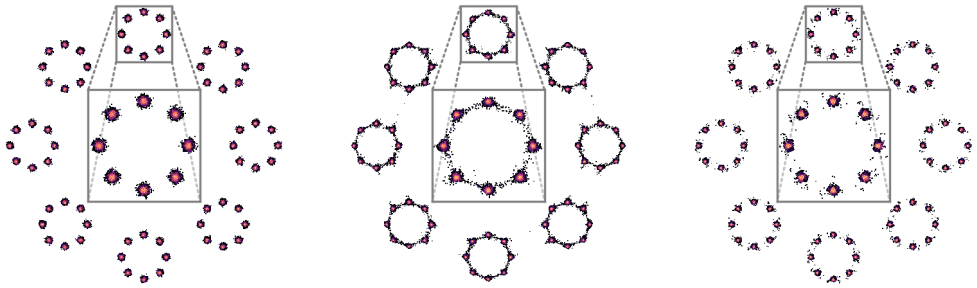
where $h_n = \gamma_{t_{n+1}} - \gamma_{t_n}$. Recall that $\gamma_t = \sqrt{\frac{1-\alpha_t^2}{\alpha_t^2}}$, where the function α_t is a time-dependent hyperparameter of the DDM. The total derivative $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}} := \frac{d\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{d\gamma_t}$ can be decomposed as follows

$$d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \frac{\partial\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial\mathbf{x}_t}\frac{d\mathbf{x}_t}{d\gamma_t} + \frac{\partial\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t}\frac{dt}{d\gamma_t}, \tag{10}$$

where $\frac{\partial\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial\mathbf{x}_t}$ denotes the Jacobian of $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ and

$$\frac{d\mathbf{x}_t}{d\gamma_t} = \frac{\partial\mathbf{x}_t}{\partial\bar{\mathbf{x}}_t}\frac{d\bar{\mathbf{x}}_t}{d\gamma_t} + \frac{\partial\mathbf{x}_t}{\partial\gamma_t} = \frac{1}{\sqrt{\gamma_t^2+1}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \frac{\gamma_t}{1+\gamma_t^2}\mathbf{x}_t. \tag{11}$$

If not explicitly stated otherwise, we refer to the second TTM applied to the DDIM ODE, i.e., the scheme in Eq. (9), as *Higher-Order Denoising Diffusion Solver* (GENIE). Intuitively, the higher-order gradient terms used in the second TMM model the local curvature of the ODE. This translates into a Taylor formula-based extrapolation that is quadratic in time (cf. Eqs. (8) and (9)) and more accurate than linear extrapolation, as in Euler's method, thereby enabling larger time steps (see Fig. 1 for a visualization). In App. B, we also discuss the application of the third TTM to the DDIM ODE. We emphasize that TTMs are not restricted to the DDIM ODE and could just as well be applied to the Probability Flow ODE [57] (also see App. B) or neural ODEs [74, 75] more generally.



(a) Ground truth (b) DDIM (c) GENIE

Figure 2: Modeling a complex 2D toy distribution: Samples in *(b)* and *(c)* are generated via DDIM and GENIE, respectively, with 25 solver steps using the analytical score function of the ground truth distribution.

The Benefit of Higher-Order Methods: We showcase the benefit of higher-order methods on a 2D toy distribution (Fig. 2a) for which we know the score function as well as all higher-order derivatives necessary for GENIE analytically. We generate 1k different accurate "ground truth" trajectories $\mathbf{x}_t$ using DDIM with 10k steps. We compare these "ground truth" trajectories to *single* steps of DDIM and GENIE for varying step sizes Δt. We then measure the mean L_2-distance of the single steps $\hat{\mathbf{x}}_t(\Delta t)$ to the "ground truth" trajectories $\mathbf{x}_t$, and we repeat this experiment for three starting points $t \in \{0.1, 0.2, 0.5\}$. We see (Fig. 3 (*top*)) that GENIE can use larger step sizes to stay within a certain error tolerance for all starting points t. We further show samples for DDIM and GENIE, using 25 solver steps, in Fig. 2. DDIM has the undesired behavior of sampling low-density regions between modes, whereas GENIE looks like a slightly noisy version of the ground truth distribution (Fig. 2a).

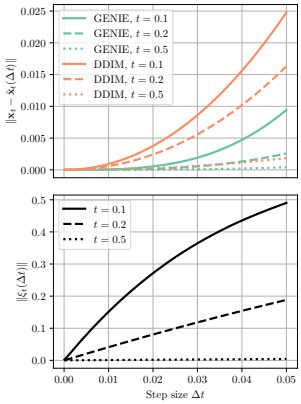


Figure 3: *Top*: Single step error using analytical score function. *Bottom*: Norm of difference $\xi_t(\Delta t)$ between analytical and approximate derivative computed via finite difference method.

Comparison to Multistep Methods: Linear multistep methods are an alternative higher-order method to solve ODEs. Liu et al. [63] applied the well-established Adams–Bashforth [AB, 77] method to the DDIM ODE. AB methods can be derived from TTMs by approximating higher-order derivatives $\frac{d^p \mathbf{y}}{dt^p}$ using the finite difference method [78]. For example, the second AB method is obtained from the second TTM by replacing $\frac{d^2 \mathbf{y}}{dt^2}$ with the first-order forward difference approximation $(f(\mathbf{y}_{t_n}, t_n) - f(\mathbf{y}_{t_{n-1}}, t_{n-1}))/h_{n-1}$. In Fig. 3 (*bottom*), we visualize the mean L_2-norm of the difference $\xi_t(\Delta t)$ between the analytical derivative $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ and its first-order forward difference approximation for varying step sizes Δt for the 2D toy distribution. The approximation is especially poor at small t for which the score function becomes complex (App. E for details on all toy experiments).

3.1 Learning Higher-Order Derivatives

The above observations inspire to apply GENIE to DDMs of more complex and high-dimensional data such as images. Regular DDMs learn a model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ for the first-order score; however, the higher-order gradient terms required for GENIE (cf. Eq. (10)) are not immediately available to us, unlike in the toy example above. Let us insert Eq. (11) into Eq. (10) and analyze the required terms more closely:

$$d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}} = \frac{1}{\sqrt{\gamma_t^2 + 1}} \underbrace{\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}_{\text{JVP}_1} - \frac{\gamma_t}{1 + \gamma_t^2} \underbrace{\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \mathbf{x}_t}_{\text{JVP}_2} + \frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t} \frac{dt}{d\gamma_t}. \tag{12}$$

We see that the full derivative decomposes into two JVP terms and one simpler time derivative term. The term $\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t}$ plays a crucial role in Eq. (12). It can be expressed as

$$\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} = -\sigma_t \frac{\partial \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \approx -\sigma_t \nabla_{\mathbf{x}_t}^\top \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t), \tag{13}$$

which means that GENIE relies on *second-order score functions* $\nabla_{\mathbf{x}_t}^\top \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ under the hood.

Given a DDM, that is, given $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$, we could compute the derivative $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ for the GENIE scheme in Eq. (9) using automatic differentiation (AD). This would, however, make a single step of GENIE at

least twice as costly as DDIM, because we would need a forward pass through the $\boldsymbol{\epsilon_\theta}$ network to compute $\boldsymbol{\epsilon_\theta}(\mathbf{x}_t, t)$ itself, and another pass to compute the JVPs and the time derivative in Eq. (12). These forward passes cannot be parallelized, since the vector-part of JVP_1 in Eq. (12) involves $\boldsymbol{\epsilon_\theta}$ itself, and needs to be known before computing the JVP. To accelerate sampling, this overhead is too expensive.

Gradient Distillation: To avoid this overhead, we propose to first distill $d_{\gamma_t}\boldsymbol{\epsilon_\theta}$ into a separate neural network. During distillation training, we can use the slow AD-based calculation of $d_{\gamma_t}\boldsymbol{\epsilon_\theta}$, but during synthesis we call the trained neural network. We build on the observation that the internal representations of the neural network modeling $\boldsymbol{\epsilon_\theta}$ (in our case a U-Net [79] architecture) can be used for downstream tasks [80, 81]: specifically, we provide the last feature layer from the $\boldsymbol{\epsilon_\theta}$ network together with its time embedding as well as $\mathbf{x}_t$ and the output $\boldsymbol{\epsilon_\theta}(\mathbf{x}_t, t)$ to a small prediction head $\boldsymbol{k_\psi}(\mathbf{x}_t, t)$ that models the different terms in Eq. (12) (see Fig. 4). The overhead generated by $\boldsymbol{k_\psi}$ is small, for instance less than 2% for our CIFAR-10 model (also see Sec. 5), and we found this approach to provide excellent performance. Note that in principle we could also train an independent deep neural network, which does not make use of the internal representations of $\boldsymbol{\epsilon_\theta}$ and could therefore theoretically be run in parallel to the $\boldsymbol{\epsilon_\theta}$ model. We justify using small prediction heads over independent neural networks because AD-based distillation training is slow: in each training iteration we first need to call the $\boldsymbol{\epsilon_\theta}$ network, then calculate the JVP terms, and only then can we call the distillation model. By modeling $d_{\gamma_t}\boldsymbol{\epsilon_\theta}$ via small prediction heads, while reusing the internal representation of the score model, we can make training relatively fast: we only need to train $\boldsymbol{k_\psi}$ for up to 50k iterations. In contrast, training score models from scratch takes roughly an order of magnitude more iterations. We leave training of independent networks to predict $d_{\gamma_t}\boldsymbol{\epsilon_\theta}$ to future work.

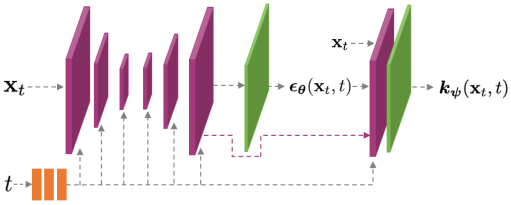


Figure 4: Our distilled model $\boldsymbol{k_\psi}$ that predicts the gradient $d_{\gamma_t}\boldsymbol{\epsilon_\theta}$ is implemented as a small additional output head on top of the first-order score model $\boldsymbol{\epsilon_\theta}$.

Mixed Network Parameterization: We found that learning $d_{\gamma_t}\boldsymbol{\epsilon_\theta}$ directly as single output of a neural network can be challenging. Assuming a single data point distribution $p_0(\mathbf{x}_0) = \delta(\mathbf{x}_0 = \mathbf{0})$, for which we know the diffused score function and all higher-order derivatives analytically, we found that the terms in Eq. (12) all behave very differently within the $t \in [0, 1]$ interval (for instance, the prefactor of JVP_1 in Eq. (12) approaches 1 as $t \to 0$, while JVP_2's prefactor vanishes). As outlined in detail in App. C.2.3, this simple single data point assumption implies an effective *mixed network parameterization*, an approach inspired by the "mixed score parametrizations" in Vahdat et al. [49] and Dockhorn et al. [60]. In particular, we model

$$\boldsymbol{k_\psi} = -\frac{1}{\gamma_t}\boldsymbol{k}_{\boldsymbol{\psi}}^{(1)} + \frac{\gamma_t}{1+\gamma_t^2}\boldsymbol{k}_{\boldsymbol{\psi}}^{(2)} + \frac{1}{\gamma_t(1+\gamma_t^2)}\boldsymbol{k}_{\boldsymbol{\psi}}^{(3)} \approx d_{\gamma_t}\boldsymbol{\epsilon_\theta}, \tag{14}$$

where $\boldsymbol{k}_{\boldsymbol{\psi}}^{(i)}(\mathbf{x}_t, t)$, $i \in \{1, 2, 3\}$, are different output channels of the neural network (i.e. the additional head on top of the $\boldsymbol{\epsilon_\theta}$ network). The three terms in Eq. (14) exactly correspond to the three terms of Eq. (12), in the same order. We show the superior performance of this parametrization in Sec. 5.3.

Learning Objective: We would like our model $\boldsymbol{k_\psi}$ to match $d_{\gamma_t}\boldsymbol{\epsilon_\theta}$ for all t and $\mathbf{x}_t$. This suggests a simple (weighted) L_2-loss for distillation, similar to regular score matching losses for DDMs [1, 57],

$$\min_{\boldsymbol{\psi}} \mathbb{E}_{t\sim\mathcal{U}[t_{\text{cutoff}},1],\mathbf{x}_0\sim p(\mathbf{x}_0),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[g_{\text{d}}(t)\|\boldsymbol{k}_\psi(\mathbf{x}_t, t) - d_{\gamma_t}\boldsymbol{\epsilon_\theta}(\mathbf{x}_t, t)\|_2^2\right], \tag{15}$$

where $\mathbf{x}_t = \alpha_t\mathbf{x}_0 + \sigma_t\boldsymbol{\epsilon}$. We choose the weighting function $g_{\text{d}}(t) = \gamma_t^2$ to counteract the division by γ_t (note $\gamma_0 = 0$) in the first and third term of our mixed network parameterization in Eq. (14). We found that this leads to a roughly constant loss over different time values t. We emphasize that during training we need to compute $d_{\gamma_t}\boldsymbol{\epsilon_\theta}$ via AD; however, at inference time we use the learned prediction head $\boldsymbol{k_\psi}$ to approximate $d_{\gamma_t}\boldsymbol{\epsilon_\theta}$ when sampling with GENIE. In App. C.2.4, we provide pseudo code for training and sampling with prediction heads $\boldsymbol{k_\psi}$.

Alternative Learning Approaches: As shown in Eq. (13), GENIE relies on second-order score functions. Recently, Meng et al. [82] directly learnt such higher-order scores with higher-order score matching objectives. Directly applying these techniques has the downside that we would need to explicitly form the higher-order score terms $\nabla_{\mathbf{x}_t}^\top\boldsymbol{\epsilon_\theta}(\mathbf{x}_t, t)$, which are very high-dimensional for data such

as images. Low-rank approximations are possible, but potentially insufficient for high performance. In our approach, we are avoiding this complication by directly modeling the lower-dimensional JVPs. We found that the methods from Meng et al. [82] can be modified to provide higher-order score matching objectives for the JVP terms required for GENIE and we briefly explored this (see App. D). However, our distillation approach with AD-based higher-order gradients worked much better. Nevertheless, this is an interesting direction for future research. To the best of our knowledge, GENIE is the first solver for the generative differential equations of DDMs that *directly* uses higher-order scores (in the form of the distilled JVPs) for generative modeling without finite difference or other approximations.

4 Related Work

Accelerated Sampling from DDMs. Several previous works address the slow sampling of DDMs: One line of work reduces and readjusts the timesteps [3, 64] used in time-discretized DDMs [1, 71]. This can be done systematically by grid search [32] or dynamic programming [83]. Bao et al. [65] speed up sampling by defining a new DDM with optimal reverse variances. DDIM [58], discussed in Sec. 2, was also introduced as a method to accelerate DDM synthesis. Further works leverage modern ODE and SDE solvers for fast synthesis from (continuous-time) DDMs: For instance, higher-order Runge–Kutta methods [57, 84] and adaptive step size SDE solvers [62] have been used. These methods are not optimally suited for the few-step synthesis regime, in which GENIE shines; see also Sec. 5. Most closely related to our work is Liu et al. [63], which simulates the DDIM ODE [58] using a higher-order linear multistep method [77]. As shown in Sec. 3, linear multistep methods can be considered an approximation of the TTMs used in GENIE. Furthermore, Tachibana et al. [61] solve the generative SDE via a higher-order Itô–Taylor method [59] and, in contrast to our work, approximate higher-order score functions with the "ideal derivative trick": they essentially assume a simple single-point data distribution for which higher-order score functions can be computed analytically.

Alternatively, sampling from DDMs can also be accelerated via learning: For instance, Watson et al. [66] learn parameters of a generalized family of DDMs by optimizing for perceptual output quality. Luhman and Luhman [68] and Salimans and Ho [69] distill a DDIM sampler into a student model, which enables sampling in as few as a single step. Xiao et al. [67] replace DDMs' Gaussian samplers with expressive generative adversarial networks, similarly allowing for few-step synthesis. GENIE can also be considered a learning-based approach, as we distill a derivative of the generative ODE into a separate neural network. However, in contrast to the mentioned methods, GENIE still solves the true underlying generative ODE, which has major advantages: for instance, it can still be used easily for classifier-guided sampling [4, 57, 70] and to efficiently encode data into latent space—a prerequisite for likelihood calculation [47, 57] and editing applications [17]. Note that the learnt sampler [66] defines a proper probabilistic generalized DDM; however, it isn't clear how it relates to the generative SDE or ODE and therefore how compatible the method is with applications such as classifier guidance.

Other approaches to accelerate DDM sampling change the diffusion itself [60, 85, 86] or train DDMs in the latent space of a Variational Autoencoder [49]. GENIE is complementary to these methods.

Higher-Order ODE Gradients beyond DDMs. TTMs [78] and other methods that leverage higher-order gradients are also applied outside the scope of DDMs. For instance, higher-order derivatives can play a crucial role when developing solvers [87] and regularization techniques [88, 89] for neural ODEs [74, 75]. Outside the field of machine learning, higher-order TTMs have been widely studied, for example, to develop solvers for stiff [90] and non-stiff [90, 91] systems.

5 Experiments

Datasets: We run experiments on four datasets: CIFAR-10 [92] (resolution 32), LSUN Bedrooms [93] (128), LSUN Church-Outdoor [93] (128), and (conditional) ImageNet [94] (64).

Architectures: Except for CIFAR-10 (we use a checkpoint by Song et al. [57]), we train our own score models using architectures introduced by previous works [1, 4]. The architecture of our prediction heads is based on (modified) BigGAN residual blocks [57, 95]. To minimize computational overhead, we only use a single residual block. See App. C for training and architecture details.

Evaluation: We measure sample quality via Fréchet Inception Distance [FID, 96] (see App. F.1).

Synthesis Strategy: We simulate the DDIM ODE from $t{=}1$ up to $t{=}10^{-3}$ using evaluation times following a quadratic function (*quadratic striding* [58]). For variance-preserving DDMs, it can be beneficial to denoise the ODE solver output at the cutoff $t{=}10^{-3}$, i.e., $\mathbf{x}_0 = \frac{\mathbf{x}_t - \sigma_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\alpha_t}$ [57, 97]. Note that the denoising step involves a score model evaluation, and therefore "loses" a function evaluation that could otherwise be used as an additional step in the ODE solver. To this end, denoising the output of the ODE solver is left as a hyperparameter of our synthesis strategy.

Analytical First Step (AFS): Every additional neural network call becomes crucial in the low number of function evaluations (NFEs) regime. We found that we can improve the performance of GENIE and all other methods evaluated on our checkpoints by replacing the learned score with the (analytical) score of $\mathcal{N}(\mathbf{0}, \boldsymbol{I}) \approx p_{t=1}(\mathbf{x}_t)$ in the first step of the ODE solver. The "gained" function evaluation can then be used as an additional step in the ODE solver. Similarly to the denoising step mentioned above, AFS is treated as a hyperparameter of our **Synthesis Strategy**. AFS details in App. F.2.

Accounting for Computational Overhead: GENIE has a slightly increased computational overhead compared to other solvers due to the prediction head $\boldsymbol{k}_\psi$. The computational overhead is increased by 1.47%, 2.83%, 14.0%, and 14.4% on CIFAR-10, ImageNet, LSUN Bedrooms, and LSUN Church-Outdoor, respectively (see also App. C.2.5). This additional overhead is always accounted for implicitly: we divide the NFEs by the computational overhead and round to the nearest integer. For example, on LSUN Bedrooms, we compare baselines with 10/15 NFEs to GENIE with 9/13 NFEs.

5.1 Image Generation

In Fig. 5 we compare our method to the most competitive baselines. In particular, on the same score model checkpoints, we compare GENIE with DDIM [58], S-PNDM [63], and F-PNDM [63]. For these four methods, we only include the best result over the two hyperparameters discussed above, namely, the denoising step and AFS (see App. F.7 for tables with all results). We also include three competitive results from the literature [64–66] that use different checkpoints and sampling strategies: for each method, we include the best result for their respective set of hyperparameters. We do not compare in this figure with Knowledge Distillation [KD, 68], Progressive Distillation [PG, 69] and Denoising Diffusion GANs [DDGAN, 67] as they do not solve the generative ODE/SDE and use fundamentally different sampling approaches with drawbacks discussed in Sec. 4.

For NFEs $\in \{10, 15, 20, 25\}$, GENIE outperforms all baselines (on the same checkpoint) on all four datasets (GENIE image samples shown in App. F.8). On CIFAR-10 and (conditional) ImageNet, GENIE also outperforms these baselines for NFEs=5, whereas DDIM outperforms GENIE slightly on the LSUN datasets (see tables in App. F.7). GENIE performs better than the three additional baselines from the literature (which use different checkpoints and sampling strategies) with the exception of the Learned Sampler [LS, 66] on CIFAR-10 for NFEs=5 and on LSUN Bedrooms for NFEs=20. The LS's advantage can be potentially due to the different checkpoint, LS learning an optimal striding schedule (whereas GENIE simply uses quadratic striding), or LS learning additional degrees of freedom within a more general class of DDMs. For CIFAR-10, we show additional baseline results in Tab. 1. In particular, we compare to a baseline where we replace the learned derivatives in GENIE with the "ideal derivative trick" proposed in Tachibana et al. [61]. We call this method "2nd TTM (ID)" and refer to App. B.1 for details of the method. We do not include commonly-used adaptive step size solvers in Fig. 5, as they are arguably not well-suited for this low NFE regime: for example, on the same CIFAR-10 checkpoint we use for GENIE, the adaptive SDE solver introduced in Jolicoeur-Martineau et al. [62] obtains an FID of 82.4 at 48 NFEs. Also on the same checkpoint, the Runge–Kutta 4(5) [84] method applied to the ProbabilityFlow ODE obtains an FID of 13.1 at 38 NFEs (solver tolerances set to 10^{-2}).

The results in Fig. 5 suggest that higher-order gradient information, as used in GENIE, can be efficiently leveraged for image synthesis. Despite using small prediction heads our distillation seems to be sufficiently accurate: for reference, replacing the distillation heads with the derivatives computed via AD, we obtain FIDs of 9.22, 4.11, 3.54, 3.46 using 10, 20, 30, and 40 NFEs, respectively (NFEs adjusted assuming an additional computational overhead of 100%). Moreover, our distillation

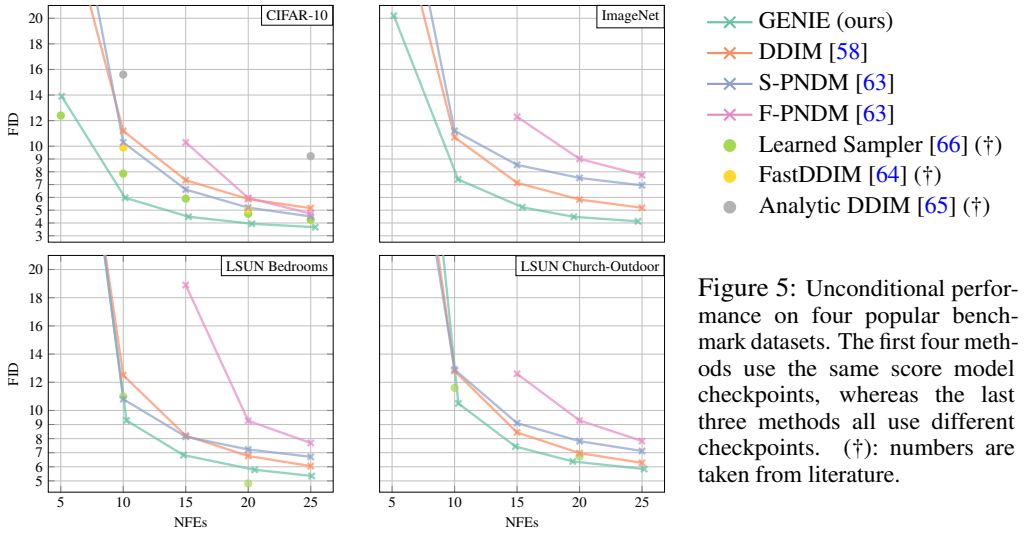


Figure 5: Unconditional performance on four popular benchmark datasets. The first four methods use the same score model checkpoints, whereas the last three methods all use different checkpoints. (†): numbers are taken from literature.

Method	NFEs=5	NFEs=10	NFEs=15	NFEs=20	NFEs=25
GENIE (ours)	13.9	**5.97**	**4.49**	**3.94**	**3.67**
DDIM [58]	29.7	11.2	7.35	5.87	5.16
S-PNDM [63]	35.9	10.3	6.61	5.20	4.51
F-PNDM [63]	N/A	N/A	10.3	5.96	4.73
2nd TTM (ID) [61]	100	49.2	29.0	19.2	14.2
Euler–Maruyama	325	230	164	112	80.3
FastDDIM [64] (†)	-	9.90	-	5.05	-
Learned Sampler [66] (†)	**12.4**	7.86	5.90	4.72	4.25
Analytic DDIM (LS) [65] (†)	-	14.0	-	-	5.71
CLD-SGM [60]	334	306	236	162	106
VESDE-PC [57]	461	461	461	461	462

Table 1: Unconditional CIFAR-10 generative performance (measured in FID). Methods above the middle line use the same score model checkpoint; methods below all use different ones. (†): numbers are taken from literature.

approach is significantly better than approximating higher-order scores using methods such as in the "ideal derivative trick" [61]. As discussed in Sec. 3, linear multistep methods such as S-PNDM [63] and F-PNDM [63] can be considered (finite difference) approximations to TTMs as used in GENIE. These approximations are inaccurate for large timesteps, which potentially explains their inferior performance when compared to GENIE. When compared to DDIM, the superior performance of GENIE seems to become less significant for large NFE: this is in line with the theory, as higher-order gradients contribute less for smaller step sizes (see the GENIE scheme in Eq. (9)). Approaches such as FastDDIM [64] and AnalyticDDIM [65], which adapt variances and discretizations of discrete-time DDMs, are useful; however, GENIE suggests that rigorous higher-order ODE solvers leveraging the continuous-time DDM formalism are still more powerful. To the best of our knowledge, the only methods that outperform GENIE abandon this ODE or SDE formulation entirely and train NFE-specific models [67, 69] which are optimized for the single use-case of image synthesis.

5.2 Guidance and Encoding

As discussed in Sec. 4, one major drawback of approaches such as KD [68], PG [69] and DDGAN [67] is that they abandon the ODE/SDE formalism, and cannot easily use methods such as classifier(-free) guidance [57, 70] or perform image encoding. However, these techniques can play an important role in synthesizing photorealistic images from DDMs [3, 4, 15, 17], as well as for image editing tasks [12, 17] and to calculate likelihoods [47, 57].

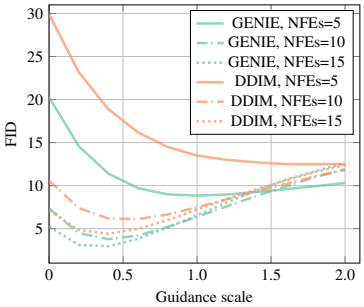


Figure 6: Sample quality as a function of guidance scale on ImageNet.

Classifier-Free Guidance [70]: We replace the unconditional model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ with $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, c, w) = (1+w)\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, c) - w\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ in the DDIM ODE (cf Eq. (6)), where $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, c)$ is a conditional model and $w > 1.0$ is the "guidance scale". GENIE then requires the derivative

$$d_{\gamma_t}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, c, w) = (1+w)d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, c) - wd_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t). \tag{16}$$

for guidance. Hence, we need to distill $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, c)$ and $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$, for which we could also share parameters [70]. We compare GENIE with DDIM on ImageNet in Fig. 6. GENIE clearly outperforms DDIM, in particular for few NFEs, and GENIE also synthesizes high-quality images (see Fig. 7).

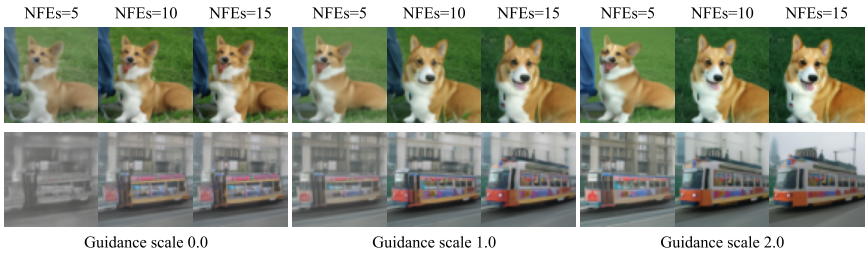


Figure 7: Classifier-free guidance for the ImageNet classes Pembroke Welsh Corgi (263) and Streetcar (829).

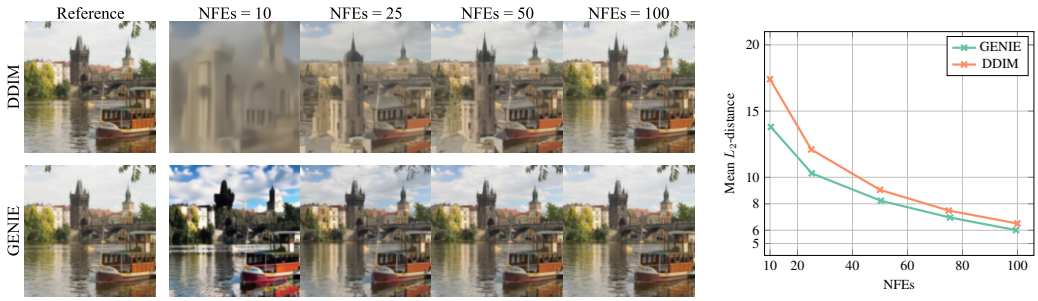


Figure 8: Encoding and subsequent decoding on LSUN Church-Outdoor. *Left:* Visual reconstruction. *Right:* L_2-distance to reference in Inception feature space [98], averaged over 100 images.

Image Encoding: We can use GENIE also to solve the generative ODE in reverse to encode given images. Therefore, we compare GENIE to DDIM on the "encode-decode" task, analyzing reconstructions for different NFEs (used twice for encoding and decoding): We find that GENIE reconstructs images much more accurately (see Fig. 8). For more details on this experiment as well as the guidance experiment above, see App. F.5 and App. F.4, respectively. We also show latent space interpolations for both GENIE and DDIM in App. F.6.

5.3 Ablation Studies

Table 2: CIFAR-10 ablation studies (measured in FID).

Ablation	NFEs=5	NFEs=10	NFEs=15	NFEs=20	NFEs=25
Standard	13.9	6.04	4.49	3.94	3.67
No mixed	14.7	6.32	4.82	4.31	4.10
No weighting	14.8	7.45	5.89	5.17	4.80
Bigger model	13.7	5.58	4.46	4.05	3.77

We perform ablation studies over architecture and training objective for the prediction heads used in GENIE: In Tab. 2, "No mixed" refers to learning $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ directly as single network output without mixed network parameterization; "No weighting" refers to setting $g_{\mathrm{d}}(t) = 1$ in Eq. (15); "Standard" uses both the mixed network parameterization and the weighting function $g_{\mathrm{d}}(t) = \gamma_t^2$. We can see that having both the mixed network parametrization and the weighting function is clearly beneficial. We also tested deeper networks in the prediction heads: for "Bigger model" we increased the number of residual blocks from one to two. The performance is roughly on par with "Standard", and we therefore opted for the smaller head due to the lower computational overhead.

6 Conclusions

We introduced GENIE, a higher-order ODE solver for DDMs. GENIE improves upon the commonly used DDIM solver by capturing the local curvature of its ODE's gradient field, which allows for larger step sizes when solving the ODE. We further propose to distill the required higher-order derivatives into a small prediction head—which we can efficiently call during inference—on top of the first-order score network. A limitation of GENIE is that it is still slightly slower than approaches that abandon the differential equation framework of DDMs altogether, which, however, comes at the considerable cost of preventing applications such as guided sampling. To overcome this limitation, in future work we plan to leverage even higher-order gradients to accelerate sampling from DDMs even further.

Broader Impact. Fast synthesis from DDMs, the goal of GENIE, can potentially make DDMs an attractive method for promising interactive generative modeling applications, such as digital content creation or real-time audio synthesis, and also reduce DDMs' environmental footprint by decreasing the computational load during inference. Although we validate GENIE on image synthesis, it could also be utilized for other tasks, which makes its broader societal impact application-dependent. In that context, it is important that practitioners apply an abundance of caution to mitigate impacts given generative modeling can also be used for malicious purposes, discussed for instance in Vaccari and Chadwick [99], Nguyen et al. [100], Mirsky and Lee [101].

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, 2020.

[2] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded Diffusion Models for High Fidelity Image Generation. *arXiv:2106.15282*, 2021.

[3] Alexander Quinn Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. In *International Conference on Machine Learning*, 2021.

[4] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Neural Information Processing Systems*, 2021.

[5] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. *arXiv:2112.10752*, 2021.

[6] Ruihan Yang, Prakhar Srivastava, and Stephan Mandt. Diffusion Probabilistic Modeling for Video Generation. *arXiv:2203.09481*, 2022.

[7] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video Diffusion Models. *arXiv:2204.03458*, 2022.

[8] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image Super-Resolution via Iterative Refinement. *arXiv:2104.07636*, 2021.

[9] Haoying Li, Yifan Yang, Meng Chang, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. SRDiff: Single Image Super-Resolution with Diffusion Probabilistic Models. *arXiv:2104.14951*, 2021.

[10] Jay Whang, Mauricio Delbracio, Hossein Talebi, Chitwan Saharia, Alexandros G. Dimakis, and Peyman Milanfar. Deblurring via Stochastic Refinement. *arXiv:2112.02475*, 2021.

[11] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising Diffusion Restoration Models. *arXiv:2201.11793*, 2022.

[12] Chenlin Meng, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Image Synthesis and Editing with Stochastic Differential Equations. *arXiv:2108.01073*, 2021.

[13] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. RePaint: Inpainting using Denoising Diffusion Probabilistic Models. *arXiv:2201.09865*, 2022.

[14] Chitwan Saharia, William Chan, Huiwen Chang, Chris A. Lee, Jonathan Ho, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Palette: Image-to-Image Diffusion Models. *arXiv:2111.05826*, 2021.

[15] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. *arXiv:2112.10741*, 2021.

[16] Omri Avrahami, Dani Lischinski, and Ohad Fried. lended Diffusion for Text-driven Editing of Natural Images. *arXiv:2111.14818*, 2021.

[17] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. *arXiv:2204.06125*, 2022.

[18] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. ILVR: Conditioning Method for Denoising Diffusion Probabilistic Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[19] Kushagra Pandey, Avideep Mukherjee, Piyush Rai, and Abhishek Kumar. DiffuseVAE: Efficient, Controllable and High-Fidelity Generation from Low-Dimensional Latents. *arXiv:2201.00308*, 2022.

[20] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion Autoencoders: Toward a Meaningful and Decodable Representation. *arXiv:2111.15640*, 2021.

[21] Xihui Liu, Dong Huk Park, Samaneh Azadi, Gong Zhang, Arman Chopikyan, Yuxiao Hu, Humphrey Shi, Anna Rohrbach, and Trevor Darrell. More Control for Free! Image Synthesis with Semantic Diffusion Guidance. *arXiv:2112.05744*, 2021.

[22] Georgios Batzolis, Jan Stanczuk, Carola-Bibiane Schönlieb, and Christian Etmann. Conditional Image Generation with Score-Based Diffusion Models. *arXiv:2111.13606*, 2021.

[23] Hiroshi Sasaki, Chris G. Willcocks, and Toby P. Breckon. UNIT-DDPM: UNpaired Image Translation with Denoising Diffusion Probabilistic Models. *arXiv:2104.05358*, 2021.

[24] Xuan Su, Jiaming Song, Chenlin Meng, and Stefano Ermon. Dual Diffusion Implicit Bridges for Image-to-Image Translation. *arXiv:2203.08382*, 2022.

[25] Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. Solving Inverse Problems in Medical Imaging with Score-Based Generative Models. In *International Conference on Learning Representations*, 2022.

[26] Cheng Peng, Pengfei Guo, S. Kevin Zhou, Vishal Patel, and Rama Chellappa. Towards performant and reliable undersampled MR reconstruction via diffusion model sampling. *arXiv:2203.04292*, 2022.

[27] Yutong Xie and Quanzheng Li. Measurement-conditioned Denoising Diffusion Probabilistic Model for Under-sampled Medical Image Reconstruction. *arXiv:2203.03623*, 2022.

[28] Guanxiong Luo, Martin Heide, and Martin Uecker. MRI Reconstruction via Data Driven Markov Chain with Joint Uncertainty Estimation. *arXiv:2202.01479*, 2022.

[29] Hyungjin Chung and Jong Chul Ye. Score-based diffusion models for accelerated MRI. *arXiv:2110.05243*, 2021.

[30] Dewei Hu, Yuankai K. Tao, and Ipek Oguz. Unsupervised Denoising of Retinal OCT with Diffusion Probabilistic Model. *arXiv:2201.11760*, 2022.

[31] Hyungjin Chung, Byeongsu Sim, and Jong Chul Ye. Come-Closer-Diffuse-Faster: Accelerating Conditional Diffusion Models for Inverse Problems through Stochastic Contraction. *arXiv:2112.05146*, 2021.

[32] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. WaveGrad: Estimating Gradients for Waveform Generation. In *International Conference on Learning Representations*, 2021.

[33] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A Versatile Diffusion Model for Audio Synthesis. In *International Conference on Learning Representations*, 2021.

[34] Myeonghun Jeong, Hyeongju Kim, Sung Jun Cheon, Byoung Jin Choi, and Nam Soo Kim. Diff-TTS: A Denoising Diffusion Model for Text-to-Speech. *arXiv preprint arXiv:2104.01409*, 2021.

[35] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J. Weiss, Mohammad Norouzi, Najim Dehak, and William Chan. WaveGrad 2: Iterative Refinement for Text-to-Speech Synthesis. *arXiv:2106.09660*, 2021.

[36] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech. In *International Conference on Machine Learning*, 2021.

[37] Songxiang Liu, Dan Su, and Dong Yu. DiffGAN-TTS: High-Fidelity and Efficient Text-to-Speech with Denoising Diffusion GANs. *arXiv:2201.11972*, 2022.

[38] Shitong Luo and Wei Hu. Diffusion Probabilistic Models for 3D Point Cloud Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[39] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning Gradient Fields for Shape Generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[40] Linqi Zhou, Yilun Du, and Jiajun Wu. 3D Shape Generation and Completion through Point-Voxel Diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[41] Zhaoyang Lyu, Zhifeng Kong, Xudong XU, Liang Pan, and Dahua Lin. A Conditional Point Diffusion-Refinement Paradigm for 3D Point Cloud Completion. In *International Conference on Learning Representations*, 2022.

[42] Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you should treat it like one. *arXiv:2201.12204*, 2022.

[43] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation. In *International Conference on Machine Learning*, 2021.

[44] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation. In *International Conference on Learning Representations*, 2022.

[45] Tian Xie, Xiang Fu, Octavian-Eugen Ganea, Regina Barzilay, and Tommi S. Jaakkola. Crystal Diffusion Variational Autoencoder for Periodic Material Generation. In *International Conference on Learning Representations*, 2022.

[46] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based Generative Modeling of Graphs via the System of Stochastic Differential Equations. *arXiv:2202.02514*, 2022.

[47] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum Likelihood Training of Score-Based Diffusion Models. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[48] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational Diffusion Models. In *Advances in Neural Information Processing Systems*, 2021.

[49] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based Generative Modeling in Latent Space. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[50] Chin-Wei Huang, Jae Hyun Lim, and Aaron Courville. A Variational Perspective on Diffusion-Based Generative Models and Score Matching. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[51] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[52] Jongmin Yoon, Sung Ju Hwang, and Juho Lee. Adversarial Purification with Score-based Generative Models. In *International Conference on Machine Learning*, 2021.

[53] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation. In *Advances in Neural Information Processing Systems*, 2021.

[54] Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian Score-Based Generative Modeling. *arXiv:2202.02763*, 2022.

[55] Pedro Sanchez and Sotirios A. Tsaftaris. Diffusion Causal Models for Counterfactual Estimation. In *First Conference on Causal Learning and Reasoning*, 2022.

[56] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion Models for Adversarial Purification. In *International Conference on Machine Learning (ICML)*, 2022.

[57] 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.

[58] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*, 2021.

[59] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, Berlin, 1992.

[60] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-Based Generative Modeling with Critically-Damped Langevin Diffusion. In *International Conference on Learning Representations*, 2022.

[61] Hideyuki Tachibana, Mocho Go, Muneyoshi Inahara, Yotaro Katayama, and Yotaro Watanabe. Itô-Taylor Sampling Scheme for Denoising Diffusion Probabilistic Models using Ideal Derivatives. *arXiv:2112.13339*, 2021.

[62] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta Go Fast When Generating Data with Score-Based Models. *arXiv:2105.14080*, 2021.

[63] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo Numerical Methods for Diffusion Models on Manifolds. In *International Conference on Learning Representations*, 2022.

[64] Zhifeng Kong and Wei Ping. On Fast Sampling of Diffusion Probabilistic Models. *arXiv:2106.00132*, 2021.

[65] 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*, 2022.

[66] 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*, 2022.

[67] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the Generative Learning Trilemma with Denoising Diffusion GANs. In *International Conference on Learning Representations*, 2022.

[68] Eric Luhman and Troy Luhman. Knowledge Distillation in Iterative Generative Models for Improved Sampling Speed. *arXiv:2101.02388*, 2021.

[69] Tim Salimans and Jonathan Ho. Progressive Distillation for Fast Sampling of Diffusion Models. In *International Conference on Learning Representations*, 2022.

[70] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[71] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, 2015.

[72] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[73] Ulrich G Haussmann and Etienne Pardoux. Time Reversal of Diffusions. *The Annals of Probability*, pages 1188–1205, 1986.

[74] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems*, 2018.

[75] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. *International Conference on Learning Representations*, 2019.

[76] Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[77] John Charles Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 2016.

[78] Peter E Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, 1992.

[79] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[80] Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception Prioritized Training of Diffusion Models. *arXiv:2204.00227*, 2022.

[81] Dmitry Baranchuk, Andrey Voynov, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Label-Efficient Semantic Segmentation with Diffusion Models. In *International Conference on Learning Representations*, 2022.

[82] Chenlin Meng, Yang Song, Wenzhe Li, and Stefano Ermon. Estimating High Order Gradients of the Data Distribution by Denoising. *Advances in Neural Information Processing Systems*, 34, 2021.

[83] Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. Learning to Efficiently Sample from Diffusion Probabilistic Models. *arXiv:2106.03802*, 2021.

[84] J. R. Dormand and P. J. Prince. A family of embedded Runge–Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

[85] Eliya Nachmani, Robin San Roman, and Lior Wolf. Non Gaussian Denoising Diffusion Models. *arXiv:2106.07582*, 2021.

[86] Max WY Lam, Jun Wang, Rongjie Huang, Dan Su, and Dong Yu. Bilateral Denoising Diffusion Models. *arXiv:2108.11514*, 2021.

[87] Franck Djeumou, Cyrus Neary, Eric Goubault, Sylvie Putot, and Ufuk Topcu. Taylor-Lagrange Neural Ordinary Differential Equations: Toward Fast Training and Evaluation of Neural ODEs. *arXiv:2201.05715*, 2022.

[88] Jacob Kelly, Jesse Bettencourt, Matthew J Johnson, and David K Duvenaud. Learning differential equations that are easy to solve. *Advances in Neural Information Processing Systems*, 33:4370–4380, 2020.

[89] Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ODE: the world of Jacobian and kinetic regularization. In *International conference on machine learning*, pages 3154–3164. PMLR, 2020.

[90] YF Chang and George Corliss. ATOMFT: solving ODEs and DAEs using Taylor series. *Computers & Mathematics with Applications*, 28(10-12):209–233, 1994.

[91] George Corliss and YF Chang. Solving ordinary differential equations using Taylor series. *ACM Transactions on Mathematical Software (TOMS)*, 8(2):114–144, 1982.

[92] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

[93] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *arXiv preprint arXiv:1506.03365*, 2015.

[94] 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. Ieee, 2009.

[95] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*, 2019.

[96] 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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[97] Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Ioannis Mitliagkas, and Remi Tachet des Combes. Adversarial score matching and improved sampling for image generation. In *International Conference on Learning Representations*, 2021.

[98] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[99] Cristian Vaccari and Andrew Chadwick. Deepfakes and Disinformation: Exploring the Impact of Synthetic Political Video on Deception, Uncertainty, and Trust in News. *Social Media + Society*, 6(1): 2056305120903408, 2020.

[100] Thanh Thi Nguyen, Quoc Viet Hung Nguyen, Cuong M. Nguyen, Dung Nguyen, Duc Thanh Nguyen, and Saeid Nahavandi. Deep Learning for Deepfakes Creation and Detection: A Survey. *arXiv:1909.11573*, 2021.

[101] Yisroel Mirsky and Wenke Lee. The Creation and Detection of Deepfakes: A Survey. *ACM Comput. Surv.*, 54(1), 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Sec. 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We did not derive novel theoretical results. We rather propose a novel higher-order solver for sampling from denoising diffusion generative models.
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will release code and instructions to reproduce all experiments upon acceptance of the manuscript. The internal guidelines of our institution prevent us from releasing code at this stage.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provide all model training and evaluation details in the Appendix, including all hyperparameters.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Following standard conventions in the related generative modeling literature, we do not report error bars. Furthermore, we avoid running similar setups repeatedly to save computational resources.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see App. F.3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] On CIFAR-10, we use a checkpoint provided by Song et al. [57]. For baseline comparisons, we run publicly available code: in particular, we make use of the repositories `https://github.com/yang-song/score_sde_pytorch` and `https://github.com/nv-tlabs/CLD-SGM` from previous, publicly available papers, which we cite (Song et al. [57] and Dockhorn et al. [60], respectively). We also cite all used datasets: CIFAR-10 [92], LSUN Bedrooms [93], LSUN Church-Outdoor [93], and ImageNet [94].
   (b) Did you mention the license of the assets? [Yes] In the Appendix, we mention the licenses of all codes we are running for baseline comparisons and evaluation. See App. F.1.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We are only using publicly available datasets, which have been widely used in the generative modeling literature as standard benchmarks (CIFAR-10, ImageNet, LSUN Church-Outdoor, LSUN Bedrooms) and avoid using face image datasets.
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]