GENIE: Higher-Order Denoising Diffusion Solvers

Tim Dockhorn[1,2,3,*] **Arash Vahdat**[1] **Karsten Kreis**[1]

[1]NVIDIA [2]University of Waterloo [3]Vector Institute

tim.dockhorn@uwaterloo.ca, {avahdat,kkreis}@nvidia.com

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 previous 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. Project page and code: https://nv-tlabs.github.io/GENIE.

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

[*]Work done during internship at NVIDIA.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

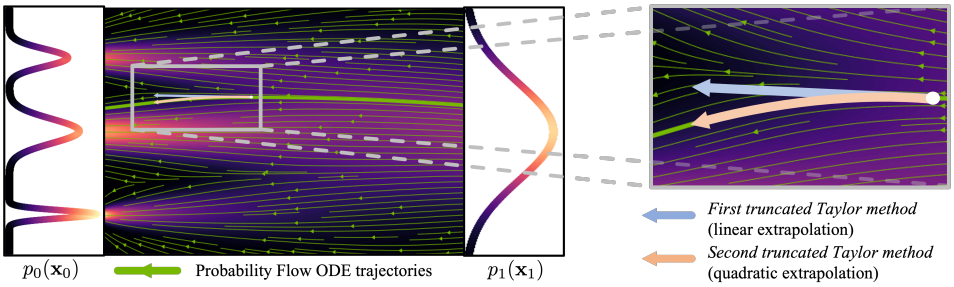


Figure 1: Our novel *Higher-Order Denoising Diffusion Solver* (GENIE) relies on the second *truncated Taylor method* (TTM) to simulate a (re-parametrized) Probability Flow ODE for sampling from denoising diffusion models. The second TTM captures the local curvature of the ODE's gradient field and enables 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 previous 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 $h_n = t_{n+1} - t_n$ (see App. B.1 for a truncation error analysis with respect to the exact ODE solution). Note that the first TTM is simply Euler's method. Applying the second TTM to the DDIM ODE results in the following scheme:

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

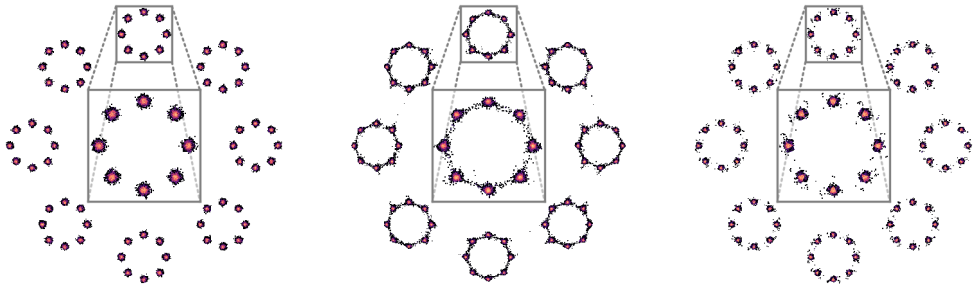
where $h_n = \gamma_{t_{n+1}} - \gamma_{t_n}$. Recall that $\gamma_t = \sqrt{\frac{1-\alpha_t^2}{\alpha_t^2}}$, where the function α_t is a time-dependent hyperparameter of the DDM. The total derivative $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}} := \frac{d\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{d\gamma_t}$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

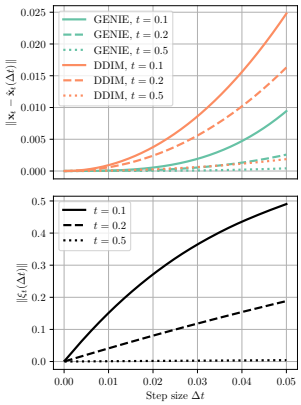


Figure 3: *Top*: Single step error using analytical score function. *Bottom*: Norm of difference $\xi_t(\Delta t)$ between analytical and approximate derivative computed via finite difference method.

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:

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

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 $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ network to compute $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ 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 $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ 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 $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ into a separate neural network. During distillation training, we can use the slow AD-based calculation of $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$, but during synthesis we call the trained neural network. We build on the observation that the internal representations of the neural network modeling $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ (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 $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ network together with its time embedding as well as $\mathbf{x}_t$ and the output $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ to a small prediction head $\boldsymbol{k}_{\boldsymbol{\psi}}(\mathbf{x}_t, t)$ that models the different terms in Eq. (12) (see Fig. 4). The overhead generated by $\boldsymbol{k}_{\boldsymbol{\psi}}$ 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 $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ and could therefore theoretically be run in parallel to the $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ 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 $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ network, then calculate the JVP terms, and only then can we call the distillation model. By modeling $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ via small prediction heads, while reusing the internal representation of the score model, we can make training relatively fast: we only need to train $\boldsymbol{k}_{\boldsymbol{\psi}}$ 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 $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ to future work.

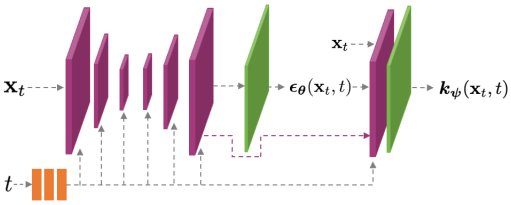


Figure 4: Our distilled model $\boldsymbol{k}_{\boldsymbol{\psi}}$ that predicts the gradient $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is implemented as a small additional output head on top of the first-order score model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$. Purple layers are used both in $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ and $\boldsymbol{k}_{\boldsymbol{\psi}}$; green layers are specific for $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ and $\boldsymbol{k}_{\boldsymbol{\psi}}$.

Mixed Network Parameterization: We found that learning $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ 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

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

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 $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ 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: Ideally, we would like our model $\boldsymbol{k}_{\boldsymbol{\psi}}$ to match $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ exactly, for all $t \in [0, T]$ and $\mathbf{x}_t$ in the diffused data distribution, which the generative ODE trajectories traverse. This suggests a simple (weighted) L_2-loss, similar to regular score matching losses for DDMs [1, 57]:

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

for diffused data points $\alpha_t\mathbf{x}_0 + \sigma_t\boldsymbol{\epsilon}$ and $g_{\text{d}}(t) = \gamma_t^2$ to counteract the $1/\gamma_t$ in the first and third terms of Eq. (14). This leads to a roughly constant loss over different time values t. During training we compute $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ via AD; however, at inference time we use the learned prediction head $\boldsymbol{k}_{\boldsymbol{\psi}}$ to approximate $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$. In App. C.2.4, we provide pseudo code for training and sampling with heads $\boldsymbol{k}_{\boldsymbol{\psi}}$. Note that our distillation objective is consistent and principled: if $\boldsymbol{k}_{\psi}$ matches $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ exactly, the resulting GENIE algorithm recovers the second TTM *exactly* (extended discussion in App. B.4).

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 $\nabla_{\mathbf{x}_t}^{\top}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$, 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, they propose to use an "ideal derivative trick" to approximate higher-order score functions. In App. B.2, we show that applying this ideal derivative approximation to the DDIM ODE does not have any effect: the "ideal derivatives" are zero by construction. Note that in GENIE, we in fact use the DDIM ODE, rather than, for example, the regular Probability Flow ODE [57], as the base ODE for GENIE.

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.

Concurrent Works. Zhang and Chen [92] motivate the DDIM ODE from an exponential integrator perspective applied to the Probability Flow ODE and propose to apply existing solvers from the numerical ODE literature, namely, Runge–Kutta and linear multistepping, to the DDIM ODE directly. Lu et al. [93] similarly recognize the semi-linear structure of the Probability Flow ODE, derive dedicated solvers, and introduce new step size schedulers to accelerate DDM sampling. Karras et al. [94] propose new fast solvers, both deterministic and stochastic, specifically designed for the differential equations arising in DDMs. Both Zhang et al. [95] and Karras et al. [94] realize that the DDIM ODE has "straight line solution trajectories" for spherical normal data and single data points—this exactly corresponds to our derivation that the higher-order terms in the DDIM ODE are zero in such a setting (see App. B.2). Bao et al. [96] learn covariance matrices for DDM sampling using prediction heads somewhat similar to the ones in GENIE; in App. G.1, we thoroughly discuss the differences between GENIE and the method proposed in Bao et al. [96].

5 Experiments

Datasets: We run experiments on five datasets: CIFAR-10 [97] (resolution 32), LSUN Bedrooms [98] (128), LSUN Church-Outdoor [98] (128), (conditional) ImageNet [99] (64), and AFHQv2 [100] (512). On AFHQv2 we only consider the subset of cats; referred to as "Cats" in the remainder of this work.

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, 101]. 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, 102] (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, 103]. 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 $\boldsymbol{k}_{\boldsymbol{\psi}}$. 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.6 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 (see detailed results in App. F.6 and GENIE image samples in App. F.7). 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.6). GENIE also 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 LSUN Bedrooms for NFEs=20. Though LS uses a learned striding schedule on LSUN Bedrooms (whereas GENIE simply uses quadratic striding), the LS's advantage is most likely due to the different checkpoint. In Tab. 1, we investigate the effect of optimizing the striding schedule, via learning (LS) or grid search (DDIM & GENIE), on CIFAR-10 and find that its significance decreases rapidly with increased NFEs (also see App. F.6 for details). In Tab. 1, we also show additional baseline results; however, 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 adaptive Runge–Kutta 4(5) [84] method applied to the ProbabilityFlow ODE achieves an FID of 13.1 at 38 NFEs (solver tolerances set to 10^{-2}).

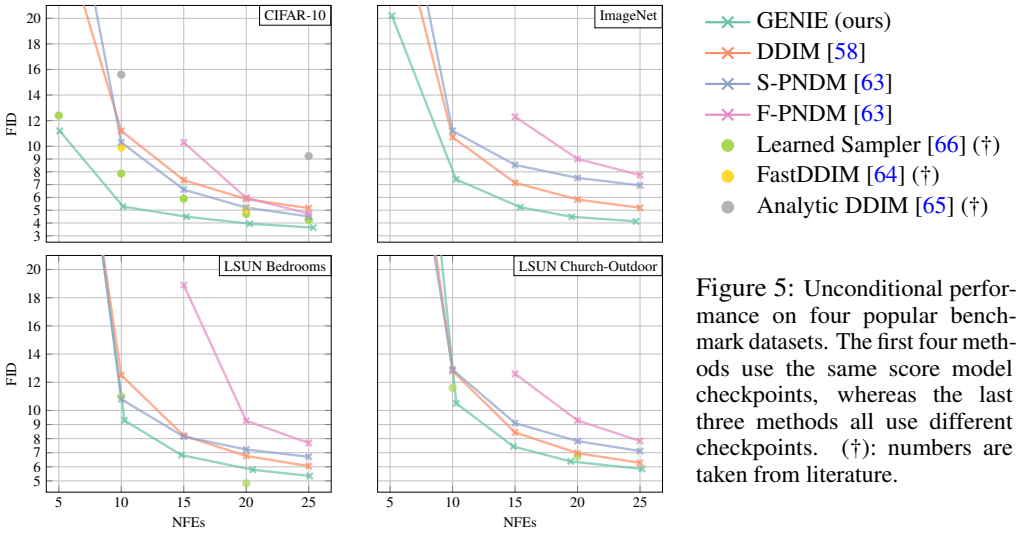


Figure 5: Unconditional performance on four popular benchmark datasets. The first four methods use the same score model checkpoints, whereas the last three methods all use different checkpoints. (†): numbers are taken from literature.

Method	NFEs=5	NFEs=10	NFEs=15	NFEs=20	NFEs=25
GENIE (ours) (∗)	**11.2**	**5.28**	**4.49**	**3.94**	**3.64**
GENIE (ours)	13.9	5.97	**4.49**	**3.94**	3.67
DDIM [58] (∗)	27.6	11.2	7.35	5.87	5.16
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
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
Learned Sampler [66] (†)	14.3	8.15	5.94	4.89	4.47
Analytic DDIM [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. (∗): methods either learn an optimal striding schedule (Learned Sampler) or do a small grid search over striding schedules (DDIM & GENIE); also see App. F.6

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%). 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 can be 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].

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

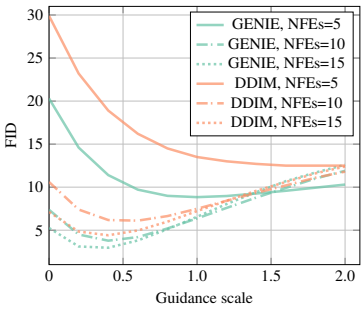


Figure 6: Sample quality as a function of guidance scale on ImageNet.

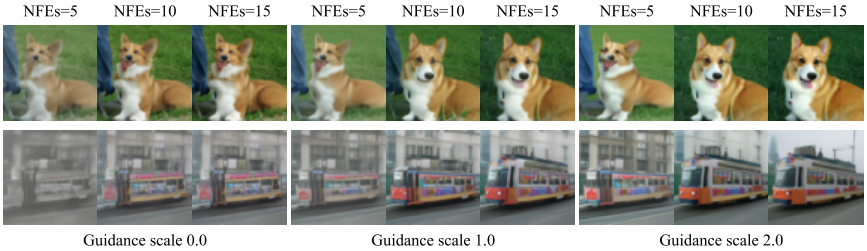


Figure 7: Classifier-free guidance for the ImageNet classes Pembroke Welsh Corgi (263) and Streetcar (829).

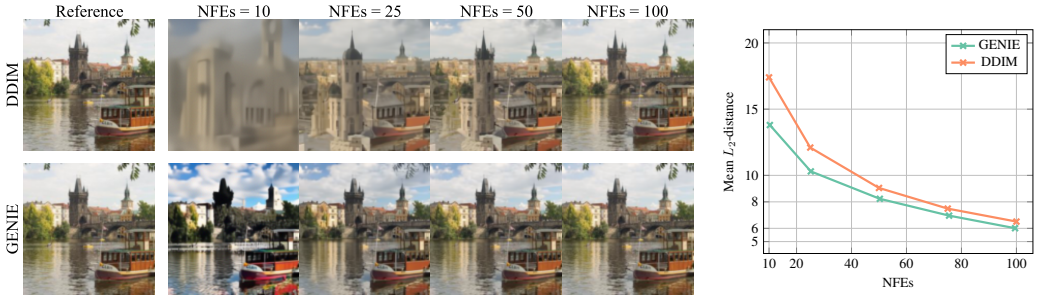


Figure 8: Encoding and subsequent decoding on LSUN Church-Outdoor. *Left:* Visual reconstruction. *Right:* L_2-distance to reference in Inception feature space [104], averaged over 100 images.

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

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).

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.4 and App. F.3, respectively. We also show latent space interpolations for both GENIE and DDIM in App. F.5.

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.

5.4 Upsampling

Table 3: Cats (upsampler) generative performance (measured in FID).

Method	NFEs=5	NFEs=10	NFEs=15
GENIE (ours)	**5.53**	**4.90**	**4.83**
DDIM [58]	9.47	6.64	5.85
S-PNDM [63]	14.6	11.0	8.83
F-PNDM [63]	N/A	N/A	11.7

Cascaded diffusion model pipelines [2] and DDM-based super-resolution [8] have become crucial ingredients in DDMs for large-scale image generation [105]. Hence, we also explore the applicability of GENIE in this setting. We train a 128×128 base model as well as a $128 \times 128 \rightarrow 512 \times 512$ diffusion upsampler [2, 8] on Cats. In Tab. 3, we compare the generative performance of GENIE to other fast samplers for the upsampler (in isolation). We find that GENIE performs very well on this task: with only five NFEs GENIE outperforms all other methods at NFEs=15. We show upsampled samples for GENIE with NFEs=5 in Fig. 9. For more quantitative and qualitative results, we refer to App. F.6 and App. F.7, respectively. Training and inference details for the score model and the GENIE prediction head, for both base model and upsampler, can be found in App. C.

Figure 9: High-resolution images generated with the $128 \times 128 \rightarrow 512 \times 512$ GENIE upsampler using only five neural network calls. For the two images at the top, the upsampler is conditioned on test images from the Cats dataset. For the two images at the bottom, the upsampler is conditioned on samples from the 128×128 GENIE base model (generated using 25 NFEs); an upsampler neural network evaluation is roughly four times as expensive as a base model evaluation.

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, future work could leverage even higher-order gradients to accelerate sampling from DDMs even further (also see App. G.2).

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 [106], Nguyen et al. [107], Mirsky and Lee [108].

Acknowledgements

This work was funded by NVIDIA. Tim Dockhorn acknowledges additional funding from the Vector Institute Research Grant, not in direct support of this work. We thank Yaoliang Yu for discussions.

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] Qinsheng Zhang and Yongxin Chen. Fast Sampling of Diffusion Models with Exponential Integrator. *arXiv:2204.13902*, 2022.

[93] 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:2206.00927*, 2022.

[94] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. *arXiv:2206.00364*, 2022.

[95] Qinsheng Zhang, Molei Tao, and Yongxin Chen. gDDIM: Generalized denoising diffusion implicit models. *arXiv:2206.05564*, 2022.

[96] Fan Bao, Chongxuan Li, Jiacheng Sun, Jun Zhu, and Bo Zhang. Estimating the Optimal Covariance with Imperfect Mean in Diffusion Probabilistic Models. *arXiv:2206.07309*, 2022.

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

[98] 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.

[99] 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.

[100] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse Image Synthesis for Multiple Domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8188–8197, 2020.

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

[102] 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.

[103] 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.

[104] 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.

[105] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. *arXiv:2205.11487*, 2022.

[106] 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.

[107] 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.

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

[109] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[110] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

[111] Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing Generative Models via Precision and Recall. *Advances in Neural Information Processing Systems*, 31, 2018.

[112] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved Precision and Recall Metric for Assessing Generative Models. *Advances in Neural Information Processing Systems*, 32, 2019.

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 provide code and instructions to reproduce all experiments in the near future.
   (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.8.
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 [97], LSUN Bedrooms [98], LSUN Church-Outdoor [98], ImageNet [99], and AFHQv2 [100].
   (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, AFHQv2) 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]

Contents

1 Introduction **1**

2 Background **2**

3 Higher-Order Denoising Diffusion Solver **3**

3.1 Learning Higher-Order Derivatives 4

4 Related Work **6**

5 Experiments **7**

5.1 Image Generation 7

5.2 Guidance and Encoding 8

5.3 Ablation Studies 9

5.4 Upsampling 9

6 Conclusions **10**

References **10**

A DDIM ODE **20**

B Synthesis from Denoising Diffusion Models via Truncated Taylor Methods **21**

B.1 Theoretical Bounds for the Truncated Taylor Method 21

B.2 Approximate Higher-Order Derivatives via the "Ideal Derivative Trick" 21

B.3 3rd TTM Applied to the DDIM ODE 23

B.4 GENIE is Consistent and Principled 24

C Model and Implementation Details **27**

C.1 Score Models 27

C.2 Prediction Heads 27

C.2.1 Model Architecture 28

C.2.2 Training Details 28

C.2.3 Mixed Network Parameterization 28

C.2.4 Pseudocode 29

C.2.5 Measuring Computational Overhead 30

D Learning Higher-Order Gradients without Automatic Differentiation and Distillation **31**

E Toy Experiments **32**

F Image Experiments **34**

F.1 Evaluation Metrics, Baselines, and Datasets 34

F.2 Analytical First Step (AFS) 34

F.3 Classifier-Free Guidance 34
F.4 Encoding 35
F.5 Latent Space Interpolation 35
F.6 Extended Quantitative Results 35
F.7 Extended Qualitative Results 38
F.8 Computational Resources 38

G Miscellaneous **46**
G.1 Connection to Bao et al. [96] 46
G.2 Combining GENIE with Progressive Distillation 47

A DDIM ODE

The DDIM ODE has previously been shown [58, 69] to be a re-parameterization of the Probability Flow ODE [57]. In this section, we show an alternative presentation to the ones given in Song et al. [58] and Salimans and Ho [69]. We start from the Probability Flow ODE for variance-preserving continuous-time DDMs [57], i.e.,

$$d\mathbf{x}_t = -\tfrac{1}{2}\beta_t \left[\mathbf{x}_t + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] dt, \tag{17}$$

where $\beta_t = -\frac{d}{dt}\log \alpha_t^2$ and $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is the *score function*. Replacing the unknown score function with a learned score model $\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$, we obtain the approximate Probability Flow ODE

$$d\mathbf{x}_t = -\tfrac{1}{2}\beta_t \left[\mathbf{x}_t + \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right] dt. \tag{18}$$

Let us now define $\gamma_t = \sqrt{\frac{1-\alpha_t^2}{\alpha_t^2}}$ and $\bar{\mathbf{x}}_t = \mathbf{x}_t\sqrt{1+\gamma_t^2}$, and take the (total) derivative of $\bar{\mathbf{x}}_t$ with respect to γ_t:

$$\frac{d\bar{\mathbf{x}}_t}{d\gamma_t} = \frac{\partial \bar{\mathbf{x}}_t}{\partial \mathbf{x}_t}\frac{d\mathbf{x}_t}{d\gamma_t} + \frac{\partial \bar{\mathbf{x}}_t}{\partial \gamma_t} \tag{19}$$

$$= \sqrt{1+\gamma_t^2}\frac{d\mathbf{x}_t}{d\gamma_t} + \frac{\gamma_t}{\sqrt{1+\gamma_t^2}}\mathbf{x}_t. \tag{20}$$

The derivative $\frac{d\mathbf{x}_t}{d\gamma_t}$ can be computed as follows

$$\frac{d\mathbf{x}_t}{d\gamma_t} = \frac{d\mathbf{x}_t}{dt}\frac{dt}{d\gamma_t} \quad \text{(by chain rule)} \tag{21}$$

$$= -\frac{1}{2}\beta_t \left[\mathbf{x}_t + \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\frac{dt}{d\gamma_t} \quad \text{(inserting Eq. (18))} \tag{22}$$

$$= \frac{1}{2}\frac{d\log\alpha_t^2}{dt}\left[\mathbf{x}_t + \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\frac{dt}{d\gamma_t} \quad \text{(by definition of } \beta_t\text{)} \tag{23}$$

$$= \frac{1}{2}\frac{d\log\alpha_t^2}{d\gamma_t}\left[\mathbf{x}_t + \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right] \quad \text{(by chain rule)} \tag{24}$$

$$= \frac{1}{2}\frac{d\log\alpha_t^2}{d\alpha_t^2}\frac{d\alpha_t^2}{d\gamma_t}\left[\mathbf{x}_t + \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right] \quad \text{(by chain rule)} \tag{25}$$

$$= \frac{1}{2}\frac{1}{\alpha_t^2}\frac{d\alpha_t^2}{d\gamma_t}\left[\mathbf{x}_t + \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]. \tag{26}$$

We can write α_t^2 as a function of γ_t, i.e., $\alpha_t^2 = \left(\gamma_t^2+1\right)^{-1}$, and therefore

$$\frac{d\alpha_t^2}{d\gamma_t} = -\frac{2\gamma_t}{\left(\gamma_t^2+1\right)^2}. \tag{27}$$

Inserting Eq. (27) into Eq. (26), we obtain

$$\frac{d\mathbf{x}_t}{d\gamma_t} = -\frac{\gamma_t}{\gamma_t^2+1}\left[\mathbf{x}_t + \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]. \tag{28}$$

Lastly, inserting Eq. (28) into Eq. (20), we have

$$\frac{d\bar{\mathbf{x}}_t}{d\gamma_t} = -\frac{\gamma_t}{\sqrt{\gamma_t^2+1}}\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \tag{29}$$

Letting $\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \coloneqq -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t}$, where $\sigma_t = \sqrt{1-\alpha_t^2} = \frac{\gamma_t}{\sqrt{\gamma_t^2+1}}$, denote a particular parameterization of the score model, we obtain the approximate generative DDIM ODE as

$$\frac{d\bar{\mathbf{x}}_t}{d\gamma_t} = \frac{\gamma_t}{\sqrt{\gamma_t^2+1}}\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t} \tag{30}$$

$$= \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t). \tag{31}$$

B Synthesis from Denoising Diffusion Models via Truncated Taylor Methods

In this work, we propose Higher-Order Denoising Diffusion Solvers (GENIE). GENIE is based on the *truncated Taylor method* (TTM) [78]. As outlined in Sec. 3, 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

$$\mathbf{y}_{t_{n+1}} = \mathbf{y}_{t_n} + h_n \frac{d\mathbf{y}}{dt}|_{(\mathbf{y}_{t_n}, t_n)} + \cdots + \frac{1}{p!} h_n^p \frac{d^p\mathbf{y}}{dt^p}|_{(\mathbf{y}_{t_n}, t_n)}, \tag{32}$$

where $h_n = t_{n+1} - t_n$. To generate samples from denoising diffusion models, we can, for example, apply the second TTM to the (approximate) Probability Flow ODE or the (approximate) DDIM ODE, resulting in the following respective schemes:

$$\mathbf{x}_{t_{n+1}} = \mathbf{x}_{t_n} + (t_{n+1} - t_n)\boldsymbol{f}(\mathbf{x}_{t_n}, t_n) + \frac{1}{2}(t_{n+1} - t_n)^2 \frac{d\boldsymbol{f}}{dt}|_{(\mathbf{x}_{t_n}, t_n)}, \tag{33}$$

where $\boldsymbol{f}(\mathbf{x}_t, t) = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t - \frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t}\right]$, and

$$\bar{\mathbf{x}}_{t_{n+1}} = \bar{\mathbf{x}}_{t_n} + (\gamma_{t_{n+1}} - \gamma_{t_n})\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) + \frac{1}{2}(\gamma_{t_{n+1}} - \gamma_{t_n})^2 \frac{d\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{d\gamma_t}|_{(\mathbf{x}_{t_n}, t_n)}. \tag{34}$$

In this work, we generate samples from DDMs using the scheme in Eq. (34). We distill the derivative $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}} := \frac{d\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{d\gamma_t}$ into a small neural network $\boldsymbol{k}_{\boldsymbol{\psi}}$. For training, $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is computed via automatic differentiation, however, during inference, we can efficiently query the trained network $\boldsymbol{k}_{\boldsymbol{\psi}}$.

B.1 Theoretical Bounds for the Truncated Taylor Method

Consider the p-TTM for a general ODE $\frac{d\mathbf{y}}{dt} = \boldsymbol{f}(\mathbf{y}, t)$:

$$\mathbf{y}_{t_{n+1}} = \mathbf{y}_{t_n} + h_n \frac{d\mathbf{y}}{dt}|_{(\mathbf{y}_{t_n}, t_n)} + \cdots + \frac{1}{p!} h_n^p \frac{d^p\mathbf{y}}{dt^p}|_{(\mathbf{y}_{t_n}, t_n)}. \tag{35}$$

We represent, the exact solution $\mathbf{y}(t_{n+1})$ using the $(p+2)$-th Taylor expansion

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + h_n \frac{d\mathbf{y}}{dt}|_{(\mathbf{y}_{t_n}, t_n)} + \cdots + \frac{1}{p!} h_n^p \frac{d^p\mathbf{y}}{dt^p}|_{(\mathbf{y}_{t_n}, t_n)} + \frac{1}{(p+1)!} h_n^{p+1} \frac{d^{p+1}\mathbf{y}}{dt^{p+1}}|_{(\mathbf{y}_{t_n}, t_n)} + \mathcal{O}(h_n^{p+2}). \tag{36}$$

The local truncation error (LTE) introduced by the p-th TTM is given by the difference between the two equations above

$$\|\mathbf{y}_{t_{n+1}} - \mathbf{y}(t_{n+1})\| = \|\frac{1}{(p+1)!} h_n^{p+1} \frac{d^{p+1}\mathbf{y}}{dt^{p+1}}|_{(\mathbf{y}_{t_n}, t_n)} + \mathcal{O}(h_n^{p+2})\|. \tag{37}$$

For small h_n, the LTE is proportional to h_n^{p+1}. Consequently, using higher orders p implies lower errors, as h_n usually is a small time step.

In conclusion, this demonstrates that it is preferable to use higher-order methods with lower errors when aiming to accurately solve ODEs like the Probability Flow ODE or the DDIM ODE of diffusion models.

B.2 Approximate Higher-Order Derivatives via the "Ideal Derivative Trick"

Tachibana et al. [61] sample from DDMs using (an approximation to) a higher-order Itô-Taylor method [59]. In their scheme, they approximate higher-order score functions with the "ideal derivative trick", essentially assuming simple single-point ($\mathbf{x}_0$) data distributions, for which higher-order score functions can be computed analytically (more formally, their approximation corresponds to ignoring the expectation over the full data distribution when learning the score function. They assume that for any $\mathbf{x}_t$, there is a single unique $\mathbf{x}_0$ from the input data to be predicted with the score model). In that case, further assuming the score model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ is learnt perfectly (i.e., it perfectly predicts the noise that was used to generate $\mathbf{x}_t$ from $\mathbf{x}_0$), one has

$$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \frac{\mathbf{x}_t - \alpha_t \mathbf{x}_0}{\sigma_t}. \tag{38}$$

This expression can now be used to analytically calculate approximate spatial and time derivatives (also see App. F.1 and App. F.2 in Tachibana et al. [61]):

$$\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \approx \frac{\partial}{\partial \mathbf{x}_t}\left(\frac{\mathbf{x}_t - \alpha_t \mathbf{x}_0}{\sigma_t}\right) = \frac{1}{\sigma_t}\boldsymbol{I}, \tag{39}$$

and

$$\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t} \approx \frac{\partial}{\partial t}\left(\frac{\mathbf{x}_t - \alpha_t \mathbf{x}_0}{\sigma_t}\right) = -\frac{\mathbf{x}_t - \alpha_t \mathbf{x}_0}{\sigma_t^2}\frac{d\sigma_t}{dt} - \frac{\mathbf{x}_0}{\sigma_t}\frac{d\alpha_t}{dt}. \tag{40}$$

Rearranging Eq. (38), we have

$$\mathbf{x}_0 \approx \frac{\mathbf{x}_t - \sigma_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\alpha_t}. \tag{41}$$

Inserting this expression, Eq. (40) becomes

$$\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t} \approx \frac{\frac{d \log \alpha_t^2}{dt}}{2\sigma_t}\left(\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t} - \mathbf{x}_t\right). \tag{42}$$

We will now proceed to show that the "ideal derivative trick", i.e. using the approximations in Eqs. (39) and (42), results in $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}} = \mathbf{0}$.

As in Sec. 3, the total derivative $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is composed as

$$d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t}\frac{d\mathbf{x}_t}{d\gamma_t} + \frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t}\frac{dt}{d\gamma_t}. \tag{43}$$

Inserting the "ideal derivative trick", the above becomes

$$d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \frac{1}{\sigma_t}\left(\frac{1}{2}\frac{1}{\alpha_t^2}\frac{d\alpha_t^2}{d\gamma_t}\left[\mathbf{x}_t - \frac{\boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t)}{\sigma_t}\right]\right) + \left(\frac{\frac{d \log \alpha_t^2}{dt}}{2\sigma_t}\left(\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t} - \mathbf{x}_t\right)\right)\frac{dt}{d\gamma_t}, \tag{44}$$

where we have inserted Eq. (26) for $\frac{d\mathbf{x}_t}{d\gamma_t}$ and used the usual parameterization $\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) := -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t}$. Using $\frac{d \log \alpha_t^2}{dt} = \frac{1}{\alpha_t^2}\frac{d\alpha_t^2}{dt}$ and $\frac{d\alpha_t^2}{dt}\frac{dt}{d\gamma_t} = \frac{d\alpha_t^2}{d\gamma_t}$, we can see that the right-hand side of Eq. (44) is $\mathbf{0}$. Hence, applying the second TTM to the DDIM ODE and using the "ideal derivative trick" is equivalent to the first TTM (Euler's method) applied to the DDIM ODE. We believe that this is potentially a reason why the DDIM solver [58], Euler's method applied to the DDIM ODE, shows such great empirical performance: it can be interpreted as an approximate ("ideal derivative trick") second order ODE solver. On the other hand, our derivation also implies that the "ideal derivative trick" used in the second TTM for the DDIM ODE does not actually provide any benefit over the standard DDIM solver, because all additional second-order terms vanish. Hence, to improve upon regular DDIM, the "ideal derivative trick" is insufficient and we need to learn the higher-order score terms more accurately without such coarse approximations, as we do in our work.

Furthermore, it is interesting to show that we do not obtain the same cancellation effect when applying the "ideal derivative trick" to the Probability Flow ODE in Eq. (18): Let $\boldsymbol{f}(\mathbf{x}_t, t) = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t - \frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t}\right]$ (right-hand side of Probability Flow ODE), then

$$\frac{d\boldsymbol{f}}{dt}|_{(\mathbf{x}_t, t)} = \frac{\beta'(t)}{\beta(t)}\boldsymbol{f}(\mathbf{x}_t, t) - \frac{1}{2}\beta(t)\frac{d}{dt}\left[\mathbf{x}_t - \frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t}\right] \tag{45}$$

$$= \left[\frac{\beta'(t)}{\beta(t)} - \frac{1}{2}\beta(t)\right]\boldsymbol{f}(\mathbf{x}_t, t) + \frac{1}{2}\beta(t)\left(\frac{\frac{d\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{dt}}{\sigma_t} - \sigma_t^{-2}\frac{d\sigma_t}{dt}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right), \tag{46}$$

where $\beta'(t) := \frac{d\beta(t)}{dt}$. Using the "ideal derivative trick", we have $\frac{d\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{dt} = d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\, d_t\gamma_t \approx \mathbf{0}$, and therefore the above becomes

$$\frac{d\boldsymbol{f}}{dt}|_{(\mathbf{x}_t, t)} \approx \left[\frac{\beta'(t)}{\beta(t)} - \frac{1}{2}\beta(t)\right]\boldsymbol{f}(\mathbf{x}_t, t) - \frac{\beta(t)}{2\sigma_t^2}\frac{d\sigma_t}{dt}\boldsymbol{\epsilon}(\mathbf{x}_t, t). \tag{47}$$

The derivative $\frac{d\sigma_t}{dt}$ can be computed as follows

$$\frac{d\sigma_t}{dt} = \frac{1}{2\sigma_t}\frac{d\sigma_t^2}{dt} \tag{48}$$

$$= \frac{1}{2\sigma_t}\frac{d}{dt}\left(1 - e^{-\int_0^t \beta(t')\,dt'}\right) \tag{49}$$

$$= \frac{\beta(t)e^{-\int_0^t \beta(t')\,dt'}}{2\sigma_t}. \tag{50}$$

Putting everything back together, we have

$$\frac{d\boldsymbol{f}}{dt}|_{(\mathbf{x}_t,t)} = \left[\frac{\beta'(t)}{2\sigma_t} + \frac{\beta^2(t)}{4\sigma_t} - \frac{\beta^2(t)e^{-\int_0^t \beta(t')\,dt'}}{4\sigma_t^3}\right]\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) + \left[-\frac{\beta'(t)}{2} + \frac{\beta^2(t)}{4}\right]\mathbf{x}_t, \tag{51}$$

which is clearly not $\mathbf{0}$ for all $\mathbf{x}_t$ and t. Hence, in contrast to the DDIM ODE, applying Euler's method to the Probability Flow ODE does not lead to an approximate (in the sense of the "ideal derivative trick") second order ODE solver.

Note that very related observations have been made in the concurrent works Karras et al. [94] and Zhang et al. [95]. These works notice that when the data distribution consist only of a single data point or a spherical Gaussian distribution, then the solution trajectories of the generative DDIM ODE are straight lines. In fact, this exactly corresponds to our observation that in such a setting we have $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}} = \mathbf{0}$, as shown above in the analysis of the "ideal derivatives approximation". Note in that context that our above derivation considers the "single data point" distribution assumption, but also applies to the setting where the data is a spherical normal distribution (only σ_t would be different, which would not affect the derivation).

B.3 3rd TTM Applied to the DDIM ODE

As promised in Sec. 3, we show here how to apply the third TTM to the DDIM ODE, resulting in the following scheme:

$$\bar{\mathbf{x}}_{t_{n+1}} = \bar{\mathbf{x}}_{t_n} + h_n\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n},t_n) + \frac{1}{2}h_n^2\frac{d\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{d\gamma_t}|_{(\mathbf{x}_{t_n},t_n)} + \frac{1}{6}h_n^3\frac{d^2\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{d\gamma_t^2}|_{(\mathbf{x}_{t_n},t_n)}, \tag{52}$$

where $h_n = (\gamma_{t_{n+1}} - \gamma_{t_n})$. In the remainder of this section, we derive a computable formula for $\frac{d^2\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{d\gamma_t^2}$, only containing partial derivatives.

Using the chain rule, we have

$$\frac{d^2\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{d\gamma_t^2}|_{(\mathbf{x}_t,t)} = \frac{\partial d_\gamma\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial\mathbf{x}_t}\frac{d\mathbf{x}_t}{d\gamma_t} + \frac{\partial d_\gamma\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t}\frac{dt}{d\gamma_t}, \tag{53}$$

where, using Eq. (43),

$$\frac{\partial d_\gamma\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial\mathbf{x}_t} = \frac{\partial^2\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial\mathbf{x}^2}\frac{d\mathbf{x}_t}{d\gamma_t} + \frac{\partial\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial\mathbf{x}_t}\left(\frac{1}{\sqrt{\gamma_t^2+1}}\frac{\partial\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial\mathbf{x}_t} - \frac{\gamma_t}{1+\gamma_t^2}\boldsymbol{I}\right) + \frac{\partial^2\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t\partial\mathbf{x}_t}\frac{dt}{d\gamma_t}, \tag{54}$$

and

$$\frac{\partial d_\gamma\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t} = \frac{\partial}{\partial t}\left(\frac{\partial\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial\mathbf{x}_t}\frac{d\mathbf{x}_t}{d\gamma_t}\right) + \frac{\partial}{\partial t}\left(\frac{\partial\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t}\frac{dt}{d\gamma_t}\right). \tag{55}$$

The remaining terms in Eq. (55) can be computed as

$$\frac{\partial}{\partial t}\left(\frac{\partial\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t}\frac{dt}{d\gamma_t}\right) = \frac{\partial^2\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t^2}\frac{dt}{d\gamma_t} + \frac{\partial\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t}\frac{d\left(\frac{dt}{d\gamma_t}\right)}{dt}, \tag{56}$$

and

$$\frac{\partial}{\partial t}\left(\frac{\partial\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial\mathbf{x}_t}\frac{d\mathbf{x}_t}{d\gamma_t}\right) = \frac{\partial^2\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t\,\partial\mathbf{x}_t}\frac{d\mathbf{x}_t}{d\gamma_t} + \frac{\partial\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial\mathbf{x}_t}\frac{\partial\left(\frac{d\mathbf{x}_t}{d\gamma_t}\right)}{\partial t} \tag{57}$$

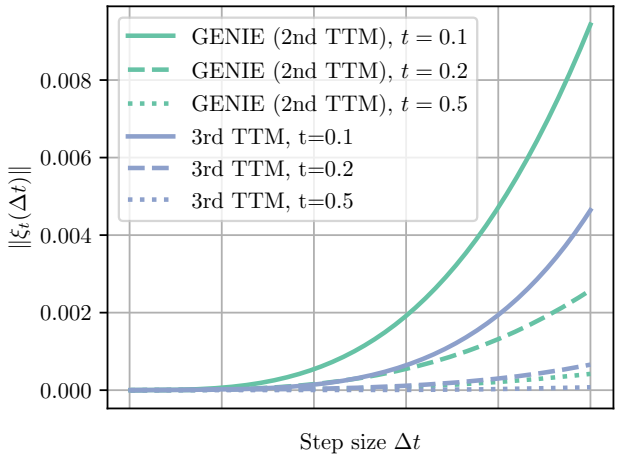


Figure 10: Single step error using analytical score function. See also Fig. 3 (*top*).

where, inserting Eq. (28) for $\frac{d\mathbf{x}_t}{d\gamma_t}$ as well as using the usual parameterization $\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \coloneqq -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\sigma_t}$,

$$\frac{\partial\left(\frac{d\mathbf{x}_t}{d\gamma_t}\right)}{\partial t} = \frac{\partial}{\partial t}\left(-\frac{\gamma_t}{\gamma_t^2+1}\left[\mathbf{x}_t - \frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\sigma_t}\right]\right) \tag{58}$$

$$= \frac{\partial\left(\frac{1}{\sqrt{\gamma_t^2+1}}\right)}{\partial t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) + \frac{1}{\sqrt{\gamma_t^2+1}}\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t} - \frac{\partial\left(\frac{\gamma_t}{1+\gamma_t^2}\right)}{\partial t}\mathbf{x}_t \quad \left(\text{using } \sigma_t = \frac{\gamma_t}{\sqrt{\gamma_t^2+1}}\right) \tag{59}$$

$$= \left(-\frac{\gamma_t}{(\gamma_t^2+1)^{3/2}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) + \frac{\gamma_t^2-1}{(\gamma_t^2+1)^2}\mathbf{x}_t\right)\frac{d\gamma_t}{dt} + \frac{1}{\sqrt{\gamma_t^2+1}}\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t}. \tag{60}$$

We now have a formula for $\frac{d^2\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{d\gamma_t^2}$ containing only partial derivatives, and therefore we can compute $\frac{d^2\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}{d\gamma_t^2}$ using automatic differentiation. Note that we could follow the same procedure to compute even higher derivatives of $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$.

We repeat the 2D toy distribution single step error experiment from Sec. 3 (see also Fig. 3 (*top*) and App. E for details). As expected, in Fig. 10 we can clearly see that the third TTM improves upon the second TTM.

In Fig. 11, we compare the second TTM to the third TTM applied to the DDIM ODE on CIFAR-10. Both for the second and the third TTM, we compute all partial derivatives using automatic differentiation (without distillation). It appears that for using 15 or less steps in the ODE solver, the second TTM performs better than the third TTM. We believe that this could potentially be due to our score model $s_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ not being accurate enough, in contrast to the above 2D toy distribution experiment, where we have access to the analytical score function. Furthermore, note that when we train $s_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ via score matching, we never regularize (higher-order) derivatives of the neural network, and therefore there is no incentive for them to be well-behaved. It would be interesting to see if, besides having more accurate score models, regularization techniques such as spectral regularization [109] could potentially alleviate this issue. Also the higher-order score matching techniques derived by Meng et al. [82] could help to learn higher-order derivates of the score functions more accurately. We leave this exploration to future work.

B.4 GENIE is Consistent and Principled

GENIE is a consistent and principled approach to developing a higher-order ODE solver for sampling from diffusion models: GENIE's design consists of two parts: (1) We are building on the second Truncated Taylor Method (TTM), which is a well-studied ODE solver (see Kloeden and Platen [78])

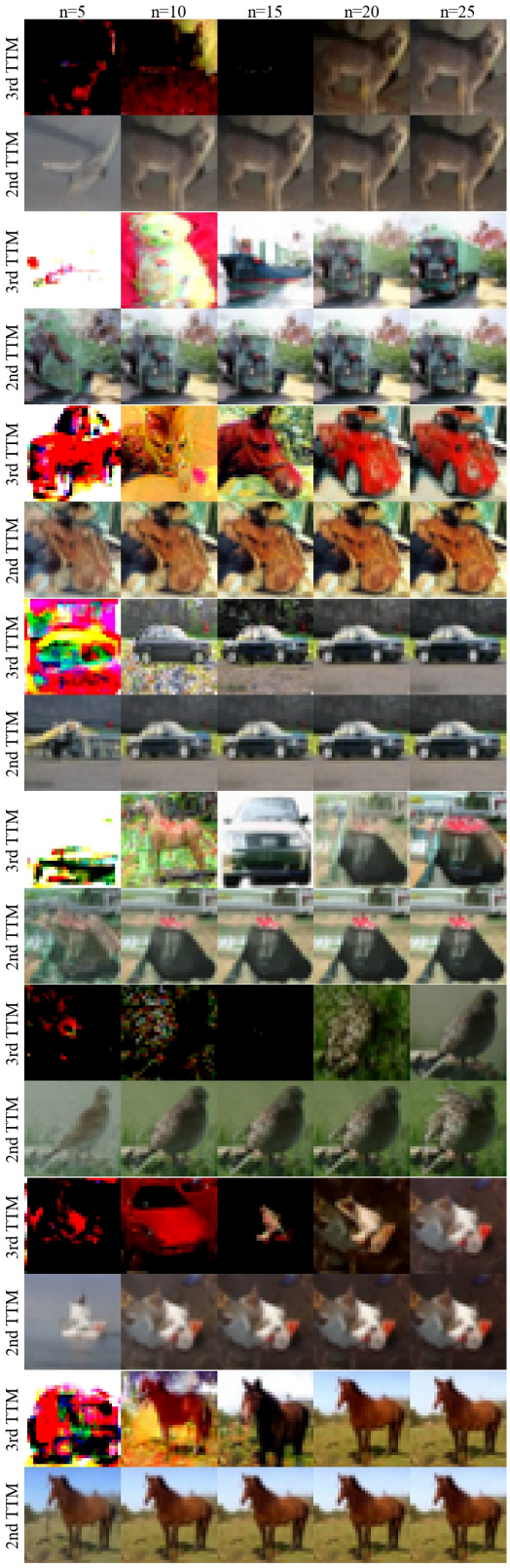


Figure 11: Qualitative comparison of the second and the third TTMs applied to the DDIM ODE on CIFAR-10 (all necessary derivatives calculated with automatic differentiation). The number of steps in the ODE solver is denoted as n.

with provable local and global truncation errors (see also App. B.1). Therefore, if during inference we had access to the ground truth second-order ODE derivatives, which are required for the second TTM, GENIE would simply correspond to the *exact* second TTM.

(2) In principle, we could calculate the exact second-order derivatives *during inference* using automatic differentiation. However, this is too slow for competitive sampling speeds, as it requires additional backward passes through the first-order score network. Therefore, in practice, we use the learned prediction heads $\mathbf{k}_\psi(\mathbf{x}_t, t)$.

Consequently, if $\mathbf{k}_\psi(\mathbf{x}_t, t)$ modeled the ground truth second-order derivatives exactly, i.e. $\mathbf{k}_\psi(\mathbf{x}_t, t) = d_{\gamma_t}\epsilon_\theta(\mathbf{x}_t, t)$ for all $\mathbf{x}_t$ and t, we would obtain a rigorous second-order solver based on the TTM, following (1) above.

In practice, distillation will not be perfect. However, given the above analysis, optimizing a neural network $\mathbf{k}_\psi(\mathbf{x}_t, t)$ towards $d_{\gamma_t}\epsilon_\theta(\mathbf{x}_t, t)$ is well motivated and theoretically grounded. In particular, *during training* we are calculating *exact* ODE gradients using automatic differentiation on the first-order score model as distillation targets. Therefore, in the limit of infinite neural network capacity and perfect optimization, we could in theory minimize our distillation objective function (Eq. (15)) perfectly and obtain $\mathbf{k}_\psi(\mathbf{x}_t, t) = d_{\gamma_t}\epsilon_\theta(\mathbf{x}_t, t)$.

Also recall that regular denoising score matching itself, on which all diffusion models rely, follows the exact same argument. In particular, denoising score matching also minimizes a "simple" (weighted) L_2-loss between a trainable score model $\mathbf{s}_\theta(\mathbf{x}_t, t)$ and the spatial derivative of the log-perturbation kernel, i.e., $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t \mid \mathbf{x}_0)$. From this perspective, denoising score matching itself also simply tries to "distill" (spatial) derivatives into a model. If we perfectly optimized the denoising score matching objective, we would obtain a diffusion model that models the data distribution exactly, but in practice, similar to GENIE, we never achieve that due to imperfect optimization and finite-capacity neural networks. Nevertheless, denoising score matching similarly is a well-defined and principled method, precisely because of that theoretical limit in which the distribution can be reproduced exactly.

We would also like to point out that other, established higher-order methods for diffusion model sampling with the generative ODE, such as linear multistep methods [63], make approximations, too, which can be worse in fact. In particular, multistep methods *always* approximate higher-order derivatives in the TTM using finite differences which is crude for large step sizes, as can be seen in Fig. 3 (*bottom*). From this perspective, if our distillation is sufficiently accurate, GENIE can be expected to be more accurate than such multistep methods.

C Model and Implementation Details

C.1 Score Models

We train *variance-preserving* DDMs [57] for which $\sigma_t^2 = 1 - \alpha_t^2$. We follow Song et al. [57] and set $\beta(t) = 0.1 + 19.9t$; note that $\alpha_t = e^{-\frac{1}{2}\int_0^t \beta(t')\,dt'}$. All score models are parameterized as either $\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) := -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\sigma_t}$ ($\boldsymbol{\epsilon}$-prediction) or $\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) := -\frac{\alpha_t \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)+\sigma_t \mathbf{x}_t}{\sigma_t}$ ($\mathbf{v}$-prediction), where $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ and $\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ are U-Nets [79]. The $\boldsymbol{\epsilon}$-prediction model is trained using the following score matching objective [1]

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}[t_{\text{cutoff}},1],\mathbf{x}_0\sim p(\mathbf{x}_0),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})} \left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\|_2^2\right], \quad \mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}. \tag{61}$$

The $\mathbf{v}$-prediction model is trained using the following score matching objective [69]

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}[t_{\text{cutoff}},1],\mathbf{x}_0\sim p(\mathbf{x}_0),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})} \left[\|\tfrac{\boldsymbol{\epsilon}-\sigma_t\mathbf{x}_t}{\alpha_t} - \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\|_2^2\right], \quad \mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}, \tag{62}$$

which is referred to as "SNR+1" weighting [69]. The neural network $\mathbf{v}_{\boldsymbol{\theta}}$ is now effectively tasked with predicting $\mathbf{v} := \alpha_t \boldsymbol{\epsilon} - \sigma_t \mathbf{x}_0$.

CIFAR-10: On this dataset, we do not train our own score model, but rather use a checkpoint[2] provided by Song et al. [57]. The model is based on the DDPM++ architecture introduced in Song et al. [57] and predicts $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$.

LSUN Bedrooms and LSUN Church-Outdoor: Both datasets use exactly the same model structure. The model structure is based on the DDPM architecture introduced in Ho et al. [1] and predicts $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$.

ImageNet: This model is based on the architecture introduced in Dhariwal and Nichol [4]. We make a small change to the architecture and replace its sinusoidal time embedding by a Gaussian Fourier projection time embedding [57]. The model is class-conditional and we follow Dhariwal and Nichol [4] and simply add the class embedding to the (Gaussian Fourier projection) time embedding. The model predicts $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$.

Cats (Base): This model is based on the architecture introduced in Dhariwal and Nichol [4]. We make a small change to the architecture and replace its sinusoidal time embedding by a Gaussian Fourier projection time embedding [57]. The model predicts $\mathbf{v}_{\boldsymbol{\theta}}$.

Cats (Upsampler): This model is based on the architecture introduced in Dhariwal and Nichol [4]. We make a small change to the architecture and replace its sinusoidal time embedding by a Gaussian Fourier projection time embedding [57]. The upsampler is conditioned on noisy upscaled lower-resolution images, which are concatenated to the regular channels that form the synthesized outputs of the diffusion model. Therefore, we expand the number of input channels from three to six. We use augmentation conditioning [105] to noise the lower-resolution image. In particular, we upscale $\alpha_{t'}\mathbf{x}_{\text{low}} + \sigma_{t'}\mathbf{z}$, where $\mathbf{x}_{\text{low}}$ is the clean lower-resolution image. During training t' is sampled from $\mathcal{U}[t_{\text{cutoff}}, 1]$. During inference, t' is a hyper-parameter which we set to 0.1 for all experiments.

We use two-independent Gaussian Fourier projection embeddings for t and t' and concatenate them before feeding them into the layers of the U-Net.

Model Hyperparameters and Training Details: All model hyperparameters and training details can be found in Tab. 4.

C.2 Prediction Heads

We model the derivative $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ using a small prediction head $\boldsymbol{k}_{\boldsymbol{\psi}}$ on top of the first-order score model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$. In particular, we provide the last feature layer from the $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ network together with its time embedding as well as $\mathbf{x}_t$ and the output of $\boldsymbol{\epsilon}(\mathbf{x}_t, t)$ to the prediction head (see Fig. 4 for a visualization). We found modeling $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ to be effective even for our Cats models that learn to predict $\mathbf{v} = \alpha_t\boldsymbol{\epsilon} - \sigma_t\mathbf{x}_0$ rather than $\boldsymbol{\epsilon}$. Directly learning $d_{\gamma_t}\mathbf{v}_{\boldsymbol{\theta}}$ and adapting the mixed network parameterization (see App. C.2.3) could potentially improve results further. We leave this exploration to future work.

We provide additional details on our architecture next.

[2]The checkpoint can be found at https://drive.google.com/file/d/16_-Ahc6ImZV5ClUcOvM5Iivf8OJ1VSif/view?usp=sharing.

Table 4: Model hyperparameters and training details. The CIFAR-10 model is taken from Song et al. [57]; all other models are trained by ourselves.

Hyperparameter	CIFAR-10	LSUN Bedrooms	LSUN Church-Outdoor	ImageNet	Cats (Base)	Cats (Upsampler)
Model						
Data dimensionality (in pixels)	32	128	128	64	128	512
Residual blocks per resolution	8	2	2	3	2	2
Attention resolutions	16	16	16	8	(8, 16)	(8, 16)
Base channels	128	128	128	192	96	192
Channel multipliers	1,2,3,4	1,1,2,2,4,4,4	1,1,2,2,4,4,4	1,2,3,4	1,2,2,3,3	1,1,2,2,3,3,4
EMA rate	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
# of head channels	N/A	N/A	N/A	64	64	64
# of parameters	107M	148M	148M	283M	200M	80.2M
Base architecture	DDPM++ [57]	DDPM [1]	DDPM [1]	[4]	[4]	[4]
Prediction	$\boldsymbol{\epsilon}$	$\boldsymbol{\epsilon}$	$\boldsymbol{\epsilon}$	$\boldsymbol{\epsilon}$	$\mathbf{v}$	$\mathbf{v}$
Training						
# of iterations	400k	300k	300k	400k	400k	150k
# of learning rate warmup iterations	100k	100k	100k	100k	100k	100k
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Mixed precision training	✗	✓	✓	✓	✓	✓
Learning rate	10^{-4}	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	10^{-4}	10^{-4}
Gradient norm clipping	1.0	1.0	1.0	1.0	1.0	1.0
Dropout	0.1	0.0	0.0	0.1	0.1	0.1
Batch size	128	256	256	1024	128	64
t_{cutoff}	10^{-5}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}

C.2.1 Model Architecture

The architecture of our prediction heads is based on (modified) BigGAN residual blocks [57, 101]. To minimize computational overhead, we only use a single residual block.

In particular, we concatenate the last feature layer with $\mathbf{x}_t$ as well as $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ and feed it into a convolutional layer. For the upsampler, we also condition on the noisy up-scaled lower resolution image. We experimented with normalizing the feature layer before concatenation. The output of the convolutional layer as well as the time embedding are then fed to the residual block. Similar to U-Nets used in score models, we normalize the output of the residual block and apply an activation function. Lastly, the signal is fed to another convolutional layer that brings the number of channels to a desired value (in our case nine, three for each $\boldsymbol{k}_{\boldsymbol{\psi}}^{(i)}$, $i \in \{1, 2, 3\}$, in Eq. (66)).

All model hyperparameters can be found in Tab. 5. We also include the additional computational overhead induced by the prediction heads in Tab. 5; see App. C.2.5 for details on how we measured the overhead.

C.2.2 Training Details

We train for 50k iterations using Adam [110]. We experimented with two base learning rates: 10^{-4} and $5 \cdot 10^{-5}$. We furthermore tried two "optimization setups": (linearly) warming up the learning rate in the first 10k iterations (score models are often trained by warming up the learning rate in the first 100k iterations) or, following Salimans and Ho [69], linearly decaying the learning rate to 0 in the entire 50k iterations of training; we respectively refer to these two setups as "warmup" and "decay". We measure the FID every 5k iterations and use the best checkpoint.

Note that we have to compute the Jacobian-vector products in Eq. (12) via automatic differentiation during training. We repeatedly found that computing the derivative $\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t}$ via automatic differentiation leads to numerical instability (`NaN`) for small t when using mixed precision training. For simplicity, we turned off mixed precision training altogether. However, training performance could have been optimized by only turning off mixed precision training for the derivative $\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t}$.

All training details can be found in Tab. 5.

C.2.3 Mixed Network Parameterization

Our mixed network parameterization is derived from a simple single data point assumption, i.e., $p_t(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \mathbf{0}, \sigma_t^2 \boldsymbol{I})$. This assumption leads to $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \frac{\mathbf{x}_t}{\sigma_t}$ which we can plug into the three

Table 5: Model hyperparameters and training details for the prediction heads.

Hyperparameter	CIFAR-10	LSUN Bedrooms	LSUN Church-Outdoor	ImageNet	Cats (Base)	Cats (Upsampler)
Model						
Data dimensionality	32	128	128	64	128	512
EMA rate	0	0	0	0	0	0
Number of channels	128	128	128	196	196	92
# of parameters	526k	526k	526k	1.17M	1.17M	302k
Normalize $\mathbf{x}_{\text{embed}}$	✗	✗	✓	✗	✗	✗
Training						
# of iterations	20k	40k	35k	15k	20k	20k
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Optimization setup	Decay	Warmup	Warmup	Warmup	Warmup	Warmup
Mixed precision training	✗	✗	✗	✗	✗	✗
Learning rate	$5 \cdot 10^{-5}$	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Gradient norm clipping	1.0	1.0	1.0	1.0	1.0	1.0
Dropout	0.0	0.0	0.0	0.0	0.0	0.0
Batch size	128	256	256	256	64	16
t_{cutoff}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}
Inference						
Add. comp. overhead	1.47%	14.0%	14.4%	2.83%	7.55%	13.3%

terms of Eq. (12):

$$\frac{1}{\sqrt{\gamma_t^2+1}} \frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \frac{1}{\sqrt{\gamma_t^2+1}} \frac{\mathbf{x}_t}{\sigma_t^2} = \frac{1}{\gamma_t} \frac{\mathbf{x}_t}{\sigma_t}, \tag{63}$$

and

$$-\frac{\gamma_t}{1+\gamma_t^2} \frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \mathbf{x}_t \approx -\frac{\gamma_t}{\sigma_t\left(1+\gamma_t^2\right)} \mathbf{x}_t = -\frac{\gamma_t}{1+\gamma_t^2} \frac{\mathbf{x}_t}{\sigma_t}, \tag{64}$$

and finally

$$\frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t} \frac{dt}{d\gamma_t} \approx -\frac{\mathbf{x}_t}{\sigma_t^2} \frac{d\sigma_t}{dt} \frac{dt}{d\gamma_t} = -\frac{\gamma_t^2+1}{\gamma_t^2} \mathbf{x}_t \frac{1}{\left(\gamma_t^2+1\right)^{3/2}} = -\frac{1}{\gamma_t(1+\gamma_t^2)} \frac{\mathbf{x}_t}{\sigma_t}, \tag{65}$$

where we have used $\sigma_t = \frac{\gamma_t}{\sqrt{\gamma_t^2+1}}$. This derivation therefore implies the following mixed network parameterization

$$\boldsymbol{k}_{\boldsymbol{\psi}} = -\frac{1}{\gamma_t} \boldsymbol{k}_{\boldsymbol{\psi}}^{(1)} + \frac{\gamma_t}{1+\gamma_t^2} \boldsymbol{k}_{\boldsymbol{\psi}}^{(2)} + \frac{1}{\gamma_t(1+\gamma_t^2)} \boldsymbol{k}_{\boldsymbol{\psi}}^{(3)} \approx d_{\gamma_t} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}, \tag{66}$$

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 $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ network). To provide additional intuition, we basically replaced the $-\frac{\mathbf{x}_t}{\sigma_t}$ terms in Eqs. (63) to (65) by neural networks. However, we know that for approximately Normal data $\frac{\mathbf{x}_t}{\sigma_t} \approx \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$, where $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ predicts "noise" values $\boldsymbol{\epsilon}$ that were drawn from a standard Normal distribution and are therefore varying on a well-behaved scale. Consequently, up to the Normal data assumption, we can also expect our prediction heads $\boldsymbol{k}_{\boldsymbol{\psi}}^{(i)}(\mathbf{x}_t, t)$ in the parameterization in Eq. (66) to predict well-behaved output values, which should make training stable. This *mixed network parameterization* approach is inspired by the mixed score parameterization from Vahdat et al. [49] and Dockhorn et al. [60].

C.2.4 Pseudocode

In this section, we provide pseudocode for training our prediction heads $\boldsymbol{k}_{\boldsymbol{\psi}}$ and using them for sampling with GENIE. In Alg. 1, the analytical $\frac{dt}{d\gamma_t}$ is an implicit hyperparameter of the DDM as it depends on α_t. For our choice of $\alpha_t = e^{-\frac{1}{2}\int_0^t 0.1+19.9t' \, dt'}$ (see App. C.1), we have

$$\frac{dt}{d\gamma_t} = \frac{\frac{2\gamma_t}{19.9(\gamma_t^2+1)}}{\sqrt{\left(\frac{0.1}{19.9}\right)^2 + \frac{2\log(\gamma_t^2+1)}{19.9}}}, \tag{67}$$

where $\gamma_t = \sqrt{\frac{1-\alpha_t^2}{\alpha_t^2}}$.

In Alg. 2, we are free to use any time discretization $t_0 = 1.0 > t_1 > \cdots > t_N = t_{\text{cutoff}}$. When referring to "linear striding" in this work, we mean the time discretization $t_n = 1.0 - (1.0 - t_{\text{cutoff}})\frac{n}{N}$. When referring to "quadratic striding" in this work, we mean the time discretization $t_n = \left(1.0 - (1.0 - \sqrt{t_{\text{cutoff}}})\frac{n}{N}\right)^2$.

Algorithm 1 Training prediction heads $\boldsymbol{k}_{\boldsymbol{\psi}}$

Input: Score model $\boldsymbol{s}_{\boldsymbol{\theta}} := -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t}$, number of training iterations N.
Output: Trained prediction head $\boldsymbol{k}_{\boldsymbol{\psi}}$.

for $n = 1$ **to** N **do**
  Sample $\mathbf{x}_0 \sim p_0(\mathbf{x}_0)$, $t \sim \mathcal{U}[t_{\text{cutoff}}, 1]$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  Set $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$
  Compute $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$
  Compute the exact spatial Jacobian-vector product $\text{JVP}_{\text{s}} = \frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t}\left(\frac{1}{\sqrt{\gamma_t^2+1}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \frac{\gamma_t}{1+\gamma_t^2}\mathbf{x}_t\right)$ via automatic differentiation
  Compute the exact temporal Jacobian-vector product $\text{JVP}_{\text{t}} = \frac{\partial \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t}\frac{dt}{d\gamma_t}$ via automatic differentiation ($\frac{dt}{d\gamma_t}$ can be computed analytically)
  Compute $\boldsymbol{k}_{\boldsymbol{\psi}}(\mathbf{x}_t, t)$ using the mixed parameterization in Eq. (66)
  Update weights $\boldsymbol{\psi}$ to minimize $\gamma_t^2 \|\boldsymbol{k}_{\boldsymbol{\psi}}(\mathbf{x}_t, t) - d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\|_2^2$, where $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \text{JVP}_{\text{s}} - \text{JVP}_{\text{t}}$
end for

Algorithm 2 GENIE sampling

Input: Score model $\boldsymbol{s}_{\boldsymbol{\theta}} := -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t}$, prediction head $\boldsymbol{k}_{\boldsymbol{\psi}}$, number of sampler steps N, time discretization $\{t_n\}_{n=0}^N$.
Output: Generated GENIE output sample $\mathbf{y}$.
Sample $\mathbf{x}_{t_0} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
Set $\bar{\mathbf{x}}_{t_0} = \sqrt{1+\gamma_{t_0}^2}\mathbf{x}_{t_0}$ ▷ *Note that* $\bar{\mathbf{x}}_{t_n} = \sqrt{1+\gamma_{t_n}^2}\mathbf{x}_{t_n}$ *for all* t_n
for $n = 0$ **to** $N-1$ **do**
  if AFS and $n = 0$ **then**
    $\bar{\mathbf{x}}_{t_{n+1}} = \bar{\mathbf{x}}_{t_n} + (\gamma_{t_{n+1}} - \gamma_{t_n})\mathbf{x}_{t_n}$
  else
    $\bar{\mathbf{x}}_{t_{n+1}} = \bar{\mathbf{x}}_{t_n} + (\gamma_{t_{n+1}} - \gamma_{t_n})\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) + \frac{1}{2}(\gamma_{t_{n+1}} - \gamma_{t_n})^2\boldsymbol{k}_{\boldsymbol{\psi}}(\mathbf{x}_{t_n}, t_n)$
  end if
  $\mathbf{x}_{t_{n+1}} = \frac{\bar{\mathbf{x}}_{t_{n+1}}}{\sqrt{1+\gamma_{t_{n+1}}^2}}$
end for
if Denoising **then**
  $\mathbf{y} = \frac{\mathbf{x}_{t_N} - \sigma_{t_N}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_{t_N}, t_N)}{\alpha_{t_N}}$
else
  $\mathbf{y} = \mathbf{x}_{t_N}$
end if

C.2.5 Measuring Computational Overhead

Our prediction heads induce a slight computational overhead since their forward pass has to occur after the forward pass of the score model. We measure the overhead as follows: first, we measure the inference time of the score model itself. We do five forward passes to "warm-up" the model and then subsequently synchronize via `torch.cuda.synchronize()`. We then measure the total wall-clock time of 50 forward passes. We then repeat this process using a combined forward pass: first the score model and subsequently the prediction head. We choose the batch size to (almost) fill the entire GPU memory. In particular we chose batch sizes of 512, 128, 128, 64, 64, and 8, for CIFAR-10, LSUN Bedrooms, LSUN Church-Outdoor, ImageNet, Cats (base), and Cats (upsampler), respectively. The computational overhead for each model is reported in Tab. 5. This measurement was carried out on a single NVIDIA 3080 Ti GPU.

D Learning Higher-Order Gradients without Automatic Differentiation and Distillation

In this work, we learn the derivative $d_{\gamma_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$, which includes a spatial and a temporal Jacobian-vector product, by distillation based on automatic differentiation (AD). We now derive an alternative learning objective for the spatial Jacobian-vector product (JVP) which does not require any AD. We start with the following (conditional) expectation

$$\mathbb{E}\left[\alpha_t^2 \mathbf{x}_0\mathbf{x}_t^\top - \alpha_t\left[\mathbf{x}_0\mathbf{x}_t^\top + \mathbf{x}_t\mathbf{x}_0^\top\right] \mid \mathbf{x}_t, t\right] = -\mathbf{x}_t\mathbf{x}_t^\top + \sigma_t^4 \boldsymbol{S}_2(\mathbf{x}_t, t) + \sigma_t^4 \boldsymbol{s}_1(\mathbf{x}_t, t)\boldsymbol{s}_1(\mathbf{x}_t, t)^\top + \sigma_t^2 \boldsymbol{I}, \tag{68}$$

where $\boldsymbol{s}_1(\mathbf{x}_t, t) \coloneqq \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ and $\boldsymbol{S}_2(\mathbf{x}_t, t) \coloneqq \nabla_{\mathbf{x}_t}^\top \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$. The above formula is derived in Meng et al. [Theorem 1, 82]. Adding $\mathbf{x}_t\mathbf{x}_t^\top$ to Eq. (68) and subsequently dividing by σ_t^2, we have

$$\mathbb{E}\left[\frac{\alpha_t^2}{\sigma_t^2}\mathbf{x}_0\mathbf{x}_t^\top - \frac{\alpha_t}{\sigma_t^2}\left[\mathbf{x}_0\mathbf{x}_t^\top + \mathbf{x}_t\mathbf{x}_0^\top\right] + \frac{1}{\sigma_t^2}\mathbf{x}_t\mathbf{x}_t^\top \mid \mathbf{x}_t, t\right] = \sigma_t^2 \boldsymbol{S}_2(\mathbf{x}_t, t) + \sigma_t^2 \boldsymbol{s}_1(\mathbf{x}_t, t)\boldsymbol{s}_1(\mathbf{x}_t, t)^\top + \boldsymbol{I}, \tag{69}$$

where we could pull the $\frac{1}{\sigma_t^2}\mathbf{x}_t\mathbf{x}_t^\top$ term into the expectation because it is conditioned on t and $\mathbf{x}_t$. Using $\mathbf{x}_t = \alpha_t\mathbf{x}_0 + \sigma_t\boldsymbol{\epsilon}$, we can rewrite the above as

$$\mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top \mid \mathbf{x}_t, t\right] = \sigma_t^2 \boldsymbol{S}_2(\mathbf{x}_t, t) + \sigma_t^2 \boldsymbol{s}_1(\mathbf{x}_t, t)\boldsymbol{s}_1(\mathbf{x}_t, t)^\top + \boldsymbol{I}. \tag{70}$$

For an arbitrary $\boldsymbol{v} \coloneqq \boldsymbol{v}(\mathbf{x}_t, t)$, we then have

$$\mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top \boldsymbol{v} \mid \mathbf{x}_t, t\right] = \sigma_t^2 \boldsymbol{S}_2(\mathbf{x}_t, t)\boldsymbol{v} + \sigma_t^2 \boldsymbol{s}_1(\mathbf{x}_t, t)\boldsymbol{s}_1(\mathbf{x}_t, t)^\top \boldsymbol{v} + \boldsymbol{v}. \tag{71}$$

Therefore, we can develop a score matching-like learning objective for the (general) spatial JVP $\boldsymbol{o}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \boldsymbol{S}_2(\mathbf{x}, t)\boldsymbol{v}$ as

$$\mathbb{E}_{t\sim\mathcal{U}[t_{\text{cutoff}},1], \mathbf{x}_0\sim p(\mathbf{x}_0), \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[g_{\text{no-ad}}(t)\|\boldsymbol{o}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) + \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)^\top \boldsymbol{v} + \frac{1}{\sigma_t^2}\boldsymbol{v} - \boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top \boldsymbol{v}\|_2^2\right], \tag{72}$$

for some weighting function $g_{\text{no-ad}}(t)$. Setting $\boldsymbol{v}(\mathbf{x}_t, t) = -\sigma_t\left(\frac{1}{\sqrt{\gamma_t^2+1}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \frac{\gamma_t}{1+\gamma_t^2}\mathbf{x}_t\right)$, would recover the spatial JVP needed for the computation of $d_{\gamma_t}\boldsymbol{\epsilon}$. In the initial phase of this project, we briefly experimented with learning the spatial JVP using this approach; however, we found that our distillation approach worked significantly better.

E Toy Experiments

For all toy experiments in Sec. 3, we consider the following ground truth distribution:

$$p_0(\mathbf{x}_0) = \frac{1}{8}\sum_{i=1}^{8} p_0^{(i)}(\mathbf{x}_0), \tag{73}$$

where

$$p_0^{(i)}(\mathbf{x}_0) = \frac{1}{8}\sum_{j=1}^{8} \mathcal{N}(\mathbf{x}_0, s_1\boldsymbol{\mu}_i + s_1 s_2 \boldsymbol{\mu}_j, \sigma^2 \boldsymbol{I}). \tag{74}$$

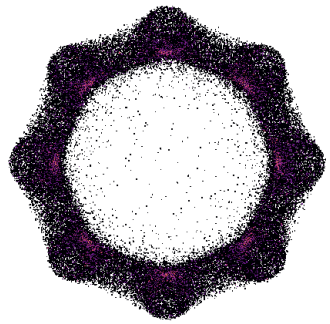
We set $\sigma = 10^{-2}$, $s_1 = 0.9$, $s_2 = 0.2$, and

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \boldsymbol{\mu}_2 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \quad \boldsymbol{\mu}_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{\mu}_4 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

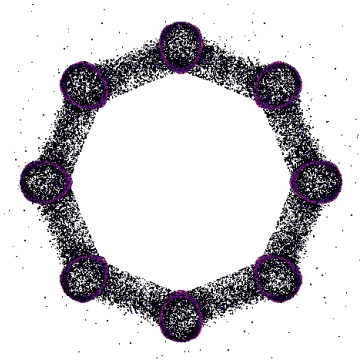
$$\boldsymbol{\mu}_5 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \quad \boldsymbol{\mu}_6 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}, \quad \boldsymbol{\mu}_7 = \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \quad \boldsymbol{\mu}_8 = \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

The ground truth distribution is visualized in Fig. 2a. Note that we can compute the score functions (and all its derivatives) analytically for Gaussian mixture distributions.

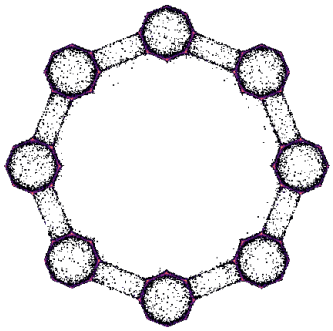
In Fig. 2, we compared DDIM to GENIE for sampling using the analytical score function of the ground truth distribution with 25 solver steps. In Fig. 12, we repeated this experiment for 5, 10, 15, and 20 solver steps. We found that in particular for $n = 10$ both solvers generate samples in interesting patterns.



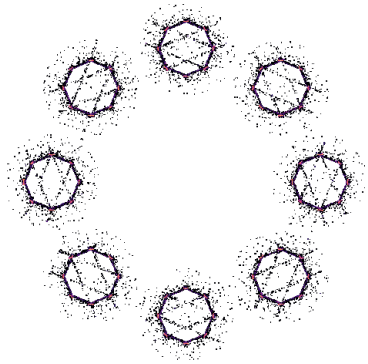
(a) DDIM, $n = 5$



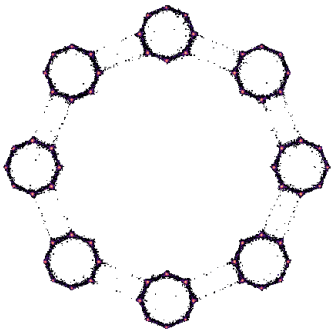
(b) GENIE, $n = 5$



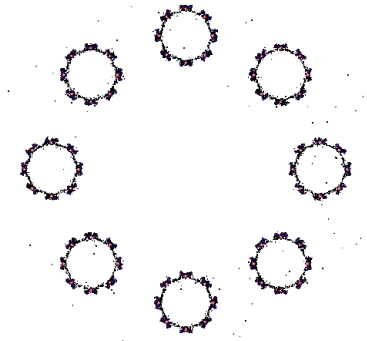
(c) DDIM, $n = 10$



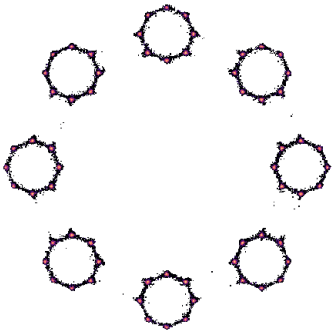
(d) GENIE, $n = 10$



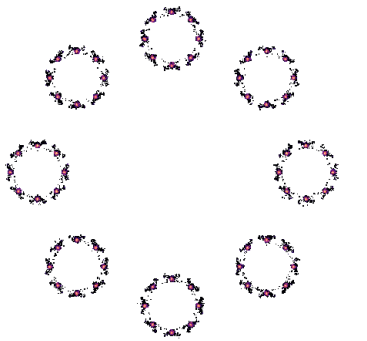
(e) DDIM, $n = 15$



(f) GENIE, $n = 15$



(g) DDIM, $n = 20$



(h) GENIE, $n = 20$

Figure 12: Modeling a complex 2D toy distribution: Samples are generated with DDIM and GENIE with n solver steps using the analytical score function of the ground truth distribution (visualized in Fig. 2a). Zoom in for details.

F Image Experiments

F.1 Evaluation Metrics, Baselines, and Datasets

Metrics: We quantitatively measure sample quality via Fréchet Inception Distance [FID, 102]. It is common practice to use 50k samples from the training set for reference statistics. We follow this practice for all datasets except for ImageNet and Cats. For ImageNet, we follow Dhariwal and Nichol [4] and use the entire training set for reference statistics. For the small Cats dataset, we use the training as well as the validation set for reference statistics.

Baselines: We run baseline experiments using two publicly available repositories. The `score_sde_pytorch` repository is licensed according to the Apache License 2.0; see also their license file here. The `CLD-SGM` repository is licensed according to the NVIDIA Source Code License; see also their license file here.

Datasets: We link here the websites of the datasets used in this experiment: CIFAR-10, LSUN datasets, ImageNet, and AFHQv2.

F.2 Analytical First Step (AFS)

The forward process of DDMs generally converges to an analytical distribution. This analytical distribution is then used to sample from DDMs, defining the initial condition for the generative ODE/SDE. For example, for variance-preserving DDMs, we have $p_1(\mathbf{x}_1) \approx \mathcal{N}(\mathbf{x}_1; \mathbf{0}, \boldsymbol{I})$.

In this work, we try to minimize the computational complexity of sampling from DDMs, and therefore operate in a low NFE regime. In this regime, every additional function evaluation makes a significant difference. We therefore experimented with replacing the learned score with the (analytical score) of $\mathcal{N}(\mathbf{0}, \boldsymbol{I}) \approx p_1(\mathbf{x}_1)$ in the first step of the ODE solver. This "gained" function evaluation can then be used as an additional step in the ODE solver later.

In particular, we have

$$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_1, 1) \approx \mathbf{x}_1, \tag{75}$$

and $\frac{d\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_1,1)}{d\gamma_1} \approx \mathbf{0}$ as shown below:

$$\begin{aligned}
\frac{d\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_1, 1)}{d\gamma_1} &\approx \frac{d\mathbf{x}_t}{d\gamma_t}\Big|_{t=1} && (76)\\
&= -\frac{\gamma_t}{\gamma_t^2 + 1}\left[\mathbf{x}_t + \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\big|_{t=1} \quad \text{(using Eq. (28))} && (77)\\
&\approx \mathbf{0} \quad \text{(using normal assumption } \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx -\mathbf{x}_t) && (78)
\end{aligned}$$

Given this, the AFS step becomes identical to the Euler update that uses the Normal score function for $\mathbf{x}_1$. This step is shown in the pseudocode in Alg. 2.

F.3 Classifier-Free Guidance

As discussed in Sec. 5.2, to guide diffusion sampling towards particular classes, we replace $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ with

$$\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, c, w) = (1 + w)\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, c) - w\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \tag{79}$$

where $w > 1.0$ is the "guidance scale", in the DDIM ODE. We experiment with classifier-free guidance on ImageNet. In Eq. (79) we re-use the conditional ImageNet score model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, c)$ trained before (see App. C.1 for details), and train an additional unconditional ImageNet score model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ using the exact same setup (and simply setting the class embedding to zero). We also re-use the conditional prediction head trained on top of the conditional ImageNet score model and train an additional prediction head for the unconditional model. Note that for both the score models as well as the prediction heads, we could share parameters between the models to reduce computational complexity [70]. The modified GENIE scheme for classifier-free guidance is then given as

$$\bar{\mathbf{x}}_{t_{n+1}} = \bar{\mathbf{x}}_{t_n} + (\gamma_{t_{n+1}} - \gamma_{t_n})\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n, c, w) + \frac{1}{2}(\gamma_{t_{n+1}} - \gamma_{t_n})^2\hat{\boldsymbol{k}}_{\boldsymbol{\psi}}(\mathbf{x}_{t_n}, t_n, c, w), \tag{80}$$

where

$$\hat{\boldsymbol{k}}_{\boldsymbol{\psi}}(\mathbf{x}_{t_n}, t_n, c, w) = (1 + w)\boldsymbol{k}_{\boldsymbol{\psi}}(\mathbf{x}_{t_n}, t_n, c) - w\boldsymbol{k}_{\boldsymbol{\psi}}(\mathbf{x}_{t_n}, t_n). \tag{81}$$

F.4 Encoding

To encode a data point $\mathbf{x}_0$ into latent space, we first "diffuse" the data point to $t = 10^{-3}$, i.e., $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We subsequently simulate the generative ODE (backwards) from $t = 10^{-3}$ to $t = 1$, obtaining the latent point $\mathbf{x}_1$.

To decode a latent point $\mathbf{x}_1$, we simulate the generative ODE (forwards) from $t = 1.0$ to $t = 10^{-3}$. We then denoise the data point, i.e., $\mathbf{x}_0 = \frac{\mathbf{x}_t - \sigma_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\alpha_t}$. Note that denoising is generally optional to sample from DDMs; however, for our encoding-decoding experiment we always used denoising in the decoding part to match the inital "diffusion" in the encoding part.

F.5 Latent Space Interpolation

We can use encoding to perform latent space interpolation of two data points $\mathbf{x}_0^{(0)}$ and $\mathbf{x}_0^{(1)}$. We first encode both data points, following the encoding setup from App. F.4, and obtain $\mathbf{x}_1^{(0)}$ and $\mathbf{x}_1^{(1)}$, respectively. We then perform spherical interpolation of the latent codes:

$$\mathbf{x}_1^{(b)} = \mathbf{x}_1^{(0)} \sqrt{1 - \mathrm{b}} + \mathbf{x}_2^{(1)} \sqrt{\mathrm{b}}, \quad \mathrm{b} \in [0, 1]. \tag{82}$$

Subsequently, we decode the latent code $\mathbf{x}_1^{(b)}$ following the decoding setup from App. F.4. In Fig. 13, we show latent space interpolations for LSUN Church-Outdoor and LSUN Bedrooms.

F.6 Extended Quantitative Results

In this section, we show additional quantitative results not presented in the main paper. In particular, we show results for all four hyperparameter combinations (binary choice of AFS and binary choice of denoising) for methods evaluated by ourselves. For these methods (i.e., GENIE, DDIM, S-PNDM, F-PNDM, Euler–Maruyama), we follow the **Synthesis Strategy** outlined in Sec. 5, with the exception that we use linear striding instead of quadratic striding for S-PNDM [63] and F-PNDM [63]. To apply quadratic striding to these two methods, one would have to derive the Adams–Bashforth methods for non-constant step sizes which is beyond the scope of our work.

Results can be found in Tabs. 8 to 13. As expected, AFS can considerably improve results for almost all methods, in particular for NFEs ≤ 15. Denoising, on the other hand, is more important for larger NFEs. For our Cats models, we initially found that denoising hurts performance, and therefore did not further test it in all settings.

Recall Scores. We quantify the sample diversity of GENIE and other fast samplers using the recall score [111]. In particular, we follow DDGAN [67] and use the improved recall score [112]; results on CIFAR-10 can be found in Tab. 6. As expected, we can see that for all methods recall scores suffer as the NFEs decrease. Compared to the baselines, GENIE achieves excellent recall scores, being on par with F-PNDM for NFE≥ 15. However, F-PNDM cannot be run for NFE$\leq$10 (due to its additional Runge–Kutta warm-up iterations). Overall, these results confirm that GENIE offers strong sample diversity when compared to other common samplers using the same score model checkpoint.

Striding Schedule Grid Search. As discussed in Sec. 5 the fixed quadratic striding schedule (for choosing the times t for evaluating the model during synthesis under fixed NFE budgets) used in GENIE may be sub-optimal, in particular for small NFEs. To explore this, we did a small grid search over three different striding schedules. As described in App. C.2.4, the quadratic striding schedule can be written as $t_n = \left(1.0 - (1.0 - \sqrt{t_{\text{cutoff}}})\frac{n}{N}\right)^2$, and easily be generalized to

$$t_n = \left(1.0 - (1.0 - t_{\text{cutoff}}^{1/\rho})\frac{n}{N}\right)^{\rho}, \rho > 1. \tag{83}$$

In particular, besides the quadratic schedule $\rho = 2$, we also tested the two additional values $\rho = 1.5$ and $\rho = 2.5$. We tested these schedules on GENIE as well as DDIM [58]; note that the other two comptetive baselines, S-PNDM [63] and F-PNDM [63], rely on linear striding, and therefore a grid search is not applicable. We show results for GENIE and DDIM in Tab. 7; for each combination of solver and NFE we applied the best synthesis strategy (whether or not we use denoising and/or the analytical first step) of quadratic striding ($\rho = 2.0$) also to $\rho = 1.5$ and $\rho = 2.5$. As can be seen in the table, $\rho = 1.5$ improves for both DDIM and GENIE for NFE=5 (over the quadratic schedule

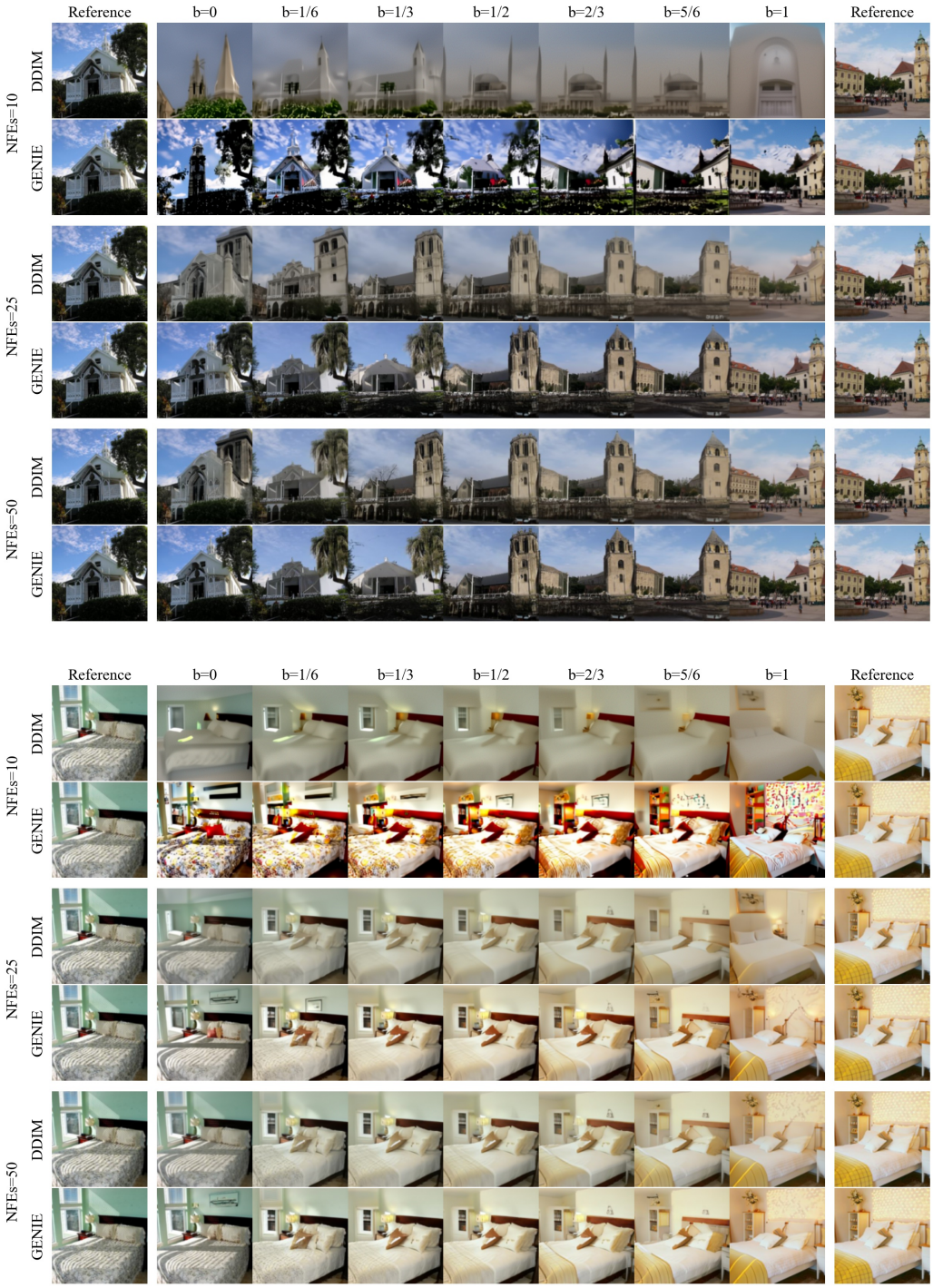


Figure 13: Latent space interpolations for LSUN Church-Outdoor (Top) and LSUN Bedrooms (Bottom). Note that $b = 0$ and $b = 1$ correspond to the decodings of the encoded reference images. Since this encode-decode loop is itself not perfect, the references are not perfectly reproduced at $b = 0$ and $b = 1$.

Table 6: Unconditional CIFAR-10 generative performance, measured in Recall (higher values are better). All methods use the same score model checkpoint.

Method	AFS	Denoising	NFEs=5	NFEs=10	NFEs=15	NFEs=20	NFEs=25
GENIE (ours)	✗	✗	0.28	0.48	0.54	0.56	0.56
	✗	✓	0.21	0.45	0.52	0.56	0.57
	✓	✗	0.27	0.47	0.53	0.56	0.56
	✓	✓	0.19	0.46	0.53	0.55	0.56
DDIM [58]	✗	✗	0.10	0.27	0.38	0.43	0.46
	✗	✓	0.07	0.24	0.35	0.42	0.46
	✓	✗	0.08	0.27	0.38	0.43	0.46
	✓	✓	0.04	0.24	0.36	0.42	0.45
S-PNDM [63]	✗	✗	0.06	0.30	0.43	0.49	0.52
	✗	✓	0.02	0.25	0.39	0.46	0.50
	✓	✗	0.11	0.33	0.45	0.50	0.53
	✓	✓	0.06	0.29	0.41	0.47	0.51
F-PNDM [63]	✗	✗	N/A	N/A	0.55	0.57	0.58
	✗	✓	N/A	N/A	0.52	0.56	0.57
	✓	✗	N/A	N/A	0.55	0.58	0.59
	✓	✓	N/A	N/A	0.54	0.56	0.57
Euler–Maruyama	✗	✗	0.00	0.00	0.00	0.02	0.08
	✗	✓	0.00	0.00	0.00	0.03	0.06
	✓	✗	0.00	0.00	0.00	0.03	0.09
	✓	✓	0.00	0.00	0.00	0.03	0.09

Table 7: Unconditional CIFAR-10 generative performance (measured in FID) using our GENIE and DDIM [58] with different striding schedules using exponents $\rho \in \{1.5, 2.0, 2.5\}$.

Method	ρ	NFEs=5	NFEs=10	NFEs=15	NFEs=20	NFEs=25
GENIE	1.5	**11.2**	**5.28**	5.03	4.35	3.97
	2.0	13.9	5.97	**4.49**	**3.94**	3.67
	2.5	17.8	7.19	4.57	**3.94**	**3.64**
DDIM	1.5	**27.6**	13.5	8.97	7.20	6.15
	2.0	29.7	**11.2**	**7.35**	**5.87**	**5.16**
	2.5	33.2	13.4	8.28	6.36	5.39

$\rho = 2$), whereas larger ρ are preferred for larger NFE. The improvement of GENIE from 13.9 to 11.2 FID for NFE=5 is significant.

Discretization Errors of GENIE compared to other Fast Samplers. We compute discretization errors, in particular local and global truncation errors, of GENIE and compare to existing faster solvers. We are using the CIFAR-10 model. We initially sample 100 latent vectors $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and then, starting from those latent vectors, synthesize 100 approximate ground truth trajectories (GTTs) using DDIM with 1k NFEs (for that many steps, the discretization error is negligible; hence, we can treat this as a pseudo ground truth).

We then synthesize 100 sample trajectories for DDIM [58], S-PNDM [63], F-PNDM [63], and GENIE (for NFEs=$\{5, 10, 15, 20, 25\}$, similar to the main experiments) using the same latent vectors as starting points that were used to generate the GTTs. DDIM, S-PNDM, and F-PNDM are training-free methods that can be run on the exact same score model, which also our GENIE relies on. Thereby, we are able to isolate discretization errors from errors in the learnt score function. We then compute the average L_2-distance (in Inception feature space [104]) between the output image of the fast samplers and the "output" of the pseudo GTT. As can be seen in Fig. 14, GENIE outperforms the three other methods on all NFEs.

Comparing the local truncation error (LTE) of different higher-order solvers can unfortunately not be done in a fair manner. Similar to DDIM, GENIE only needs the current value and a single NFE to predict the next step. In contrast, multistep methods rely on a history of predictions and Runge–Kutta methods rely on multiple NFEs to predict the next step. Thus, we can only fairly compare the LTE of

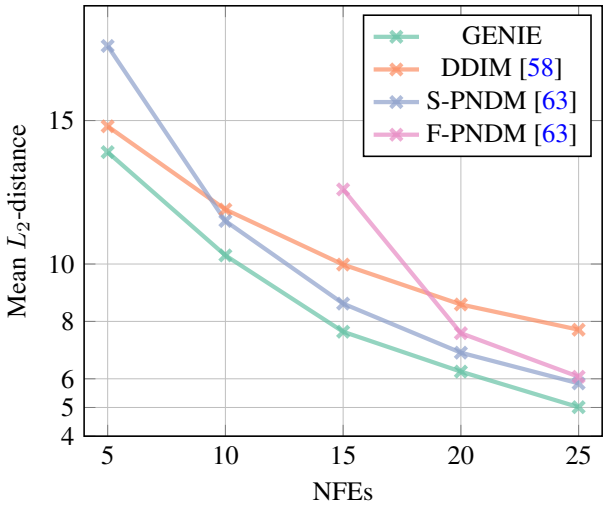


Figure 14: **Global Truncation Error:** L_2-distance of generated outputs by the fast samplers to the (approximate) ground truth (computed using DDIM with 1k NFEs) in Inception feature space [104]. Results are averaged over 100 samples.

GENIE to the LTE of DDIM. In particular, we compute LTEs at three starting times $t \in \{0.1, 0.2, .5\}$ (similar to what we did in Fig. 3). For each t, we then compare one step predictions for different step sizes Δt against the ground truth trajectory (L_2-distance in data space averaged over 100 predictions; since we are not operating directly in image space at these intermediate t, using inception feature would not make sense here). As expected, we can see in Fig. 15 that GENIE has smaller LTE than DDIM for all starting times t.

F.7 Extended Qualitative Results

In this section, we show additional qualitative comparisons of DDIM and GENIE on LSUN Church-Outdoor (Fig. 16), ImageNet (Fig. 17), and Cats (upsampler conditioned on test set images) (Fig. 18 and Fig. 19). In all figures, we can see that samples generated with GENIE generally exhibit finer details as well as sharper contrast and are less blurry compared to standard DDIM.

In Fig. 20 and Fig. 21, we show additional high-resolution images generated with the GENIE Cats upsampler using base model samples and test set samples, respectively.

F.8 Computational Resources

The total amount of compute used in this research project is roughly 163k GPU hours. We used an in-house GPU cluster of V100 NVIDIA GPUs.

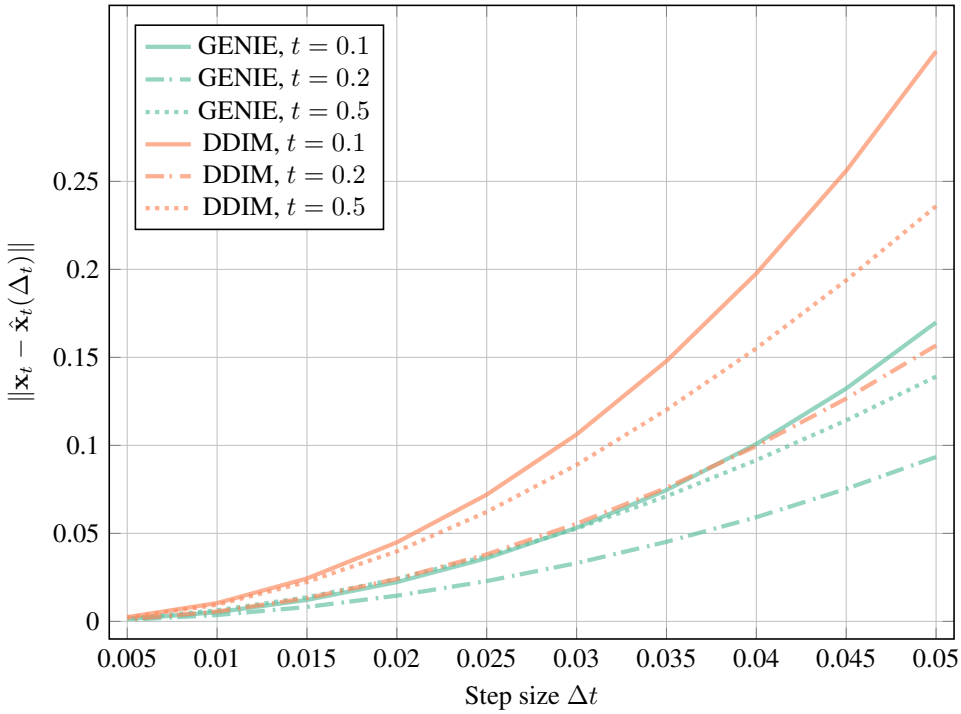


Figure 15: **Local Truncation Error:** Single step (local discretization) error, measured in L_2-distance to (approximate) ground truth (computed using DDIM with 1k NFEs) in data space and averaged over 100 samples, for GENIE and DDIM for three starting time points $t \in \{0.1, 0.2, 0.5\}$ (this is, the t from which a small step with size Δt is taken).

Table 8: 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. This table is an extension of Tab. 1.

Method	AFS	Denoising	NFEs=5	NFEs=10	NFEs=15	NFEs=20	NFEs=25
GENIE (ours)	✗	✗	15.4	**5.97**	4.70	4.30	4.10
	✗	✓	23.5	6.91	4.74	4.02	3.72
	✓	✗	13.9	6.04	4.76	4.33	4.18
	✓	✓	17.9	6.27	**4.49**	**3.94**	**3.67**
DDIM [58]	✗	✗	30.1	11.6	7.56	6.00	5.27
	✗	✓	37.9	13.9	8.76	6.77	5.76
	✓	✗	29.7	11.2	7.35	5.87	5.16
	✓	✓	35.2	12.8	8.17	6.39	5.49
S-PNDM [63]	✗	✗	60.2	12.1	7.16	5.48	4.62
	✗	✓	101	17.2	10.8	8.74	7.62
	✓	✗	35.9	10.3	6.61	5.20	4.51
	✓	✓	56.8	14.9	10.2	8.37	7.35
F-PNDM [63]	✗	✗	N/A	N/A	12.1	6.58	4.89
	✗	✓	N/A	N/A	19.5	10.6	8.43
	✓	✗	N/A	N/A	10.3	5.96	4.73
	✓	✓	N/A	N/A	15.7	10.9	8.52
Euler–Maruyama	✗	✗	364	236	178	121	85.0
	✗	✓	391	235	191	129	89.9
	✓	✗	325	230	164	112	80.3
	✓	✓	364	235	176	120	83.6
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 9: Conditional ImageNet generative performance (measured in FID).

Method	AFS	Denoising	NFEs=5	NFEs=10	NFEs=15	NFEs=20	NFEs=25
GENIE (ours)	✗	✗	23.4	8.35	6.13	5.36	5.00
	✗	✓	35.4	7.59	**5.23**	**4.48**	**4.13**
	✓	✗	21.6	8.92	6.59	5.73	5.27
	✓	✓	**20.2**	**7.41**	5.36	4.68	4.27
DDIM [58]	✗	✗	39.0	14.5	9.47	7.57	6.64
	✗	✓	39.8	11.1	7.17	5.83	5.19
	✓	✗	37.4	14.7	9.73	7.86	6.92
	✓	✓	30.0	10.7	7.14	5.93	5.35
S-PNDM [63]	✗	✗	57.9	15.2	10.0	8.12	7.20
	✗	✓	60.6	12.2	8.69	7.59	6.94
	✓	✗	39.0	13.7	9.75	8.08	7.22
	✓	✓	35.5	11.2	8.54	7.52	6.94
F-PNDM [63]	✗	✗	N/A	N/A	13.9	9.45	7.87
	✗	✓	N/A	N/A	14.5	9.45	8.05
	✓	✗	N/A	N/A	12.5	9.01	7.74
	✓	✓	N/A	N/A	12.3	9.26	7.86

Table 10: Unconditional LSUN Bedrooms generative performance (measured in FID). Methods above the middle line use the same score model checkpoint; Learned Sampler uses a different one. (†): numbers are taken from literature.

Method	AFS	Denoising	NFEs=5	NFEs=10	NFEs=15	NFEs=20	NFEs=25
GENIE (ours)	✗	✗	74.1	17.1	13.3	11.6	11.1
	✗	✓	115	11.4	7.18	5.80	**5.35**
	✓	✗	55.9	18.4	14.1	12.3	11.6
	✓	✓	47.3	**9.29**	**6.83**	5.79	5.40
DDIM [58]	✗	✗	69.6	27.1	19.0	15.8	14.2
	✗	✓	81.0	16.3	9.18	7.12	6.20
	✓	✗	62.1	27.1	19.3	16.3	14.6
	✓	✓	42.5	12.5	8.21	6.77	6.05
S-PNDM [63]	✗	✗	70.4	22.1	15.7	13.5	12.4
	✗	✓	88.9	12.2	8.40	7.33	6.80
	✓	✗	48.0	20.2	15.2	13.4	12.4
	✓	✓	45.0	10.8	8.14	7.23	6.71
F-PNDM [63]	✗	✗	N/A	N/A	36.1	18.5	14.6
	✗	✓	N/A	N/A	26.8	9.85	7.86
	✓	✗	N/A	N/A	29.4	17.5	14.3
	✓	✓	N/A	N/A	18.9	9.27	7.69
Learned Sampler [66] (†)	✗	✓	**29.2**	11.0	-	**4.82**	-

Table 11: Unconditional LSUN Church-Outdoor generative performance (measured in FID). Methods above the middle line use the same score model checkpoint; Learned Sampler uses a different one. (†): numbers are taken from literature.

Method	AFS	Denoising	NFEs=5	NFEs=10	NFEs=15	NFEs=20	NFEs=25
GENIE (ours)	✗	✗	97.2	25.4	15.9	11.6	9.57
	✗	✓	147	13.7	11.7	8.52	7.28
	✓	✗	47.8	13.6	10.6	9.17	8.28
	✓	✓	60.3	**10.5**	**7.44**	**6.38**	**5.84**
DDIM [58]	✗	✗	81.5	28.5	16.7	11.9	9.9
	✗	✓	110	25.3	11.5	8.53	7.35
	✓	✗	44.0	17.4	12.5	10.2	9.07
	✓	✓	45.8	12.8	8.44	6.97	6.28
S-PNDM [63]	✗	✗	59.4	18.7	13.3	11.4	10.4
	✗	✓	87.5	14.8	9.54	7.98	7.21
	✓	✗	40.7	17.0	12.8	11.2	10.3
	✓	✓	48.8	12.9	9.10	7.82	7.12
F-PNDM [63]	✗	✗	N/A	N/A	15.5	12.0	10.6
	✗	✓	N/A	N/A	15.7	9.78	7.99
	✓	✗	N/A	N/A	15.2	11.8	10.4
	✓	✓	N/A	N/A	12.6	9.29	7.83
Learned Sampler [66] (†)	✗	✓	**30.2**	11.6	-	6.74	-

Table 12: Cats (base model) generative performance (measured in FID).

Method	AFS	NFEs=10	NFEs=15	NFEs=20	NFEs=25
GENIE (ours)	✗	**12.2**	**8.74**	**7.40**	6.84
	✓	13.3	9.07	7.76	**6.76**
DDIM [58]	✗	12.7	9.89	8.66	7.98
	✓	13.6	10.0	8.73	7.87
S-PNDM [63]	✗	12.8	11.6	10.8	10.4
	✓	12.5	11.3	10.7	10.2
F-PNDM [63]	✗	N/A	12.8	10.4	10.6
	✓	N/A	11.8	10.4	10.3

Table 13: Cats (upsampler) generative performance (measured in FID).

Method	AFS	NFEs=5	NFEs=10	NFEs=15
GENIE (ours)	✗	7.03	4.93	**4.83**
	✓	**5.53**	**4.90**	4.91
DDIM [58]	✗	11.3	7.16	5.99
	✓	9.47	6.64	5.85
S-PNDM [63]	✗	16.7	12.1	8.83
	✓	14.6	11.0	9.01
F-PNDM [63]	✗	N/A	N/A	12.9
	✓	N/A	N/A	11.7

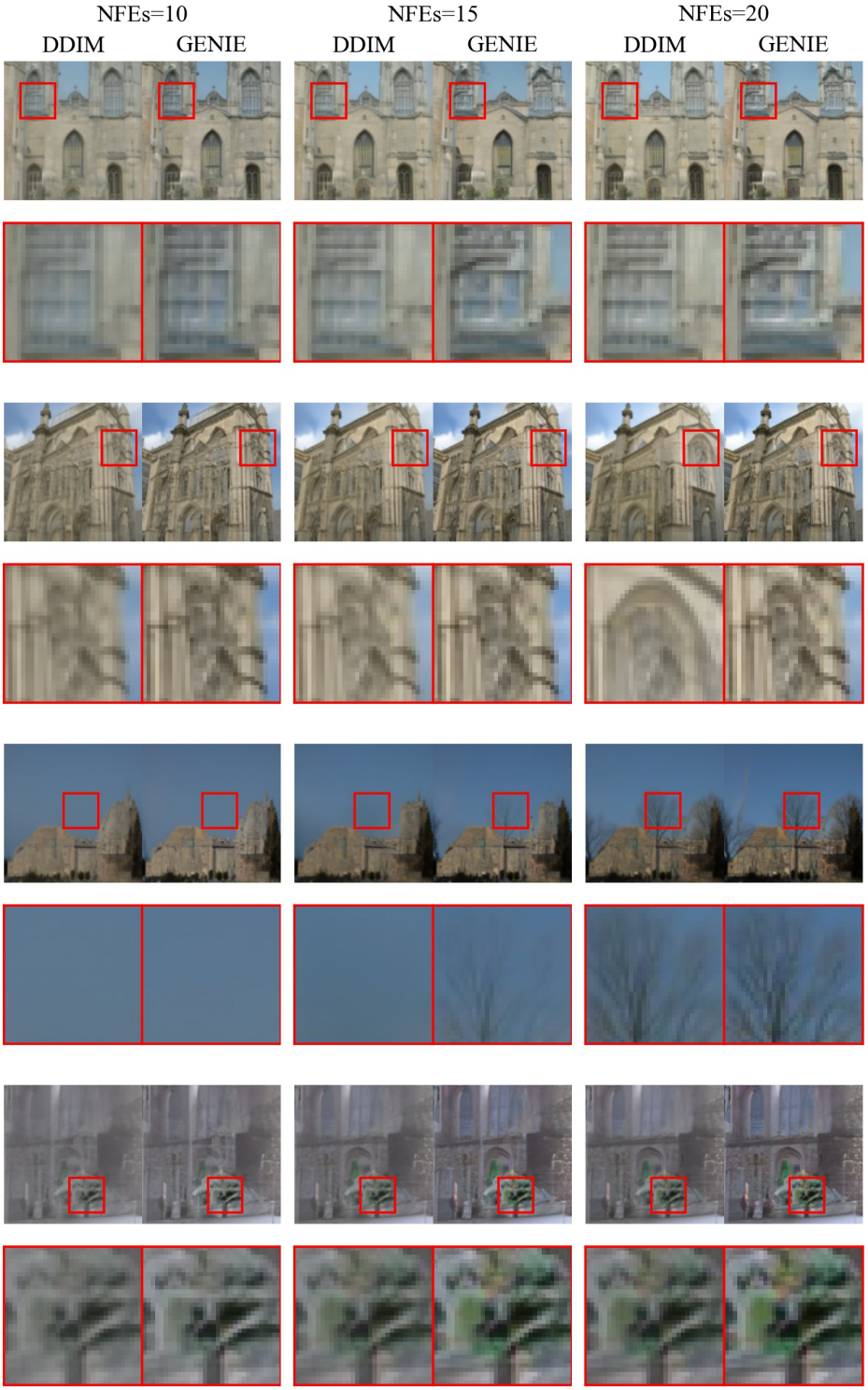


Figure 16: Additional samples on LSUN Church-Outdoor with zoom-in on details. GENIE often results in sharper and higher contrast samples compared to DDIM.

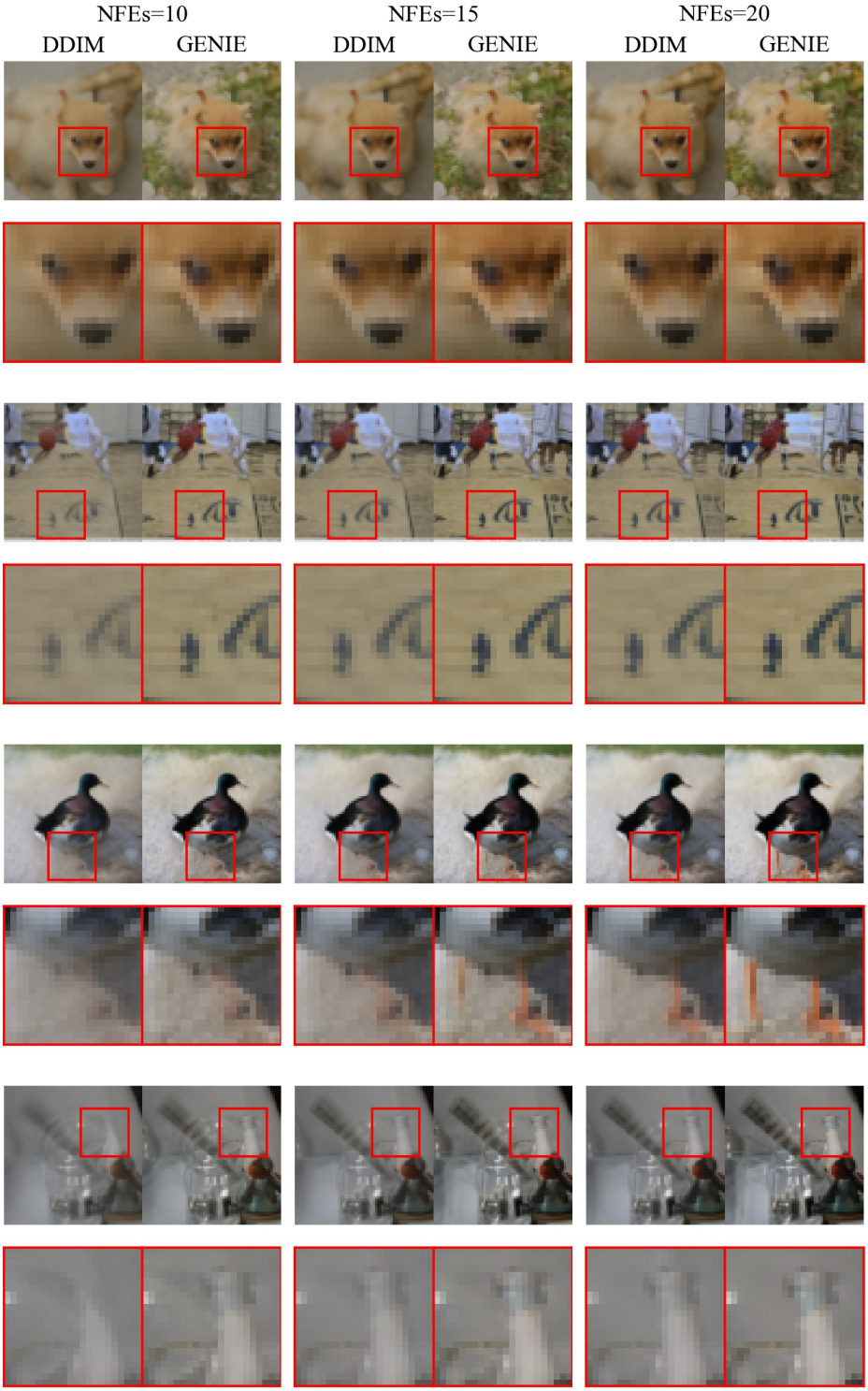


Figure 17: Additional samples on ImageNet with zoom-in on details. GENIE often results in sharper and higher contrast samples compared to DDIM.

Figure 18: Additional samples on Cats with zoom-in on details. GENIE often results in sharper and higher contrast samples compared to DDIM.

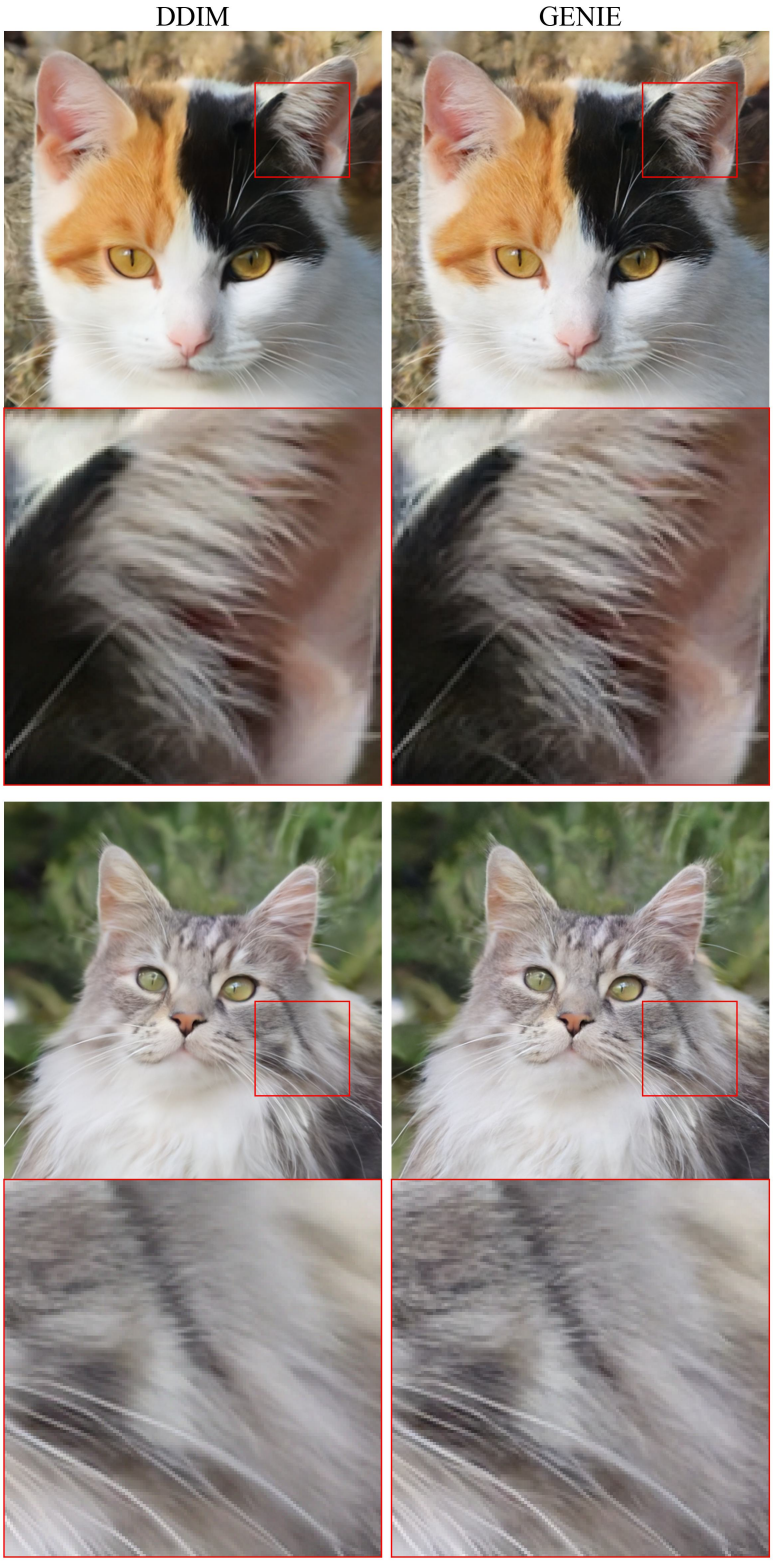


Figure 19: Additional samples on Cats with zoom-in on details. GENIE often results in sharper and higher contrast samples compared to DDIM.

Figure 20: End-to-end samples on Cats. The GENIE base model uses 25 function evaluations and the GENIE upsampler only uses five function evaluations. An upsampler evaluation is roughly four times as expensive as a base model evaluation.

G Miscellaneous

G.1 Connection to Bao et al. [96]

The concurrent Bao et al. [96] learn covariance matrices for diffusion model sampling using prediction heads somewhat similar to the ones in GENIE. Specifically, both Bao et al. [96] and GENIE use small prediction heads that operate on top of the large first-order score predictor. However, we would like to stress multiple differences: (i) Bao et al. [96] learn the DDM's sampling covariance matrices, while we learn higher-order ODE gradients. More generally, Bao et al. [96] rely on stochastic diffusion model sampling, while we use the ODE formulation. (ii) Most importantly, in our case we can resort to directly learning the low-dimensional JVPs without low-rank or diagonal matrix approximations or other assumptions. Similar techniques are not directly applicable in Bao et al. [96]'s setting. In detail, this is because in their case the relevant matrices (obtained after Cholesky or another applicable decomposition of the covariance) do not act on regular vectors but random noise variables. In other words, instead of using a deterministic JVP predictor (which takes $\mathbf{x}_t$ and t as inputs), as in GENIE, Bao et al. [96] would require to model an entire distribution for each $\mathbf{x}_t$ and t without explicitly forming high-dimensional Cholesky decomposition-based matrices, if they wanted to do something somewhat analogous to GENIE's novel JVP-based approach. As a consequence, Bao et al. [96] take another route to keeping the dimensionality of the additional network outputs manageable in practice. In particular, they resort to assuming a diagonal covariance matrix in their experiments. By directly learning JVPs, we never have to rely on such potentially limiting assumptions. (iii) Experimentally,

Figure 21: Upsampling 128×128 test set images using the GENIE upsampler with only five function evaluations.

Bao et al. [96] also consider fast sampling with few neural network calls. However, GENIE generally outperforms them (see, for example, their CIFAR10 results in their Table 2 for 10 and 25 NFE). This might indeed be due to the assumptions made by Bao et al. [96], which we avoid. Furthermore, their stochastic vs. our deterministic sampling may play a role, too.

G.2 Combining GENIE with Progressive Distillation

We speculate that GENIE could potentially be combined with Progressive Distillation [69]: In every distillation stage of [69], one could quickly train a small GENIE prediction head to model higher-order ODE gradients. This would then allow for larger and/or more accurate steps, whose results represent the distillation target (teacher) in the progressive distillation protocol. This may also reduce the number of required distillation stages. Overall, this could potentially speed up the cumbersome stage-wise distillation and maybe also lead to an accuracy and performance improvement. In particular, we could replace the DDIM predictions in Algorithm 2 of [69] with improved GENIE predictions.

Note that this approach would not be possible with multistep methods as proposed by Liu et al. [63]. Such techniques could not be used here, because they require the history of previous predictions, which are not available in the progressive distillation training scheme.

We leave exploration of this direction to future work.