Consistency Models

Yang Song [1] **Prafulla Dhariwal** [1] **Mark Chen** [1] **Ilya Sutskever** [1]

Abstract

Diffusion models have significantly advanced the fields of image, audio, and video generation, but they depend on an iterative sampling process that causes slow generation. To overcome this limitation, we propose *consistency models*, a new family of models that generate high quality samples by directly mapping noise to data. They support fast one-step generation by design, while still allowing multistep sampling to trade compute for sample quality. They also support zero-shot data editing, such as image inpainting, colorization, and super-resolution, without requiring explicit training on these tasks. Consistency models can be trained either by distilling pre-trained diffusion models, or as standalone generative models altogether. Through extensive experiments, we demonstrate that they outperform existing distillation techniques for diffusion models in one- and few-step sampling, achieving the new state-of-the-art FID of 3.55 on CIFAR-10 and 6.20 on ImageNet 64×64 for one-step generation. When trained in isolation, consistency models become a new family of generative models that can outperform existing one-step, non-adversarial generative models on standard benchmarks such as CIFAR-10, ImageNet 64×64 and LSUN 256×256.

1. Introduction

Diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; 2020; Ho et al., 2020; Song et al., 2021), also known as score-based generative models, have achieved unprecedented success across multiple fields, including image generation (Dhariwal & Nichol, 2021; Nichol et al., 2021; Ramesh et al., 2022; Saharia et al., 2022; Rombach et al., 2022), audio synthesis (Kong et al., 2020; Chen et al., 2021; Popov et al., 2021), and video generation (Ho et al., 2022b;a). A key feature of diffusion models is the iterative sampling process which progressively removes noise from random initial vectors. This iterative process provides a flexible trade-off of compute and sample quality, as using extra compute for more iterations usually yields samples of better quality. It is also the crux of many zero-shot data editing capabilities of diffusion models, enabling them to solve challenging inverse problems ranging from image inpainting, colorization, stroke-guided image editing, to Computed Tomography and Magnetic Resonance Imaging (Song & Ermon, 2019; Song et al., 2021; 2022; 2023; Kawar et al., 2021; 2022; Chung et al., 2023; Meng et al., 2021). However, compared to single-step generative models like GANs (Goodfellow et al., 2014), VAEs (Kingma & Welling, 2014; Rezende et al., 2014), or normalizing flows (Dinh et al., 2015; 2017; Kingma & Dhariwal, 2018), the iterative generation procedure of diffusion models typically requires 10–2000 times more compute for sample generation (Song & Ermon, 2020; Ho et al., 2020; Song et al., 2021; Zhang & Chen, 2022; Lu et al., 2022), causing slow inference and limited real-time applications.

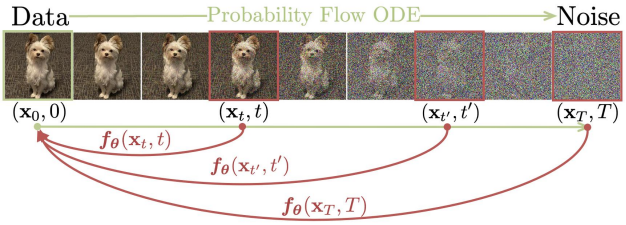


Figure 1: Given a Probability Flow (PF) ODE that smoothly converts data to noise, we learn to map any point (*e.g.*, $\mathbf{x}_t$, $\mathbf{x}_{t'}$, and $\mathbf{x}_T$) on the ODE trajectory to its origin (*e.g.*, $\mathbf{x}_0$) for generative modeling. Models of these mappings are called consistency models, as their outputs are trained to be consistent for points on the same trajectory.

Our objective is to create generative models that facilitate efficient, single-step generation without sacrificing important advantages of iterative sampling, such as trading compute for sample quality when necessary, as well as performing zero-shot data editing tasks. As illustrated in Fig. 1, we build on top of the probability flow (PF) ordinary differential equation (ODE) in continuous-time diffusion models (Song et al., 2021), whose trajectories smoothly transition

[1]OpenAI, San Francisco, CA 94110, USA. Correspondence to: Yang Song <songyang@openai.com>.

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

the data distribution into a tractable noise distribution. We propose to learn a model that maps any point at any time step to the trajectory's starting point. A notable property of our model is self-consistency: *points on the same trajectory map to the same initial point*. We therefore refer to such models as **consistency models**. Consistency models allow us to generate data samples (initial points of ODE trajectories, *e.g.*, $\mathbf{x}_0$ in Fig. 1) by converting random noise vectors (endpoints of ODE trajectories, *e.g.*, $\mathbf{x}_T$ in Fig. 1) with only one network evaluation. Importantly, by chaining the outputs of consistency models at multiple time steps, we can improve sample quality and perform zero-shot data editing at the cost of more compute, similar to what iterative sampling enables for diffusion models.

To train a consistency model, we offer two methods based on enforcing the self-consistency property. The first method relies on using numerical ODE solvers and a pre-trained diffusion model to generate pairs of adjacent points on a PF ODE trajectory. By minimizing the difference between model outputs for these pairs, we can effectively distill a diffusion model into a consistency model, which allows generating high-quality samples with one network evaluation. By contrast, our second method eliminates the need for a pre-trained diffusion model altogether, allowing us to train a consistency model in isolation. This approach situates consistency models as an independent family of generative models. Importantly, neither approach necessitates adversarial training, and they both place minor constraints on the architecture, allowing the use of flexible neural networks for parameterizing consistency models.

We demonstrate the efficacy of consistency models on several image datasets, including CIFAR-10 (Krizhevsky et al., 2009), ImageNet 64×64 (Deng et al., 2009), and LSUN 256×256 (Yu et al., 2015). Empirically, we observe that as a distillation approach, consistency models outperform existing diffusion distillation methods like progressive distillation (Salimans & Ho, 2022) across a variety of datasets in few-step generation: On CIFAR-10, consistency models reach new state-of-the-art FIDs of 3.55 and 2.93 for one-step and two-step generation; on ImageNet 64×64, it achieves record-breaking FIDs of 6.20 and 4.70 with one and two network evaluations respectively. When trained as standalone generative models, consistency models can match or surpass the quality of one-step samples from progressive distillation, despite having no access to pre-trained diffusion models. They are also able to outperform many GANs, and existing non-adversarial, single-step generative models across multiple datasets. Furthermore, we show that consistency models can be used to perform a wide range of zero-shot data editing tasks, including image denoising, interpolation, inpainting, colorization, super-resolution, and stroke-guided image editing (SDEdit, Meng et al. (2021)).

2. Diffusion Models

Consistency models are heavily inspired by the theory of continuous-time diffusion models (Song et al., 2021; Karras et al., 2022). Diffusion models generate data by progressively perturbing data to noise via Gaussian perturbations, then creating samples from noise via sequential denoising steps. Let $p_{\text{data}}(\mathbf{x})$ denote the data distribution. Diffusion models start by diffusing $p_{\text{data}}(\mathbf{x})$ with a stochastic differential equation (SDE) (Song et al., 2021)

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{\mu}(\mathbf{x}_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\mathbf{w}_t, \tag{1}$$

where $t \in [0, T]$, $T > 0$ is a fixed constant, $\boldsymbol{\mu}(\cdot, \cdot)$ and $\sigma(\cdot)$ are the drift and diffusion coefficients respectively, and $\{\mathbf{w}_t\}_{t\in[0,T]}$ denotes the standard Brownian motion. We denote the distribution of $\mathbf{x}_t$ as $p_t(\mathbf{x})$ and as a result $p_0(\mathbf{x}) \equiv p_{\text{data}}(\mathbf{x})$. A remarkable property of this SDE is the existence of an ordinary differential equation (ODE), dubbed the *Probability Flow (PF) ODE* by Song et al. (2021), whose solution trajectories sampled at t are distributed according to $p_t(\mathbf{x})$:

$$\mathrm{d}\mathbf{x}_t = \left[\boldsymbol{\mu}(\mathbf{x}_t, t) - \frac{1}{2}\sigma(t)^2 \nabla \log p_t(\mathbf{x}_t)\right]\mathrm{d}t. \tag{2}$$

Here $\nabla \log p_t(\mathbf{x})$ is the *score function* of $p_t(\mathbf{x})$; hence diffusion models are also known as *score-based generative models* (Song & Ermon, 2019; 2020; Song et al., 2021).

Typically, the SDE in Eq. (1) is designed such that $p_T(\mathbf{x})$ is close to a tractable Gaussian distribution $\pi(\mathbf{x})$. We hereafter adopt the settings in Karras et al. (2022), where $\boldsymbol{\mu}(\mathbf{x}, t) = \mathbf{0}$ and $\sigma(t) = \sqrt{2t}$. In this case, we have $p_t(\mathbf{x}) = p_{\text{data}}(\mathbf{x}) \otimes \mathcal{N}(\mathbf{0}, t^2\boldsymbol{I})$, where $\otimes$ denotes the convolution operation, and $\pi(\mathbf{x}) = \mathcal{N}(\mathbf{0}, T^2\boldsymbol{I})$. For sampling, we first train a *score model* $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t) \approx \nabla \log p_t(\mathbf{x})$ via *score matching* (Hyvärinen & Dayan, 2005; Vincent, 2011; Song et al., 2019; Song & Ermon, 2019; Ho et al., 2020), then plug it into Eq. (2) to obtain an empirical estimate of the PF ODE, which takes the form of

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -t\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t). \tag{3}$$

We call Eq. (3) the *empirical PF ODE*. Next, we sample $\hat{\mathbf{x}}_T \sim \pi = \mathcal{N}(\mathbf{0}, T^2\boldsymbol{I})$ to initialize the empirical PF ODE and solve it backwards in time with any numerical ODE solver, such as Euler (Song et al., 2020; 2021) and Heun solvers (Karras et al., 2022), to obtain the solution trajectory $\{\hat{\mathbf{x}}_t\}_{t\in[0,T]}$. The resulting $\hat{\mathbf{x}}_0$ can then be viewed as an approximate sample from the data distribution $p_{\text{data}}(\mathbf{x})$. To avoid numerical instability, one typically stops the solver at $t = \epsilon$, where ϵ is a fixed small positive number, and accepts $\hat{\mathbf{x}}_\epsilon$ as the approximate sample. Following Karras et al. (2022), we rescale image pixel values to $[-1, 1]$, and set $T = 80, \epsilon = 0.002$.

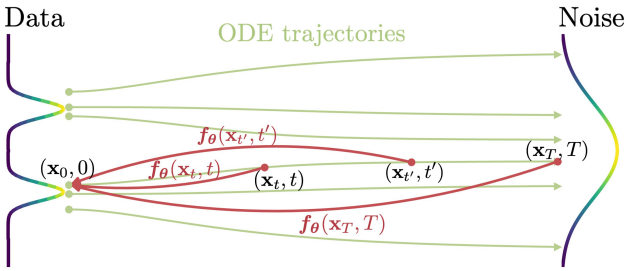


Figure 2: Consistency models are trained to map points on any trajectory of the PF ODE to the trajectory's origin.

Diffusion models are bottlenecked by their slow sampling speed. Clearly, using ODE solvers for sampling requires iterative evaluations of the score model $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)$, which is computationally costly. Existing methods for fast sampling include faster numerical ODE solvers (Song et al., 2020; Zhang & Chen, 2022; Lu et al., 2022; Dockhorn et al., 2022), and distillation techniques (Luhman & Luhman, 2021; Salimans & Ho, 2022; Meng et al., 2022; Zheng et al., 2022). However, ODE solvers still need more than 10 evaluation steps to generate competitive samples. Most distillation methods like Luhman & Luhman (2021) and Zheng et al. (2022) rely on collecting a large dataset of samples from the diffusion model prior to distillation, which itself is computationally expensive. To our best knowledge, the only distillation approach that does not suffer from this drawback is progressive distillation (PD, Salimans & Ho (2022)), with which we compare consistency models extensively in our experiments.

3. Consistency Models

We propose consistency models, a new type of models that support single-step generation at the core of its design, while still allowing iterative generation for trade-offs between sample quality and compute, and zero-shot data editing. Consistency models can be trained in either the distillation mode or the isolation mode. In the former case, consistency models distill the knowledge of pre-trained diffusion models into a single-step sampler, significantly improving other distillation approaches in sample quality, while allowing zero-shot image editing applications. In the latter case, consistency models are trained in isolation, with no dependence on pre-trained diffusion models. This makes them an independent new class of generative models.

Below we introduce the definition, parameterization, and sampling of consistency models, plus a brief discussion on their applications to zero-shot data editing.

Definition Given a solution trajectory $\{\mathbf{x}_t\}_{t\in[\epsilon,T]}$ of the PF ODE in Eq. (2), we define the *consistency function* as $\boldsymbol{f} : (\mathbf{x}_t, t) \mapsto \mathbf{x}_\epsilon$. A consistency function has the property of *self-consistency*: its outputs are consistent for arbitrary pairs of $(\mathbf{x}_t, t)$ that belong to the same PF ODE trajectory, *i.e.*, $\boldsymbol{f}(\mathbf{x}_t, t) = \boldsymbol{f}(\mathbf{x}_{t'}, t')$ for all $t, t' \in [\epsilon, T]$. As illustrated in Fig. 2, the goal of a *consistency model*, symbolized as $\boldsymbol{f}_{\boldsymbol{\theta}}$, is to estimate this consistency function $\boldsymbol{f}$ from data by learning to enforce the self-consistency property (details in Sections 4 and 5). Note that a similar definition is used for neural flows (Biloš et al., 2021) in the context of neural ODEs (Chen et al., 2018). Compared to neural flows, however, we do not enforce consistency models to be invertible.

Parameterization For any consistency function $\boldsymbol{f}(\cdot, \cdot)$, we have $\boldsymbol{f}(\mathbf{x}_\epsilon, \epsilon) = \mathbf{x}_\epsilon$, *i.e.*, $\boldsymbol{f}(\cdot, \epsilon)$ is an identity function. We call this constraint the *boundary condition*. All consistency models have to meet this boundary condition, as it plays a crucial role in the successful training of consistency models. This boundary condition is also the most confining architectural constraint on consistency models. For consistency models based on deep neural networks, we discuss two ways to implement this boundary condition *almost for free*. Suppose we have a free-form deep neural network $F_{\boldsymbol{\theta}}(\mathbf{x}, t)$ whose output has the same dimensionality as $\mathbf{x}$. The first way is to simply parameterize the consistency model as

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) = \begin{cases} \mathbf{x} & t = \epsilon \\ F_{\boldsymbol{\theta}}(\mathbf{x}, t) & t \in (\epsilon, T] \end{cases}. \tag{4}$$

The second method is to parameterize the consistency model using skip connections, that is,

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) = c_{\text{skip}}(t)\mathbf{x} + c_{\text{out}}(t)F_{\boldsymbol{\theta}}(\mathbf{x}, t), \tag{5}$$

where $c_{\text{skip}}(t)$ and $c_{\text{out}}(t)$ are differentiable functions such that $c_{\text{skip}}(\epsilon) = 1$, and $c_{\text{out}}(\epsilon) = 0$. This way, the consistency model is differentiable at $t = \epsilon$ if $F_{\boldsymbol{\theta}}(\mathbf{x}, t), c_{\text{skip}}(t), c_{\text{out}}(t)$ are all differentiable, which is critical for training continuous-time consistency models (Appendices B.1 and B.2). The parameterization in Eq. (5) bears strong resemblance to many successful diffusion models (Karras et al., 2022; Balaji et al., 2022), making it easier to borrow powerful diffusion model architectures for constructing consistency models. We therefore follow the second parameterization in all experiments.

Sampling With a well-trained consistency model $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot, \cdot)$, we can generate samples by sampling from the initial distribution $\hat{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, T^2\boldsymbol{I})$ and then evaluating the consistency model for $\hat{\mathbf{x}}_\epsilon = \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_T, T)$. This involves only one forward pass through the consistency model and therefore *generates samples in a single step*. Importantly, one can also evaluate the consistency model multiple times by alternating denoising and noise injection steps for improved sample quality. Summarized in Algorithm 1, this *multistep* sampling procedure provides the flexibility to trade compute for sample quality. It also has important applications in zero-shot data editing. In practice, we find time points

Algorithm 1 Multistep Consistency Sampling

Input: Consistency model $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot,\cdot)$, sequence of time points $\tau_1 > \tau_2 > \cdots > \tau_{N-1}$, initial noise $\hat{\mathbf{x}}_T$
$\mathbf{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_T, T)$
for $n = 1$ **to** $N-1$ **do**
  Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\hat{\mathbf{x}}_{\tau_n} \leftarrow \mathbf{x} + \sqrt{\tau_n^2 - \epsilon^2}\mathbf{z}$
  $\mathbf{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{\tau_n}, \tau_n)$
end for
Output: $\mathbf{x}$

$\{\tau_1, \tau_2, \cdots, \tau_{N-1}\}$ in Algorithm 1 with a greedy algorithm, where the time points are pinpointed one at a time using ternary search to optimize the FID of samples obtained from Algorithm 1. This assumes that given prior time points, the FID is a unimodal function of the next time point. We find this assumption to hold empirically in our experiments, and leave the exploration of better strategies as future work.

Zero-Shot Data Editing Similar to diffusion models, consistency models enable various data editing and manipulation applications in zero shot; they do not require explicit training to perform these tasks. For example, consistency models define a one-to-one mapping from a Gaussian noise vector to a data sample. Similar to latent variable models like GANs, VAEs, and normalizing flows, consistency models can easily interpolate between samples by traversing the latent space (Fig. 11). As consistency models are trained to recover $\mathbf{x}_\epsilon$ from any noisy input $\mathbf{x}_t$ where $t \in [\epsilon, T]$, they can perform denoising for various noise levels (Fig. 12). Moreover, the multistep generation procedure in Algorithm 1 is useful for solving certain inverse problems in zero shot by using an iterative replacement procedure similar to that of diffusion models (Song & Ermon, 2019; Song et al., 2021; Ho et al., 2022b). This enables many applications in the context of image editing, including inpainting (Fig. 10), colorization (Fig. 8), super-resolution (Fig. 6b) and stroke-guided image editing (Fig. 13) as in SDEdit (Meng et al., 2021). In Section 6.3, we empirically demonstrate the power of consistency models on many zero-shot image editing tasks.

4. Training Consistency Models via Distillation

We present our first method for training consistency models based on distilling a pre-trained score model $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)$. Our discussion revolves around the empirical PF ODE in Eq. (3), obtained by plugging the score model $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)$ into the PF ODE. Consider discretizing the time horizon $[\epsilon, T]$ into $N-1$ sub-intervals, with boundaries $t_1 = \epsilon < t_2 < \cdots < t_N = T$. In practice, we follow Karras et al. (2022) to determine the boundaries with the formula $t_i = (\epsilon^{1/\rho} + {}^{i-1}/_{N-1}(T^{1/\rho} - \epsilon^{1/\rho}))^\rho$, where $\rho = 7$. When N is sufficiently large, we can obtain an accurate estimate of $\mathbf{x}_{t_n}$ from $\mathbf{x}_{t_{n+1}}$ by running one discretization step of a numerical ODE solver. This estimate, which we denote as $\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}$, is defined by

$$\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n} := \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi}), \quad (6)$$

where $\Phi(\cdots; \boldsymbol{\phi})$ represents the update function of a one-step ODE solver applied to the empirical PF ODE. For example, when using the Euler solver, we have $\Phi(\mathbf{x}, t; \boldsymbol{\phi}) = -t\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)$ which corresponds to the following update rule

$$\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n} = \mathbf{x}_{t_{n+1}} - (t_n - t_{n+1})t_{n+1}\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}}, t_{n+1}).$$

For simplicity, we only consider one-step ODE solvers in this work. It is straightforward to generalize our framework to multistep ODE solvers and we leave it as future work.

Due to the connection between the PF ODE in Eq. (2) and the SDE in Eq. (1) (see Section 2), one can sample along the distribution of ODE trajectories by first sampling $\mathbf{x} \sim p_{\text{data}}$, then adding Gaussian noise to $\mathbf{x}$. Specifically, given a data point $\mathbf{x}$, we can generate a pair of adjacent data points $(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, \mathbf{x}_{t_{n+1}})$ on the PF ODE trajectory efficiently by sampling $\mathbf{x}$ from the dataset, followed by sampling $\mathbf{x}_{t_{n+1}}$ from the transition density of the SDE $\mathcal{N}(\mathbf{x}, t^2_{n+1}\boldsymbol{I})$, and then computing $\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}$ using one discretization step of the numerical ODE solver according to Eq. (6). Afterwards, we train the consistency model by minimizing its output differences on the pair $(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, \mathbf{x}_{t_{n+1}})$. This motivates our following *consistency distillation* loss for training consistency models.

Definition 1. *The consistency distillation loss is defined as*

$$\mathcal{L}^N_{CD}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) := \mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n))], \quad (7)$$

where the expectation is taken with respect to $\mathbf{x} \sim p_{data}$, $n \sim \mathcal{U}[\![1, N-1]\!]$, and $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t^2_{n+1}\boldsymbol{I})$. Here $\mathcal{U}[\![1, N-1]\!]$ denotes the uniform distribution over $\{1, 2, \cdots, N-1\}$, $\lambda(\cdot) \in \mathbb{R}^+$ is a positive weighting function, $\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}$ is given by Eq. (6), $\boldsymbol{\theta}^-$ denotes a running average of the past values of $\boldsymbol{\theta}$ during the course of optimization, and $d(\cdot,\cdot)$ is a metric function that satisfies $\forall \mathbf{x}, \mathbf{y} : d(\mathbf{x}, \mathbf{y}) \geqslant 0$ and $d(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$.

Unless otherwise stated, we adopt the notations in Definition 1 throughout this paper, and use $\mathbb{E}[\cdot]$ to denote the expectation over all random variables. In our experiments, we consider the squared ℓ_2 distance $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2$, ℓ_1 distance $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1$, and the Learned Perceptual Image Patch Similarity (LPIPS, Zhang et al. (2018)). We find $\lambda(t_n) \equiv 1$ performs well across all tasks and datasets. In practice, we minimize the objective by stochastic gradient descent on the model parameters $\boldsymbol{\theta}$, while updating $\boldsymbol{\theta}^-$ with exponential moving average (EMA). That is, given a decay

Algorithm 2 Consistency Distillation (CD)

Input: dataset $\mathcal{D}$, initial model parameter $\boldsymbol{\theta}$, learning rate η, ODE solver $\Phi(\cdot, \cdot; \boldsymbol{\phi})$, $d(\cdot, \cdot)$, $\lambda(\cdot)$, and μ
$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
repeat
  Sample $\mathbf{x} \sim \mathcal{D}$ and $n \sim \mathcal{U}[\![1, N-1]\!]$
  Sample $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t_{n+1}^2 \boldsymbol{I})$
  $\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n} \leftarrow \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi})$
  $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) \leftarrow$
    $\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n))$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$
  $\boldsymbol{\theta}^- \leftarrow \operatorname{stopgrad}(\mu \boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta})$
until convergence

Algorithm 3 Consistency Training (CT)

Input: dataset $\mathcal{D}$, initial model parameter $\boldsymbol{\theta}$, learning rate η, step schedule $N(\cdot)$, EMA decay rate schedule $\mu(\cdot)$, $d(\cdot, \cdot)$, and $\lambda(\cdot)$
$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$ and $k \leftarrow 0$
repeat
  Sample $\mathbf{x} \sim \mathcal{D}$, and $n \sim \mathcal{U}[\![1, N(k)-1]\!]$
  Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) \leftarrow$
    $\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n))$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$
  $\boldsymbol{\theta}^- \leftarrow \operatorname{stopgrad}(\mu(k)\boldsymbol{\theta}^- + (1-\mu(k))\boldsymbol{\theta})$
  $k \leftarrow k + 1$
until convergence

rate $0 \leqslant \mu < 1$, we perform the following update after each optimization step:

$$\boldsymbol{\theta}^- \leftarrow \operatorname{stopgrad}(\mu \boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta}). \tag{8}$$

The overall training procedure is summarized in Algorithm 2. In alignment with the convention in deep reinforcement learning (Mnih et al., 2013; 2015; Lillicrap et al., 2015) and momentum based contrastive learning (Grill et al., 2020; He et al., 2020), we refer to $\boldsymbol{f}_{\boldsymbol{\theta}^-}$ as the "target network", and $\boldsymbol{f}_{\boldsymbol{\theta}}$ as the "online network". We find that compared to simply setting $\boldsymbol{\theta}^- = \boldsymbol{\theta}$, the EMA update and "stopgrad" operator in Eq. (8) can greatly stabilize the training process and improve the final performance of the consistency model.

Below we provide a theoretical justification for consistency distillation based on asymptotic analysis.

Theorem 1. *Let $\Delta t \coloneqq \max_{n \in [\![1, N-1]\!]} \{|t_{n+1} - t_n|\}$, and $\boldsymbol{f}(\cdot, \cdot; \boldsymbol{\phi})$ be the consistency function of the empirical PF ODE in Eq. (3). Assume $\boldsymbol{f}_{\boldsymbol{\theta}}$ satisfies the Lipschitz condition: there exists $L > 0$ such that for all $t \in [\epsilon, T]$, $\mathbf{x}$, and $\mathbf{y}$, we have $\|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) - \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{y}, t)\|_2 \leqslant L \|\mathbf{x} - \mathbf{y}\|_2$. Assume further that for all $n \in [\![1, N-1]\!]$, the ODE solver called at t_{n+1} has local error uniformly bounded by $O((t_{n+1} - t_n)^{p+1})$ with $p \geqslant 1$. Then, if $\mathcal{L}^N_{CD}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, we have*

$$\sup_{n, \mathbf{x}} \|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t_n) - \boldsymbol{f}(\mathbf{x}, t_n; \boldsymbol{\phi})\|_2 = O((\Delta t)^p).$$

Proof. The proof is based on induction and parallels the classic proof of global error bounds for numerical ODE solvers (Süli & Mayers, 2003). We provide the full proof in Appendix A.2. □

Since $\boldsymbol{\theta}^-$ is a running average of the history of $\boldsymbol{\theta}$, we have $\boldsymbol{\theta}^- = \boldsymbol{\theta}$ when the optimization of Algorithm 2 converges. That is, the target and online consistency models will eventually match each other. If the consistency model additionally achieves zero consistency distillation loss, then Theorem 1 implies that, under some regularity conditions, the estimated consistency model can become arbitrarily accurate, as long as the step size of the ODE solver is sufficiently small. Importantly, our boundary condition $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, \epsilon) \equiv \mathbf{x}$ precludes the trivial solution $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) \equiv \mathbf{0}$ from arising in consistency model training.

The consistency distillation loss $\mathcal{L}^N_{\text{CD}}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$ can be extended to hold for infinitely many time steps ($N \to \infty$) if $\boldsymbol{\theta}^- = \boldsymbol{\theta}$ or $\boldsymbol{\theta}^- = \operatorname{stopgrad}(\boldsymbol{\theta})$. The resulting continuous-time loss functions do not require specifying N nor the time steps $\{t_1, t_2, \cdots, t_N\}$. Nonetheless, they involve Jacobian-vector products and require forward-mode automatic differentiation for efficient implementation, which may not be well-supported in some deep learning frameworks. We provide these continuous-time distillation loss functions in Theorems 3 to 5, and relegate details to Appendix B.1.

5. Training Consistency Models in Isolation

Consistency models can be trained without relying on any pre-trained diffusion models. This differs from existing diffusion distillation techniques, making consistency models a new independent family of generative models.

Recall that in consistency distillation, we rely on a pre-trained score model $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)$ to approximate the ground truth score function $\nabla \log p_t(\mathbf{x})$. It turns out that we can avoid this pre-trained score model altogether by leveraging the following unbiased estimator (Lemma 1 in Appendix A):

$$\nabla \log p_t(\mathbf{x}_t) = -\mathbb{E}\left[\frac{\mathbf{x}_t - \mathbf{x}}{t^2} \,\middle|\, \mathbf{x}_t\right],$$

where $\mathbf{x} \sim p_{\text{data}}$ and $\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}; t^2 \boldsymbol{I})$. That is, given $\mathbf{x}$ and $\mathbf{x}_t$, we can estimate $\nabla \log p_t(\mathbf{x}_t)$ with $-(\mathbf{x}_t - \mathbf{x})/t^2$.

This unbiased estimate suffices to replace the pre-trained diffusion model in consistency distillation when using the Euler method as the ODE solver in the limit of $N \to \infty$, as

justified by the following result.

Theorem 2. *Let* $\Delta t := \max_{n \in [\![1, N-1]\!]} \{|t_{n+1} - t_n|\}$. *Assume* d *and* $\boldsymbol{f}_{\boldsymbol{\theta}^-}$ *are both twice continuously differentiable with bounded second derivatives, the weighting function* $\lambda(\cdot)$ *is bounded, and* $\mathbb{E}[\|\nabla \log p_{t_n}(\mathbf{x}_{t_n})\|_2^2] < \infty$. *Assume further that we use the Euler ODE solver, and the pre-trained score model matches the ground truth,* i.e., $\forall t \in [\epsilon, T] : \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t) \equiv \nabla \log p_t(\mathbf{x})$. *Then,*

$$\mathcal{L}_{CD}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = \mathcal{L}_{CT}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-) + o(\Delta t), \tag{9}$$

where the expectation is taken with respect to $\mathbf{x} \sim p_{data}$, $n \sim \mathcal{U}[\![1, N-1]\!]$, *and* $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t_{n+1}^2 \boldsymbol{I})$. *The consistency training objective, denoted by* $\mathcal{L}_{CT}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$, *is defined as*

$$\mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n))], \tag{10}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. *Moreover,* $\mathcal{L}_{CT}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-) \geqslant O(\Delta t)$ *if* $\inf_N \mathcal{L}_{CD}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) > 0$.

Proof. The proof is based on Taylor series expansion and properties of score functions (Lemma 1). A complete proof is provided in Appendix A.3. □

We refer to Eq. (10) as the *consistency training* (CT) loss. Crucially, $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$ only depends on the online network $\boldsymbol{f}_{\boldsymbol{\theta}}$, and the target network $\boldsymbol{f}_{\boldsymbol{\theta}^-}$, while being completely agnostic to diffusion model parameters $\boldsymbol{\phi}$. The loss function $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) \geqslant O(\Delta t)$ decreases at a slower rate than the remainder $o(\Delta t)$ and thus will dominate the loss in Eq. (9) as $N \to \infty$ and $\Delta t \to 0$.

For improved practical performance, we propose to progressively increase N during training according to a schedule function $N(\cdot)$. The intuition (*cf*., Fig. 3d) is that the consistency training loss has less "variance" but more "bias" with respect to the underlying consistency distillation loss (*i.e*., the left-hand side of Eq. (9)) when N is small (*i.e*., Δt is large), which facilitates faster convergence at the beginning of training. On the contrary, it has more "variance" but less "bias" when N is large (*i.e*., Δt is small), which is desirable when closer to the end of training. For best performance, we also find that μ should change along with N, according to a schedule function $\mu(\cdot)$. The full algorithm of consistency training is provided in Algorithm 3, and the schedule functions used in our experiments are given in Appendix C.

Similar to consistency distillation, the consistency training loss $\mathcal{L}_{\text{CT}}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$ can be extended to hold in continuous time (*i.e*., $N \to \infty$) if $\boldsymbol{\theta}^- = \text{stopgrad}(\boldsymbol{\theta})$, as shown in Theorem 6. This continuous-time loss function does not require schedule functions for N or μ, but requires forward-mode automatic differentiation for efficient implementation. Unlike the discrete-time CT loss, there is no undesirable "bias" associated with the continuous-time objective, as we effectively take $\Delta t \to 0$ in Theorem 2. We relegate more details to Appendix B.2.

6. Experiments

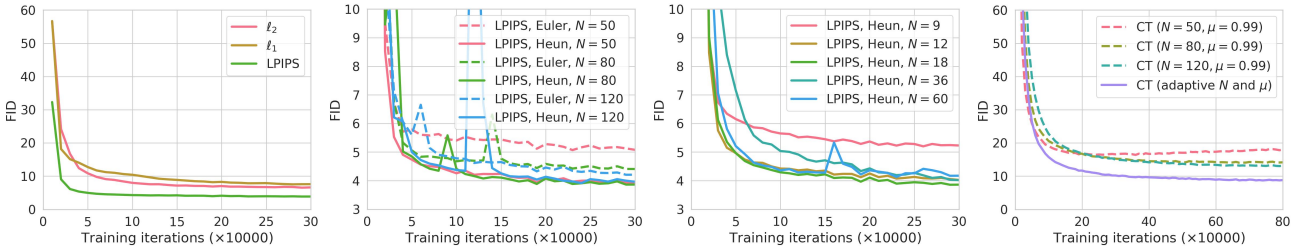
We employ consistency distillation and consistency training to learn consistency models on real image datasets, including CIFAR-10 (Krizhevsky et al., 2009), ImageNet 64×64 (Deng et al., 2009), LSUN Bedroom 256×256, and LSUN Cat 256×256 (Yu et al., 2015). Results are compared according to Fréchet Inception Distance (FID, Heusel et al. (2017), lower is better), Inception Score (IS, Salimans et al. (2016), higher is better), Precision (Prec., Kynkäänniemi et al. (2019), higher is better), and Recall (Rec., Kynkäänniemi et al. (2019), higher is better). Additional experimental details are provided in Appendix C.

6.1. Training Consistency Models

We perform a series of experiments on CIFAR-10 to understand the effect of various hyperparameters on the performance of consistency models trained by consistency distillation (CD) and consistency training (CT). We first focus on the effect of the metric function $d(\cdot, \cdot)$, the ODE solver, and the number of discretization steps N in CD, then investigate the effect of the schedule functions $N(\cdot)$ and $\mu(\cdot)$ in CT.

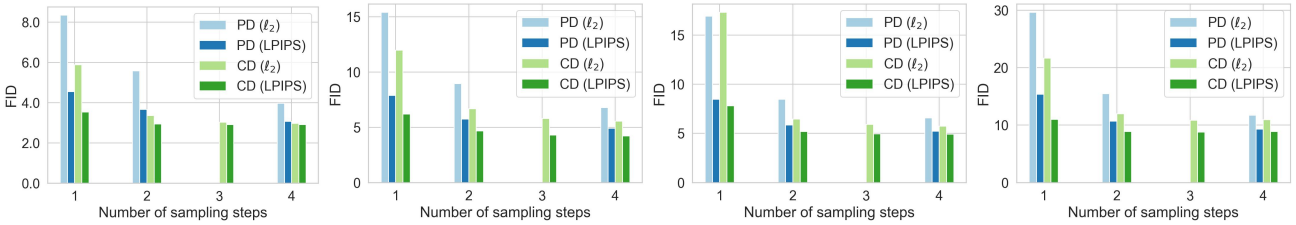
To set up our experiments for CD, we consider the squared ℓ_2 distance $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2$, ℓ_1 distance $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1$, and the Learned Perceptual Image Patch Similarity (LPIPS, Zhang et al. (2018)) as the metric function. For the ODE solver, we compare Euler's forward method and Heun's second order method as detailed in Karras et al. (2022). For the number of discretization steps N, we compare $N \in \{9, 12, 18, 36, 50, 60, 80, 120\}$. All consistency models trained by CD in our experiments are initialized with the corresponding pre-trained diffusion models, whereas models trained by CT are randomly initialized.

As visualized in Fig. 3a, the optimal metric for CD is LPIPS, which outperforms both ℓ_1 and ℓ_2 by a large margin over all training iterations. This is expected as the outputs of consistency models are images on CIFAR-10, and LPIPS is specifically designed for measuring the similarity between natural images. Next, we investigate which ODE solver and which discretization step N work the best for CD. As shown in Figs. 3b and 3c, Heun ODE solver and $N = 18$ are the best choices. Both are in line with the recommendation of Karras et al. (2022) despite the fact that we are training consistency models, not diffusion models. Moreover, Fig. 3b shows that with the same N, Heun's second order solver uniformly outperforms Euler's first order solver. This corroborates with Theorem 1, which states that the optimal consistency models trained by higher order ODE solvers have smaller estimation errors with the same N. The results of Fig. 3c also indicate that once N is sufficiently large, the performance of CD becomes insensitive to N. Given these insights, we hereafter use LPIPS and Heun ODE solver for CD unless otherwise stated. For N in CD, we follow the



(a) Metric functions in CD. (b) Solvers and N in CD. (c) N with Heun solver in CD. (d) Adaptive N and μ in CT.

Figure 3: Various factors that affect consistency distillation (CD) and consistency training (CT) on CIFAR-10. The best configuration for CD is LPIPS, Heun ODE solver, and $N = 18$. Our adaptive schedule functions for N and μ make CT converge significantly faster than fixing them to be constants during the course of optimization.



(a) CIFAR-10 (b) ImageNet 64×64 (c) Bedroom 256×256 (d) Cat 256×256

Figure 4: Multistep image generation with consistency distillation (CD). CD outperforms progressive distillation (PD) across all datasets and sampling steps. The only exception is single-step generation on Bedroom 256×256.

suggestions in Karras et al. (2022) on CIFAR-10 and ImageNet 64×64. We tune N separately on other datasets (details in Appendix C).

Due to the strong connection between CD and CT, we adopt LPIPS for our CT experiments throughout this paper. Unlike CD, there is no need for using Heun's second order solver in CT as the loss function does not rely on any particular numerical ODE solver. As demonstrated in Fig. 3d, the convergence of CT is highly sensitive to N—smaller N leads to faster convergence but worse samples, whereas larger N leads to slower convergence but better samples upon convergence. This matches our analysis in Section 5, and motivates our practical choice of progressively growing N and μ for CT to balance the trade-off between convergence speed and sample quality. As shown in Fig. 3d, adaptive schedules of N and μ significantly improve the convergence speed and sample quality of CT. In our experiments, we tune the schedules $N(\cdot)$ and $\mu(\cdot)$ separately for images of different resolutions, with more details in Appendix C.

6.2. Few-Step Image Generation

Distillation In current literature, the most directly comparable approach to our consistency distillation (CD) is progressive distillation (PD, Salimans & Ho (2022)); both are thus far the only distillation approaches that *do not construct synthetic data before distillation*. In stark contrast, other distillation techniques, such as knowledge distillation (Luhman & Luhman, 2021) and DFNO (Zheng et al., 2022), have to prepare a large synthetic dataset by generating numerous samples from the diffusion model with expensive numerical ODE/SDE solvers. We perform comprehensive comparison for PD and CD on CIFAR-10, ImageNet 64×64, and LSUN 256×256, with all results reported in Fig. 4. All methods distill from an EDM (Karras et al., 2022) model that we pre-trained in-house. We note that across all sampling iterations, *using the LPIPS metric uniformly improves PD compared to the squared ℓ_2 distance in the original paper of Salimans & Ho (2022)*. Both PD and CD improve as we take more sampling steps. We find that CD uniformly outperforms PD across all datasets, sampling steps, and metric functions considered, except for single-step generation on Bedroom 256×256, where CD with ℓ_2 slightly underperforms PD with ℓ_2. As shown in Table 1, CD even outperforms distillation approaches that require synthetic dataset construction, such as Knowledge Distillation (Luhman & Luhman, 2021) and DFNO (Zheng et al., 2022).

Direct Generation In Tables 1 and 2, we compare the sample quality of consistency training (CT) with other generative models using one-step and two-step generation. We also include PD and CD results for reference. Both tables report PD results obtained from the ℓ_2 metric function, as this is the default setting used in the original paper of Salimans

Table 1: Sample quality on CIFAR-10. *Methods that require synthetic data construction for distillation.

METHOD	NFE (↓)	FID (↓)	IS (↑)
Diffusion + Samplers			
DDIM (Song et al., 2020)	50	4.67	
DDIM (Song et al., 2020)	20	6.84	
DDIM (Song et al., 2020)	10	8.23	
DPM-solver-2 (Lu et al., 2022)	10	5.94	
DPM-solver-fast (Lu et al., 2022)	10	4.70	
3-DEIS (Zhang & Chen, 2022)	10	**4.17**	
Diffusion + Distillation			
Knowledge Distillation* (Luhman & Luhman, 2021)	1	9.36	
DFNO* (Zheng et al., 2022)	1	4.12	
1-Rectified Flow (+distill)* (Liu et al., 2022)	1	6.18	9.08
2-Rectified Flow (+distill)* (Liu et al., 2022)	1	4.85	9.01
3-Rectified Flow (+distill)* (Liu et al., 2022)	1	5.21	8.79
PD (Salimans & Ho, 2022)	1	8.34	8.69
CD	1	**3.55**	**9.48**
PD (Salimans & Ho, 2022)	2	5.58	9.05
CD	2	**2.93**	**9.75**
Direct Generation			
BigGAN (Brock et al., 2019)	1	14.7	9.22
Diffusion GAN (Xiao et al., 2022)	1	14.6	8.93
AutoGAN (Gong et al., 2019)	1	12.4	8.55
E2GAN (Tian et al., 2020)	1	11.3	8.51
ViTGAN (Lee et al., 2021)	1	6.66	9.30
TransGAN (Jiang et al., 2021)	1	9.26	9.05
StyleGAN2-ADA (Karras et al., 2020)	1	2.92	**9.83**
StyleGAN-XL (Sauer et al., 2022)	1	**1.85**	
Score SDE (Song et al., 2021)	2000	2.20	**9.89**
DDPM (Ho et al., 2020)	1000	3.17	9.46
LSGM (Vahdat et al., 2021)	147	2.10	
PFGM (Xu et al., 2022)	110	2.35	9.68
EDM (Karras et al., 2022)	35	**2.04**	9.84
1-Rectified Flow (Liu et al., 2022)	1	378	1.13
Glow (Kingma & Dhariwal, 2018)	1	48.9	3.92
Residual Flow (Chen et al., 2019)	1	46.4	
GLFlow (Xiao et al., 2019)	1	44.6	
DenseFlow (Grcić et al., 2021)	1	34.9	
DC-VAE (Parmar et al., 2021)	1	17.9	8.20
CT	1	**8.70**	**8.49**
CT	2	**5.83**	**8.85**

Table 2: Sample quality on ImageNet 64×64, and LSUN Bedroom & Cat 256×256. †Distillation techniques.

METHOD	NFE (↓)	FID (↓)	Prec. (↑)	Rec. (↑)
ImageNet 64×64				
PD† (Salimans & Ho, 2022)	1	15.39	0.59	0.62
DFNO† (Zheng et al., 2022)	1	8.35		
CD†	1	6.20	0.68	0.63
PD† (Salimans & Ho, 2022)	2	8.95	0.63	**0.65**
CD†	2	**4.70**	**0.69**	0.64
ADM (Dhariwal & Nichol, 2021)	250	**2.07**	0.74	0.63
EDM (Karras et al., 2022)	79	2.44	0.71	**0.67**
BigGAN-deep (Brock et al., 2019)	1	4.06	**0.79**	0.48
CT	1	13.0	0.71	0.47
CT	2	11.1	0.69	0.56
LSUN Bedroom 256×256				
PD† (Salimans & Ho, 2022)	1	16.92	0.47	0.27
PD† (Salimans & Ho, 2022)	2	8.47	0.56	**0.39**
CD†	1	7.80	0.66	0.34
CD†	2	**5.22**	**0.68**	**0.39**
DDPM (Ho et al., 2020)	1000	4.89	0.60	0.45
ADM (Dhariwal & Nichol, 2021)	1000	**1.90**	0.66	**0.51**
EDM (Karras et al., 2022)	79	3.57	0.66	0.45
PGGAN (Karras et al., 2018)	1	8.34		
PG-SWGAN (Wu et al., 2019)	1	8.0		
TDPM (GAN) (Zheng et al., 2023)	1	5.24		
StyleGAN2 (Karras et al., 2020)	1	2.35	0.59	0.48
CT	1	16.0	0.60	0.17
CT	2	7.85	**0.68**	0.33
LSUN Cat 256×256				
PD† (Salimans & Ho, 2022)	1	29.6	0.51	0.25
PD† (Salimans & Ho, 2022)	2	15.5	0.59	0.36
CD†	1	11.0	0.65	0.36
CD†	2	**8.84**	**0.66**	**0.40**
DDPM (Ho et al., 2020)	1000	17.1	0.53	0.48
ADM (Dhariwal & Nichol, 2021)	1000	**5.57**	0.63	**0.52**
EDM (Karras et al., 2022)	79	6.69	**0.70**	0.43
PGGAN (Karras et al., 2018)	1	37.5		
StyleGAN2 (Karras et al., 2020)	1	7.25	0.58	0.43
CT	1	20.7	0.56	0.23
CT	2	11.7	0.63	0.36

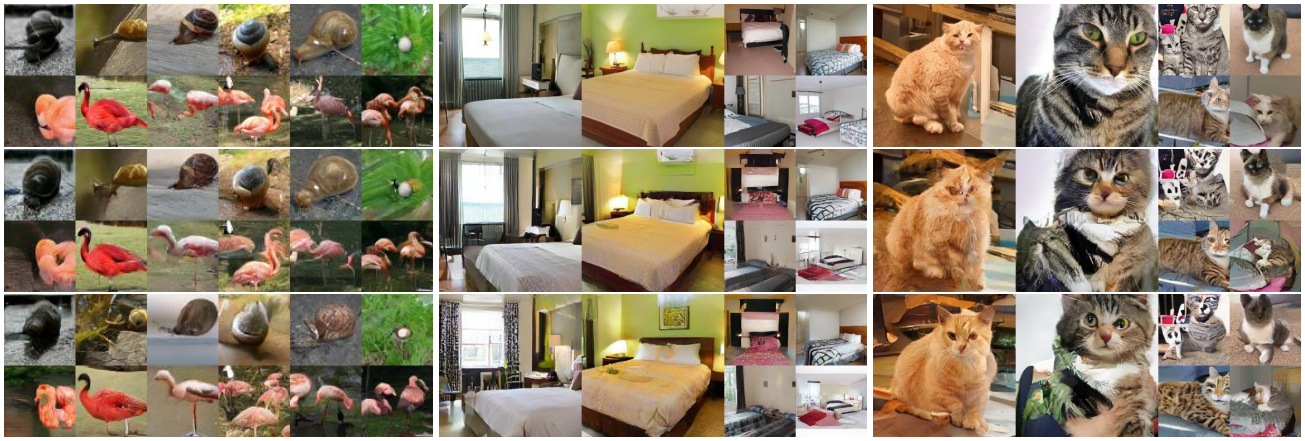
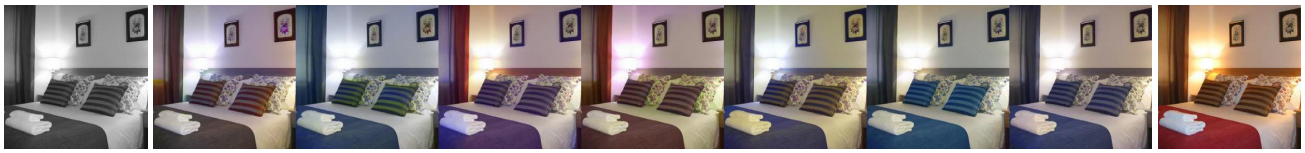
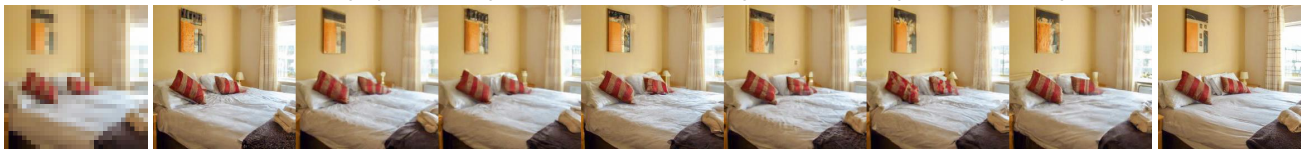


Figure 5: Samples generated by EDM (*top*), CT + single-step generation (*middle*), and CT + 2-step generation (*Bottom*). All corresponding images are generated from the same initial noise.



(a) *Left*: The gray-scale image. *Middle*: Colorized images. *Right*: The ground-truth image.



(b) *Left*: The downsampled image (32×32). *Middle*: Full resolution images (256×256). *Right*: The ground-truth image (256×256).

(c) *Left*: A stroke input provided by users. *Right*: Stroke-guided image generation.

Figure 6: Zero-shot image editing with a consistency model trained by consistency distillation on LSUN Bedroom 256×256.

& Ho (2022). For fair comparison, we ensure PD and CD distill the same EDM models. In Tables 1 and 2, we observe that CT outperforms existing single-step, non-adversarial generative models, *i.e.*, VAEs and normalizing flows, by a significant margin on CIFAR-10. Moreover, *CT achieves comparable quality to one-step samples from PD without relying on distillation*. In Fig. 5, we provide EDM samples (top), single-step CT samples (middle), and two-step CT samples (bottom). In Appendix E, we show additional samples for both CD and CT in Figs. 14 to 21. Importantly, *all samples obtained from the same initial noise vector share significant structural similarity*, even though CT and EDM models are trained independently from one another. This indicates that CT is less likely to suffer from mode collapse, as EDMs do not.

6.3. Zero-Shot Image Editing

Similar to diffusion models, consistency models allow zero-shot image editing by modifying the multistep sampling process in Algorithm 1. We demonstrate this capability with a consistency model trained on the LSUN bedroom dataset using consistency distillation. In Fig. 6a, we show such a consistency model can colorize gray-scale bedroom images at test time, even though it has never been trained on colorization tasks. In Fig. 6b, we show the same consistency model can generate high-resolution images from low-resolution inputs. In Fig. 6c, we additionally demonstrate that it can generate images based on stroke inputs created by humans, as in SDEdit for diffusion models (Meng et al., 2021). Again, this editing capability is zero-shot, as the model has not been trained on stroke inputs. In Appendix D, we additionally demonstrate the zero-shot capability of consistency models on inpainting (Fig. 10), interpolation (Fig. 11) and denoising (Fig. 12), with more examples on colorization (Fig. 8), super-resolution (Fig. 9) and stroke-guided image generation (Fig. 13).

7. Conclusion

We have introduced consistency models, a type of generative models that are specifically designed to support one-step and few-step generation. We have empirically demonstrated that our consistency distillation method outshines the existing distillation techniques for diffusion models on multiple image benchmarks and small sampling iterations. Furthermore, as a standalone generative model, consistency models generate better samples than existing single-step generation models except for GANs. Similar to diffusion models, they also allow zero-shot image editing applications such as inpainting, colorization, super-resolution, denoising, interpolation, and stroke-guided image generation.

In addition, consistency models share striking similarities with techniques employed in other fields, including deep Q-learning (Mnih et al., 2015) and momentum-based contrastive learning (Grill et al., 2020; He et al., 2020). This offers exciting prospects for cross-pollination of ideas and methods among these diverse fields.

Acknowledgements

We thank Alex Nichol for reviewing the manuscript and providing valuable feedback, Chenlin Meng for providing stroke inputs needed in our stroke-guided image generation experiments, and the OpenAI Algorithms team.

References

Balaji, Y., Nah, S., Huang, X., Vahdat, A., Song, J., Kreis, K., Aittala, M., Aila, T., Laine, S., Catanzaro, B., Karras, T., and Liu, M.-Y. ediff-i: Text-to-image diffusion models with ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Biloš, M., Sommer, J., Rangapuram, S. S., Januschowski, T., and Günnemann, S. Neural flows: Efficient alternative to neural odes. *Advances in Neural Information Processing Systems*, 34:21325–21337, 2021.

Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1xsqj09Fm.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations (ICLR)*, 2021.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural Ordinary Differential Equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Chen, R. T., Behrmann, J., Duvenaud, D. K., and Jacobsen, J.-H. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, pp. 9916–9926, 2019.

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OnD9zGAGT0k.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. *International Conference in Learning Representations Workshop Track*, 2015.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Dockhorn, T., Vahdat, A., and Kreis, K. Genie: Higher-order denoising diffusion solvers. *arXiv preprint arXiv:2210.05475*, 2022.

Gong, X., Chang, S., Jiang, Y., and Wang, Z. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3224–3234, 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Grcić, M., Grubišić, I., and Šegvić, S. Densely connected normalizing flows. *Advances in Neural Information Processing Systems*, 34:23968–23982, 2021.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems*, 33, 2020.

Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J., et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022a.

Ho, J., Salimans, T., Gritsenko, A. A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2022b. URL https://openreview.net/forum?id=BBelR2NdDZ5.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research (JMLR)*, 6(4), 2005.

Jiang, Y., Chang, S., and Wang, Z. Transgan: Two pure transformers can make one strong gan, and that can scale up. *Advances in Neural Information Processing Systems*, 34:14745–14758, 2021.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk99zCeAb.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of stylegan. 2020.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022.

Kawar, B., Vaksman, G., and Elad, M. Snips: Solving noisy inverse problems stochastically. *arXiv preprint arXiv:2105.14951*, 2021.

Kawar, B., Elad, M., Ermon, S., and Song, J. Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems*, 2022.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10215–10224. 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. DiffWave: A Versatile Diffusion Model for Audio Synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., and Aila, T. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

Lee, K., Chang, H., Jiang, L., Zhang, H., Tu, Z., and Liu, C. Vitgan: Training gans with vision transformers. *arXiv preprint arXiv:2107.04589*, 2021.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *arXiv preprint arXiv:2206.00927*, 2022.

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

Meng, C., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. Sdedit: Image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

Meng, C., Gao, R., Kingma, D. P., Ermon, S., Ho, J., and Salimans, T. On distillation of guided diffusion models. *arXiv preprint arXiv:2210.03142*, 2022.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Parmar, G., Li, D., Lee, K., and Tu, Z. Dual contradistinctive generative autoencoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 823–832, 2021.

Popov, V., Vovk, I., Gogoryan, V., Sadekova, T., and Kudinov, M. Grad-TTS: A diffusion probabilistic model for text-to-speech. *arXiv preprint arXiv:2105.06337*, 2021.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 1278–1286, 2014.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

Sauer, A., Schwarz, K., and Geiger, A. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep Unsupervised Learning Using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265, 2015.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Song, J., Vahdat, A., Mardani, M., and Kautz, J. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9_gsMA8MRKQ.

Song, Y. and Ermon, S. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, pp. 11918–11930, 2019.

Song, Y. and Ermon, S. Improved Techniques for Training Score-Based Generative Models. *Advances in Neural Information Processing Systems*, 33, 2020.

Song, Y., Garg, S., Shi, J., and Ermon, S. Sliced score matching: A scalable approach to density and score estimation. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, pp. 204, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Song, Y., Shen, L., Xing, L., and Ermon, S. Solving inverse problems in medical imaging with score-based generative models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vaRCHVj0uGI.

Süli, E. and Mayers, D. F. *An introduction to numerical analysis*. Cambridge university press, 2003.

Tian, Y., Wang, Q., Huang, Z., Li, W., Dai, D., Yang, M., Wang, J., and Fink, O. Off-policy reinforcement learning for efficient and effective gan architecture search. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pp. 175–192. Springer, 2020.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

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

Wu, J., Huang, Z., Acharya, D., Li, W., Thoma, J., Paudel, D. P., and Gool, L. V. Sliced wasserstein generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3713–3722, 2019.

Xiao, Z., Yan, Q., and Amit, Y. Generative latent flow. *arXiv preprint arXiv:1905.10485*, 2019.

Xiao, Z., Kreis, K., and Vahdat, A. Tackling the generative learning trilemma with denoising diffusion GANs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=JprM0p-q0Co.

Xu, Y., Liu, Z., Tegmark, M., and Jaakkola, T. S. Poisson flow generative models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=voV_TRqcWh.

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Zhang, Q. and Chen, Y. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Zheng, H., Nie, W., Vahdat, A., Azizzadenesheli, K., and Anandkumar, A. Fast sampling of diffusion models via operator learning. *arXiv preprint arXiv:2211.13449*, 2022.

Zheng, H., He, P., Chen, W., and Zhou, M. Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=HDxgaKk956l.

Contents

Appendices

A. Proofs

A.1. Notations

We use $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ to denote a consistency model parameterized by $\boldsymbol{\theta}$, and $\boldsymbol{f}(\mathbf{x}, t; \boldsymbol{\phi})$ the consistency function of the empirical PF ODE in Eq. (3). Here $\boldsymbol{\phi}$ symbolizes its dependency on the pre-trained score model $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)$. For the consistency function of the PF ODE in Eq. (2), we denote it as $\boldsymbol{f}(\mathbf{x}, t)$. Given a multi-variate function $\boldsymbol{h}(\mathbf{x}, \mathbf{y})$, we let $\partial_1 \boldsymbol{h}(\mathbf{x}, \mathbf{y})$ denote the Jacobian of $\boldsymbol{h}$ over $\mathbf{x}$, and analogously $\partial_2 \boldsymbol{h}(\mathbf{x}, \mathbf{y})$ denote the Jacobian of $\boldsymbol{h}$ over $\mathbf{y}$. Unless otherwise stated, $\mathbf{x}$ is supposed to be a random variable sampled from the data distribution $p_{\text{data}}(\mathbf{x})$, n is sampled uniformly at random from $[\![1, N-1]\!]$, and $\mathbf{x}_{t_n}$ is sampled from $\mathcal{N}(\mathbf{x}; t_n^2 \boldsymbol{I})$. Here $[\![1, N-1]\!]$ represents the set of integers $\{1, 2, \cdots, N-1\}$. Furthermore, recall that we define

$$\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n} \coloneqq \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1}) \Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi}),$$

where $\Phi(\cdots; \boldsymbol{\phi})$ denotes the update function of a one-step ODE solver for the empirical PF ODE defined by the score model $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)$. By default, $\mathbb{E}[\cdot]$ denotes the expectation over all relevant random variables in the expression.

A.2. Consistency Distillation

Theorem 1. *Let $\Delta t \coloneqq \max_{n \in [\![1, N-1]\!]} \{|t_{n+1} - t_n|\}$, and $\boldsymbol{f}(\cdot, \cdot; \boldsymbol{\phi})$ be the consistency function of the empirical PF ODE in Eq. (3). Assume $\boldsymbol{f}_{\boldsymbol{\theta}}$ satisfies the Lipschitz condition: there exists $L > 0$ such that for all $t \in [\epsilon, T]$, $\mathbf{x}$, and $\mathbf{y}$, we have $\|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) - \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{y}, t)\|_2 \leqslant L \|\mathbf{x} - \mathbf{y}\|_2$. Assume further that for all $n \in [\![1, N-1]\!]$, the ODE solver called at t_{n+1} has local error uniformly bounded by $O((t_{n+1} - t_n)^{p+1})$ with $p \geqslant 1$. Then, if $\mathcal{L}^N_{CD}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, we have*

$$\sup_{n, \mathbf{x}} \|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t_n) - \boldsymbol{f}(\mathbf{x}, t_n; \boldsymbol{\phi})\|_2 = O((\Delta t)^p).$$

Proof. From $\mathcal{L}^N_{\text{CD}}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, we have

$$\mathcal{L}^N_{\text{CD}}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = \mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n))] = 0. \tag{11}$$

According to the definition, we have $p_{t_n}(\mathbf{x}_{t_n}) = p_{\text{data}}(\mathbf{x}) \otimes \mathcal{N}(\mathbf{0}, t_n^2 \boldsymbol{I})$ where $t_n \geqslant \epsilon > 0$. It follows that $p_{t_n}(\mathbf{x}_{t_n}) > 0$ for every $\mathbf{x}_{t_n}$ and $1 \leqslant n \leqslant N$. Therefore, Eq. (11) entails

$$\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)) \equiv 0. \tag{12}$$

Because $\lambda(\cdot) > 0$ and $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$, this further implies that

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) \equiv \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n). \tag{13}$$

Now let $\boldsymbol{e}_n$ represent the error vector at t_n, which is defined as

$$\boldsymbol{e}_n \coloneqq \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) - \boldsymbol{f}(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi}).$$

We can easily derive the following recursion relation

$$\boldsymbol{e}_{n+1} = \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi})$$

$$\begin{aligned}
&\overset{(i)}{=} \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n) - \boldsymbol{f}(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi}) \\
&= \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) + \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) - \boldsymbol{f}(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi}) \\
&= \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) + \boldsymbol{e}_n, \qquad (14)
\end{aligned}$$

where (i) is due to Eq. (13) and $\boldsymbol{f}(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi}) = \boldsymbol{f}(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi})$. Because $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot, t_n)$ has Lipschitz constant L, we have

$$\begin{aligned}
\|\boldsymbol{e}_{n+1}\|_2 &\leqslant \|\boldsymbol{e}_n\|_2 + L \left\| \hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n} - \mathbf{x}_{t_n} \right\|_2 \\
&\overset{(i)}{=} \|\boldsymbol{e}_n\|_2 + L \cdot O((t_{n+1} - t_n)^{p+1}) \\
&= \|\boldsymbol{e}_n\|_2 + O((t_{n+1} - t_n)^{p+1}),
\end{aligned}$$

where (i) holds because the ODE solver has local error bounded by $O((t_{n+1} - t_n)^{p+1})$. In addition, we observe that $\boldsymbol{e}_1 = \mathbf{0}$, because

$$\begin{aligned}
\boldsymbol{e}_1 &= \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_1}, t_1) - \boldsymbol{f}(\mathbf{x}_{t_1}, t_1; \boldsymbol{\phi}) \\
&\overset{(i)}{=} \mathbf{x}_{t_1} - \boldsymbol{f}(\mathbf{x}_{t_1}, t_1; \boldsymbol{\phi}) \\
&\overset{(ii)}{=} \mathbf{x}_{t_1} - \mathbf{x}_{t_1} \\
&= \mathbf{0}.
\end{aligned}$$

Here (i) is true because the consistency model is parameterized such that $\boldsymbol{f}(\mathbf{x}_{t_1}, t_1; \boldsymbol{\phi}) = \mathbf{x}_{t_1}$ and (ii) is entailed by the definition of $\boldsymbol{f}(\cdot, \cdot; \boldsymbol{\phi})$. This allows us to perform induction on the recursion formula Eq. (14) to obtain

$$\begin{aligned}
\|\boldsymbol{e}_n\|_2 &\leqslant \|\boldsymbol{e}_1\|_2 + \sum_{k=1}^{n-1} O((t_{k+1} - t_k)^{p+1}) \\
&= \sum_{k=1}^{n-1} O((t_{k+1} - t_k)^{p+1}) \\
&= \sum_{k=1}^{n-1} (t_{k+1} - t_k) O((t_{k+1} - t_k)^{p}) \\
&\leqslant \sum_{k=1}^{n-1} (t_{k+1} - t_k) O((\Delta t)^{p}) \\
&= O((\Delta t)^p) \sum_{k=1}^{n-1} (t_{k+1} - t_k) \\
&= O((\Delta t)^p)(t_n - t_1) \\
&\leqslant O((\Delta t)^p)(T - \epsilon) \\
&= O((\Delta t)^p),
\end{aligned}$$

which completes the proof. □

A.3. Consistency Training

The following lemma provides an unbiased estimator for the score function, which is crucial to our proof for Theorem 2.

Lemma 1. *Let* $\mathbf{x} \sim p_{data}(\mathbf{x})$, $\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}; t^2\boldsymbol{I})$, *and* $p_t(\mathbf{x}_t) = p_{data}(\mathbf{x}) \otimes \mathcal{N}(\mathbf{0}, t^2\boldsymbol{I})$. *We have* $\nabla \log p_t(\mathbf{x}) = -\mathbb{E}[\frac{\mathbf{x}_t - \mathbf{x}}{t^2} \mid \mathbf{x}_t]$.

Proof. According to the definition of $p_t(\mathbf{x}_t)$, we have $\nabla \log p_t(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log \int p_{\text{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x}) \, \mathrm{d}\mathbf{x}$, where $p(\mathbf{x}_t \mid \mathbf{x}) = \mathcal{N}(\mathbf{x}_t; \mathbf{x}, t^2\boldsymbol{I})$. This expression can be further simplified to yield

$$\nabla \log p_t(\mathbf{x}_t) = \frac{\int p_{\text{data}}(\mathbf{x}) \nabla_{\mathbf{x}_t} p(\mathbf{x}_t \mid \mathbf{x}) \, \mathrm{d}\mathbf{x}}{\int p_{\text{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x}) \, \mathrm{d}\mathbf{x}}$$

$$
\begin{aligned}
&= \frac{\int p_{\text{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x}) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid \mathbf{x}) \, \mathrm{d}\mathbf{x}}{\int p_{\text{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x}) \, \mathrm{d}\mathbf{x}} \\
&= \frac{\int p_{\text{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x}) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid \mathbf{x}) \, \mathrm{d}\mathbf{x}}{p_t(\mathbf{x}_t)} \\
&= \int \frac{p_{\text{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x})}{p_t(\mathbf{x}_t)} \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid \mathbf{x}) \, \mathrm{d}\mathbf{x} \\
&\overset{(i)}{=} \int p(\mathbf{x} \mid \mathbf{x}_t) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid \mathbf{x}) \, \mathrm{d}\mathbf{x} \\
&= \mathbb{E}[\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid \mathbf{x}) \mid \mathbf{x}_t] \\
&= -\mathbb{E}\left[\frac{\mathbf{x}_t - \mathbf{x}}{t^2} \mid \mathbf{x}_t \right],
\end{aligned}
$$

where (i) is due to Bayes' rule. □

Theorem 2. *Let $\Delta t \coloneqq \max_{n \in [\![1, N-1]\!]} \{|t_{n+1} - t_n|\}$. Assume d and $\boldsymbol{f}_{\boldsymbol{\theta}^-}$ are both twice continuously differentiable with bounded second derivatives, the weighting function $\lambda(\cdot)$ is bounded, and $\mathbb{E}[\|\nabla \log p_{t_n}(\mathbf{x}_{t_n})\|_2^2] < \infty$. Assume further that we use the Euler ODE solver, and the pre-trained score model matches the ground truth,* i.e., *$\forall t \in [\epsilon, T] : \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t) \equiv \nabla \log p_t(\mathbf{x})$. Then,*

$$
\mathcal{L}_{CD}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = \mathcal{L}_{CT}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-) + o(\Delta t),
$$

where the expectation is taken with respect to $\mathbf{x} \sim p_{data}$, $n \sim \mathcal{U}[\![1, N-1]\!]$, and $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t_{n+1}^2 \boldsymbol{I})$. The consistency training objective, denoted by $\mathcal{L}_{CT}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$, is defined as

$$
\mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n))],
$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Moreover, $\mathcal{L}_{CT}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-) \geqslant O(\Delta t)$ if $\inf_N \mathcal{L}_{CD}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) > 0$.

Proof. With Taylor expansion, we have

$$
\begin{aligned}
&\mathcal{L}_{\text{CD}}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = \mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n)] \\
=&\mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}} + (t_{n+1} - t_n) t_{n+1} \nabla \log p_{t_{n+1}}(\mathbf{x}_{t_{n+1}}), t_n))] \\
=&\mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1}) + \partial_1 \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})(t_{n+1} - t_n) t_{n+1} \nabla \log p_{t_{n+1}}(\mathbf{x}_{t_{n+1}}) \\
&\qquad + \partial_2 \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})(t_n - t_{n+1}) + o(|t_{n+1} - t_n|))] \\
=&\mathbb{E}\{\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})) + \lambda(t_n) \partial_2 d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1}))[\\
&\quad \partial_1 \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})(t_{n+1} - t_n) t_{n+1} \nabla \log p_{t_{n+1}}(\mathbf{x}_{t_{n+1}}) + \partial_2 \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})(t_n - t_{n+1}) + o(|t_{n+1} - t_n|)]\} \\
=&\mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1}))] \\
&\quad + \mathbb{E}\{\lambda(t_n) \partial_2 d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1}))[\partial_1 \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})(t_{n+1} - t_n) t_{n+1} \nabla \log p_{t_{n+1}}(\mathbf{x}_{t_{n+1}})]\} \\
&\quad + \mathbb{E}\{\lambda(t_n) \partial_2 d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1}))[\partial_2 \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})(t_n - t_{n+1})]\} + \mathbb{E}[o(|t_{n+1} - t_n|)].
\end{aligned} \tag{15}
$$

Then, we apply Lemma 1 to Eq. (15) and use Taylor expansion in the reverse direction to obtain

$$
\begin{aligned}
&\mathcal{L}_{\text{CD}}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) \\
=&\mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1}))] \\
&\quad + \mathbb{E}\left\{ \lambda(t_n) \partial_2 d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})) \left[\partial_1 \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})(t_n - t_{n+1}) t_{n+1} \mathbb{E}\left[\frac{\mathbf{x}_{t_{n+1}} - \mathbf{x}}{t_{n+1}^2} \middle| \mathbf{x}_{t_{n+1}} \right] \right] \right\} \\
&\quad + \mathbb{E}\{\lambda(t_n) \partial_2 d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1}))[\partial_2 \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})(t_n - t_{n+1})]\} + \mathbb{E}[o(|t_{n+1} - t_n|)] \\
\overset{(i)}{=}&\mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1}))] \\
&\quad + \mathbb{E}\left\{ \lambda(t_n) \partial_2 d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})) \left[\partial_1 \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})(t_n - t_{n+1}) t_{n+1} \left(\frac{\mathbf{x}_{t_{n+1}} - \mathbf{x}}{t_{n+1}^2} \right) \right] \right\}
\end{aligned}
$$

$$
\begin{aligned}
&+\mathbb{E}\{\lambda(t_n)\partial_2 d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}},t_{n+1}))[\partial_2\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}},t_{n+1})(t_n-t_{n+1})]\}+\mathbb{E}[o(|t_{n+1}-t_n|)]\\
=&\mathbb{E}\Bigg[\lambda(t_n)d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}},t_{n+1}))\\
&+\lambda(t_n)\partial_2 d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}},t_{n+1}))\left[\partial_1\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}},t_{n+1})(t_n-t_{n+1})t_{n+1}\left(\frac{\mathbf{x}_{t_{n+1}}-\mathbf{x}}{t_{n+1}^2}\right)\right]\\
&+\lambda(t_n)\partial_2 d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}},t_{n+1}))[\partial_2\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}},t_{n+1})(t_n-t_{n+1})]+o(|t_{n+1}-t_n|)\Bigg]\\
&+\mathbb{E}[o(|t_{n+1}-t_n|)]\\
=&\mathbb{E}\left[\lambda(t_n)d\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}\left(\mathbf{x}_{t_{n+1}}+(t_n-t_{n+1})t_{n+1}\frac{\mathbf{x}_{t_{n+1}}-\mathbf{x}}{t_{n+1}^2},t_n\right)\right)\right]+\mathbb{E}[o(|t_{n+1}-t_n|)]\\
=&\mathbb{E}\left[\lambda(t_n)d\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}\left(\mathbf{x}_{t_{n+1}}+(t_n-t_{n+1})\frac{\mathbf{x}_{t_{n+1}}-\mathbf{x}}{t_{n+1}},t_n\right)\right)\right]+\mathbb{E}[o(|t_{n+1}-t_n|)]\\
=&\mathbb{E}\left[\lambda(t_n)d\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}\left(\mathbf{x}+t_{n+1}\mathbf{z}+(t_n-t_{n+1})\mathbf{z},t_n\right)\right)\right]+\mathbb{E}[o(|t_{n+1}-t_n|)]\\
=&\mathbb{E}\left[\lambda(t_n)d\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}\left(\mathbf{x}+t_n\mathbf{z},t_n\right)\right)\right]+\mathbb{E}[o(|t_{n+1}-t_n|)]\\
=&\mathbb{E}\left[\lambda(t_n)d\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}\left(\mathbf{x}+t_n\mathbf{z},t_n\right)\right)\right]+\mathbb{E}[o(\Delta t)]\\
=&\mathbb{E}\left[\lambda(t_n)d\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}\left(\mathbf{x}+t_n\mathbf{z},t_n\right)\right)\right]+o(\Delta t)\\
=&\mathcal{L}_{\mathrm{CT}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-)+o(\Delta t),
\end{aligned}
\tag{16}
$$

where (i) is due to the law of total expectation, and $\mathbf{z}\coloneqq\frac{\mathbf{x}_{t_{n+1}}-\mathbf{x}}{t_{n+1}}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})$. This implies $\mathcal{L}_{\mathrm{CD}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-;\boldsymbol{\phi})=\mathcal{L}_{\mathrm{CT}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-)+o(\Delta t)$ and thus completes the proof for Eq. (9). Moreover, we have $\mathcal{L}_{\mathrm{CT}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-)\geqslant O(\Delta t)$ whenever $\inf_N\mathcal{L}_{\mathrm{CD}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-;\boldsymbol{\phi})>0$. Otherwise, $\mathcal{L}_{\mathrm{CT}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-)<O(\Delta t)$ and thus $\lim_{\Delta t\to 0}\mathcal{L}_{\mathrm{CD}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-;\boldsymbol{\phi})=0$, which is a clear contradiction to $\inf_N\mathcal{L}_{\mathrm{CD}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-;\boldsymbol{\phi})>0$. □

Remark 1. *When the condition $\mathcal{L}_{CT}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-)\geqslant O(\Delta t)$ is not satisfied, such as in the case where $\boldsymbol{\theta}^-=\mathrm{stopgrad}(\boldsymbol{\theta})$, the validity of $\mathcal{L}_{CT}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-)$ as a training objective for consistency models can still be justified by referencing the result provided in Theorem 6.*

B. Continuous-Time Extensions

The consistency distillation and consistency training objectives can be generalized to hold for infinite time steps ($N\to\infty$) under suitable conditions.

B.1. Consistency Distillation in Continuous Time

Depending on whether $\boldsymbol{\theta}^-=\boldsymbol{\theta}$ or $\boldsymbol{\theta}^-=\mathrm{stopgrad}(\boldsymbol{\theta})$ (same as setting $\mu=0$), there are two possible continuous-time extensions for the consistency distillation objective $\mathcal{L}_{\mathrm{CD}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-;\boldsymbol{\phi})$. Given a twice continuously differentiable metric function $d(\mathbf{x},\mathbf{y})$, we define $\boldsymbol{G}(\mathbf{x})$ as a matrix, whose (i,j)-th entry is given by

$$
[\boldsymbol{G}(\mathbf{x})]_{ij}\coloneqq\left.\frac{\partial^2 d(\mathbf{x},\mathbf{y})}{\partial y_i\partial y_j}\right|_{\mathbf{y}=\mathbf{x}}.
$$

Similarly, we define $\boldsymbol{H}(\mathbf{x})$ as

$$
[\boldsymbol{H}(\mathbf{x})]_{ij}\coloneqq\left.\frac{\partial^2 d(\mathbf{y},\mathbf{x})}{\partial y_i\partial y_j}\right|_{\mathbf{y}=\mathbf{x}}.
$$

The matrices $\boldsymbol{G}$ and $\boldsymbol{H}$ play a crucial role in forming continuous-time objectives for consistency distillation. Additionally, we denote the Jacobian of $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x},t)$ with respect to $\mathbf{x}$ as $\frac{\partial\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x},t)}{\partial\mathbf{x}}$.

When $\boldsymbol{\theta}^-=\boldsymbol{\theta}$ (with no stopgrad operator), we have the following theoretical result.

Theorem 3. *Let $t_n=\tau(\frac{n-1}{N-1})$, where $n\in[\![1,N]\!]$, and $\tau(\cdot)$ is a strictly monotonic function with $\tau(0)=\epsilon$ and $\tau(1)=T$. Assume τ is continuously differentiable in $[0,1]$, d is three times continuously differentiable with bounded third derivatives,*

and $\boldsymbol{f}_{\boldsymbol{\theta}}$ is twice continuously differentiable with bounded first and second derivatives. Assume further that the weighting function $\lambda(\cdot)$ is bounded, and $\sup_{\mathbf{x},t\in[\epsilon,T]} \|\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x},t)\|_2 < \infty$. Then with the Euler solver in consistency distillation, we have

$$\lim_{N\to\infty}(N-1)^2\mathcal{L}_{CD}^N(\boldsymbol{\theta},\boldsymbol{\theta};\boldsymbol{\phi}) = \mathcal{L}_{CD}^\infty(\boldsymbol{\theta},\boldsymbol{\theta};\boldsymbol{\phi}), \tag{17}$$

where $\mathcal{L}_{CD}^\infty(\boldsymbol{\theta},\boldsymbol{\theta};\boldsymbol{\phi})$ is defined as

$$\frac{1}{2}\mathbb{E}\left[\frac{\lambda(t)}{[(\tau^{-1})'(t)]^2}\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t} - t\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial \mathbf{x}_t}\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t,t)\right)^{\mathsf{T}}\boldsymbol{G}(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t))\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t} - t\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial \mathbf{x}_t}\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t,t)\right)\right]. \tag{18}$$

Here the expectation above is taken over $\mathbf{x}\sim p_{data}$, $u\sim\mathcal{U}[0,1]$, $t=\tau(u)$, and $\mathbf{x}_t\sim\mathcal{N}(\mathbf{x},t^2\boldsymbol{I})$.

Proof. Let $\Delta u = \frac{1}{N-1}$ and $u_n = \frac{n-1}{N-1}$. First, we can derive the following equation with Taylor expansion:

$$\begin{aligned}
&\boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}) = \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}} + t_{n+1}\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}},t_{n+1})\tau'(u_n)\Delta u, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})\\
=&t_{n+1}\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}}\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}},t_{n+1})\tau'(u_n)\Delta u - \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})}{\partial t_{n+1}}\tau'(u_n)\Delta u + O((\Delta u)^2),
\end{aligned} \tag{19}$$

Note that $\tau'(u_n) = \frac{1}{\tau^{-1}(t_{n+1})}$. Then, we apply Taylor expansion to the consistency distillation loss, which gives

$$\begin{aligned}
&(N-1)^2\mathcal{L}_{\text{CD}}^N(\boldsymbol{\theta},\boldsymbol{\theta};\boldsymbol{\phi}) = \frac{1}{(\Delta u)^2}\mathcal{L}_{\text{CD}}^N(\boldsymbol{\theta},\boldsymbol{\theta};\boldsymbol{\phi}) = \frac{1}{(\Delta u)^2}\mathbb{E}[\lambda(t_n)d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n)]\\
\overset{(i)}{=}&\frac{1}{2(\Delta u)^2}\Bigg(\mathbb{E}\{\lambda(t_n)\tau'(u_n)^2[\boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})]^{\mathsf{T}}\boldsymbol{G}(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}))\\
&\qquad\cdot[\boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})]\} + \mathbb{E}[O(|\Delta u|^3)]\Bigg)\\
\overset{(ii)}{=}&\frac{1}{2}\mathbb{E}\Bigg[\lambda(t_n)\tau'(u_n)^2\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})}{\partial t_{n+1}} - t_{n+1}\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}}\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}},t_{n+1})\right)^{\mathsf{T}}\boldsymbol{G}(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}))\\
&\qquad\cdot\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})}{\partial t_{n+1}} - t_{n+1}\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}}\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}},t_{n+1})\right)\Bigg] + \mathbb{E}[O(|\Delta u|)]\\
=&\frac{1}{2}\mathbb{E}\Bigg[\frac{\lambda(t_n)}{[(\tau^{-1})'(t_n)]^2}\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})}{\partial t_{n+1}} - t_{n+1}\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}}\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}},t_{n+1})\right)^{\mathsf{T}}\boldsymbol{G}(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}))\\
&\qquad\cdot\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})}{\partial t_{n+1}} - t_{n+1}\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}}\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}},t_{n+1})\right)\Bigg] + \mathbb{E}[O(|\Delta u|)]
\end{aligned} \tag{20}$$

where we obtain (i) by expanding $d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}),\cdot)$ to second order and observing $d(\mathbf{x},\mathbf{x})\equiv 0$ and $\nabla_{\mathbf{y}}d(\mathbf{x},\mathbf{y})|_{\mathbf{y}=\mathbf{x}}\equiv\mathbf{0}$. We obtain (ii) using Eq. (19). By taking the limit for both sides of Eq. (20) as $\Delta u\to 0$ or equivalently $N\to\infty$, we arrive at Eq. (17), which completes the proof. □

Remark 2. *Although Theorem 3 assumes the Euler ODE solver for technical simplicity, we believe an analogous result can be derived for more general solvers, since all ODE solvers should perform similarly as $N\to\infty$. We leave a more general version of Theorem 3 as future work.*

Remark 3. *Theorem 3 implies that consistency models can be trained by minimizing $\mathcal{L}_{CD}^\infty(\boldsymbol{\theta},\boldsymbol{\theta};\boldsymbol{\phi})$. In particular, when $d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2^2$, we have*

$$\mathcal{L}_{CD}^\infty(\boldsymbol{\theta},\boldsymbol{\theta};\boldsymbol{\phi}) = \mathbb{E}\left[\frac{\lambda(t)}{[(\tau^{-1})'(t)]^2}\left\|\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial t} - t\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\partial \mathbf{x}_t}\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t,t)\right\|_2^2\right]. \tag{21}$$

However, this continuous-time objective requires computing Jacobian-vector products as a subroutine to evaluate the loss function, which can be slow and laborious to implement in deep learning frameworks that do not support forward-mode automatic differentiation.

Remark 4. *If $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ matches the ground truth consistency function for the empirical PF ODE of $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)$, then*

$$\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t)}{\partial t} - t \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t)}{\partial \mathbf{x}} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t) \equiv 0$$

and therefore $\mathcal{L}_{CD}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$. This can be proved by noting that $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \equiv \mathbf{x}_\epsilon$ for all $t \in [\epsilon, T]$, and then taking the time-derivative of this identity:

$$\begin{aligned}
&\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \equiv \mathbf{x}_\epsilon \\
\iff &\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} + \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t} \equiv 0 \\
\iff &\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} [-t \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t)] + \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t} \equiv 0 \\
\iff &\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t} - t \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) \equiv 0.
\end{aligned}$$

The above observation provides another motivation for $\mathcal{L}_{CD}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi})$, as it is minimized if and only if the consistency model matches the ground truth consistency function.

For some metric functions, such as the ℓ_1 norm, the Hessian $\boldsymbol{G}(\mathbf{x})$ is zero so Theorem 3 is vacuous. Below we show that a non-vacuous statement holds for the ℓ_1 norm with just a small modification of the proof for Theorem 3.

Theorem 4. *Let $t_n = \tau(\frac{n-1}{N-1})$, where $n \in [\![1, N]\!]$, and $\tau(\cdot)$ is a strictly monotonic function with $\tau(0) = \epsilon$ and $\tau(1) = T$. Assume τ is continuously differentiable in $[0, 1]$, and $\boldsymbol{f}_{\boldsymbol{\theta}}$ is twice continuously differentiable with bounded first and second derivatives. Assume further that the weighting function $\lambda(\cdot)$ is bounded, and $\sup_{\mathbf{x}, t \in [\epsilon, T]} \left\| \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t) \right\|_2 < \infty$. Suppose we use the Euler ODE solver, and set $d(\mathbf{x}, \mathbf{y}) = \left\| \mathbf{x} - \mathbf{y} \right\|_1$ in consistency distillation. Then we have*

$$\lim_{N \to \infty} (N-1) \mathcal{L}_{CD}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = \mathcal{L}_{CD,\, \ell_1}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}), \tag{22}$$

where

$$\mathcal{L}_{CD,\, \ell_1}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) := \mathbb{E}\left[\frac{\lambda(t)}{(\tau^{-1})'(t)} \left\| t \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\partial t} \right\|_1 \right]$$

where the expectation above is taken over $\mathbf{x} \sim p_{data}$, $u \sim \mathcal{U}[0, 1]$, $t = \tau(u)$, and $\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}, t^2 \boldsymbol{I})$.

Proof. Let $\Delta u = \frac{1}{N-1}$ and $u_n = \frac{n-1}{N-1}$. We have

$$\begin{aligned}
&(N-1)\mathcal{L}_{\mathrm{CD}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = \frac{1}{\Delta u} \mathcal{L}_{\mathrm{CD}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = \frac{1}{\Delta u} \mathbb{E}[\lambda(t_n) \| \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) \|_1] \\
\overset{(i)}{=} &\frac{1}{\Delta u} \mathbb{E}\left[\lambda(t_n) \left\| t_{n+1} \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) \tau'(u_n) - \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial t_{n+1}} \tau'(u_n) + O((\Delta u)^2) \right\|_1 \right] \\
= &\mathbb{E}\left[\lambda(t_n) \tau'(u_n) \left\| t_{n+1} \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial t_{n+1}} + O(\Delta u) \right\|_1 \right] \\
= &\mathbb{E}\left[\frac{\lambda(t_n)}{(\tau^{-1})'(t_n)} \left\| t_{n+1} \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial t_{n+1}} + O(\Delta u) \right\|_1 \right]
\end{aligned} \tag{23}$$

where (i) is obtained by plugging Eq. (19) into the previous equation. Taking the limit for both sides of Eq. (23) as $\Delta u \to 0$ or equivalently $N \to \infty$ leads to Eq. (22), which completes the proof. □

Remark 5. *According to Theorem 4, consistency models can be trained by minimizing $\mathcal{L}_{CD,\, \ell_1}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi})$. Moreover, the same reasoning in Remark 4 can be applied to show that $\mathcal{L}_{CD,\, \ell_1}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$ if and only if $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \mathbf{x}_\epsilon$ for all $\mathbf{x}_t \in \mathbb{R}^d$ and $t \in [\epsilon, T]$.*

In the second case where $\boldsymbol{\theta}^- = \mathrm{stopgrad}(\boldsymbol{\theta})$, we can derive a so-called "pseudo-objective" whose gradient matches the gradient of $\mathcal{L}_{\mathrm{CD}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$ in the limit of $N \to \infty$. Minimizing this pseudo-objective with gradient descent gives another way to train consistency models via distillation. This pseudo-objective is provided by the theorem below.

Theorem 5. *Let $t_n = \tau(\frac{n-1}{N-1})$, where $n \in [\![1, N]\!]$, and $\tau(\cdot)$ is a strictly monotonic function with $\tau(0) = \epsilon$ and $\tau(1) = T$. Assume τ is continuously differentiable in $[0, 1]$, d is three times continuously differentiable with bounded third derivatives, and $\boldsymbol{f}_{\boldsymbol{\theta}}$ is twice continuously differentiable with bounded first and second derivatives. Assume further that the weighting function $\lambda(\cdot)$ is bounded, $\sup_{\mathbf{x}, t\in[\epsilon,T]} \|\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)\|_2 < \infty$, and $\sup_{\mathbf{x}, t\in[\epsilon,T]} \|\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2 < \infty$. Suppose we use the Euler ODE solver, and $\boldsymbol{\theta}^- = \operatorname{stopgrad}(\boldsymbol{\theta})$ in consistency distillation. Then,*

$$\lim_{N\to\infty} (N-1)\nabla_{\boldsymbol{\theta}} \mathcal{L}^N_{CD}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = \nabla_{\boldsymbol{\theta}} \mathcal{L}^\infty_{CD}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}), \tag{24}$$

where

$$\mathcal{L}^\infty_{CD}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) \coloneqq \mathbb{E}\left[\frac{\lambda(t)}{(\tau^{-1})'(t)} \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)^\mathsf{T} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t)) \left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t)}{\partial t} - t \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) \right) \right]. \tag{25}$$

Here the expectation above is taken over $\mathbf{x} \sim p_{data}$, $u \sim \mathcal{U}[0, 1]$, $t = \tau(u)$, and $\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}, t^2\boldsymbol{I})$.

Proof. We denote $\Delta u = \frac{1}{N-1}$ and $u_n = \frac{n-1}{N-1}$. First, we leverage Taylor series expansion to obtain

$$\begin{aligned}
&(N-1)\mathcal{L}^N_{\text{CD}}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = \frac{1}{\Delta u}\mathcal{L}^N_{\text{CD}}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = \frac{1}{\Delta u}\mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)] \\
\overset{(i)}{=}& \frac{1}{2\Delta u}\Bigg(\mathbb{E}\{\lambda(t_n)[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)]^\mathsf{T} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)) \\
&\qquad \cdot [\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)]\} + \mathbb{E}[O(|\Delta u|^3)] \Bigg) \\
=& \frac{1}{2\Delta u}\mathbb{E}\{\lambda(t_n)[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)]^\mathsf{T} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n))[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)]\} + \mathbb{E}[O(|\Delta u|^2)]
\end{aligned} \tag{26}$$

where (i) is derived by expanding $d(\cdot, \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n))$ to second order and leveraging $d(\mathbf{x}, \mathbf{x}) \equiv 0$ and $\nabla_{\mathbf{y}} d(\mathbf{y}, \mathbf{x})|_{\mathbf{y}=\mathbf{x}} \equiv \mathbf{0}$. Next, we compute the gradient of Eq. (26) with respect to $\boldsymbol{\theta}$ and simplify the result to obtain

$$\begin{aligned}
&(N-1)\nabla_{\boldsymbol{\theta}}\mathcal{L}^N_{\text{CD}}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = \frac{1}{\Delta u}\nabla_{\boldsymbol{\theta}}\mathcal{L}^N_{\text{CD}}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) \\
=& \frac{1}{2\Delta u}\nabla_{\boldsymbol{\theta}}\mathbb{E}\{\lambda(t_n)[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)]^\mathsf{T} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n))[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)]\} + \mathbb{E}[O(|\Delta u|^2)] \\
\overset{(i)}{=}& \frac{1}{\Delta u}\mathbb{E}\{\lambda(t_n)[\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1})]^\mathsf{T} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n))[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)]\} + \mathbb{E}[O(|\Delta u|^2)] \\
\overset{(ii)}{=}& \frac{1}{\Delta u}\mathbb{E}\Bigg\{ \lambda(t_n)[\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1})]^\mathsf{T} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)) \Bigg[t_{n+1} \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) \tau'(u_n) \Delta u \\
&\qquad - \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial t_{n+1}} \tau'(u_n) \Delta u \Bigg] \Bigg\} + \mathbb{E}[O(|\Delta u|)] \\
=& \mathbb{E}\Bigg\{ \lambda(t_n)[\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1})]^\mathsf{T} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)) \Bigg[t_{n+1} \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) \tau'(u_n) \\
&\qquad - \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial t_{n+1}} \tau'(u_n) \Bigg] \Bigg\} + \mathbb{E}[O(|\Delta u|)] \\
=& \nabla_{\boldsymbol{\theta}}\mathbb{E}\Bigg\{ \lambda(t_n)[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1})]^\mathsf{T} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)) \Bigg[t_{n+1} \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) \tau'(u_n) \\
&\qquad - \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial t_{n+1}} \tau'(u_n) \Bigg] \Bigg\} + \mathbb{E}[O(|\Delta u|)] \\
=& \nabla_{\boldsymbol{\theta}}\mathbb{E}\Bigg\{ \frac{\lambda(t_n)}{(\tau^{-1})'(t_n)} [\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1})]^\mathsf{T} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)) \Bigg[t_{n+1} \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial \mathbf{x}_{t_{n+1}}} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) \\
&\qquad - \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}}, t_{n+1})}{\partial t_{n+1}} \Bigg] \Bigg\} + \mathbb{E}[O(|\Delta u|)]
\end{aligned} \tag{27}$$

Here (i) results from the chain rule, and (ii) follows from Eq. (19) and $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) \equiv \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}, t)$, since $\boldsymbol{\theta}^- = \operatorname{stopgrad}(\boldsymbol{\theta})$. Taking the limit for both sides of Eq. (28) as $\Delta u \to 0$ (or $N \to \infty$) yields Eq. (24), which completes the proof. □

Remark 6. *When $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2$, the pseudo-objective $\mathcal{L}_{CD}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$ can be simplified to*

$$\mathcal{L}_{CD}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = 2\mathbb{E}\left[\frac{\lambda(t)}{(\tau^{-1})'(t)} \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)^{\mathsf{T}} \left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t)}{\partial t} - t \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t)\right)\right]. \tag{28}$$

Remark 7. *The objective $\mathcal{L}_{CD}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$ defined in Theorem 5 is only meaningful in terms of its gradient—one cannot measure the progress of training by tracking the value of $\mathcal{L}_{CD}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$, but can still apply gradient descent to this objective to distill consistency models from pre-trained diffusion models. Because this objective is not a typical loss function, we refer to it as the "pseudo-objective" for consistency distillation.*

Remark 8. *Following the same reasoning in Remark 4, we can easily derive that $\mathcal{L}_{CD}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = 0$ and $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{CD}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = \mathbf{0}$ if $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ matches the ground truth consistency function for the empirical PF ODE that involves $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)$. However, the converse does not hold true in general. This distinguishes $\mathcal{L}_{CD}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$ from $\mathcal{L}_{CD}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi})$, the latter of which is a true loss function.*

B.2. Consistency Training in Continuous Time

A remarkable observation is that the pseudo-objective in Theorem 5 can be estimated without any pre-trained diffusion models, which enables direct consistency training of consistency models. More precisely, we have the following result.

Theorem 6. *Let $t_n = \tau(\frac{n-1}{N-1})$, where $n \in [\![1, N]\!]$, and $\tau(\cdot)$ is a strictly monotonic function with $\tau(0) = \epsilon$ and $\tau(1) = T$. Assume τ is continuously differentiable in $[0, 1]$, d is three times continuously differentiable with bounded third derivatives, and $\boldsymbol{f}_{\boldsymbol{\theta}}$ is twice continuously differentiable with bounded first and second derivatives. Assume further that the weighting function $\lambda(\cdot)$ is bounded, $\mathbb{E}[\|\nabla \log p_{t_n}(\mathbf{x}_{t_n})\|_2^2] < \infty$, $\sup_{\mathbf{x}, t \in [\epsilon, T]} \|\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2 < \infty$, and $\boldsymbol{\phi}$ represents diffusion model parameters that satisfy $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t) \equiv \nabla \log p_t(\mathbf{x})$. Then if $\boldsymbol{\theta}^- = \operatorname{stopgrad}(\boldsymbol{\theta})$, we have*

$$\lim_{N \to \infty} (N-1) \nabla_{\boldsymbol{\theta}} \mathcal{L}_{CD}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = \lim_{N \to \infty} (N-1) \nabla_{\boldsymbol{\theta}} \mathcal{L}_{CT}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{CT}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-), \tag{29}$$

where $\mathcal{L}_{CD}^{N}$ uses the Euler ODE solver, and

$$\mathcal{L}_{CT}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) := \mathbb{E}\left[\frac{\lambda(t)}{(\tau^{-1})'(t)} \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)^{\mathsf{T}} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t)) \left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t)}{\partial t} + \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \cdot \frac{\mathbf{x}_t - \mathbf{x}}{t}\right)\right]. \tag{30}$$

Here the expectation above is taken over $\mathbf{x} \sim p_{data}$, $u \sim \mathcal{U}[0, 1]$, $t = \tau(u)$, and $\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}, t^2 \boldsymbol{I})$.

Proof. The proof mostly follows that of Theorem 5. First, we leverage Taylor series expansion to obtain

$$\begin{aligned}
&(N-1)\mathcal{L}_{\mathrm{CT}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = \frac{1}{\Delta u} \mathcal{L}_{\mathrm{CT}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = \frac{1}{\Delta u} \mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n))] \\
&\overset{(i)}{=} \frac{1}{2\Delta u} \Bigg(\mathbb{E}\{\lambda(t_n) [\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n)]^{\mathsf{T}} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n)) \\
&\qquad \cdot [\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n)]\} + \mathbb{E}[O(|\Delta u|^3)] \Bigg) \\
&= \frac{1}{2\Delta u} \mathbb{E}\{\lambda(t_n) [\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n)]^{\mathsf{T}} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n)) \\
&\qquad \cdot [\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n)]\} + \mathbb{E}[O(|\Delta u|^2)]
\end{aligned} \tag{31}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, (i) is derived by first expanding $d(\cdot, \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n))$ to second order, and then noting that $d(\mathbf{x}, \mathbf{x}) \equiv 0$ and $\nabla_{\mathbf{y}} d(\mathbf{y}, \mathbf{x})|_{\mathbf{y}=\mathbf{x}} \equiv \mathbf{0}$. Next, we compute the gradient of Eq. (31) with respect to $\boldsymbol{\theta}$ and simplify the result to obtain

$$\begin{aligned}
&(N-1)\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{CT}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = \frac{1}{\Delta u} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CT}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) \\
&= \frac{1}{2\Delta u} \nabla_{\boldsymbol{\theta}} \mathbb{E}\{\lambda(t_n) [\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n)]^{\mathsf{T}} \boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n)) \\
&\qquad \cdot [\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n)]\} + \mathbb{E}[O(|\Delta u|^2)]
\end{aligned}$$

$$\overset{(i)}{=}\frac{1}{\Delta u}\mathbb{E}\{\lambda(t_n)[\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1})]^{\mathsf{T}}\boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n)) \cdot[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1})-\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n)]\}+\mathbb{E}[O(|\Delta u|^2)] \tag{32}$$

$$\begin{aligned}
&\overset{(ii)}{=}\frac{1}{\Delta u}\mathbb{E}\Big\{\lambda(t_n)[\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1})]^{\mathsf{T}}\boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n))\Big[\tau'(u_n)\Delta u\partial_1\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n)\mathbf{z}\\
&\qquad+\partial_2\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n)\tau'(u_n)\Delta u\Big]\Big\}+\mathbb{E}[O(|\Delta u|)]\\
=&\mathbb{E}\Big\{\lambda(t_n)\tau'(u_n)[\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1})]^{\mathsf{T}}\boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n))\Big[\partial_1\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n)\mathbf{z}\\
&\qquad+\partial_2\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n)\Big]\Big\}+\mathbb{E}[O(|\Delta u|)]\\
=&\nabla_{\boldsymbol{\theta}}\mathbb{E}\Big\{\lambda(t_n)\tau'(u_n)[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1})]^{\mathsf{T}}\boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n))\Big[\partial_1\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n)\mathbf{z}\\
&\qquad+\partial_2\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n)\Big]\Big\}+\mathbb{E}[O(|\Delta u|)]\\
=&\nabla_{\boldsymbol{\theta}}\mathbb{E}\Big\{\lambda(t_n)\tau'(u_n)[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})]^{\mathsf{T}}\boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_n},t_n))\Big[\partial_1\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_n},t_n)\frac{\mathbf{x}_{t_n}-\mathbf{x}}{t_n}+\partial_2\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_n},t_n)\Big]\Big\}+\mathbb{E}[O(|\Delta u|)]\\
=&\nabla_{\boldsymbol{\theta}}\mathbb{E}\Big\{\frac{\lambda(t_n)}{(\tau^{-1})'(t_n)}[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})]^{\mathsf{T}}\boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_n},t_n))\Big[\partial_1\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_n},t_n)\frac{\mathbf{x}_{t_n}-\mathbf{x}}{t_n}+\partial_2\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_n},t_n)\Big]\Big\}+\mathbb{E}[O(|\Delta u|)]
\end{aligned} \tag{33}$$

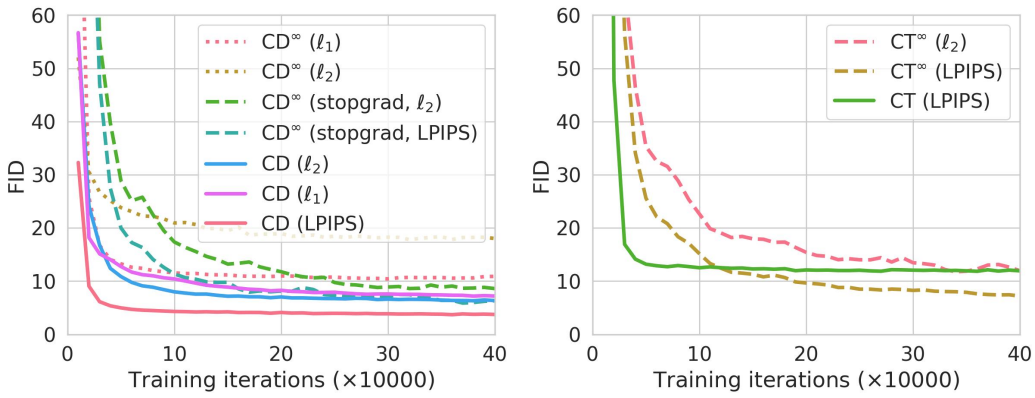
Here (i) results from the chain rule, and (ii) follows from Taylor expansion. Taking the limit for both sides of Eq. (33) as $\Delta u\to 0$ or $N\to\infty$ yields the second equality in Eq. (29).

Now we prove the first equality. Applying Taylor expansion again, we obtain

$$\begin{aligned}
&(N-1)\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{CD}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-;\boldsymbol{\phi})=\frac{1}{\Delta u}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{CD}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-;\boldsymbol{\phi})=\frac{1}{\Delta u}\nabla_{\boldsymbol{\theta}}\mathbb{E}[\lambda(t_n)d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n))]\\
=&\frac{1}{\Delta u}\mathbb{E}[\lambda(t_n)\nabla_{\boldsymbol{\theta}}d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n))]\\
=&\frac{1}{\Delta u}\mathbb{E}[\lambda(t_n)\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})^{\mathsf{T}}\partial_1 d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1}),\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n))]\\
=&\frac{1}{\Delta u}\mathbb{E}\Big\{\lambda(t_n)\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})^{\mathsf{T}}\Big[\partial_1 d(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n),\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n))\\
&\qquad+\boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n))(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})-\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n))+O(|\Delta u|^2)\Big]\Big\}\\
=&\frac{1}{\Delta u}\mathbb{E}\{\lambda(t_n)\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})^{\mathsf{T}}[\boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n))(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})-\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n))]+O(|\Delta u|^2)\}\\
=&\frac{1}{\Delta u}\mathbb{E}\{\lambda(t_n)\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}},t_{n+1})^{\mathsf{T}}[\boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n))(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_{t_{n+1}},t_{n+1})-\boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}},t_n))]+O(|\Delta u|^2)\}\\
\overset{(i)}{=}&\frac{1}{\Delta u}\mathbb{E}\{\lambda(t_n)[\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1})]^{\mathsf{T}}\boldsymbol{H}(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n))\\
&\qquad\cdot[\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1})-\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n)]\}+\mathbb{E}[O(|\Delta u|^2)]
\end{aligned}$$

where (i) holds because $\mathbf{x}_{t_{n+1}}=\mathbf{x}+t_{n+1}\mathbf{z}$ and $\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}=\mathbf{x}_{t_{n+1}}-(t_n-t_{n+1})t_{n+1}\frac{-(\mathbf{x}_{t_{n+1}}-\mathbf{x})}{t_{n+1}^2}=\mathbf{x}_{t_{n+1}}+(t_n-t_{n+1})\mathbf{z}=\mathbf{x}+t_n\mathbf{z}$. Because (i) matches Eq. (32), we can use the same reasoning procedure from Eq. (32) to Eq. (33) to conclude $\lim_{N\to\infty}(N-1)\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{CD}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-;\boldsymbol{\phi})=\lim_{N\to\infty}(N-1)\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{CT}}^N(\boldsymbol{\theta},\boldsymbol{\theta}^-)$, completing the proof. □

Remark 9. *Note that $\mathcal{L}_{CT}^{\infty}(\boldsymbol{\theta},\boldsymbol{\theta}^-)$ does not depend on the diffusion model parameter $\boldsymbol{\phi}$ and hence can be optimized without any pre-trained diffusion models.*



(a) Consistency Distillation (b) Consistency Training

Figure 7: Comparing discrete consistency distillation/training algorithms with continuous counterparts.

Remark 10. *When* $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2$*, the continuous-time consistency training objective becomes*

$$\mathcal{L}_{CT}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = 2\mathbb{E}\left[\frac{\lambda(t)}{(\tau^{-1})'(t)}\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)^{\mathsf{T}}\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t)}{\partial t} + \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t, t)}{\partial \mathbf{x}_t} \cdot \frac{\mathbf{x}_t - \mathbf{x}}{t}\right)\right]. \tag{34}$$

Remark 11. *Similar to* $\mathcal{L}_{CD}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$ *in Theorem 5,* $\mathcal{L}_{CT}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$ *is a pseudo-objective; one cannot track training by monitoring the value of* $\mathcal{L}_{CT}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$*, but can still apply gradient descent on this loss function to train a consistency model* $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ *directly from data. Moreover, the same observation in Remark 8 holds true:* $\mathcal{L}_{CT}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = 0$ *and* $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{CT}^{\infty}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = \mathbf{0}$ *if* $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ *matches the ground truth consistency function for the PF ODE.*

B.3. Experimental Verifications

To experimentally verify the efficacy of our continuous-time CD and CT objectives, we train consistency models with a variety of loss functions on CIFAR-10. All results are provided in Fig. 7. We set $\lambda(t) = (\tau^{-1})'(t)$ for all continuous-time experiments. Other hyperparameters are the same as in Table 3. We occasionally modify some hyperparameters for improved performance. For distillation, we compare the following objectives:

- CD (ℓ_2): Consistency distillation $\mathcal{L}_{\text{CD}}^{N}$ with $N = 18$ and the ℓ_2 metric.
- CD (ℓ_1): Consistency distillation $\mathcal{L}_{\text{CD}}^{N}$ with $N = 18$ and the ℓ_1 metric. We set the learning rate to 2e-4.
- CD (LPIPS): Consistency distillation $\mathcal{L}_{\text{CD}}^{N}$ with $N = 18$ and the LPIPS metric.
- CD$^{\infty}$ (ℓ_2): Consistency distillation $\mathcal{L}_{\text{CD}}^{\infty}$ in Theorem 3 with the ℓ_2 metric. We set the learning rate to 1e-3 and dropout to 0.13.
- CD$^{\infty}$ (ℓ_1): Consistency distillation $\mathcal{L}_{\text{CD}}^{\infty}$ in Theorem 4 with the ℓ_1 metric. We set the learning rate to 1e-3 and dropout to 0.3.
- CD$^{\infty}$ (stopgrad, ℓ_2): Consistency distillation $\mathcal{L}_{\text{CD}}^{\infty}$ in Theorem 5 with the ℓ_2 metric. We set the learning rate to 5e-6.
- CD$^{\infty}$ (stopgrad, LPIPS): Consistency distillation $\mathcal{L}_{\text{CD}}^{\infty}$ in Theorem 5 with the LPIPS metric. We set the learning rate to 5e-6.

We did not investigate using the LPIPS metric in Theorem 3 because minimizing the resulting objective would require back-propagating through second order derivatives of the VGG network used in LPIPS, which is computationally expensive and prone to numerical instability. As revealed by Fig. 7a, the stopgrad version of continuous-time distillation (Theorem 5) works better than the non-stopgrad version (Theorem 3) for both the LPIPS and ℓ_2 metrics, and the LPIPS metric works the best for all distillation approaches. Additionally, discrete-time consistency distillation outperforms continuous-time

Table 3: Hyperparameters used for training CD and CT models

Hyperparameter	CIFAR-10		ImageNet 64×64		LSUN 256×256	
	CD	CT	CD	CT	CD	CT
Learning rate	4e-4	4e-4	8e-6	8e-6	1e-5	1e-5
Batch size	512	512	2048	2048	2048	2048
μ	0		0.95		0.95	
μ_0		0.9		0.95		0.95
s_0		2		2		2
s_1		150		200		150
N	18		40		40	
ODE solver	Heun		Heun		Heun	
EMA decay rate	0.9999	0.9999	0.999943	0.999943	0.999943	0.999943
Training iterations	800k	800k	600k	800k	600k	1000k
Mixed-Precision (FP16)	No	No	Yes	Yes	Yes	Yes
Dropout probability	0.0	0.0	0.0	0.0	0.0	0.0
Number of GPUs	8	8	64	64	64	64

consistency distillation, possibly due to the larger variance in continuous-time objectives, and the fact that one can use effective higher-order ODE solvers in discrete-time objectives.

For consistency training (CT), we find it important to initialize consistency models from a pre-trained EDM model in order to stabilize training when using continuous-time objectives. We hypothesize that this is caused by the large variance in our continuous-time loss functions. For fair comparison, we thus initialize all consistency models from the same pre-trained EDM model on CIFAR-10 for both discrete-time and continuous-time CT, even though the former works well with random initialization. We leave variance reduction techniques for continuous-time CT to future research.

We empirically compare the following objectives:

- CT (LPIPS): Consistency training $\mathcal{L}_{\text{CT}}^{N}$ with $N = 120$ and the LPIPS metric. We set the learning rate to 4e-4, and the EMA decay rate for the target network to 0.99. We do not use the schedule functions for N and μ here because they cause slower learning when the consistency model is initialized from a pre-trained EDM model.
- CT$^{\infty}$ (ℓ_2): Consistency training $\mathcal{L}_{\text{CT}}^{\infty}$ with the ℓ_2 metric. We set the learning rate to 5e-6.
- CT$^{\infty}$ (LPIPS): Consistency training $\mathcal{L}_{\text{CT}}^{\infty}$ with the LPIPS metric. We set the learning rate to 5e-6.

As shown in Fig. 7b, the LPIPS metric leads to improved performance for continuous-time CT. We also find that continuous-time CT outperforms discrete-time CT with the same LPIPS metric. This is likely due to the bias in discrete-time CT, as $\Delta t > 0$ in Theorem 2 for discrete-time objectives, whereas continuous-time CT has no bias since it implicitly drives Δt to 0.

C. Additional Experimental Details

Model Architectures We follow Song et al. (2021); Dhariwal & Nichol (2021) for model architectures. Specifically, we use the NCSN++ architecture in Song et al. (2021) for all CIFAR-10 experiments, and take the corresponding network architectures from Dhariwal & Nichol (2021) when performing experiments on ImageNet 64×64, LSUN Bedroom 256×256 and LSUN Cat 256×256.

Parameterization for Consistency Models We use the same architectures for consistency models as those used for EDMs. The only difference is we slightly modify the skip connections in EDM to ensure the boundary condition holds for consistency models. Recall that in Section 3 we propose to parameterize a consistency model in the following form:

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) = c_{\text{skip}}(t)\mathbf{x} + c_{\text{out}}(t)F_{\boldsymbol{\theta}}(\mathbf{x}, t).$$

In EDM (Karras et al., 2022), authors choose

$$c_{\text{skip}}(t) = \frac{\sigma_{\text{data}}^2}{t^2 + \sigma_{\text{data}}^2}, \quad c_{\text{out}}(t) = \frac{\sigma_{\text{data}} t}{\sqrt{\sigma_{\text{data}}^2 + t^2}},$$

where $\sigma_{\text{data}} = 0.5$. However, this choice of c_{skip} and c_{out} does not satisfy the boundary condition when the smallest time instant $\epsilon \neq 0$. To remedy this issue, we modify them to

$$c_{\text{skip}}(t) = \frac{\sigma_{\text{data}}^2}{(t-\epsilon)^2 + \sigma_{\text{data}}^2}, \quad c_{\text{out}}(t) = \frac{\sigma_{\text{data}}(t-\epsilon)}{\sqrt{\sigma_{\text{data}}^2 + t^2}},$$

which clearly satisfies $c_{\text{skip}}(\epsilon) = 1$ and $c_{\text{out}}(\epsilon) = 0$.

Schedule Functions for Consistency Training As discussed in Section 5, consistency generation requires specifying schedule functions $N(\cdot)$ and $\mu(\cdot)$ for best performance. Throughout our experiments, we use schedule functions that take the form below:

$$N(k) = \left\lceil \sqrt{\frac{k}{K}((s_1+1)^2 - s_0^2) + s_0^2} - 1 \right\rceil + 1$$
$$\mu(k) = \exp\left(\frac{s_0 \log \mu_0}{N(k)}\right),$$

where K denotes the total number of training iterations, s_0 denotes the initial discretization steps, $s_1 > s_0$ denotes the target discretization steps at the end of training, and $\mu_0 > 0$ denotes the EMA decay rate at the beginning of model training.

Training Details In both consistency distillation and progressive distillation, we distill EDMs (Karras et al., 2022). We trained these EDMs ourselves according to the specifications given in Karras et al. (2022). The original EDM paper did not provide hyperparameters for the LSUN Bedroom 256×256 and Cat 256×256 datasets, so we mostly used the same hyperparameters as those for the ImageNet 64×64 dataset. The difference is that we trained for 600k and 300k iterations for the LSUN Bedroom and Cat datasets respectively, and reduced the batch size from 4096 to 2048.

We used the same EMA decay rate for LSUN 256×256 datasets as for the ImageNet 64×64 dataset. For progressive distillation, we used the same training settings as those described in Salimans & Ho (2022) for CIFAR-10 and ImageNet 64×64. Although the original paper did not test on LSUN 256×256 datasets, we used the same settings for ImageNet 64×64 and found them to work well.

In all distillation experiments, we initialized the consistency model with pre-trained EDM weights. For consistency training, we initialized the model randomly, just as we did for training the EDMs. We trained all consistency models with the Rectified Adam optimizer (Liu et al., 2019), with no learning rate decay or warm-up, and no weight decay. We also applied EMA to the weights of the online consistency models in both consistency distillation and consistency training, as well as to the weights of the training online consistency models according to Karras et al. (2022). For LSUN 256×256 datasets, we chose the EMA decay rate to be the same as that for ImageNet 64×64, except for consistency distillation on LSUN Bedroom 256×256, where we found that using zero EMA worked better.

When using the LPIPS metric on CIFAR-10 and ImageNet 64×64, we rescale images to resolution 224×224 with bilinear upsampling before feeding them to the LPIPS network. For LSUN 256×256, we evaluated LPIPS without rescaling inputs. In addition, we performed horizontal flips for data augmentation for all models and on all datasets. We trained all models on a cluster of Nvidia A100 GPUs. Additional hyperparameters for consistency training and distillation are listed in Table 3.

D. Additional Results on Zero-Shot Image Editing

With consistency models, we can perform a variety of zero-shot image editing tasks. As an example, we present additional results on colorization (Fig. 8), super-resolution (Fig. 9), inpainting (Fig. 10), interpolation (Fig. 11), denoising (Fig. 12), and stroke-guided image generation (SDEdit, Meng et al. (2021), Fig. 13). The consistency model used here is trained via consistency distillation on the LSUN Bedroom 256×256.

All these image editing tasks, except for image interpolation and denoising, can be performed via a small modification to the multistep sampling algorithm in Algorithm 1. The resulting pseudocode is provided in Algorithm 4. Here $\mathbf{y}$ is a reference image that guides sample generation, $\mathbf{\Omega}$ is a binary mask, $\odot$ computes element-wise products, and $\boldsymbol{A}$ is an invertible linear transformation that maps images into a latent space where the conditional information in $\mathbf{y}$ is infused into the iterative

Algorithm 4 Zero-Shot Image Editing

Input: Consistency model $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot,\cdot)$, sequence of time points $t_1 > t_2 > \cdots > t_N$, reference image $\mathbf{y}$, invertible linear transformation $\boldsymbol{A}$, and binary image mask $\boldsymbol{\Omega}$
$\mathbf{y} \leftarrow \boldsymbol{A}^{-1}[(\boldsymbol{A}\mathbf{y}) \odot (1 - \boldsymbol{\Omega}) + \mathbf{0} \odot \boldsymbol{\Omega}]$
Sample $\mathbf{x} \sim \mathcal{N}(\mathbf{y}, t_1^2 \boldsymbol{I})$
$\mathbf{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t_1)$
$\mathbf{x} \leftarrow \boldsymbol{A}^{-1}[(\boldsymbol{A}\mathbf{y}) \odot (1 - \boldsymbol{\Omega}) + (\boldsymbol{A}\mathbf{x}) \odot \boldsymbol{\Omega}]$
for $n = 2$ **to** N **do**
  Sample $\mathbf{x} \sim \mathcal{N}(\mathbf{x}, (t_n^2 - \epsilon^2)\boldsymbol{I})$
  $\mathbf{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t_n)$
  $\mathbf{x} \leftarrow \boldsymbol{A}^{-1}[(\boldsymbol{A}\mathbf{y}) \odot (1 - \boldsymbol{\Omega}) + (\boldsymbol{A}\mathbf{x}) \odot \boldsymbol{\Omega}]$
end for
Output: $\mathbf{x}$

generation procedure by masking with $\boldsymbol{\Omega}$. Unless otherwise stated, we choose

$$t_i = \left(T^{1/\rho} + \frac{i-1}{N-1}(\epsilon^{1/\rho} - T^{1/\rho})\right)^{\rho}$$

in our experiments, where $N = 40$ for LSUN Bedroom 256×256.

Below we describe how to perform each task using Algorithm 4.

Inpainting When using Algorithm 4 for inpainting, we let $\mathbf{y}$ be an image where missing pixels are masked out, $\boldsymbol{\Omega}$ be a binary mask where 1 indicates the missing pixels, and $\boldsymbol{A}$ be the identity transformation.

Colorization The algorithm for image colorization is similar, as colorization becomes a special case of inpainting once we transform data into a decoupled space. Specifically, let $\mathbf{y} \in \mathbb{R}^{h \times w \times 3}$ be a gray-scale image that we aim to colorize, where all channels of $\mathbf{y}$ are assumed to be the same, *i.e.*, $\mathbf{y}[:,:,0] = \mathbf{y}[:,:,1] = \mathbf{y}[:,:,2]$ in NumPy notation. In our experiments, each channel of this gray scale image is obtained from a colorful image by averaging the RGB channels with

$$0.2989R + 0.5870G + 0.1140B.$$

We define $\boldsymbol{\Omega} \in \{0,1\}^{h \times w \times 3}$ to be a binary mask such that

$$\boldsymbol{\Omega}[i,j,k] = \begin{cases} 1, & k = 1 \text{ or } 2 \\ 0, & k = 0 \end{cases}.$$

Let $\boldsymbol{Q} \in \mathbb{R}^{3 \times 3}$ be an orthogonal matrix whose first column is proportional to the vector $(0.2989, 0.5870, 0.1140)$. This orthogonal matrix can be obtained easily via QR decomposition, and we use the following in our experiments

$$\boldsymbol{Q} = \begin{pmatrix} 0.4471 & -0.8204 & 0.3563 \\ 0.8780 & 0.4785 & 0 \\ 0.1705 & -0.3129 & -0.9343 \end{pmatrix}.$$

We then define the linear transformation $\boldsymbol{A} : \mathbf{x} \in \mathbb{R}^{h \times w \times 3} \mapsto \mathbf{y} \in \mathbb{R}^{h \times w \times 3}$, where

$$\mathbf{y}[i,j,k] = \sum_{l=0}^{2} \mathbf{x}[i,j,l]\boldsymbol{Q}[l,k].$$

Because $\boldsymbol{Q}$ is orthogonal, the inversion $\boldsymbol{A}^{-1} : \mathbf{y} \in \mathbb{R}^{h \times w} \mapsto \mathbf{x} \in \mathbb{R}^{h \times w \times 3}$ is easy to compute, where

$$\mathbf{x}[i,j,k] = \sum_{l=0}^{2} \mathbf{y}[i,j,l]\boldsymbol{Q}[k,l].$$

With $\boldsymbol{A}$ and $\boldsymbol{\Omega}$ defined as above, we can now use Algorithm 4 for image colorization.

Super-resolution With a similar strategy, we employ Algorithm 4 for image super-resolution. For simplicity, we assume that the down-sampled image is obtained by averaging non-overlapping patches of size $p \times p$. Suppose the shape of full resolution images is $h \times w \times 3$. Let $\mathbf{y} \in \mathbb{R}^{h \times w \times 3}$ denote a low-resolution image naively up-sampled to full resolution, where pixels in each non-overlapping patch share the same value. Additionally, let $\mathbf{\Omega} \in \{0, 1\}^{h/p \times w/p \times p^2 \times 3}$ be a binary mask such that

$$\mathbf{\Omega}[i, j, k, l] = \begin{cases} 1, & k \geqslant 1 \\ 0, & k = 0 \end{cases}.$$

Similar to image colorization, super-resolution requires an orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{p^2 \times p^2}$ whose first column is $(1/p, 1/p, \cdots, 1/p)$. This orthogonal matrix can be obtained with QR decomposition. To perform super-resolution, we define the linear transformation $\boldsymbol{A} : \mathbf{x} \in \mathbb{R}^{h \times w \times 3} \mapsto \mathbf{y} \in \mathbb{R}^{h/p \times w/p \times p^2 \times 3}$, where

$$\mathbf{y}[i, j, k, l] = \sum_{m=0}^{p^2-1} \mathbf{x}[i \times p + (m - m \bmod p)/p, j \times p + m \bmod p, l]\boldsymbol{Q}[m, k].$$

The inverse transformation $\boldsymbol{A}^{-1} : \mathbf{y} \in \mathbb{R}^{h/p \times w/p \times p^2 \times 3} \mapsto \mathbf{x} \in \mathbb{R}^{h \times w \times 3}$ is easy to derive, with

$$\mathbf{x}[i, j, k, l] = \sum_{m=0}^{p^2-1} \mathbf{y}[i \times p + (m - m \bmod p)/p, j \times p + m \bmod p, l]\boldsymbol{Q}[k, m].$$

Above definitions of $\boldsymbol{A}$ and $\mathbf{\Omega}$ allow us to use Algorithm 4 for image super-resolution.

Stroke-guided image generation We can also use Algorithm 4 for stroke-guided image generation as introduced in SDEdit (Meng et al., 2021). Specifically, we let $\mathbf{y} \in \mathbb{R}^{h \times w \times 3}$ be a stroke painting. We set $\boldsymbol{A} = \boldsymbol{I}$, and define $\mathbf{\Omega} \in \mathbb{R}^{h \times w \times 3}$ as a matrix of ones. In our experiments, we set $t_1 = 5.38$ and $t_2 = 2.24$, with $N = 2$.

Denoising It is possible to denoise images perturbed with various scales of Gaussian noise using a single consistency model. Suppose the input image $\mathbf{x}$ is perturbed with $\mathcal{N}(\mathbf{0}; \sigma^2 \boldsymbol{I})$. As long as $\sigma \in [\epsilon, T]$, we can evaluate $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma)$ to produce the denoised image.

Interpolation We can interpolate between two images generated by consistency models. Suppose the first sample $\mathbf{x}_1$ is produced by noise vector $\mathbf{z}_1$, and the second sample $\mathbf{x}_2$ is produced by noise vector $\mathbf{z}_2$. In other words, $\mathbf{x}_1 = \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{z}_1, T)$ and $\mathbf{x}_2 = \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{z}_2, T)$. To interpolate between $\mathbf{x}_1$ and $\mathbf{x}_2$, we first use spherical linear interpolation to get

$$\mathbf{z} = \frac{\sin[(1 - \alpha)\psi]}{\sin(\psi)}\mathbf{z}_1 + \frac{\sin(\alpha\psi)}{\sin(\psi)}\mathbf{z}_2,$$

where $\alpha \in [0, 1]$ and $\psi = \arccos(\frac{\mathbf{z}_1^\top \mathbf{z}_2}{\|\mathbf{z}_1\|_2 \|\mathbf{z}_2\|_2})$, then evaluate $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{z}, T)$ to produce the interpolated image.

E. Additional Samples from Consistency Models

We provide additional samples from consistency distillation (CD) and consistency training (CT) on CIFAR-10 (Figs. 14 and 18), ImageNet 64×64 (Figs. 15 and 19), LSUN Bedroom 256×256 (Figs. 16 and 20) and LSUN Cat 256×256 (Figs. 17 and 21).

Figure 8: Gray-scale images (left), colorized images by a consistency model (middle), and ground truth (right).

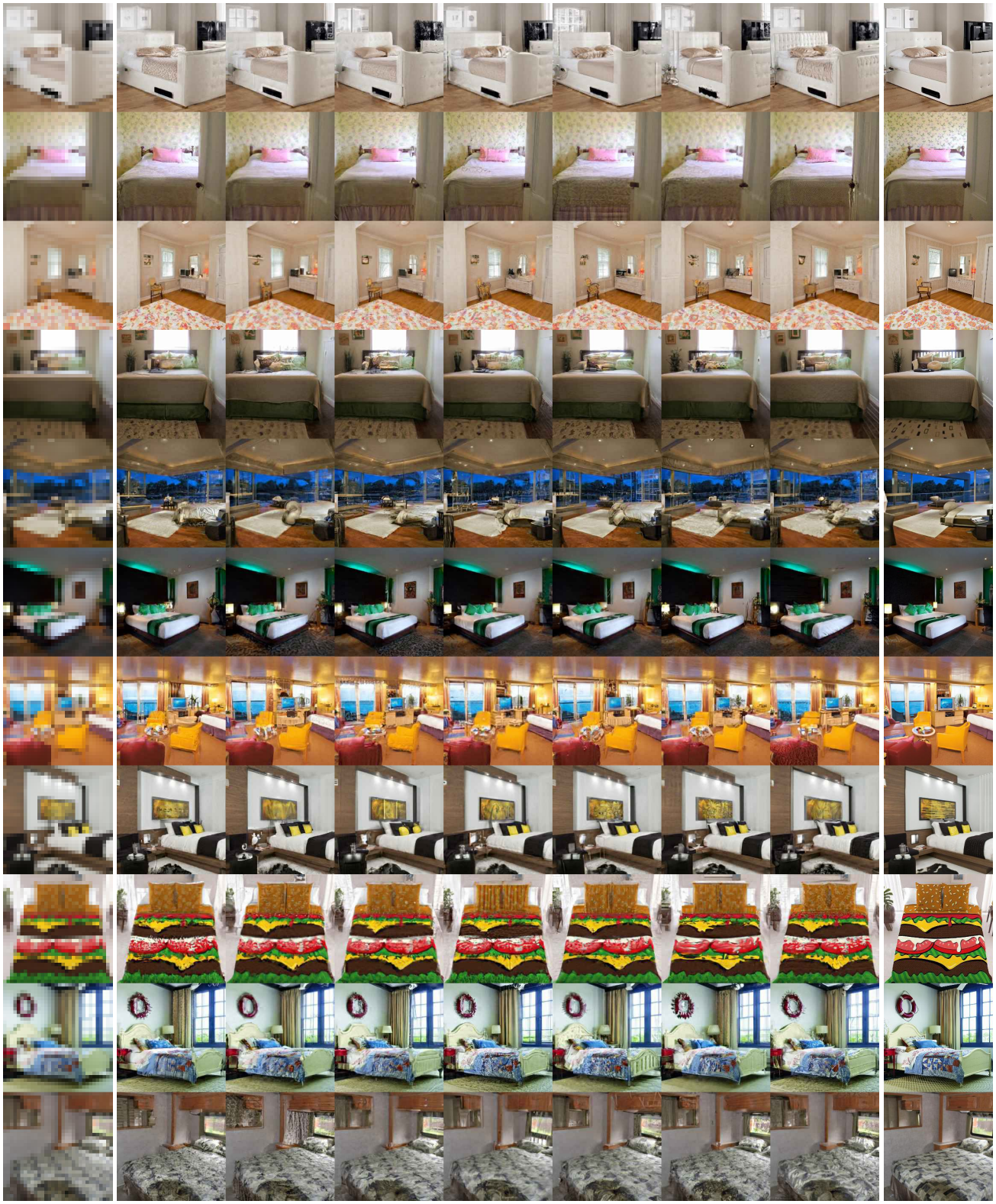


Figure 9: Downsampled images of resolution 32×32 (left), full resolution (256×256) images generated by a consistency model (middle), and ground truth images of resolution 256×256 (right).

Figure 10: Masked images (left), imputed images by a consistency model (middle), and ground truth (right).

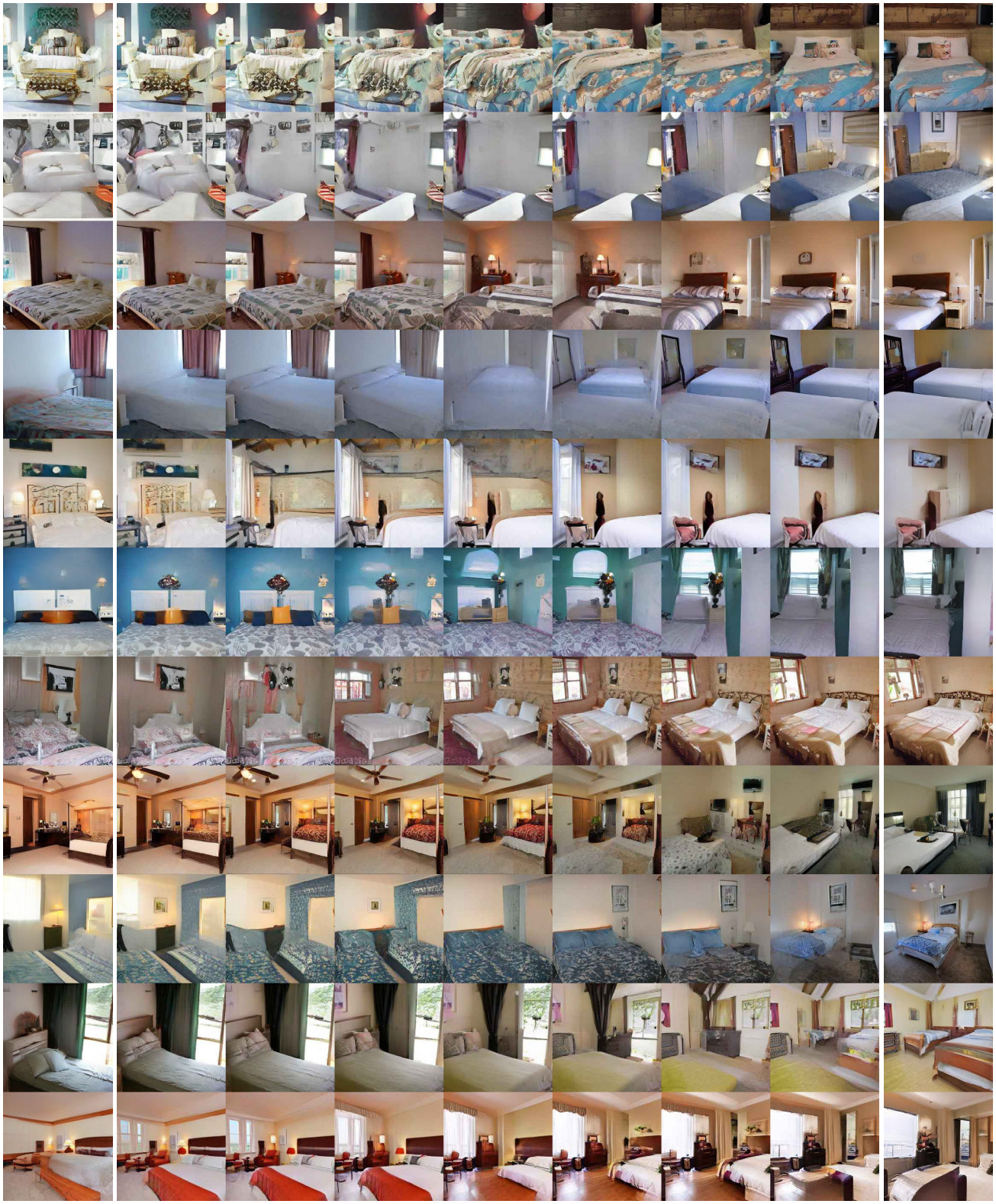


Figure 11: Interpolating between leftmost and rightmost images with spherical linear interpolation. All samples are generated by a consistency model trained on LSUN Bedroom 256×256.

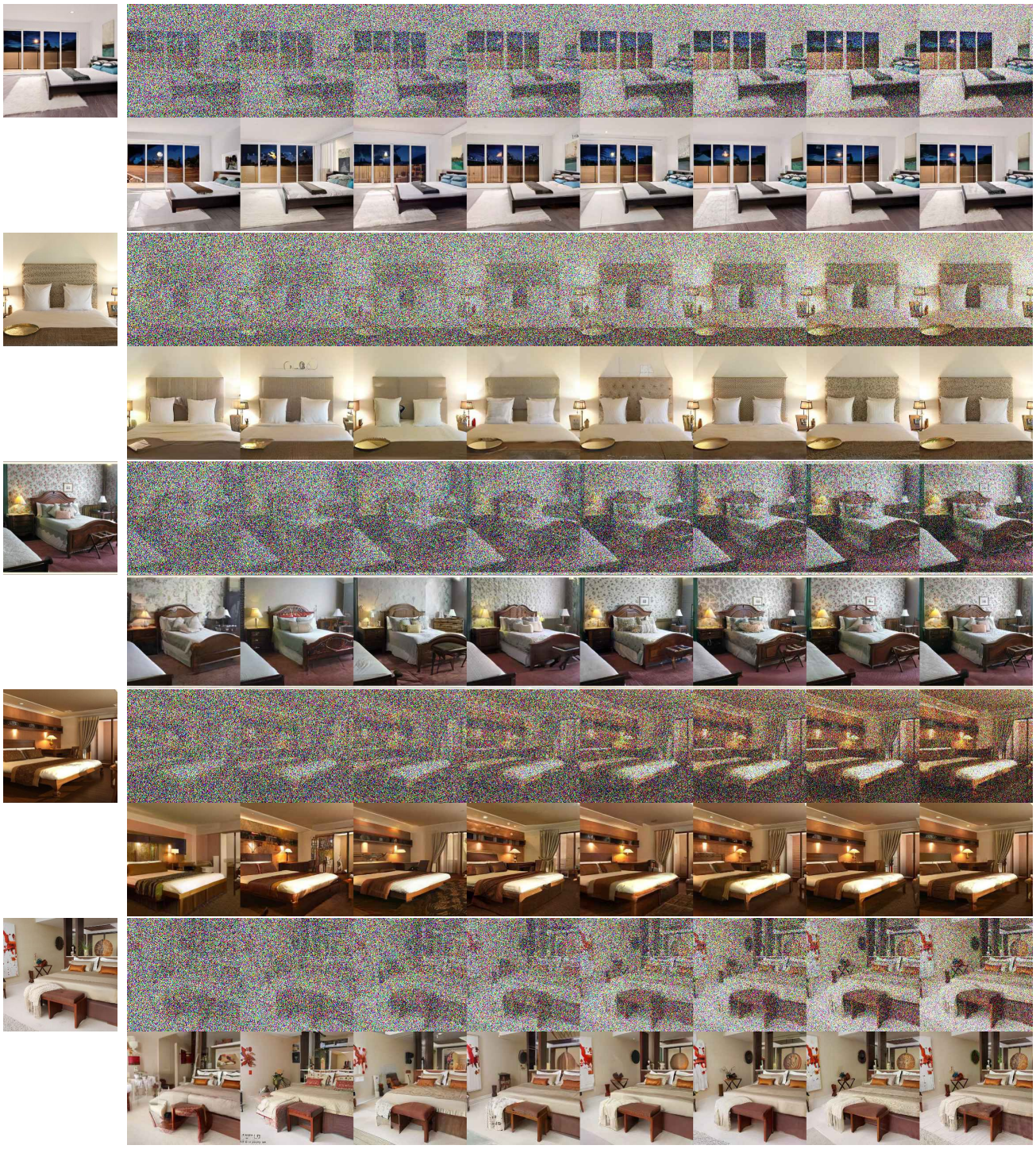
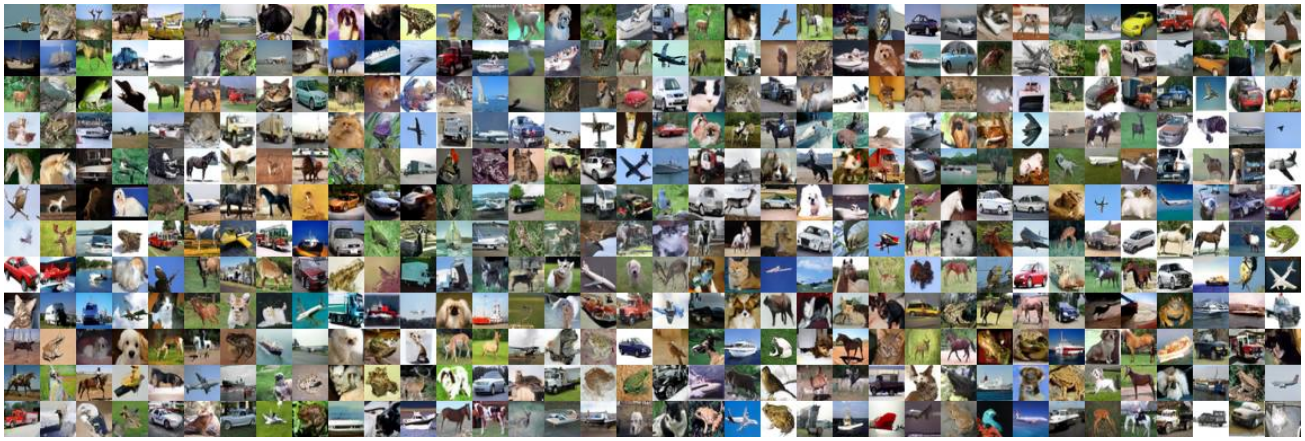


Figure 12: Single-step denoising with a consistency model. The leftmost images are ground truth. For every two rows, the top row shows noisy images with different noise levels, while the bottom row gives denoised images.

Figure 13: SDEdit with a consistency model. The leftmost images are stroke painting inputs. Images on the right side are the results of stroke-guided image generation (SDEdit).

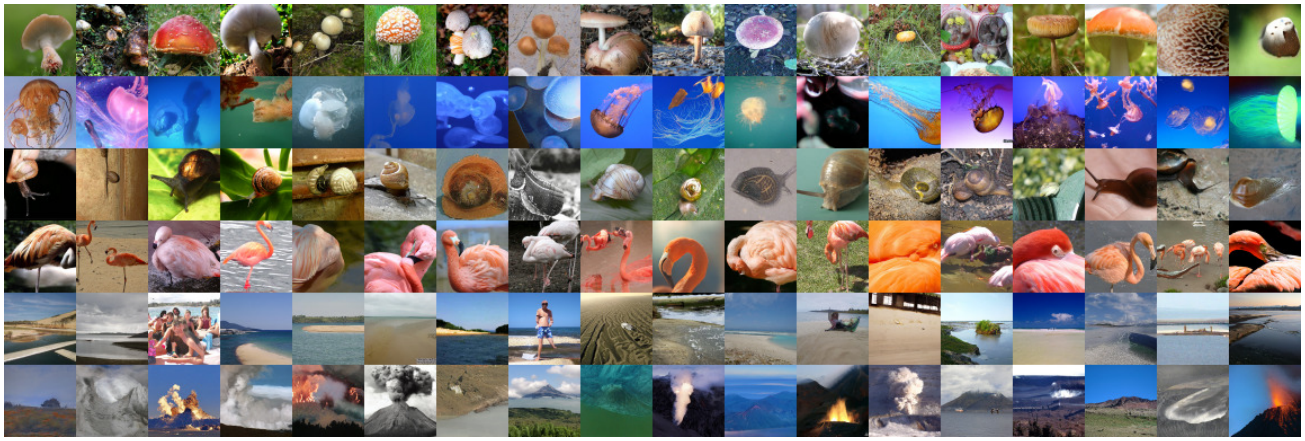
(a) EDM (FID=2.04)



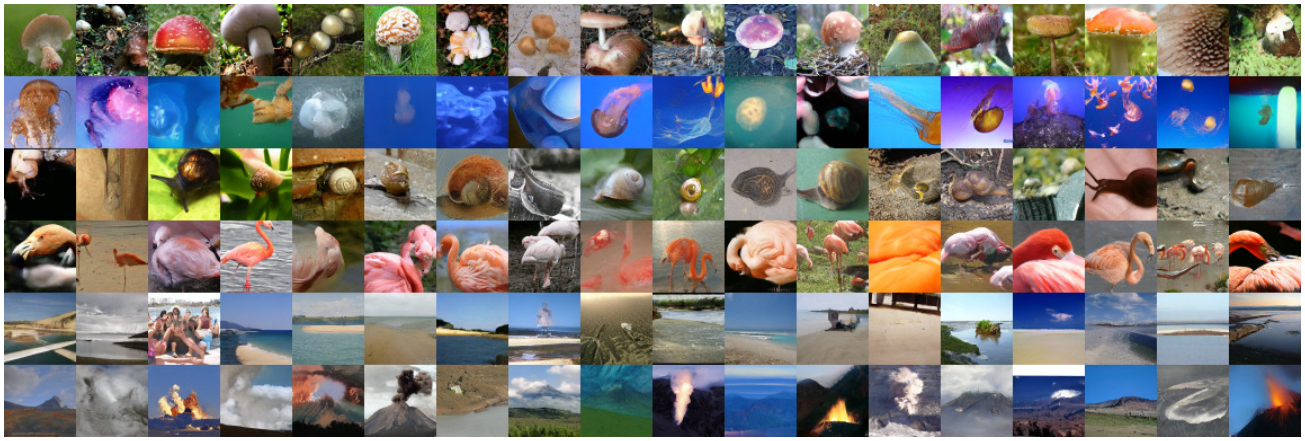
(b) CD with single-step generation (FID=3.55)

(c) CD with two-step generation (FID=2.93)

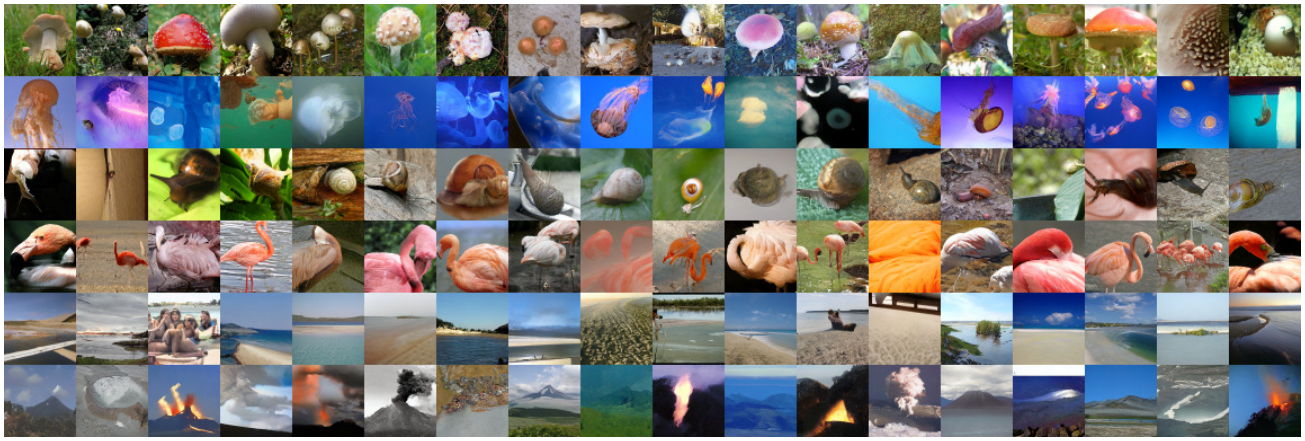
Figure 14: Uncurated samples from CIFAR-10 32×32. All corresponding samples use the same initial noise.



(a) EDM (FID=2.44)



(b) CD with single-step generation (FID=6.20)



(c) CD with two-step generation (FID=4.70)

Figure 15: Uncurated samples from ImageNet 64×64. All corresponding samples use the same initial noise.

(a) EDM (FID=3.57)

(b) CD with single-step generation (FID=7.80)

(c) CD with two-step generation (FID=5.22)

Figure 16: Uncurated samples from LSUN Bedroom 256×256. All corresponding samples use the same initial noise.

(a) EDM (FID=6.69)

(b) CD with single-step generation (FID=10.99)

(c) CD with two-step generation (FID=8.84)

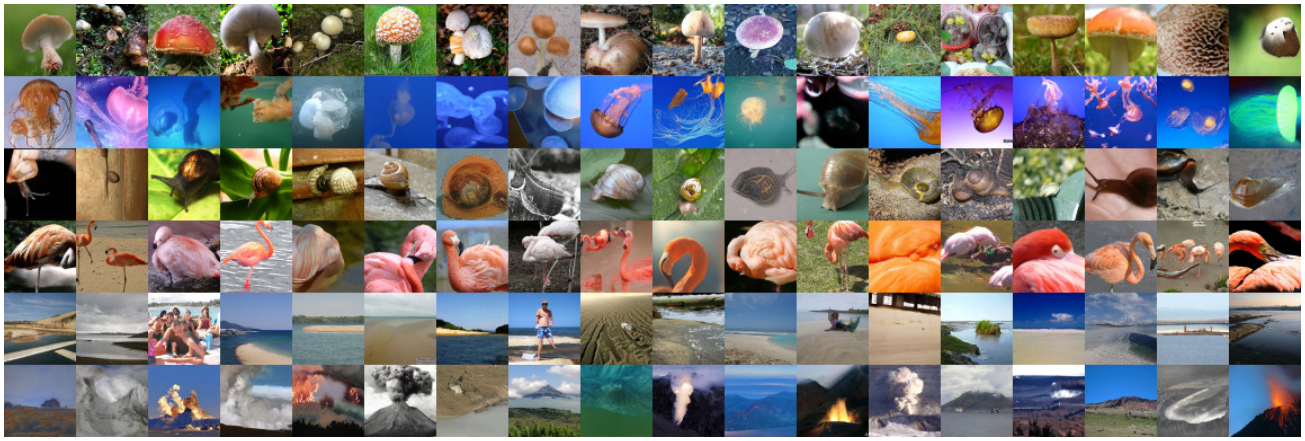
Figure 17: Uncurated samples from LSUN Cat 256×256. All corresponding samples use the same initial noise.

(a) EDM (FID=2.04)

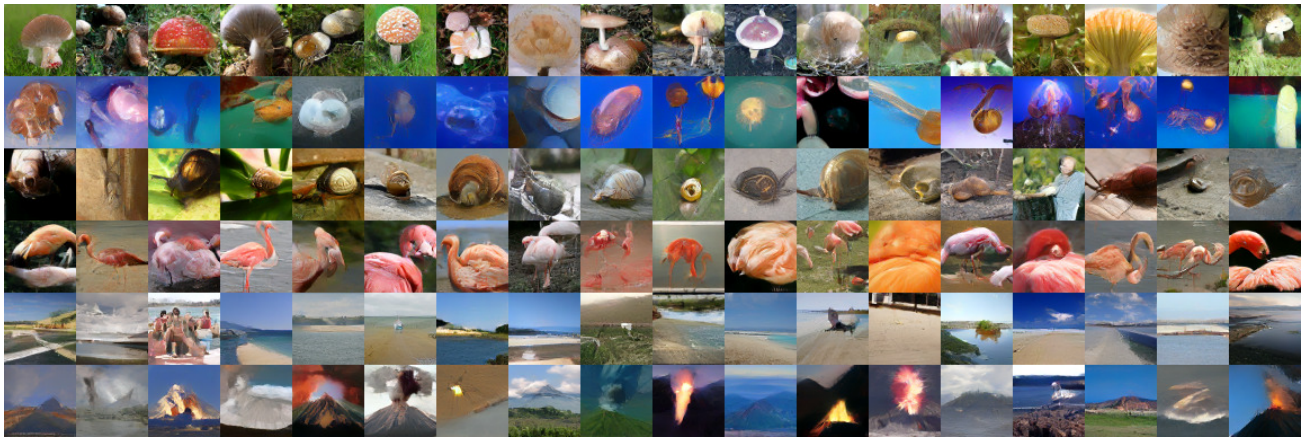
(b) CT with single-step generation (FID=8.73)

(c) CT with two-step generation (FID=5.83)

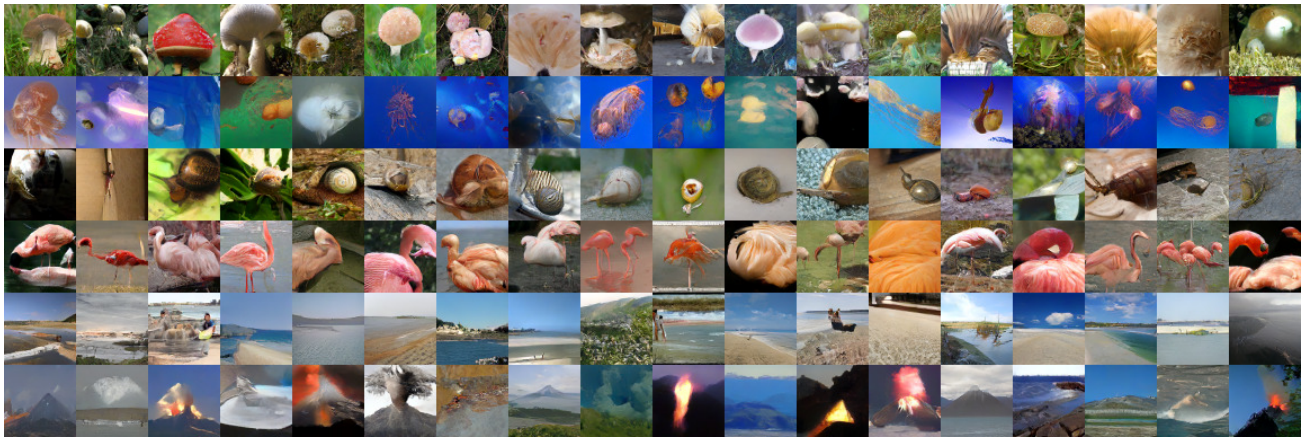
Figure 18: Uncurated samples from CIFAR-10 32×32. All corresponding samples use the same initial noise.



(a) EDM (FID=2.44)



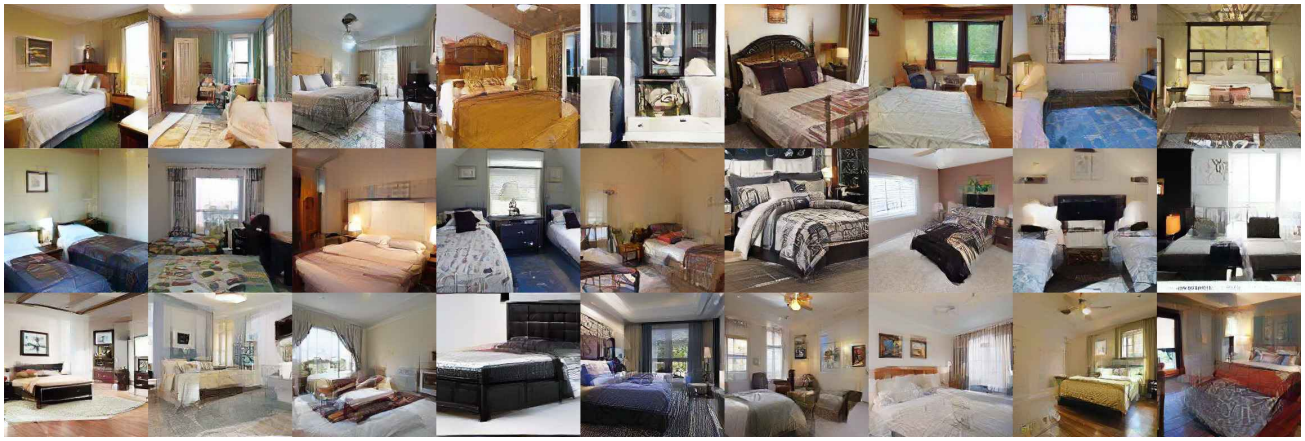
(b) CT with single-step generation (FID=12.96)



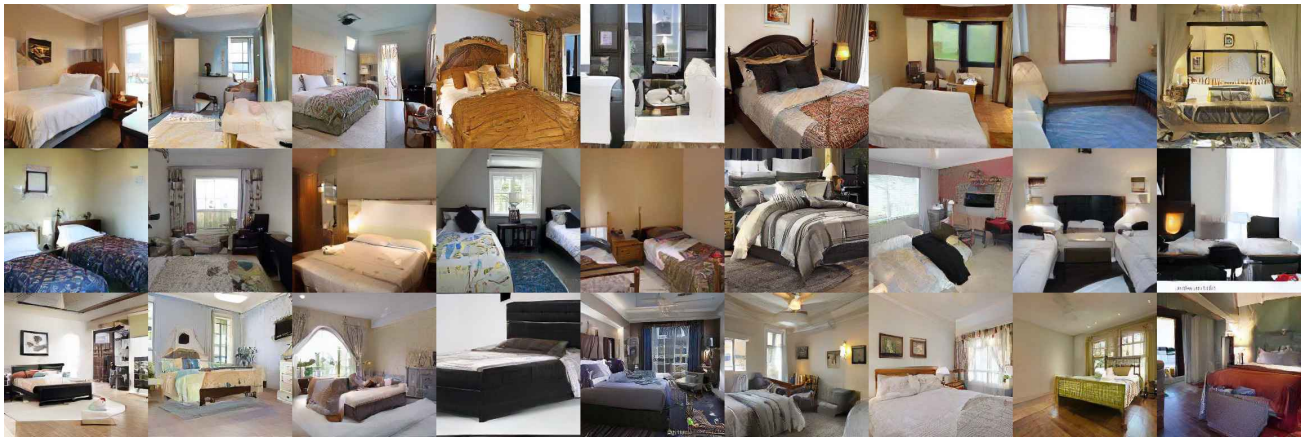
(c) CT with two-step generation (FID=11.12)

Figure 19: Uncurated samples from ImageNet 64×64. All corresponding samples use the same initial noise.

(a) EDM (FID=3.57)



(b) CT with single-step generation (FID=16.00)



(c) CT with two-step generation (FID=7.80)

Figure 20: Uncurated samples from LSUN Bedroom 256×256. All corresponding samples use the same initial noise.

(a) EDM (FID=6.69)

(b) CT with single-step generation (FID=20.70)

(c) CT with two-step generation (FID=11.76)

Figure 21: Uncurated samples from LSUN Cat 256×256. All corresponding samples use the same initial noise.