Simple and Fast Distillation of Diffusion Models

Zhenyu Zhou[1,2] **Defang Chen**[3†] **Can Wang**[1,2] **Chun Chen**[1,2] **Siwei Lyu**[3]
[1]Zhejiang University, State Key Laboratory of Blockchain and Data Security
[2]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
[3]University at Buffalo, State University of New York
{zhyzhou, defchern}@zju.edu.cn

Abstract

Diffusion-based generative models have demonstrated their powerful performance across various tasks, but this comes at a cost of the slow sampling speed. To achieve both efficient and high-quality synthesis, various distillation-based accelerated sampling methods have been developed recently. However, they generally require time-consuming fine tuning with elaborate designs to achieve satisfactory performance in a specific number of function evaluation (NFE), making them difficult to employ in practice. To address this issue, we propose **S**imple and **F**ast **D**istillation (SFD) of diffusion models, which simplifies the paradigm used in existing methods and largely shortens their fine-tuning time up to 1000×. We begin with a vanilla distillation-based sampling method and boost its performance to state of the art by identifying and addressing several small yet vital factors affecting the synthesis efficiency and quality. Our method can also achieve sampling with variable NFEs using a single distilled model. Extensive experiments demonstrate that SFD strikes a good balance between the sample quality and fine-tuning costs in few-step image generation task. For example, SFD achieves 4.53 FID (NFE=2) on CIFAR-10 with only **0.64 hours** of fine-tuning on a single NVIDIA A100 GPU. Our code is available at `https://github.com/zju-pi/diff-sampler`.

1 Introduction

Diffusion models have attracted increasing interest in recent years due to their remarkable generative abilities across various domains, including image [41, 44, 42], video [14, 2], audio [20, 24], and molecular structures [54]. These models progressively transform a noisy input into a realistic output through iterative denoising steps. Diffusion models are preferred over other generative models [9, 19] for their high-quality synthesis, stable training and a strong theoretical foundation rooted in stochastic differential equations [51]. However, achieving high-quality synthesis with diffusion models typically requires hundreds to thousands of sampling steps, resulting in slow sampling speeds and a significant challenge for practical applications.

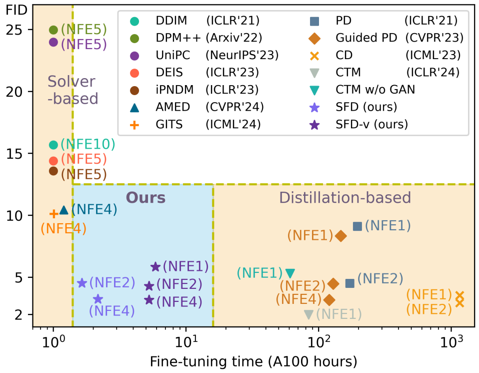


Figure 1: *Comparison of acceleration methods on diffusion models. For better visualization, the time axis is shifted by adding one hour to the actual time required. Our method achieves good performance with a small fine-tuning cost. Note that it takes about* **200 hours** *to train a diffusion model from scratch in this setting.*

[†] Corresponding author. Work partially done during Defang's time at Zhejiang University.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

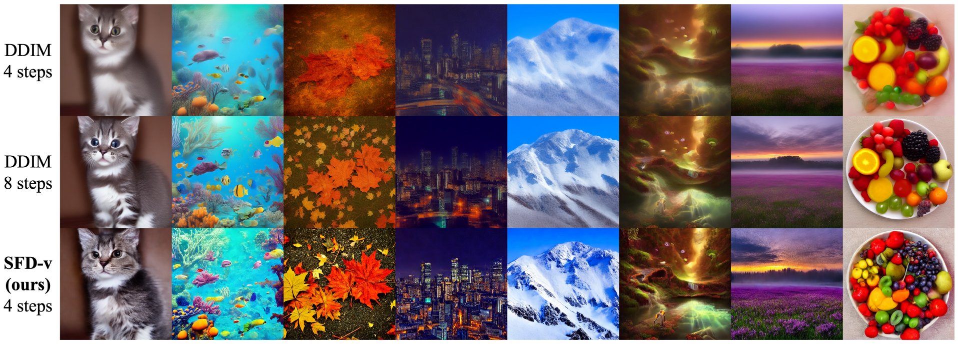


Figure 2: Comparison of synthesized images by Stable Diffusion v1.5 [41] with guidance scale 7.5.

Recent years have witnessed significant progress in accelerating the sampling of diffusion models [48, 30, 58, 60, 8, 38, 63, 4, 31, 45, 34, 50, 17, 52, 7]. These methods typically fall into two categories: *solver-based methods* and *distillation-based methods*. Solver-based methods [48, 30, 58, 16, 60, 8, 38, 63, 4] consider sampling from diffusion models as solving differential equations, and employ fast numerical solvers to accelerate high-quality synthesis. However, these methods are limited by inherent truncation errors, and the sample quality becomes degraded when the number of function evaluations (NFEs) is relatively small (e.g., NFE ≤ 5). Distillation-based methods, on the other hand, retain the structure of the original (teacher) diffusion model but aim to create a simplified (student) model that streamlines the iterative refinement process of diffusion models [31, 45, 34, 50, 17, 52, 7]. Extreme distillation-based methods even establish a direct one-to-one mapping between the implicit data distribution and a pre-specified noise distribution [31, 27, 50, 10, 56]. Although distillation-based methods have demonstrated impressive results, often outperforming solver-based methods in sampling quality given the total NFE budge is less than 5, they require expensive computational resources to fine tuning pre-trained diffusion models. As illustrated in Figure 1, the necessary time generally exceeds one hundred GPU hours, *even reaching the same order of magnitude required for training a diffusion model from scratch*. We attribute the time-consuming fine-tuning process to the following two factors:

- **The mismatch between fine-tuning and sampling steps**. There often exists significant fine-tuning costs in existing distillation-based methods that does not effectively contribute to the final performance due to the step mismatch. For example, progressive distillation [45, 34, 1] fine-tunes the diffusion model at thousands of timestamps but only a few steps (e.g., 8 or fewer) are used in sampling. Besides, consistency-based distillation [50] spends most fine-tuning efforts to ensure the consistency property [5], yet only 1 or 2 steps are used in sampling. Such inconsistencies waste excessive efforts in the fine-tuning process.
- **The complex optimization objectives**. The optimization objectives of distillation-based methods are getting increasingly complex, including the use of LPIPS [59, 50, 17, 56], adversarial training [46, 17] as well as various regularization terms [17, 56]. Despite the improved results, these additional components complicate the fine-tuning process.

In this paper, we introduce **S**imple and **F**ast **D**istillation (SFD) of diffusion models, which aims to achieve fast and high-quality synthesis with diffusion models in a few sampling steps, at minimal fine-tuning cost. Starting from the general framework behind distillation-based methods, we address the issue of step mismatch by fine-tuning only a small number of timestamps that will be used in sampling, which significantly improves the fine-tuning efficiency. The effectiveness of this strategy is underpinned by the key observation that, fine-tuning at a specific timestamp can positively impact the gradient direction at other timestamps (Section 3.1). Our SFD is then introduced as a simplified paradigm for the distillation of diffusion models, where the student learns to mimic the teacher's sampling trajectory while minimizing accumulated errors. We release the potential of this simple framework by identifying and addressing several small yet vital factors affecting the performance (Section 3.2). Moreover, we propose a variable-NFE version of our method named SFD-v, which enables a single distilled model to achieve sampling with various steps by introducing a *step-condition* into the model (Section 3.3).

With 2 NFE, our SFD achieves a FID of 4.53 on CIFAR-10 [21] with a training cost of just **0.64 hours** on a single NVIDIA A100 GPU, which is $1000\times$ faster than consistency distillation requiring about 1156 hours (see Figure 1). Quantitative and qualitative results on additional datasets, including

ImageNet 64×64 [43], Bedroom 256×256 [57], and image generation with Stable Diffusion [41], demonstrate the effectiveness and efficiency of our methods.

2 Preliminary

2.1 Diffusion Models

Diffusion models bridge the implicit data distribution p_d and a Gaussian distribution p_n by progressively adding white Gaussian noise to the data and then iteratively reconstructing the original data from pure noise. Diffusion models are grounded in a theoretical framework based on stochastic differential equations (SDEs) [51], with the forward process injecting noise to data:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}_t, \tag{1}$$

where $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d, g(\cdot) : \mathbb{R} \to \mathbb{R}$ are drift and diffusion coefficients, and $\mathbf{w}_t \in \mathbb{R}^d$ denotes the Wiener process [37]. The backward process reconstructs the original data from the noisy input, which can be achieved with a *reverse-time* SDE that shares the same marginals determined by the forward SDE, i.e., $\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_\mathbf{x} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}_t$, where $\nabla_\mathbf{x} \log p_t(\mathbf{x})$ is known as *score function* [15, 33]. The reverse-time SDE can be further simplified to the *probability flow ordinary differential equation* (PF-ODE) [51, 16, 3], $\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\nabla_\mathbf{x} \log p_t(\mathbf{x})]\mathrm{d}t$. In particular, we consider $\mathbf{f}(\mathbf{x}, t) = \mathbf{0}$ and $g(t) = \sqrt{2t}$ in this paper, i.e.,

$$\mathrm{d}\mathbf{x} = -t\nabla_\mathbf{x} \log p_t(\mathbf{x})\mathrm{d}t, \tag{2}$$

The score function is estimated as $\nabla_\mathbf{x} \log p_t(\mathbf{x}) \approx -\boldsymbol{\epsilon}_\theta(\mathbf{x}, t)/t$ with a noise-prediction model $\boldsymbol{\epsilon}_\theta(\mathbf{x}, t)$ which is obtained by minimizing a regression loss with the weighing function $\lambda(t)$ for each t [12, 48, 63]:

$$\mathcal{L}_t(\theta) = \lambda(t)\mathbb{E}_{\mathbf{x}\sim p_d, \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \|\boldsymbol{\epsilon}_\theta(\mathbf{x} + t\boldsymbol{\epsilon}, t) - \boldsymbol{\epsilon}\|_2^2. \tag{3}$$

With the noise-prediction model in place of the score function, the PF-ODE can be written as follows

$$\mathrm{d}\mathbf{x} = \boldsymbol{\epsilon}_\theta(\mathbf{x}, t)\mathrm{d}t. \tag{4}$$

Compared to the general reverse-time SDEs, the PF-ODE is preferred in practice for its conceptual simplicity and efficient sampling [51, 3]. To sample from a diffusion model with N steps, one first draws $\mathbf{x}_N \sim p_n = \mathcal{N}(\mathbf{0}, t_{\max}^2\mathbf{I})$ and then numerically solves Eq. 4 by a solver-based method [48, 29, 30, 58, 63, 4], following a hand-crafted time schedule $\Gamma(N) = \{t_0 = t_{\min}, t_1, \cdots, t_N = t_{\max}\}$. The obtained sample sequence $\{\mathbf{x}_n\}_{n=0}^N$ is called the *sampling trajectory*.

2.2 Distillation-based Diffusion Sampling

Through the lens of Eq. 4, we can interpret the noise-prediction model as a gradient field evolves over time, guiding samples towards the data distribution's manifold. Solver-based sampling methods do not change the gradient field and are convenient to implement [48, 29, 30, 58, 63, 4]. However, discretization errors prevent these methods from generating high-quality images within a few sampling steps. Distillation-based methods address this issue by fine-tuning the gradient field with the reference signals provided by a teacher (mostly a solver-based method) to build "shortcuts" on the sampling trajectory [45, 50, 34, 17]. This basic framework behind distillation-based methods, which we call *Trajectory Distillation*, is illustrated in Algorithm 1. Specifically, starting from latent encodings $\mathbf{x}_{n+1}$ and $\tilde{\mathbf{x}}_{n+1}$ $(0 \le n \le N-1)$ with a sampled n, the sampling process is written as:

$$\text{Teacher}: \quad \tilde{\mathbf{x}}_n = \mathrm{Solver}(\tilde{\mathbf{x}}_{n+1}, t_{n+1}, t_n, K; \theta), \tag{5}$$

$$\text{Student}: \quad \mathbf{x}_n^\psi = \mathrm{Euler}(\mathbf{x}_{n+1}, t_{n+1}, t_n, 1; \psi) = \mathbf{x}_{n+1} + (t_{n+1} - t_n)\boldsymbol{\epsilon}_\psi(\mathbf{x}_{n+1}, t_{n+1}), \tag{6}$$

where K is the number of teacher sampling steps taken from t_{n+1} to t_n; and ψ, θ are the parameters of the student and teacher model, respectively. In each training iteration, the student model ψ is updated with the calculated loss function $\mathcal{L}(\psi) = d(\mathbf{x}_n^\psi, \tilde{\mathbf{x}}_n)$ using a distance metric $d(\cdot, \cdot)$. $\mathrm{Solver}(\cdot, \cdot, \cdot, \cdot; \theta)$ can be any solver-based method with the fixed θ to provide reference signals. For example, in progressive distillation [45, 34], it is defined as the Euler sampler [48] with $K = 2$, while in consistency distillation [50, 17], the Heun sampler [16], $K = 1$, and a consistency loss are used.

Distillation-based methods have demonstrated impressive results but generally incur a significant computational overhead. In the following sections, we revisit the trajectory distillation framework and unlock its potential through a comprehensive assessment of the key factors affecting the performance.

Algorithm 1 Trajectory Distillation

```
repeat
    Sample $\mathbf{x}_0$ from the dataset
    Sample $n \sim \mathcal{U}(0, N-1)$
    Sample $\mathbf{x}_{n+1} \sim \mathcal{N}(\mathbf{x}_0; t_{n+1}^2\mathbf{I})$
    $\mathbf{x}_n^\psi \leftarrow \text{Euler}(\mathbf{x}_{n+1}, t_{n+1}, t_n, 1; \psi)$
    $\tilde{\mathbf{x}}_n \leftarrow \text{Solver}(\mathbf{x}_{n+1}, t_{n+1}, t_n, K; \theta)$
    $\mathcal{L}(\psi) \leftarrow d(\mathbf{x}_n^\psi, \tilde{\mathbf{x}}_n)$
    $\psi \leftarrow \psi - \eta\nabla_\psi\mathcal{L}(\psi)$
until convergence
```

Algorithm 2 SFD (ours)

```
repeat
    Sample $\mathbf{x}_N = \tilde{\mathbf{x}}_N \sim \mathcal{N}(\mathbf{0}; t_N^2\mathbf{I})$
    for $n = N-1$ to $0$ do
        $\mathbf{x}_n^\psi \leftarrow \text{Euler}(\mathbf{x}_{n+1}, t_{n+1}, t_n, 1; \psi)$
        $\tilde{\mathbf{x}}_n \leftarrow \text{Solver}(\tilde{\mathbf{x}}_{n+1}, t_{n+1}, t_n, K; \theta)$
        $\psi \leftarrow \psi - \eta\nabla_\psi d(\mathbf{x}_n^\psi, \tilde{\mathbf{x}}_n)$
        $\mathbf{x}_n \leftarrow \text{detach}(\mathbf{x}_n^\psi)$
    end for
until convergence
```

3 Method

3.1 Smooth Modification of the Gradient Field

As mentioned in Section 1, existing distillation-based methods incur significant fine-tuning costs that may not effectively contribute to the final sample quality [45, 34, 50], which is a key factor overburdening computational resources. Instead, we propose to fine-tune only a few timestamps that will be used in sampling. To validate our strategy, we initialize four different student models (denoted as MODEL(ψ_n), $0 \le n \le N-1$) from a pre-trained teacher model θ using the second-order DPM-Solver(2S) [29] with $N = 4$ and $K = 3$. We then fine-tune each MODEL(ψ_n) only on a certain timestamp t_{n+1} and make it align with the teacher predictions at the next timestamp t_n. After fine-tuning, we evaluate the performance of each MODEL(ψ_n) at all timestamps by comparing with the teacher's sampling trajectory $\{x_n\}_{n=0}^N$ under the same setting. Specifically, we calculate the L_2 distance in the following two formulas for all $0 \le n, k \le N-1$,

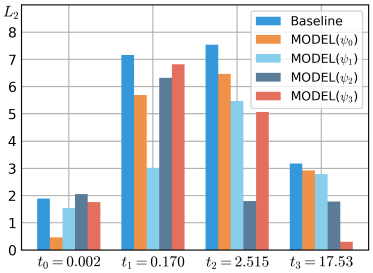


Figure 3: *MODEL(ψ_n) is trained to match the teacher's sampling trajectory at t_n but can enhance the matching at untrained timestamps. The time schedule follows the polynomial schedule with $\rho = 7, t_0 = 0.002, t_4 = 80$.*

$$\text{Baseline}: \quad \|x_n - \text{Euler}(\mathbf{x}_{n+1}, t_{n+1}, t_n, 1; \theta)\|_2, \tag{7}$$

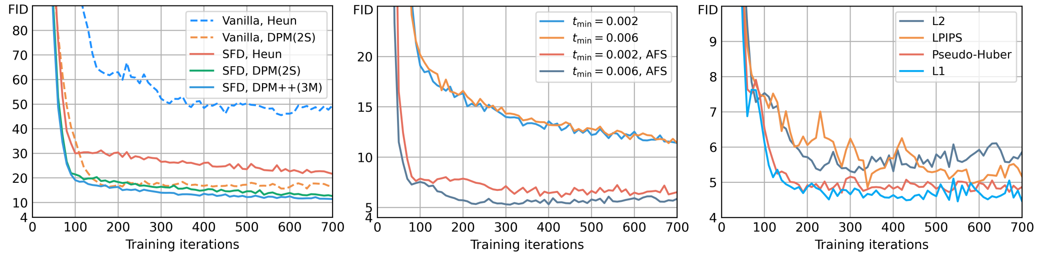
$$\text{MODEL}(\psi_k): \quad \|x_n - \text{Euler}(\mathbf{x}_{n+1}, t_{n+1}, t_n, 1; \psi_k)\|_2, \tag{8}$$

and average the results over 1,000 trajectories. As shown in Figure 3, the distance calculated using the fine-tuned models is almost consistently smaller than that of the baseline. This is remarkable since each MODEL(ψ_n) is only trained to match the teacher's sampling trajectory at a specific t_n. Yet, its performance on other timestamps is mostly improved, even though different timestamps are far apart. This indicates that trajectory distillation does not disrupt the gradient field but enhances it smoothly. Since fine-tuning at different timestamps mutually reinforces the model, fine-tuning on a fine-grained time schedule is unnecessary. This insight forms the basis of our strategy. Beyond efficiency, we will demonstrate that our approach achieves high performance in the sequel.

3.2 Simple and Fast Distillation of Diffusion Models

As for solver-based methods, the cost of a single sampling step varies depending on the design, which is commonly measured by the number of function evaluations (NFE). For the DDIM sampler [48] and other higher-order methods such as DPM-Solver++(3M) [30] and UniPC [60], one sampling step corresponds to one NFE, while two NFEs are required for the Heun sampler [16] and DPM-Solver(2S) [29]. In the following, we distill a diffusion model to achieve sampling with two NFEs ($N = 2$ by default). We configure a reasonable baseline on the CIFAR10 dataset [21] and gradually improve the performance through extensive experiments. The improved configuration is proven to be effective across different NFEs and datasets.

Default configuration. The Heun sampler is used to generate the teacher sampling trajectory instead of DDIM for efficiency, which has been demonstrated in training consistency models [50, 17]. We set



(a) Teacher solver.

(b) Timestamps.

(c) Loss metric.

Figure 4: Ablation studies of 2-NFE distillation on CIFAR10. The FID is evaluated by 50,000 generated samples with the same latent encodings and is reported every 10 iterations. We achieve the best performance with SFD, DPM-Solver++(3M) teacher, AFS, $t_{\min} = 0.006$ and L1 loss.

$K = 3$ for Heun sampler, which gives sampling trajectories with 12 NFEs. For the time schedule, if not otherwise specified, we use the polynomial schedule where $\rho = 7$, $t_{\min} = 0.002$ and $t_{\max} = 80$, following the EDM implementation [16, 50]. The squared L2 loss is used by default. For experiments in this section, we use a batch size of 128, learning rate of 5e-5 and fine-tune with 100,000 teacher sampling trajectories generated (around 780 training iterations).

From local to global. We start by analyzing the potential defects of trajectory distillation. As shown in Algorithm 1, trajectory distillation performs local fine-tuning. The term "local" indicates that the teacher model only generates part of the sampling trajectory (i.e., from t_{n+1} to t_n), and the optimization is independent across different n. This raises two defects that limit both efficiency and performance: (i) Higher-order multi-step solvers that require history evaluation records are unsupported. (ii) The student model is trained to imitate only part of the teacher's sampling trajectory. During sampling, the errors accumulate since the student is never trained to perfectly fix them. To address these issues, we view trajectory distillation from a global perspective and introduce our **S***imple and* **F***ast* **D***istillation* of diffusion models (SFD) in Algorithm 2. In each training iteration of SFD, we first generate the whole teacher sampling trajectory and let the student imitate it step by step. During this process, the student model generates its own trajectories, enabling it to learn to fix the accumulated errors. In Figure 4a, we compare both strategies (marked as "Vanilla" and "SFD") using the default configuration. It is shown that SFD exhibits better performance. In the following sections, we focus on SFD and seek to release its potential for efficient distillation of diffusion models.

Efficient solver. One of the key components that affects the efficiency of distillation-based methods is the choice of the teacher solver. To compare the performance of different solvers, we conduct experiments on both trajectory distillation and SFD with 3 representative solver-based methods: second-order Heun sampler, second-order DPM-Sovler(2S) and third-order DPM-Solver++(3M). Since history evaluations are unavailable, we exclude DPM-Solver++(3M) for trajectory distillation. The NFE of teacher sampling trajectories is kept 12 consistently ($K = 6$ is hence used for DPM-Solver++(3M)) and the results are shown in Figure 4a. DPM-Solver++(3M) stands out, and the Heun sampler is shown to be suboptimal. Therefore, for distillation-based methods with trajectory distillation involved, it is recommended to explore replacing the Heun sampler (or DDIM sampler) with DPM-Solver(2S). We leave this to future works.

Minimum and maximum timestamps. Choosing DPM-Solver++(3M) as the teacher solver, we improve SFD by adjusting the start and end timestamps during training and sampling. For the minimum timestamp $t_{\min}$, we empirically find that a slight increase improves the student and teacher sampling performance across various datasets. An ablation study of $t_{\min}$ on CIFAR10 dataset is shown in Figure 5. We increase $t_{\min}$ from 0.002 to 0.006. This change provides consistent improvements across different pre-trained diffusion models. For the maximum timestamp, we introduce analytical first step (AFS) [6, 63, 4] in the generation of student sampling trajectories, which takes one estimated step $\boldsymbol{\epsilon}_\psi(\mathbf{x}_N, t_N) \approx \mathbf{x}_N / \sqrt{1 + t_N^2}$ at the beginning of sampling to save one NFE. Therefore, to obtain a 2-NFE SFD with AFS applied, we use $N = 3$ and $K = 4$. As shown in Figure 4b, using AFS largely boosts the performance of SFD, indicating that one more inaccurate step can outperform one less step. This improvement also benefits from the nature of SFD, where the error incurred in the first step can be fixed by later steps (see the visualization in Figure 9). The detailed algorithm of SFD with AFS is included in Appendix C.

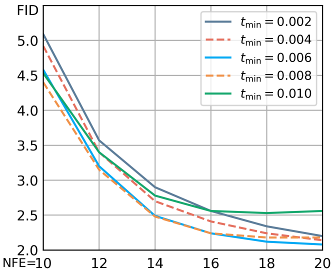


Figure 5: Ablation study on $t_{\min}$ with DPM++(3M).

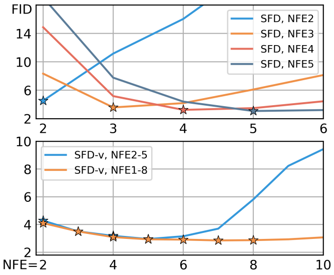


Figure 6: Extrapolation ability on untrained NFE.

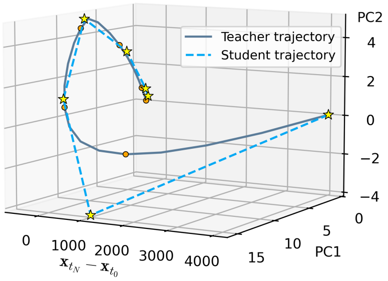


Figure 7: Visualization of the effectiveness of SFD.

Loss metric. In Figure 4c, we test various distance metrics for the loss function including squared L2 distance, L1 distance, LPIPS distance [59] and Pseudo-Huber distance [49]. Among these metrics, L1 distance outperforms. Note that the LPIPS distance is trained to evaluate the perceptual distance between two images but not corrupted ones, which may explain its suboptimal performance.

With these improvements, the SFD achieves a fast convergence with only around 300 training iterations, which only takes around **8 minutes** on a single NVIDIA A100 GPU. The performance of the obtained 2-NFE SFD is even comparable with the 2-NFE model trained by progressive distillation [45, 34], which takes more than 100 hours under our estimation. The quantitative results are included in Table 1.

Table 1: Quantitative results of the ablations.

Method	Teacher	$t_{\min}$	AFS	Loss	FID
Vanilla	Heun	0.002	N/A	L2	46.84
Vanilla	DPM(2S)	0.002	N/A	L2	16.69
SFD	Heun	0.002	False	L2	20.88
SFD	DPM(2S)	0.002	False	L2	12.50
SFD	DPM++(3M)	0.002	False	L2	11.65
SFD	DPM++(3M)	0.006	False	L2	10.93
SFD	DPM++(3M)	0.002	True	L2	7.17
SFD	DPM++(3M)	0.006	True	L2	5.67
SFD	DPM++(3M)	0.006	True	LPIPS	5.10
SFD	DPM++(3M)	0.006	True	PH	4.90
SFD	DPM++(3M)	0.006	True	L1	4.57

To verify our findings in Section 3.1, we test the extrapolation ability of SFD with untrained NFEs on CIFAR10. The results are shown in Figure 6 where the markers indicate the NFEs our methods are trained to sample with. Take "SFD, NFE4" as an example, where the SFD is only trained to sample with NFE of 4 and its performance is marked by a star. When using this SFD to sample with untrained NFEs (i.e., 2,3,5,6), even though the timestamps have never been trained in these cases, the performance is still decent and largely outperforms the DDIM sampler (DDIM with NFE of 6 gives a FID of 35.62, far exceeds the range of the figure). This empirically verifies our hypothesis that the gradient field is not disrupted but is smoothly enhanced. The change of the gradient field of a certain timestamp can also change that of nearby timestamps in a similar way.

Moreover, in Figure 7, we leverage the three-dimensional projection technique proposed in [4] to visualize the sampling trajectories generated by SFD with 5 NFEs and that of the teacher solver SFD is trained to imitate. Due to the use of AFS, the first sampling step of SFD is inaccurate. But the accumulated errors are largely reduced in the following steps thanks to the global distillation used in our SFD as discussed in Section 3.2. We include more visualised trajectories in Appendix D.2.

3.3 Towards Variable-NFE Distillation

One attractive property of diffusion models is that the sample quality can be consistently improved by increasing the sampling steps, which is currently unsupported by most distillation-based methods. Progressive distillation [45, 34] partially addresses this issue using the multi-stage training. However, the model saved in each training round only supports sampling with a certain trained step (i.e., 1, 2, 4, 8, $\cdots$). The sample quality of consistency models [50, 49] designed for the one-step sampling also deteriorates under larger sampling steps as revealed by [17]. Moreover, the unique encoding property of ODE-based diffusion sampling is corrupted in multi-step consistency models due to the noise injected in every step.

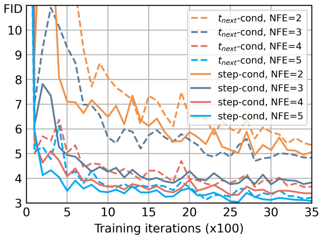
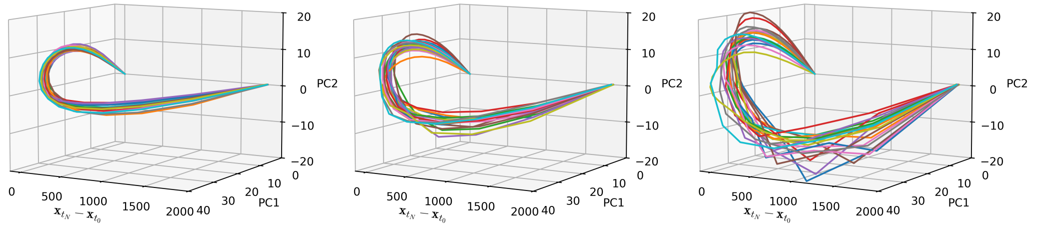


Figure 8: *Ablation study on the type of condition.*



(a) Guidance scale = 1.0. (b) Guidance scale = 4.0. (c) Guidance scale = 7.5.

Figure 9: We visualize 20 sampling trajectories generated by DPM-Solver++(2M) [30] with 20 steps using the three-dimensional projection technique proposed in [4].

To address this issue, consistency trajectory models (CTM) [17] introduce a new condition into the student model, which specifies the next time to arrive, referred to as the t_{next}-condition. Unlike CTM, introducing a *step-condition* to our SFD is more efficient. By informing the student model of the number of sampling steps, our SFD can perform sampling with different NFEs. We refer to this variable-NFE version of our method as SFD-v.

In every training iteration of SFD-v, the total number of sampling steps N is first sampled uniformly from a pre-specified list of steps. Then, the time schedule $\Gamma(N)$ is generated, and the subsequent training process is the same as training SFD. As shown in the ablation study in Figure 8, the step-condition consistently outperforms the t_{next}-condition. For the injection of step-condition, we treat it the same way as the time embedding in diffusion models (see Appendix D.1 for more details). We include the algorithm of training SFD-v in Appendix C.

3.4 Distillation under Classifier-free Guidance

Stable Diffusion [41], a latent diffusion model combined with classifier-free guidance [13], has shown to be highly effective in high-resolution image generation. The classifier-free guidance extends the flexibility of the generation of diffusion models by introducing the guidance scale ω. Given a conditioning information c, the noise-prediction model is rewritten as

$$\tilde{\boldsymbol{\epsilon}}_\theta(\mathbf{x}, t, c) = \omega \boldsymbol{\epsilon}_\theta(\mathbf{x}, t, c) + (1 - \omega)\boldsymbol{\epsilon}_\theta(\mathbf{x}, t, c = \varnothing). \tag{9}$$

However, Stable Diffusion requires a large number of network parameters and sampling steps to produce a satisfying generation. Moreover, the cost of every sampling step doubles since both conditional and unconditional evaluations are involved in Eq. 9. Distilling the Stable Diffusion model into a few steps is challenging because of source-intensive requirements and the flexibility given by the guidance scale. To address this issue, existing methods either introduce an ω-condition into their model [34, 22, 32], or simply discard the guidance scale [55, 46].

Here, we propose a new strategy with the observation on the sampling trajectories generated by Stable Diffusion under different guidance scales. Following the three-dimensional projection technique proposed in [4], we visualize the sampling trajectories generated by Stable Diffusion in its latent space using DPM-Solver++(2M) starting from 20 fixed latent encodings. As shown in Figure 9, sampling trajectories projected to the three-dimensional subspace exhibit a regular boomerang shape, which is consistent with the findings in the previous work [4]. Furthermore, we observe that the sampling trajectories become more complex as the guidance scale increases, making trajectory distillation on the high guidance scale even more challenging. This observation naturally leads to our strategy: *perform distillation with a guidance scale of 1 and sampling with any guidance scale*. Our strategy enables accelerated training since the unconditional evaluation in Eq. 9 is eliminated.

4 Experiments

4.1 Experiment Setting

Pre-trained models and datasets. Both the network parameters of student and teacher models are initialized from pre-trained diffusion models provided by EDM [16] and LDM [41]. We report quantitative as well as qualitative results on datasets with various resolutions including CIFAR10

32×32 [21], ImageNet 64×64 [43] and latent-space LSUN-Bedroom 256×256 [57]. For Stable Diffusion [41], we use the v1.5 checkpoint and generate images with a resolution of 512×512.

Training. The configuration obtained in Section 3.2 can be applied to different NFEs and datasets. Generally, in the training of SFD and SFD-v, we use DPM-Solver++(3M) [30] as the teacher solver with $K = 4$ (see Appendix D.2 for an ablation study on K). The use of adjusted $t_{\min} = 0.006$, AFS and L1 loss introduced in Section 3.2 all lead to improved results. Minor changes are needed for text-to-image generation with Stable Diffusion, where we use DPM-Solver++(2M), which is the default setting used in Stable Diffusion and $K = 3$. In this case, $t_{\min}$ is increased from 0.03 to 0.1 and the AFS is disabled due to the complex trajectory shown in Figure 9c. These experiment settings are collected in Table 6 in Appendix.

Optimization. We use Adam optimizer [18] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a batch size of 128 across all datasets. A learning rate of 1e-5 is used for ImageNet and LSUN-Bedroom while 5e-5 is used in other cases. We divide the learning rate by 10 halfway through training. Our SFD is trained with a total of 200K teacher trajectories generated (around 1.5K training iterations). We train SFD-v to enable sampling with NFE from 2 to 5, the total training iterations is multiplied by 4 accordingly. All experiments are conducted with a maximum of 4 NVIDIA A100 GPUs. To enable a batch size of 128 using Stable Diffusion, we accumulate the gradient through several rounds.

Evaluation. We measure the sample quality via Fréchet Inception Distance (FID) [11] with 50K images in general. For text-to-image generation, we use a guidance scale of 7.5 to generate 5K images with prompts from the MS-COCO [23] validation set. The FID is evaluated following the protocol in [28, 34, 46] where the validation set serves as reference images. The CLIP score is computed using the ViT-g-14 CLIP model [40] trained on LAION-2B [47].

Table 2: Results on CIFAR10 32×32.

Method	NFE	FID	Training time (A100 hours)
Solver-based Methods			
DDIM [48]	10	15.69	0
	50	2.91	0
DPM++(3M) [30]	5	24.97	0
	10	3.00	0
AMED-Plugin [63]	5	6.61	~ 0.08
	10	2.48	~ 0.11
GITS [4]	5	8.38	< 0.01
	10	2.49	~ 0.01
Diffusion Distillation			
PD [45]	1	9.12	~ 195
	2	4.51	~ 171
Guided PD [34]	1	8.34	~ 146
	2	4.48	~ 128
	4	3.18	~ 119
CD [50]	1	3.55	~ 1156
	2	2.93	~ 1156
CTM [17]	1	1.98	~ 83
CTM [17] w/o GAN loss	1	> 5	~ 60
SFD (ours) (second-stage)	1	5.83	4.88
SFD (ours)	2	4.53	0.64
	3	3.58	0.92
	4	3.24	1.17
	5	3.06	1.42
SFD-v (ours)	2	4.28	4.26
	3	3.50	
	4	3.18	
	5	2.95	

Table 3: Results on ImageNet 64×64.

Method	NFE	FID	Training time (A100 hours)
Solver-based Methods			
DDIM [48]	10	16.72	0
	50	4.09	0
DPM++(3M) [30]	5	25.49	0
	10	5.67	0
AMED-Plugin [63]	5	13.83	~ 0.18
	10	5.01	~ 0.32
GITS [4]	5	10.79	< 0.02
	10	4.48	~ 0.02
Diffusion Distillation			
PD [45]	1	15.39	< 5533
	2	8.95	< 4611
Guided PD [34]	1	22.74	< 5533
	2	9.75	< 4611
	4	4.14	< 4150
CD [50]	1	6.20	< 7867
	2	4.70	< 7867
CTM [17]	1	2.06	< 902
	2	1.90	< 902
SFD (ours) (second-stage)	1	12.89	6.86
SFD (ours)	2	10.25	3.34
	3	6.35	4.63
	4	4.99	5.98
	5	4.33	7.11
SFD-v (ours)	2	9.47	23.62
	3	5.78	
	4	4.72	
	5	4.21	

4.2 Main Results

We mainly compare our proposed SFD and SFD-v with progressive distillation [45, 34] and consistency distillation [50, 17]. In Table 2 and 3, we report unconditional and conditional results on the pixel-space image generation. To compare the training cost, we estimate the training time measured by hours consumed on a single NVIDIA A100 GPU, following the training settings in the original papers. The detail of our estimation is included in Appendix B. Our SFD achieves comparable results as progressive distillation but only requires a very small fine-tuning cost (100× to 200× speedup). At the same time, it is hard for solver-based methods to give high-quality generation within a few steps due to increased errors. In accordance with our finding in Section 3.1, the SFD-v shows consistently better results than SFD, although the training cost of SFD-v and SFD is roughly the same for each specified sampling step. These observations also apply to the performance of SFD and SFD-v on the latent-space image generation on LSUN-Bedroom shown in Table 4. In Table 5, we show the performance of our methods in terms of FID and CLIP scores with a guidance scale of 7.5. The results demonstrate the effectiveness of our strategy proposed in Section 3.4 where the training is performed with the guidance scale set to 1. The qualitative results are shown in Figure 2. We include more results in Appendix D.2.

Although generating images with one NFE is possible with SFD and SFD-v, we find it suboptimal. To address this, we propose a second-stage one-NFE distillation, initializing network parameters from a fine-tuned SFD model. In this second stage, the teacher solver is set to DDIM (as used in SFD), and we use AFS ($N = 2$) and $K = 2$. The training procedure remains the same as that of SFD. The results, marked as "second-stage", are reported in Tables 2 to 4. We provide an ablation study on the effectiveness of the second stage and the LPIPS metric [59] in Figure 10. The second-stage training significantly boosts performance and is more efficient. Additionally, combining L1 loss with LPIPS loss yields better results. Since the teacher requires a smaller NFE, the training of each iteration of the second-stage distillation is fast. Therefore, for CIFAR10/ImageNet, we perform second-stage distillation with 2000/800K sampling trajectories (around 15/6K training iterations), and the learning rate is set to 10 times larger. For LSUN-Bedroom, we use 800K trajectories and disable the LPIPS loss.

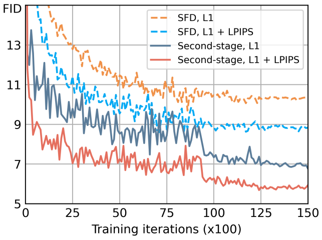


Figure 10: *Ablation study on one-NFE distillation.*

Table 4: Results on LSUN-Bedroom 256 × 256.

Method	NFE	FID
DPM++(3M) [30]	8	4.61
AMED-Plugin [63]	8	4.19
PD [45]	1	16.92
	2	8.47
CD [50]	1	7.80
	2	5.22
SFD (ours) (second-stage)	1	13.88
SFD (ours)	2	10.39
	3	6.42
	4	5.26
	5	4.73
SFD-v (ours)	2	9.25
	3	5.36
	4	4.63
	5	4.33

Table 5: Text-to-image generation with Stable Diffusion v1.5 [41]. *: Reported in the original paper [34]. We use a guidance scale of 7.5, which is the default setting in the original repository.

Method	Steps	FID-5K	CLIP Score
DPM++(2M) [30]	2	91.5 (98.8*)	0.20 (0.19*)
	4	31.1 (34.1*)	0.29 (0.29*)
	8	25.1 (25.6*)	0.32 (0.30*)
Guided PD [34]	2	37.3	0.27
	4	26.0	0.30
	8	26.9	0.30
SnapFusion [22]	8	24.2	0.30
SFD-v (ours)	2	42.9	0.24
	3	27.6	0.27
	4	24.2	0.28
	5	23.5	0.29

5 Conclusion

In this paper, we introduce **S**imple and **F**ast **D**istillation (SFD) of diffusion models to achieve fast and high-quality generation with diffusion models in a few sampling steps at minimal fine-tuning cost. Through a comprehensive investigation of several important factors, we unlock SFD's potential, achieving sample quality comparable to progressive distillation while reducing fine-tuning costs by over 100 times. To enable sampling with variable NFEs using a single distilled model, we propose SFD-v, which incorporates step-condition as an additional input. Our methods strike a good balance between sample quality and fine-tuning costs for few-step image generation, offering a new paradigm for distillation-based accelerated sampling of diffusion models.

Limitation and future work. Despite demonstrating efficient training, the FID results of our methods currently do not match those of the state-of-the-art. In future work, we plan to further explore the core mechanisms affecting the performance of trajectory distillation. Additionally, given the recent discoveries of the remarkable regular geometric structure of diffusion models [4], we aim to tailor an appropriate time schedule for our methods. We also intend to validate the effectiveness of the factors we have discussed in other distillation-based methods.

Broader impacts. Similar to existing works on content creation, our methods have the potential to be misused for malicious generation, which could have harmful social impacts. However, this risk can be mitigated through advanced deepfake detection techniques. By continuously improving these detection methods, we can help ensure the responsible and ethical use of our technology.

6 Acknowledgement

Zhenyu Zhou and Can Wang are supported by the Starry Night Science Fund of Zhejiang University Shanghai Institute for Advanced Study, China (Grant No: SN-ZJU-SIAS-001), National Natural Science Foundation of China (Grant No: U1866602) and the advanced computing resources provided by the Supercomputing Center of Hangzhou City University.

References

[1] David Berthelot, Arnaud Autef, Jierui Lin, Dian Ang Yap, Shuangfei Zhai, Siyuan Hu, Daniel Zheng, Walter Talbot, and Eric Gu. Tract: Denoising diffusion models with transitive closure time-distillation. *arXiv preprint arXiv:2303.04248*, 2023.

[2] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[3] Defang Chen, Zhenyu Zhou, Jian-Ping Mei, Chunhua Shen, Chun Chen, and Can Wang. A geometric perspective on diffusion models. *arXiv preprint arXiv:2305.19947*, 2023.

[4] Defang Chen, Zhenyu Zhou, Can Wang, Chunhua Shen, and Siwei Lyu. On the trajectory regularity of ODE-based diffusion sampling. In *International Conference on Machine Learning*, pages 7905–7934. PMLR, 2024.

[5] Giannis Daras, Yuval Dagan, Alexandros G Dimakis, and Constantinos Daskalakis. Consistent diffusion models: Mitigating sampling drift by learning to be consistent. *arXiv preprint arXiv:2302.09057*, 2023.

[6] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Genie: Higher-order denoising diffusion solvers. In *Advances in Neural Information Processing Systems*, 2022.

[7] Weilun Feng, Chuanguang Yang, Zhulin An, Libo Huang, Boyu Diao, Fei Wang, and Yongjun Xu. Relational diffusion distillation for efficient image generation. In *ACM Multimedia*, 2024.

[8] Martin Gonzalez, Nelson Fernandez Pinto, Thuy Tran, Hatem Hajri, Nader Masmoudi, et al. Seeds: Exponential sde solvers for fast high-quality sampling from diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[10] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Josh Susskind. Boot: Data-free distillation of denoising diffusion models with bootstrapping. *arXiv preprint arXiv:2306.05544*, 2023.

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

[13] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[14] Jonathan Ho, Tim Salimans, Alexey A. Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2022.

[15] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 2005.

[16] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022.

[17] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*, 2023.

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

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

[21] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[22] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *Advances in Neural Information Processing Systems*, 36, 2024.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[24] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.

[25] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[26] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2022.

[27] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[28] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*, 2023.

[29] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, 2022.

[30] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

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

[32] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

[33] Siwei Lyu. Interpretation and generalization of score matching. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 359–366, 2009.

[34] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023.

[35] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*, pages 7176–7185. PMLR, 2020.

[36] Thuan Hoang Nguyen and Anh Tran. Swiftbrush: One-step text-to-image diffusion model with variational score distillation. *arXiv preprint arXiv:2312.05239*, 2023.

[37] Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

[38] Kushagra Pandey, Maja Rudolph, and Stephan Mandt. Efficient integrators for diffusion generative models. *arXiv preprint arXiv:2310.07894*, 2023.

[39] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[40] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[42] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.

[43] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[44] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, pages 36479–36494, 2022.

[45] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

[46] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

[47] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[48] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[49] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

[50] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine learning*, 2023.

[51] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[52] Wujie Sun, Defang Chen, Can Wang, Deshi Ye, Yan Feng, and Chun Chen. Accelerating diffusion sampling with classifier-based feature distillation. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pages 810–815. IEEE, 2023.

[53] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2024.

[54] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.

[55] Yanwu Xu, Yang Zhao, Zhisheng Xiao, and Tingbo Hou. Ufogen: You forward once large scale text-to-image generation via diffusion gans. *arXiv preprint arXiv:2311.09257*, 2023.

[56] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. *arXiv preprint arXiv:2311.18828*, 2023.

[57] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[58] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *International Conference on Learning Representations*, 2023.

[59] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[60] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *arXiv preprint arXiv:2302.04867*, 2023.

[61] Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning. In *International Conference on Machine Learning*, pages 42390–42402. PMLR, 2023.

[62] Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. *arXiv preprint arXiv:2310.13268*, 2023.

[63] Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2024.

A Related Works

Solver-based methods. As sampling from diffusion models can be interpreted as solving the PF-ODE [51], various kinds of training-free solver-based methods are designed utilizing classical numerical methods including Euler's method (DDIM [48]), Heun's second method (EDM [16]), linear multi-step method (PNDM [26] and iPNDM [58]) and predictor-corrector framework (UniPC [60]). Some works focus on the semi-linear structure of the PF-ODE, approximating its solution by Taylor expansion (DPM-Solver [29], DPM-Solver++ [30] and their generalized variant SEEDS [8]) and polynomial extrapolation (DEIS [58]). Besides these training-free methods, some works further reduce the discretization error of classical numerical methods by learning intrinsic information with extra computational overhead. GENIE [6] applies the second-order truncated Taylor method and utilizes the gradient of noise-prediction model w.r.t. time distilled from the pre-trained model for accelerated sampling. DPM-Solver-v3 [62] seeks to search for the optimal parameterization for fast sampling using the Empirical Model Statistics (EMS) calculated from pre-trained models. AMED-Solver [63] achieves fast sampling resorting to the mean value theorem validated by the geometric structure of sampling trajectories and trains a small network that predicts the optimal intermediate timestamps. GITS [4] optimizes the time schedule through a dynamic programming utilizing the trajectory regularity of the diffusion models.

Aside from solver-based methods, distillation-based methods demonstrate their superiority in sampling speed and high-quality generation. Current literature can be categorized into three classes.

Trajectory distillation. Trajectory distillation originates from the primary work [31], which proposed the first one-step diffusion model with the idea of knowledge distillation. The basic framework behind is to train a student model to imitate the teacher's sampling trajectory. Progressive distillation (PD) [45, 34, 22] proposes gradually reducing the sampling steps with a multi-stage strategy. In each training round, the student model is fine-tuned to merge two DDIM steps into one step and serves as the teacher in the next training round. Following PD, trajectory matching at feature space utilizing a pre-trained classifier is shown to be effective in RCFD [52] and RDD [7]. Motivated by the idea of operator learning, DSNO [61] presents a novel way of distillation by simultaneously predicting the whole sampling trajectory with specially designed temporal convolution blocks.

Consistency distillation. Originated from consistency models [50], which is a special case of its concurrent consistent diffusion models [5], consistency distillation introduces a new way of distillation where the denoising outputs on the sampling trajectory are kept consistent. Consistency distillation shows remarkable ability in one-step generation and has become popular in distillation-based methods. Latent consistency models [32] demonstrate the effectiveness of consistency distillation in latent space. Consistency trajectory models [17] generalize the consistency to any timestamp, enabling an unrestricted traversal on sampling trajectory and producing state-of-the-art results.

Distribution matching. The idea of distribution matching for distilling diffusion models is first introduced in text-to-3D generation [39] as score distillation sampling (SDS) and is later improved to variational score distillation (VSD) [53]. Unlike the training of diffusion models, which relies on sample-wise reconstruction, distribution matching discards the coupling of noise-image pairs set by diffusion models and matches the real and reconstructed samples at a distribution level. The effectiveness of both SDS [46] and VSD [56, 36] is recently validated in image generation.

Our methods fall into the first category but differ from progressive distillation in three ways. (i) We view trajectory distillation from a global perspective, where we generate the whole teacher sampling trajectory in each iteration, and let the student model imitate it step by step. (ii) We simplify the multi-stage strategy (more than ten stages) into one or two stages. (iii) The proposed SFD-v enables sampling with different NFEs using a single distilled model. Our methods differ from DSNO in two folds. (i) Our methods enable the student model to reduce accumulated errors in previous steps, while in DSNO, the intermediate samples on the predicted sampling trajectory are not directly connected. (ii) Our methods slightly change the network architecture, while in DSNO, it is largely redesigned. Besides, the model complexity in DSNO increases as the length of the sampling trajectory increases.

B Estimation of Training Cost

In this section, we illustrate our estimation of training cost on the comparison methods shown in Table 2 and 3. For solver-based methods like AMED-Solver [63] and GITS [4], we borrow the reported training cost in original papers since similar devices are used. Efforts are put on the estimation of distillation-based methods. Due to limited resources, we do not fully re-implement these methods. Generally, after the training stabilizes, we collect the consumed time spent on several training iterations and then estimate the total training costs. Two NVIDIA A100 GPUs are used in our estimation and we double the estimated training costs to report the final results. For fair comparison, all of these methods are re-implemented with the EDM [16] repository with its provided pre-trained diffusion models. We note that the cost of training a diffusion model on CIFAR10 dataset from scratch is around 200 A100 hours.

B.1 Training Costs on CIFAR10

Progressive distillation. Following the setting in the original paper [45], we use a batch size of 128 and strictly follows the original algorithm. Adam optimizer as well as exponential moving average (EMA) is also used. After the training stabilizes, it costs around 140.5 seconds for every 320 training iterations with 2 NVIDIA A100 GPUs. As a total of 800K training iterations are required in the original paper, we estimate its training cost by $800 \times 1000 \times 140.5 \times 2/(320 \times 3600) = 195.1$ A100 hours. As for Guided PD [34], we only report the estimated training cost for its second-stage distillation, which requires a total of 600K training iterations. We roughly use the above statistics and estimate its training cost by $600 \times 1000 \times 140.5 \times 2/(320 \times 3600) = 146.4$. For the training cost under larger NFE, we scale the estimated training cost accordingly following the total number of training iterations.

Consistency distillation. The consistency distillation [50] uses a batch size of 512 and trains for a total of 800K training iterations. Following the original paper, we use the LPIPS [59] loss metric and the Rectified Adam optimizer [25]. It costs around 52 seconds for every 20 training iterations with 2 NVIDIA A100 GPUs. The training cost is thus estimated by $800 \times 1000 \times 52 \times 2/(20 \times 3600) = 1155.6$ A100 hours.

Consistency trajectory models. Following the original paper [17], we use a batch size of 256 and train with mixed-precision. For the first 50K training without the GAN loss, it costs around 21.5 seconds for every 20 training iterations and 38.5 seconds are needed for the later 50K training with GAN loss involved using 2 NVIDIA A100 GPUs. The training cost of a total of 100K iterations is estimated by $50 \times 1000 \times (21.5 + 38.5) \times 2/(20 \times 3600) = 83.3$ A100 hours. If the GAN loss is disabled during training, the training cost will be reduced to 59.7 A100 hours.

B.2 Training Costs on ImageNet

Typically, the training of ImageNet uses a large batch size (for example, the original setting is 2048 for all the methods below) which requires extensive resources (usually 64 GPUs are needed). Due to limited resources, we estimate the training cost with a batch size of 256 and multiply it by 8 which should gives an upper bound of the practical training cost.

Progressive distillation. In progressive distillation [45] as well as its guided version [34], a total of 600K training iterations are required (50K for 8 rounds and 100K for 2 rounds). With a batch size of 256, it costs around 664 seconds for every 320 training iterations with 2 NVIDIA A100 GPUs. The training cost is thus $8 \times 600 \times 1000 \times 664 \times 2/(320 \times 3600) = 5533.3$ A100 hours. For the training cost under larger NFE, the training cost is scaled accordingly as done before.

Consistency distillation. Following the original setting [50], mixed-precision optimization is applied. Other settings is similar to that in CIFAR10 illustrated above. With a batch size of 256, it costs around 59 seconds for every 20 training iterations with 2 NVIDIA A100 GPUs. As a total of 600K iterations are required, the training cost is estimated by $8 \times 600 \times 1000 \times 59 \times 2/(20 \times 3600) = 7866.7$ A100 hours.

Consistency trajectory models. Expect for a training iteration of 30K, other settings are consistent as in CIFAR10 training. For the first 10K training iterations without GAN loss, it costs around 122 seconds for every 20 training iterations with 2 NVIDIA A100 GPUs. For the later 20K training iterations with GAN loss, 142 seconds are needed. The total training cost is estimated by $8 \times 1000 \times (10 \times 122 + 20 \times 142) \times 2/(20 \times 3600) = 902.2$ A100 hours.

C Algorithms

All the algorithms involved in the main text are illustrated below.

Algorithm 3 Trajectory Distillation

```
Input: model parameters $\psi = \theta$, learning rate $\eta$, $\mathrm{Solver}(\cdot,\cdot,\cdot,\cdot)$, distance metric $d(\cdot,\cdot)$, number of student sampling steps $N$, noise schedule $\{t_n\}_{n=0}^{N}$, number of teacher sampling steps between every two noise levels $K$, dataset $\mathcal{D}$.
repeat
    Sample $\mathbf{x}_0 \sim \mathcal{D}$
    Sample $n \sim \mathcal{U}(0, N-1)$
    Sample $\mathbf{x}_{n+1} \sim \mathcal{N}(\mathbf{x}_0; t_{n+1}^2\mathbf{I})$
    $\mathbf{x}_n^{\psi} \leftarrow \mathrm{Euler}(\mathbf{x}_{n+1}, t_{n+1}, t_n, 1; \psi)$
    $\tilde{\mathbf{x}}_n \leftarrow \mathrm{Solver}(\mathbf{x}_{n+1}, t_{n+1}, t_n, K; \theta)$
    $\mathcal{L}(\psi) \leftarrow d(\mathbf{x}_n^{\psi}, \tilde{\mathbf{x}}_n)$
    $\psi \leftarrow \psi - \eta\nabla_{\psi}\mathcal{L}(\psi)$
until convergence
```

Algorithm 4 SFD (our method)

```
Input: model parameters $\psi = \theta$, learning rate $\eta$, $\mathrm{Solver}(\cdot,\cdot,\cdot,\cdot)$, distance metric $d(\cdot,\cdot)$, number of student sampling steps $N$, noise schedule $\{t_n\}_{n=0}^{N}$, number of teacher sampling steps between every two noise levels $K$.
repeat
    Sample $\mathbf{x}_N = \tilde{\mathbf{x}}_N \sim \mathcal{N}(\mathbf{0}; t_N^2\mathbf{I})$
    for $n = N-1$ to $0$ do
        $\mathbf{x}_n^{\psi} \leftarrow \mathrm{Euler}(\mathbf{x}_{n+1}, t_{n+1}, t_n, 1; \psi)$
        $\tilde{\mathbf{x}}_n \leftarrow \mathrm{Solver}(\tilde{\mathbf{x}}_{n+1}, t_{n+1}, t_n, K; \theta)$
        $\psi \leftarrow \psi - \eta\nabla_{\psi} d(\mathbf{x}_n^{\psi}, \tilde{\mathbf{x}}_n)$
        $\mathbf{x}_n \leftarrow \mathrm{detach}(\mathbf{x}_n^{\psi})$
    end for
until convergence
```

Algorithm 5 SFD with AFS

```
Input: model parameters $\psi = \theta$, learning rate $\eta$, $\mathrm{Solver}(\cdot,\cdot,\cdot,\cdot)$, distance metric $d(\cdot,\cdot)$, number of student sampling steps $N$, noise schedule $\{t_n\}_{n=0}^{N}$, number of teacher sampling steps between every two noise levels $K$.
repeat
    Sample $\mathbf{x}_N = \tilde{\mathbf{x}}_N \sim \mathcal{N}(\mathbf{0}; t_N^2\mathbf{I})$
    $\hat{\boldsymbol{\epsilon}} \leftarrow \mathbf{x}_N / \sqrt{1 + t_N^2}$
    $\mathbf{x}_{N-1} \leftarrow \mathbf{x}_N + (t_{N-1} - t_N)\hat{\boldsymbol{\epsilon}}$
    $\tilde{\mathbf{x}}_{N-1} \leftarrow \mathrm{Solver}(\tilde{\mathbf{x}}_N, t_N, t_{N-1}, K; \theta)$
    for $n = N-2$ to $0$ do
        $\mathbf{x}_n^{\psi} \leftarrow \mathrm{Euler}(\mathbf{x}_{n+1}, t_{n+1}, t_n, 1; \psi)$
        $\tilde{\mathbf{x}}_n \leftarrow \mathrm{Solver}(\tilde{\mathbf{x}}_{n+1}, t_{n+1}, t_n, K; \theta)$
        $\psi \leftarrow \psi - \eta\nabla_{\psi} d(\mathbf{x}_n^{\psi}, \tilde{\mathbf{x}}_n)$
        $\mathbf{x}_n \leftarrow \mathrm{detach}(\mathbf{x}_n^{\psi})$
    end for
until convergence
```

Algorithm 6 SFD-v (variable-NFE)

```
Input: model parameters $\psi = \theta$, learning rate $\eta$, $\mathrm{Solver}(\cdot,\cdot,\cdot,\cdot)$, distance metric $d(\cdot,\cdot)$, number of student sampling steps $N$, noise schedule $\{t_n\}_{n=0}^{N}$, number of teacher sampling steps between every two noise levels $K$, list $L$.
Initialize: inject step-condition as a new input to the model as $\boldsymbol{\epsilon}_{\psi}(\mathbf{x}, t, c, \mathrm{step} = N)$
repeat
    Sample $N \sim \mathcal{U}(L)$ and generate $\{t_n\}_{n=0}^{N}$
    Sample $\mathbf{x}_N = \tilde{\mathbf{x}}_N \sim \mathcal{N}(\mathbf{0}; t_N^2\mathbf{I})$
    for $n = N-1$ to $0$ do
        $\mathbf{x}_n^{\psi} \leftarrow \mathrm{Euler}(\mathbf{x}_{n+1}, t_{n+1}, t_n, 1; \psi)$
        $\tilde{\mathbf{x}}_n \leftarrow \mathrm{Solver}(\tilde{\mathbf{x}}_{n+1}, t_{n+1}, t_n, K; \theta)$
        $\psi \leftarrow \psi - \eta\nabla_{\psi} d(\mathbf{x}_n^{\psi}, \tilde{\mathbf{x}}_n)$
        $\mathbf{x}_n \leftarrow \mathrm{detach}(\mathbf{x}_n^{\psi})$
    end for
until convergence
```

Algorithm 7 Second-stage one-NFE distillation

```
Input: $\psi = \theta$, $\eta$, $d(\cdot,\cdot)$, $\{t_0, t_1\}$, $K$.
# AFS can be used similar to Algorithm 5
# Step-condition can be used as Algorithm 6
repeat
    Sample $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}; t_1^2\mathbf{I})$
    $\mathbf{x}_0^{\psi} \leftarrow \mathrm{Euler}(\mathbf{x}_1, t_1, t_0, 1; \psi)$
    $\mathbf{x}_0 \leftarrow \mathrm{Euler}(\mathbf{x}_1, t_1, t_0, K; \theta)$
    $\psi \leftarrow \psi - \eta\nabla_{\psi} d(\mathbf{x}_0^{\psi}, \mathbf{x}_0)$
until convergence
```

Algorithm 8 SFD/SFD-v sampling

```
Input: model parameters $\psi$, number of student sampling steps $N$, noise schedule $\{t_n\}_{n=0}^{N}$.
Initialize: Sample $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}; t_N^2\mathbf{I})$
# AFS can be used similar to Algorithm 5
# Step-condition can be used as Algorithm 6
for $n = N-1$ to $0$ do
    $\mathbf{x}_n \leftarrow \mathrm{Euler}(\mathbf{x}_{n+1}, t_{n+1}, t_n, 1; \psi)$
end for
return $\mathbf{x}_0$
```

D Additional Results

D.1 Inject Step-condition into the Model

In this section, we illustrate the detail of how to inject the step-condition into diffusion models. We take the EDM [16] models trained on CIFAR10 dataset as an example. The modifications on models trained on other datasets are similar.

Generally, we treat step-condition the same as the timestamps input to diffusion models. The step-condition go through similar operations like timestamps to obtain the step-embedding which is then added to the time-embedding in every UNetBlock. For CIFAR10 model, we use the DDPM++ backbone proposed in [51] where the positional embedding is applied to encode the input time. The detailed modifications are shown in Algorithm 9-12. Note that for class-conditional models, we do not add the class-embedding to the step-embedding which we find to be suboptimal.

Algorithm 9 Original network for CIFAR10

```
Class SongUNet(torch.nn.Module):
  def __init__():
      ...
      self.map_noise = PositionalEmbed()
      self.map_layer0 = Linear()
      self.map_layer1 = Linear()
      ...

  def forward(x, noise):
      emb = self.map_noise(noise)
      if class_conditional:
         # add class embedding to emb
      emb = silu(self.map_layer0(emb))
      emb = silu(self.map_layer1(emb))
      ...
      for every UNetBlock:
         x = UNetBlock(x, emb)
      ...
```

Algorithm 10 Network with step-condition

```
Class SongUNet(torch.nn.Module):
  def __init__():
      ...
      self.map_noise = PositionalEmbed()
      self.map_layer0 = Linear()
      self.map_layer1 = Linear()
      self.map_step = PositionalEmbed()
      self.step_layer0 = Linear()
      self.step_layer1 = Linear()
      ...

  def forward(x, noise, step):
      emb = self.map_noise(noise)
      if class_conditional:
         # add class embedding to emb
      emb = silu(self.map_layer0(emb))
      emb = silu(self.map_layer1(emb))
      emb_s = self.map_step(step)
      emb_s = silu(self.step_layer0(emb_s))
      emb_s = silu(self.step_layer1(emb_s))
      ...
      for every UNetBlock:
         x = UNetBlock(x, emb, emb_s)
      ...
```

Algorithm 11 Original UNetBlock for CIFAR10

```
Class UNetBlock(torch.nn.Module):
  def __init__():
      ...
      self.affine = Linear()
      ...

  def forward(x, emb):
      ...
      params = self.affine(emb)
      x = x + params
      x = silu(self.norm(x))
      ...
```

Algorithm 12 UNetBlock with step-condition

```
Class UNetBlock(torch.nn.Module):
  def __init__():
      ...
      self.affine = Linear()
      self.affine_s = Linear()
      ...

  def forward(x, emb, emb_s):
      ...
      params = self.affine(emb)
      params_s = self.affine(emb_s)
      x = x + params + params_s
      x = silu(self.norm(x))
      ...
```

Table 6: Experiment settings used in the main text. †: Generated teacher sampling trajectories. When training SFD-v, it refers to the average generated trajectories used for each NFE. *: We force a batch size of 128 by accumulating the gradient for 8 rounds.

Hyperparameter	CIFAR10	ImageNet	LSUN-Bedroom	Stable Diffusion
Teacher solver	DPM++(3M)	DPM++(3M)	DPM++(3M)	DPM++(2M)
K	4	4	4	3
$t_{\min}$	0.006	0.006	0.006	0.1
AFS	True	True	True	False
Generated traj.†	200K	200K	200K	100K
Learning rate	5e-5	1e-5	1e-5	5e-5
Optimizer	Adam	Adam	Adam	Adam
Loss metric	L1	L1	L1	L1
Batch size	128	128	128	16*
Mixed-Precision	True	True	True	True
Number of GPUs	4	4	4	4

Table 7: FID results on CIFAR-10. The lines with gray background are the results reported in the main text. †: Second-stage one-NFE distillation. For each row of the results of SFD-v, the number of reported results corresponds to the the length of the list of sampling steps L in Algorithm 6.

Method	NFE							
	1	2	3	4	5	6	7	8
SFD (100K)	21.14	4.57	3.66	3.26	3.06	2.97	2.87	2.85
SFD (200K)	5.83†	4.53	3.58	3.24	3.06	-	-	-
SFD-v (800K)	-	4.28	3.50	3.18	2.95	-	-	-
SFD-v (800K)	18.69	4.34	3.58	3.22	2.97	2.94	2.88	2.87
SFD-v (2000K)	11.35	4.16	3.44	3.11	2.95	-	-	-
SFD-v (2000K, conditional)	9.17	3.45	2.85	2.76	2.63	-	-	-

Table 8: Ablation study on the number of teacher sampling steps K on CIFAR10 dataset. We report pairs of FID and fine-tuning time (A100 hours).

NFE	K					
	1	2	3	4	5	6
2	64.11/0.17	18.61/0.22	6.51/0.28	4.57/0.32	4.99/0.38	5.66/0.42
3	40.15/0.26	7.76/0.32	3.60/0.39	3.66/0.46	3.84/0.53	3.97/0.59
4	25.96/0.33	5.21/0.42	3.34/0.49	3.26/0.59	3.32/0.67	3.30/0.75
5	16.39/0.41	3.78/0.51	3.10/0.61	3.06/0.71	3.09/0.82	3.17/0.92

D.2 Additional Quantitative and Qualitative results

In Table 6, we include all the hyperparameters used in our main experiments (Section 4). In Table 7, we report more quantitative results on CIFAR10 dataset [21]. By using "100K", we mean a total of 100K sampling trajectories are generated by the teacher model, which equals around 781 training iterations with a batch size of 128. The experiment settings here are basically in accordance with Table 6. For each row of the results of SFD-v, the number of reported results corresponds to the the length of the list of sampling steps L in Algorithm 6. For example, in the third row, the SFD-v is trained to sample with 2, 3, 4 and 5 NFE (each NFE is trained for 200K sampling trajectories on average), while in the fourth row it is trained to sample with 1 to 8 NFE (100 K for each NFE on average). In the last row, we also include results of conditional image generation on CIFAR10 with pre-trained model provided by EDM [16].

Following the final setting in Section 3.2, we provide an ablation study on the intermediate teacher sampling steps K in Table 8. It is shown that $K = 4$ achieves a good trade-off between FID and fine-tuning time.

During our experiments, we adhere to the time schedules utilized in previous studies (for instance, a polynomial schedule with $\rho = 7$ for pixel-space pre-trained models from EDM [16] and a linear schedule for latend-space models from LDM [41]), and find them effective. In Figure 9, we show an ablation study on CIFAR10 dataset with 2-NFE SFD trained with different polynomial coefficients.

Table 9: Ablation study on time schedule on CIFAR10 dataset.

ρ	Time schedule	FID
5	[80.00, 15.11, 1.22, 0.006]	5.50
6	[80.00, 12.63, 0.86, 0.006]	4.61
7	[80.00, 10.93, 0.67, 0.006]	4.53
8	[80.00, 9.72, 0.55, 0.006]	4.54
9	[80.00, 8.82, 0.47, 0.006]	4.60
10	[80.00, 8.13, 0.42, 0.006]	4.81

For further evaluation on fidelity and diversity, we compute precision, recall, density and coverage following standard practice [35] on CIFAR10 dataset. We use the same random seed for a fair comparison. The results are shown in Table 10. In general, while achieving considerable acceleration on image generation, our method does not sacrifice diversity.

Table 10: Evaluation on fidelity and diversity on CIFAR10 dataset.

Method	NFE	FID	Precision	Recall	Density	Coverage
SFD-v (ours)	2	4.28	0.77	0.70	1.06	0.93
	3	3.50	0.78	0.71	1.10	0.94
	4	3.18	0.79	0.71	1.13	0.94
	5	2.95	0.79	0.71	1.15	0.95
DPM++(3M) [30]	11	3.93	0.76	0.71	1.04	0.94
	15	2.64	0.76	0.73	1.03	0.95
	19	2.54	0.77	0.72	1.04	0.96
	23	2.65	0.77	0.72	1.05	0.96
	50	2.01	0.78	0.72	1.11	0.96
DDIM [48]	50	2.91	0.79	0.71	1.09	0.95
Heun [16]	50	1.96	0.79	0.72	1.10	0.96

We include more qualitative results from Figure 11 to Figure 18.

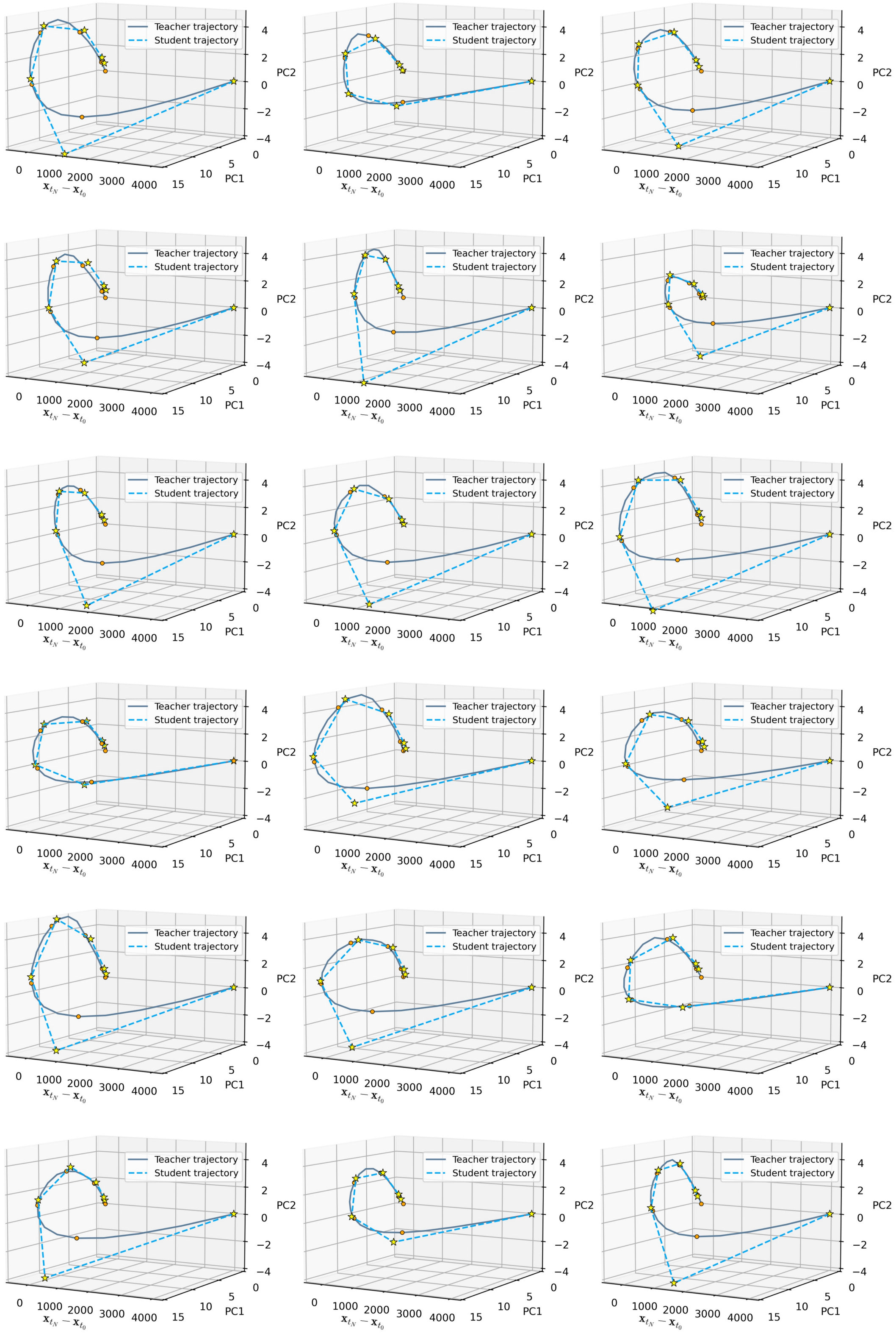


Figure 11: Visualization of the effectiveness of SFD.

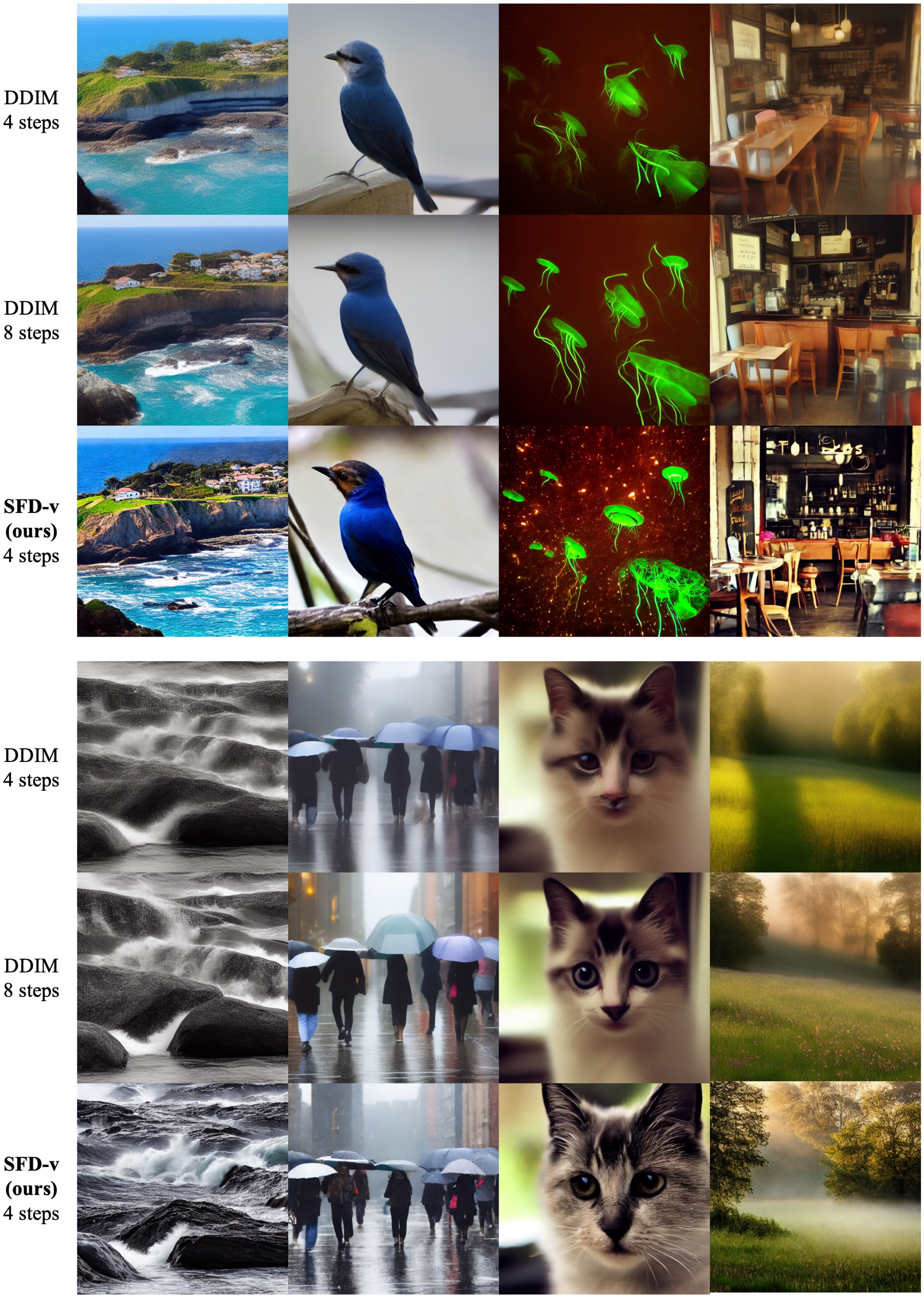


Figure 12: Qualitative results generated by Stable Diffusion v1.5 [41].

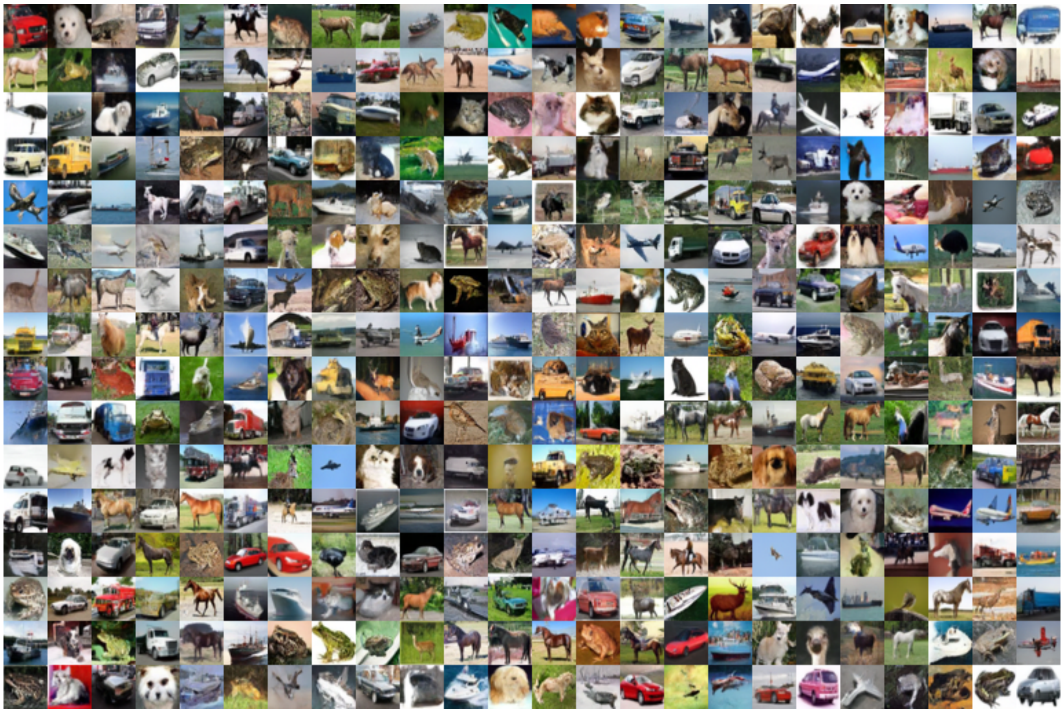


Figure 13: Uncurated qualitative results on CIFAR10. NFE=1.

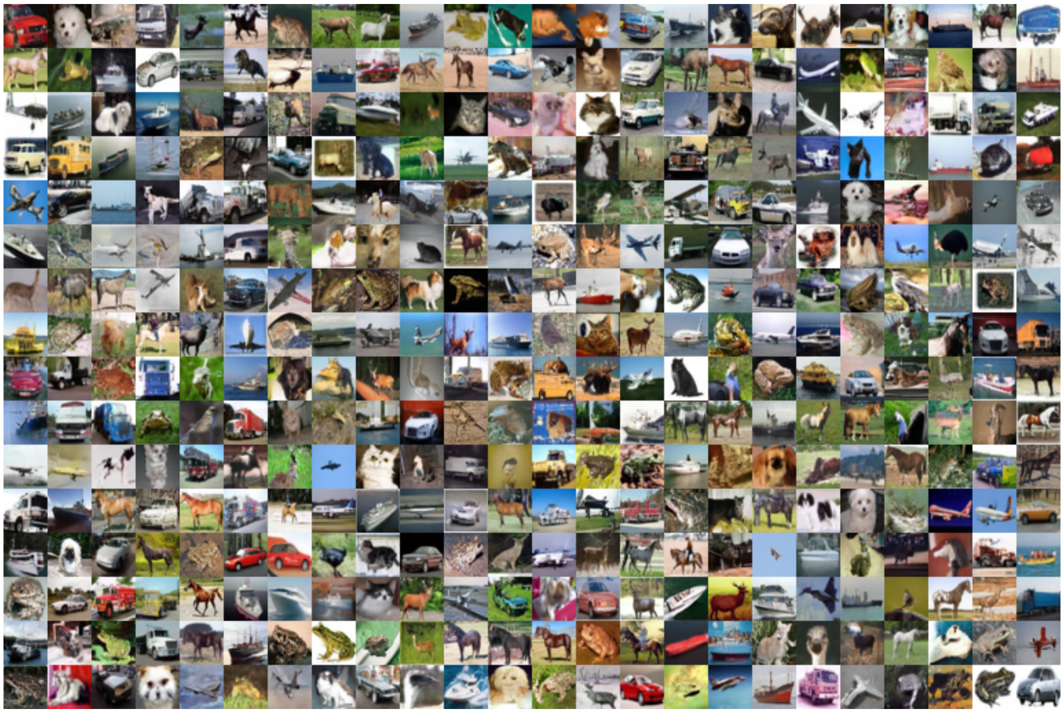


Figure 14: Uncurated qualitative results on CIFAR10. NFE=3.

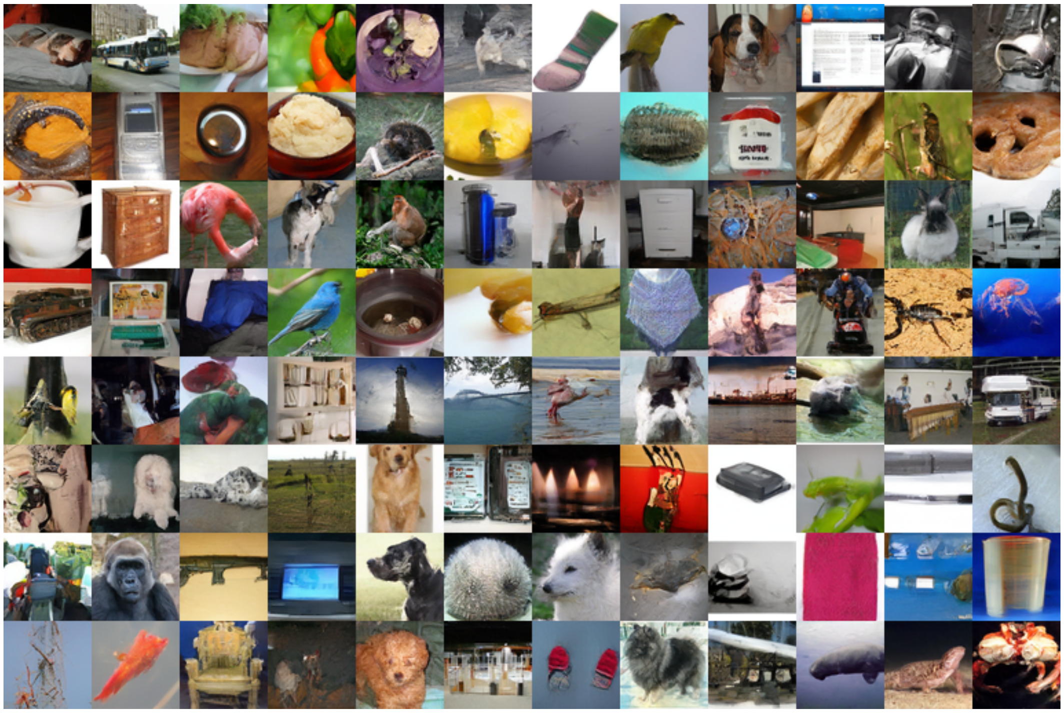


Figure 15: Uncurated qualitative results on ImageNet. NFE=1.

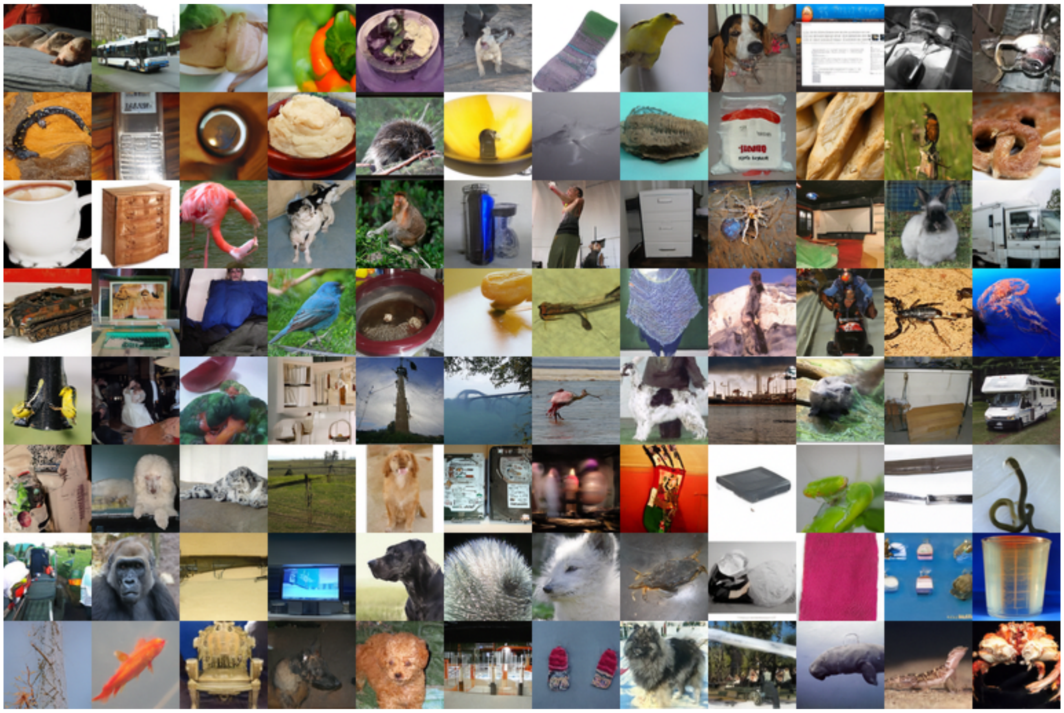


Figure 16: Uncurated qualitative results on ImageNet. NFE=3.

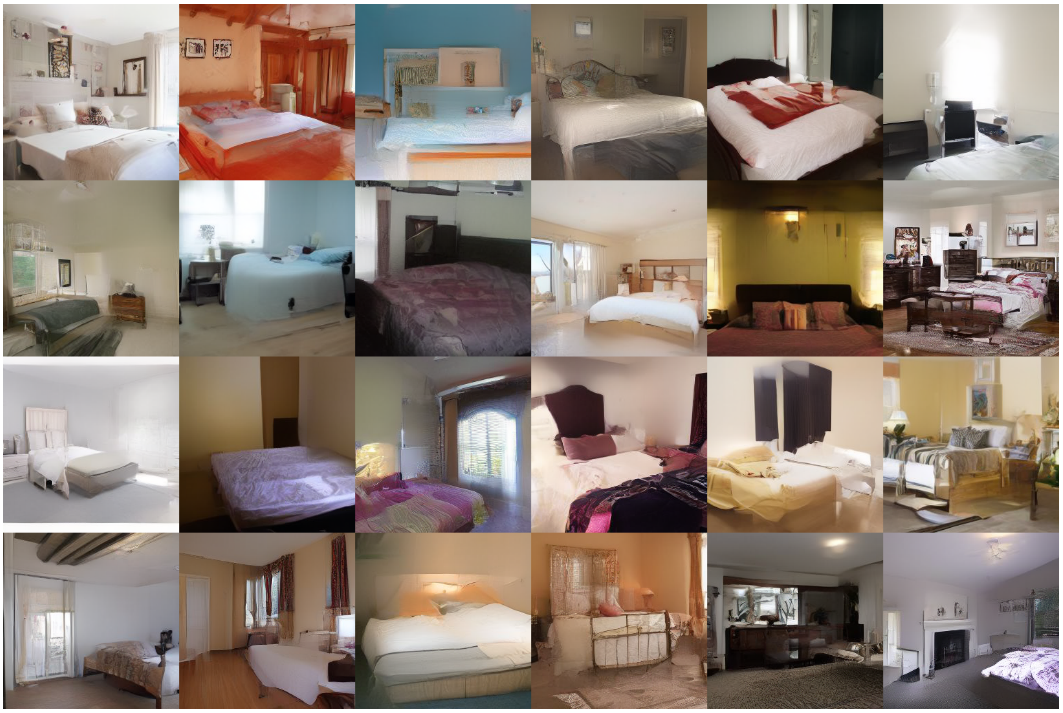


Figure 17: Uncurated qualitative results on LSUN-Bedroom. NFE=1.

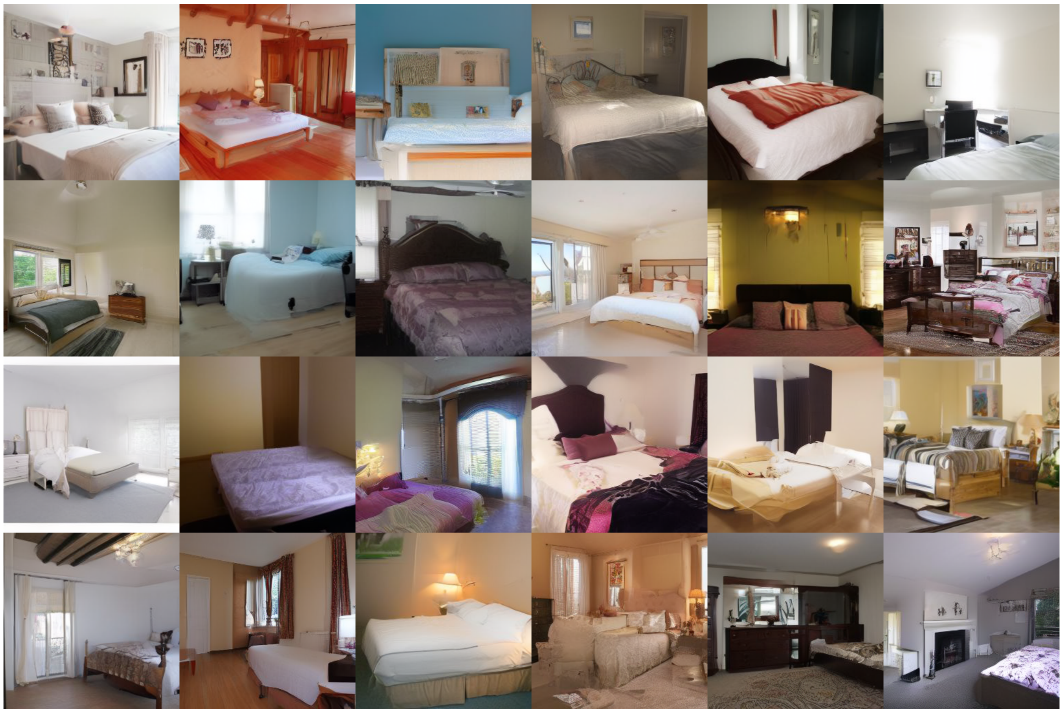


Figure 18: Uncurated qualitative results on LSUN-Bedroom. NFE=3.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The paper's contributions are aligned with the abstract and introduction.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: See Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We include all the information needed in Section 4.1 and Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code with sufficient instructions is attached in the supplemental materials and will be open sourced in the near future.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The details of training and test are fully disclosed in Section 4.1 and Table 6.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [No]

   Justification: We do not report error bars following standard conventions in the related literature. The standard deviation of the FID is small since it is averaged over 50K samples and all the evaluations in our paper share the same random seed, following [16].

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: As mentioned in the main text, we conduct all the experiments using up to 4 NVIDIA A100 GPUs. The required training time is fully specified in Table 2 and 3.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: The research conducted in the paper complies with the NeurIPS Code of Ethics in all aspects.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Section 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: We didn't describe safeguards in our paper.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cite all the used datasets. Our code is based on the repository open sourced by [63] which is licensed according to the Apache License 2.0.

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [Yes]

    Justification: We include the code in the supplementary materials.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.