# Simple and Fast Distillation of Diffusion models

A Preprint

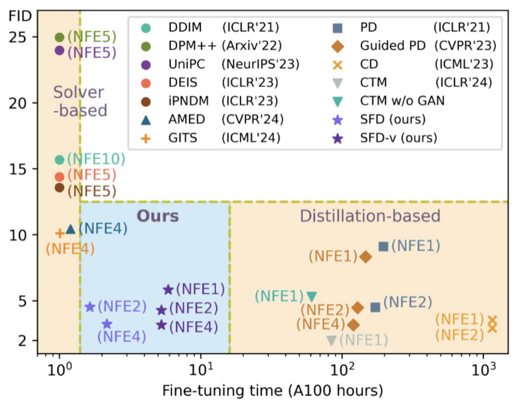


**Figure 1:** ***Comparison of acceleration methods on diffusion models. For better visualization, the time axis is shifted by adding one hour to the actual time required. Our method achieves good performance with a small fine-tuning cost. Note that it takes about* 200 hours *to train a diffusion model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

Recent years have witnessed significant progress in accelerating the sampling of diffusion models: methods mostly can be split in two categories: (I) Solver-based methods using differential equations; (II) distillation based models which has the same structure as teacher, but also expensive and often require 100 hours of training. Optimization of such distillation is also quite complex as is uses various regularization terms and adversarial training.

Overall diffusion process defined as: $dx = f(x;t)dt + g(t)dw$. Backward diffusion process is defined as

$$dx = [f(x;t) - g^2(t)\nabla_x \log p_t(x)]dt + g(t)dw$$

The reverse-time SDE can be further simplified to the probability flow ordinary differential equation:

$$dx = [f(x;t) - \frac{1}{2}g^2(t)\nabla_x \log p_t(x)]dt$$

. In this paper authors consider $f(x;t) = 0; \quad g(t) = \sqrt{2t}$ so reverse process described as $dx = -t\nabla_x \log p_x(x)dt$. The score is estimated: $\nabla_x \log p_t(x) \approx -\epsilon_\theta(x,t)/t$

A noise-prediction model $\epsilon_\theta(x,t)$ is trained by minimizing a regression loss weighted by a time-dependent function $\lambda(t)$:

$$\mathcal{L}_t(\theta) = \lambda(t)\,\mathbb{E}_{x\sim p_d,\,\epsilon\sim\mathcal{N}(0,I)}\left\|\epsilon_\theta(x+t\epsilon,t)-\epsilon\right\|_2^2. \tag{1}$$

This noise-prediction model can be used in place of the score function to define a probability flow ODE (PF-ODE):

$$\frac{dx}{dt} = \epsilon_\theta(x, t). \tag{2}$$

The PF-ODE is often preferred in practice due to its conceptual simplicity and efficient sampling compared to more general reverse-time stochastic differential equations.

To generate samples from a diffusion model using $N$ steps, one typically starts by drawing $x_N \sim p_n = \mathcal{N}(0, t_{\max}^2 I)$ and then numerically solves the PF-ODE using a solver-based method. This process follows a hand-crafted time schedule $\Gamma(N) = \{t_0 = t_{\min}, t_1, \ldots, t_N = t_{\max}\}$. The resulting sequence $\{x_n\}_{n=0}^N$ is referred to as the sampling trajectory.

The teacher and student update rules are defined as:

$$\tilde{x}_n = \text{Solver}(\tilde{x}_{n+1}, t_{n+1}, t_n, K; \theta), \tag{3}$$

$$x_n^\psi = \text{Euler}(x_{n+1}, t_{n+1}, t_n, 1; \psi) = x_{n+1} + (t_{n+1} - t_n)\, \epsilon_\psi(x_{n+1}, t_{n+1}), \tag{4}$$

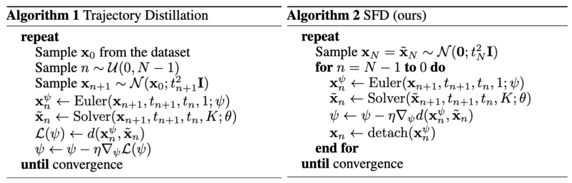
where $K$ is the number of teacher steps from $t_{n+1}$ to $t_n$, and $\theta$, $\psi$ are the parameters of the teacher and student models.

At each iteration, the student is trained using the loss:

$$\mathcal{L}(\psi) = d(x_n^\psi, \tilde{x}_n), \tag{5}$$

with a chosen distance metric $d(\cdot, \cdot)$.

The solver can be any fixed-parameter method used to generate reference samples(DDPM, DDIM etc.). For example, the Euler sampler with $K = 2$ or the Heun sampler with $K = 1$ and consistency loss.



**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
```

However, existing distillation-based methods incur significant fine-tuning costs that may not effectively contribute to the final sample quality. Authors propose to fine-tune only a few timestamps that will be used in sampling, so 4 models are defined. Firstly, authors 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 error.

Stable Diffusion, a latent diffusion model with classifier-free guidance, demonstrates strong performance in high-resolution image synthesis. The guidance mechanism introduces a scale parameter $\omega$ to blend conditional and unconditional predictions:

$$\tilde{\epsilon}_\theta(x, t, c) = \omega\, \epsilon_\theta(x, t, c) + (1 - \omega)\, \epsilon_\theta(x, t, \emptyset). \tag{6}$$

Despite its effectiveness, the model is computationally intensive due to the large number of parameters and the need for both conditional and unconditional evaluations at each sampling step.

Distillation of such models is challenging because of their high complexity and the flexibility introduced by $\omega$. Some approaches incorporate $\omega$ into the model, while others omit it entirely.

A trajectory-based strategy can simplify distillation. Sampling paths in the latent space, visualized via DPM-Solver++, reveal regular patterns that become increasingly complex with larger $\omega$. Based on this, distillation is performed at $\omega = 1$, while sampling can be conducted at arbitrary guidance scales. This reduces training cost by avoiding the unconditional branch during distillation.

Long story short, Teacher model(for example Stable Diffusion) and student model which learns the trajectory of teacher (at different timesteps save image) and also to student model another conditioning is fed: amount of overall steps to produce output(2;3;4;5 …). It is done to generalize, due to the fact that without such conditioning model will only be able to make fixed amount of steps.

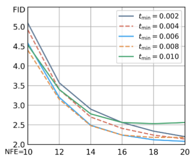


Figure 5: Ablation study on $t_{\min}$ with DPM++(3M).

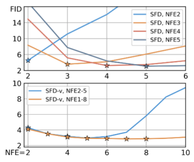


Figure 6: Extrapolation ability on untrained NFE.

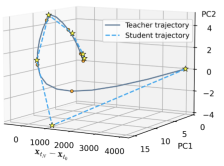


Figure 7: Visualization of the effectiveness of SFD.