Diffusion Model is Effectively Its Own Teacher

Xinyin Ma **Runpeng Yu** **Songhua Liu** **Gongfan Fang** **Xinchao Wang***
National University of Singapore
maxinyin@u.nus.edu, xinchao@nus.edu.sg

(a) DiT-XL/2 For ImageNet 512×512 (b) Stable Diffusion v1.5

Figure 1. Generated Images before (left) and after (right) our proposed self-distillation framework on DiT-XL/2 for ImageNet 512×512 and Stable Diffusion v1.5 with 10 DDIM sampling steps.

Abstract

In this paper, we introduce a novel self-distillation paradigm for improving the performance of diffusion models. Previous studies have shown that introducing a teacher to distill the diffusion model can enhance its sampling efficiency. We raise an intriguing question: can the diffusion model itself serve as its teacher to further improve the performance of itself? To this end, we propose a new paradigm called Self Step-Distillation (SSD). The core idea of SSD is to integrate the predictions or the intermediate activations of the diffusion model at each timestep with its preceding timestep through a fusion mechanism. We propose two forms, explicit SSD and implicit SSD (iSSD), to perform N-step to N-step distillation from the diffusion model itself to achieve improved image quality. We further elucidate the relationship between SSD and high-order solver, highlighting their underlying relationship. The effectiveness of SSD is validated through extensive experiments on diffusion transformers of various sizes and across different sampling steps. Our results show that this novel self-distillation paradigm can significantly enhance performance. Additionally, our method is compatible with the distillation method designed for few-step inference. Notably, with iSSD trained less than one epoch, we obtain a 32-step DiT-XL/2 achieving an FID of 1.99, outperforming the original 250-step DiT-XL/2 with an FID of 2.26. We further validate the effectiveness of our method on text-to-image diffusion models, such as Stable Diffusion, and also observe notable improvement in image quality.

1. Introduction

Diffusion-based generative models [5, 16, 51] have gained significant attention for their remarkable capability to generate diverse, high-fidelity outputs across various domains. It has been proven effective in tasks such as image synthesis [42, 44, 46], video generation [1, 17, 18] and natural language processing [27, 33]. Unlike previous generative approaches like VAEs [24] and GANs [12], Diffusion Mod-

*Corresponding author

els (DMs) rely on a unique process where noise is gradually refined into meaningful data by reversing a specified noise pathway [20]. This iterative noise-to-data transformation gives DMs exceptional versatility, allowing them to be applied successfully in image editing [21], inpainting [35], super-resolution [11], object detection [3], 3D generation [43], etc.

Recent studies have emphasized the effectiveness of knowledge distillation [15] as a tool for enhancing inference efficiency or accelerating pre-training for diffusion models [38, 50]. These methods typically achieve distillation through the use of an external teacher model or a teacher with more sampling steps. To improve inference efficiency, approaches such as progressive distillation and consistency distillation [23, 36, 47, 52] employ an extensive multi-step sampling process as a teacher model to distill the student model with fewer steps. These teacher models more accurately approximate the denoising trajectory, thereby guiding the student model to enable the few-step generation or even one-step generation [7]. Other studies [13, 19, 22] focus on distilling compact diffusion models by training smaller models under the supervision of larger, high-capacity models. Besides, [60] demonstrates that aligning model representations using guidance from an external pre-trained vision encoder can facilitate faster convergence.

The aforementioned methods can significantly enhance the sampling efficiency of diffusion models through distillation. However, the student models are constrained by the performance of the teacher model, with their training aimed at matching, but not surpassing, the teacher's performance. *This raises an intriguing question: Can distillation be leveraged to improve the generation quality of the diffusion model itself, potentially exceeding the performance of the teacher model?* This shifts the focus from traditional N-to-M step distillation or model compression for speed to a form of N-to-N step self-distillation aimed at enhancing the generation quality of the original diffusion model.

We thereby explore the unique form of self-distillation within diffusion models. While self-distillation has been studied extensively in conventional architectures [10, 39, 61, 62], its application to diffusion models remains unexplored. explore the special way that self-distillation operates in diffusion models by utilizing the properties of diffusion models. Given the temporal relationships between steps that form a sequence of predictions, we introduce Self Step-Distillation in both explicit and implicit forms. The core idea is that, with careful design, the model can enhance its performance by mimicking the next-step prediction. To achieve this, we propose a fusion of predictions and features across different steps, and redefine the computational flow for the student model. Once the student model is trained, it can be reverted to the original computational pipeline. We also illustrate how this fusion mechanism relates to high-order solvers.

We mainly conduct our experiments on diffusion transformers [40] due to their exceptional performance. We conduct experiments on DiT models ranging from DiT-S/2 to DiT-XL/2 and sampling steps from 4 to 250. With our method, a self-distilled DiT-XL/2 with 50 sampling steps surpasses the performance of the pre-trained DiT-XL/2 with 250 sampling steps (2.02 vs 2.14 in FID). To further examine the N-to-N distillation setting and show that our method is complementary to the few-step distillation, we applied our method on a progressively distilled DiT-XL/2. This resulted in an enhanced generation quality, with an FID of 1.99 achieved using 32 sampling steps. Additionally, we tested the generalizability of our method by applying it to Stable Diffusion, confirming its effectiveness in the text-to-image setting. All experiments require fewer than 10k optimization steps and a short training duration—approximately 40 minutes for DiT-XL/2 and 3 hours for Stable Diffusion. Consequently, our method can be considered a quick self-improvement technique for diffusion models, which we further show that it can operate without dependence on training data.

To summarize, the contributions of our work are:

- We demonstrate for the first time that self-distillation can enhance the performance of diffusion models itself, significantly surpassing the original model.
- We introduce a novel self-distillation framework for diffusion models, termed **S**elf **S**tep-**D**istillation. By corrupting and fusing predictions and intermediate activations, we redefine the computational flow of the teacher and student diffusion model to facilitate self-distillation.
- We analyze the underlying mechanisms of this unique self-distillation approach and reveal its connection to high-order solvers.
- Experimental results demonstrate that our self-distillation method significantly enhances the performance of diffusion models, including both DiT and Stable Diffusion, as well as distilled few-step diffusion models.

2. Related Work

Distillation in Diffusion Models. knowledge distillation [15] becomes a popular paradigm to improve the efficiency of diffusion models. Progressive Distillation [47] proposes to half the steps of sampling by matching the samples generated by N deterministic steps with 2N steps. Consistency distillation [52] enforces the self-consistency between any pair of steps and makes the boundary condition avoid trivial solutions. [36] extends it into the latent space, and [23, 54] divides the whole trajectory consistency into subphase consistency. SFD [64] solves the accumulative errors in local distillation by sampling and imitating the whole trajectory. [59] distills the diffusion model into a one-step generator by matching at the distribution level with adversarial

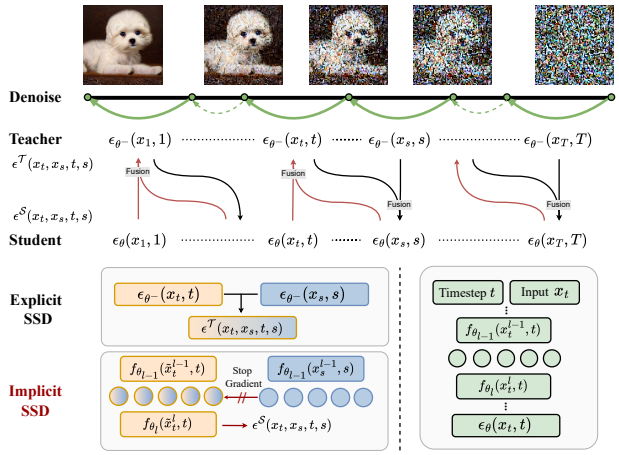


Figure 2. The overall framework of Self Step-Distillation. Here θ^- means that the weight of the model is frozen.

loss. [6] shows that the high-order term, jacobian-vector products, can be distilled into a separate neural network. Guided diffusion [38] matches the combined output of the classifier-free guidance with a single model, and reduces the sampling steps for unconditional generation and conditional generation. The adversarial objective can also be incorporated in diffusion training to stabilize the training process [30, 48]. Apart from this, another line of work uses knowledge distillation to make the diffusion model small and efficient [8, 58]. Feature-level knowledge distillation is widely adopted in those works to accelerate the post-training of these compressed diffusion models [8, 13, 22].

Efficiency in Diffusion Models. Balancing the image quality and the sampling speed is a challenging topic in diffusion models [20, 55]. (1) *Fast solver for sampling.* One common way to reduce the sampling steps is to employ a training-free solver to off-the-shelf pre-trained diffusion models. Those methods approximate the true trajectory of probability flow ODE, such as Euler's method [50], Heun's method [20], the exponential integrator with Taylor expansion [34], the unified predictor-corrector framework [63] and numerical methods [32]. (2) *Structural Efficiency.* Besides of optimizing the number of model evaluations, another way is to encourage structural efficiency. Redesign the model architectures [58], pruning [8, 29, 65] and quantization [14, 28] constructs lightweight diffusion model. Cache-based methods [37, 56] leverage the similarity between steps to reduce calculations. (3) *Pipeline Efficiency*. Breaking the sequential nature of denoising is another way to improve the efficiency of diffusion models. [49] uses Picard iteration to iteratively refine until convergence. [4, 25] decouples the images or models into multiple components and assigns them to different devices.

Self-Distillation Knowledge distillation is widely used for compressing compact networks, while self-distillation focuses on improving the performance of networks. [10] investigate distillation with a student network parameterized identically to its teacher, demonstrating that the student could significantly outperform the teacher. [61] explores distilling knowledge from deeper network layers into shallower ones. Several works [2, 39, 57] examine the underlying reasons for this phenomenon, commonly attributing it to the "dark knowledge" encoded in soft labels. [62] highlight the importance of predictive diversity, while [41] systematically investigates self-distillation carefully and interprets self-distillation as a process of learning previously unlearned perspectives of input data. Unlike these prior works, our focus is not on leveraging soft labels from teacher predictions to guide student network training. Instead, we explore a unique mechanism in diffusion models to enhance their performance.

3. Method

In this section, we elaborate on the design of our self-distillation framework. We first show our proposed framework of the self-distillation paradigm in Section 3.2, and then give the explanation about why it works in Section 3.3 and Section 3.4.

3.1. Preliminary

We begin from the forward and reverse processes in diffusion probabilistic models. Let $q_{\text{data}}(x)$ represent the data distribution. The forward process entails adding noise incrementally to a clean image $x_0 \sim q_{\text{data}}(x)$. At a given timestep t, the distribution of x_t, which includes the noise added up to that point, is defined by:

$$q(x_t \mid x_0) = \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}\right) \tag{1}$$

where α_t is a time-dependent function that controls the noise scheduling. As t approaches a sufficiently large value, the final state x_T becomes nearly the same as a Gaussian noise. For denoising, starting from a noise input $x_T \sim \mathcal{N}(0, \mathbf{I})$, an image can be progressively reconstructed through iterative sampling from the conditional distribution $q(x_{t-1}|x_t)$. This conditional probability is typically approximated by a parameterized distribution, given as $p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$. Here, $\mu_\theta(x_t, t)$ is parameterized by θ, and it follows:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(x_t, t)\right) \tag{2}$$

where $\boldsymbol{\epsilon}_\theta(x_t, t)$ represents the noise estimation function at timestep t given input x_t. The training objective for diffusion models is to minimize the difference between the predicted noise and the ground-truth noise $\boldsymbol{\epsilon}_t$ added during

the forward process. This is typically framed as a simplified version of the variational bound on the negative log-likelihood of x_0. The objective can be written as:

$$\mathcal{L}_t = \mathbb{E}_{t\sim[1,T],\mathbf{x}_0\sim q_{\text{data}}(x)}\left[\left\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right\|^2\right] \quad (3)$$

3.2. Overall Framework

Next, we introduce the design of this unique form of self-distillation within diffusion models. To achieve self-distillation, three essential components must be defined: the construction of the teacher model, the formulation of the student model, and the definition of the training objective. Given that we have only a single pre-trained model without any external supervision. This poses a critical challenge: how can a single pre-trained diffusion model be used to construct both a teacher and a student model, with the teacher model designed to be more capable than the student?

We focus on the timesteps of diffusion models, which inherently give us a sequence of predictions for one sample. Denote the model to have L layers. $x^{(l)}$ denotes the input for layer l, where $l \in \{1, 2, \ldots, L\}$. $x_t^{(1)} = x_t$ represents the input to the network at timestep t. Each layer l applies a function f_{θ_l} parameterized by θ_l, which includes operations such as multi-head self-attention and pointwise feedforward module in Transformer [53], and residual or convolution blocks in U-Net [45]. Thus, for each layer, the layer-wise computation at timestep t can be formulated as:

$$x_t^{(l)} = f_{\theta_{l-1}}\left(x_t^{(l-1)}, t\right) \quad (4)$$

Here the calculation is conditioned on the timestep t, although not all blocks in diffusion model use t as a conditional input. After all L layers have been processed, the final output of the model is the predicted noise, represented as $\boldsymbol{\epsilon}(x_t, t) = x_t^{(L+1)}$.

To construct the teacher and student models, we require the outputs for two timesteps, t and s, where s represents the preceding denoising step of t under a pre-defined solver, satisfying $s > t$. We denote the predictions of the teacher and the student model as $\boldsymbol{\epsilon}^{\mathcal{T}}\left(x_t, x_s, t, s\right)$ and $\boldsymbol{\epsilon}^{\mathcal{S}}\left(x_t, x_s, t, s\right)$, respectively. Our approach to building the teacher and student models is based on a core principle: using the predictions from a later step as stronger supervision to guide the prediction at an earlier step. The rationale behind this principle will be discussed in detail later. Based on this guideline, we develop the below two distinct forms for self-distillation.

Explicit Form: For the explicit self step-distillation, we build a teacher model by fusing the predicted noise:

$$\boldsymbol{\epsilon}^{\mathcal{T}}\left(x_t, x_s, t, s\right) = \lambda\boldsymbol{\epsilon}_{\theta^-}\left(x_t, t\right) + (1-\lambda)\boldsymbol{\epsilon}_{\theta^-}\left(x_s, s\right) \quad (5)$$

The student prediction use the predicted noise at timestep s: $\boldsymbol{\epsilon}^{\mathcal{S}}\left(x_t, x_s, t, s\right) = \boldsymbol{\epsilon}_\theta\left(x_s, s\right)$. Here, θ^- represents the model would be frozen during training.

Implicit Form: For the implicit self step-distillation, we redefine the computation flow in the model. We fuse the outputs at the same intermediate layer but different timesteps t and s, denoted $x_t^{(l)}$ and $x_s^{(l)}$ respectively. The fused output at this layer is now calculated as:

$$\tilde{x}_t^{(l)} = \lambda f_{\theta_{l-1}^-}\left(x_s^{(l-1)}, s\right) + (1-\lambda) f_{\theta_{l-1}}\left(\tilde{x}_t^{(l-1)}, t\right) \quad (6)$$

where $\tilde{x}_t^{(l-1)}$ represents the fused output from the previous layer, and $\tilde{x}_t^{(l)}$ serves as the input for the next layer, replacing the original $x_t^{(l)}$. The above equation serves as a weighted fusion of layer output in two timesteps. The final prediction $\boldsymbol{\epsilon}^{\mathcal{S}}\left(x_t, x_s, t, s\right) = \tilde{x}_t^{(L+1)}$ is obtained from these fused features, forming the prediction for the student model. With this, the computation flow of the student model is changed. In constrast, the teacher model functions as the original denoising predictor at timestep t, with its prediction output expressed as $\boldsymbol{\epsilon}^{\mathcal{T}}\left(x_t, x_s, t, s\right) = \boldsymbol{\epsilon}_{\theta^-}(x_t, t)$

Training Objective. We take the pre-trained model as both the initialization for the teacher and the student model. The teacher θ_T model remained frozen while the update of parameters is performed on the student side. The self-distillation objective here is the matching between the student $\boldsymbol{\epsilon}^{\mathcal{S}}\left(x_t, x_s, t, s\right)$ and the teacher $\boldsymbol{\epsilon}^{\mathcal{T}}\left(x_t, x_s, t, s\right)$. The distillation objective would be:

$$\min_\theta \mathbb{E}_{s\sim[1,T],x_0\sim q_{\text{data}}(x)}\left[\left\|\boldsymbol{\epsilon}^{\mathcal{T}}\left(x_t, x_s, t, s\right) - \boldsymbol{\epsilon}^{\mathcal{S}}\left(x_t, x_s, t, s\right)\right\|^2\right] \quad (7)$$

Here, x_s represents x_0 with added noise, and x_t is obtained by performing one step from x_s in the ODE solver. After the training phase is completed, no fusion operations are applied during the inference stage. The model's computation remains identical to its original configuration but benefits from the enhancements gained through the self-distillation process.

3.3. Distill from High-Order Solver

In this section, we explain why this self-distillation framework works. In order to reduce the discretization error for the first-order ODE solver, such as Euler [20], one straightforward way is to use the higher-order solver for the diffusion ODEs. The cost for this is that it needs twice or third or even more times of function evaluation. We show that our self-distillation framework serves as an approximated form for distilling it into the same network without introducing additional parameters.

We perform a single step forward from timestep s to timestep t. We compare the formulation of DPM-Solver-1 and DPM-Solver-2 [34].

$$\text{DPM-Solver-1:} x_t = \frac{\alpha_t}{\alpha_s} x_s - w_t \boldsymbol{\epsilon}_\theta \left(x_s, s\right) \tag{8}$$

$$\text{DPM-Solver-2:} x_t = \frac{\alpha_t}{\alpha_s} x_s - w_t \boldsymbol{\epsilon}_\theta \left(x_u, u\right) \tag{9}$$

Here the timestep u is an intermediate timestep between t and s. As defined in DPM-Solver-2, timestep u has $\lambda_u := \log\left(\alpha_u / \sigma_u\right)$ which is the average of those for timestep t and timestep s, and x_u denotes the state one step forward from s to u. The term w_t is a weight factor that depends only on the timestep. Using DPM-Solver-2, which is more accurate than DPM-Solver-1, the following training objective can be interpreted as encouraging the predictions of the first-order solver to align more closely with those of the second-order solver:

$$\min_\theta \mathbb{E}_{s\sim[1,T],\mathbf{x}_0\sim q_{\text{data}}(x)} \left[w_t \left\| \boldsymbol{\epsilon}_\theta \left(x_s, s\right) - \boldsymbol{\epsilon}_\theta \left(x_u, u\right) \right\|^2 \right] \tag{10}$$

Building on the above equation, if the goal is to approximate a higher-order solver using a first-order solver, the training objective simplifies to: *matching the prediction at the current step with the prediction at a subsequent step*.

3.4. Ways to approximate $\boldsymbol{\epsilon}_\theta(x_u, u)$

However, this method requires the prediction at the intermediate step between timestep s and t, which is not always feasible in N-step to N-step distillation. A feasible scenario involves using a linear interpolant noise scheduler, where $x_s = (1 - s/T)x_0 + (s/T)\boldsymbol{\epsilon}$, and the model predicts the velocity instead of the noise $\boldsymbol{\epsilon}$. Under these conditions, the objective can be reformulated to align $\boldsymbol{\epsilon}_\theta(x_s, s)$ with $\boldsymbol{\epsilon}_\theta(x_t, t)$. The corresponding results are presented in Table 1, with detailed proofs provided in the Appendix. With 10 and 250 sampling steps, consistent improvements in sample quality are observed, validating the effectiveness of Eq.10. However, as expected, performance declined in both cases when using ϵ-prediction without incorporating these properties into the noise scheduler and the specific model prediction type.

The question now arises: how we can approximate $\boldsymbol{\epsilon}_\theta(x_u, u)$ given $\boldsymbol{\epsilon}_\theta(x_s, s)$ and $\boldsymbol{\epsilon}_\theta(x_t, t)$. One straightforward way is to get the approximation of $\boldsymbol{\epsilon}_\theta(x_u, u)$ by Taylor Expansion, whose results turned to be the interpolating between $\boldsymbol{\epsilon}_\theta(x_s, s)$ and $\boldsymbol{\epsilon}_\theta(x_t, t)$. Following the assumption in [34] that $\boldsymbol{\epsilon}_\theta(x_s, s)$ is Lipschitz w.r.t x_s, we have:

$$\begin{aligned} \boldsymbol{\epsilon}_\theta \left(x_u, u\right) &= \boldsymbol{\epsilon}_\theta \left(x_s, u\right) + \left(\boldsymbol{\epsilon}_\theta \left(x_u, u\right) - \boldsymbol{\epsilon}_\theta \left(x_s, u\right)\right) \\ &= \boldsymbol{\epsilon}_\theta \left(x_s, s\right) + \boldsymbol{\epsilon}_\theta^{(1)} \left(x_s, s\right)(u - s) + \mathcal{O}\left((u-s)^2\right) \end{aligned} \tag{11}$$

Model	Sampler	NFEs	IS	FID	sFID
v-prediction with linear interpolant noise scheduler					
REPA - SiT-XL/2	SDE	250	305.7	1.42	4.70
REPA - SiT-XL/2° w/ Eq.10	SDE	250	306.3	1.41	4.68
REPA - SiT-XL/2°	ODE	10	233.9	5.85	7.82
REPA - SiT-XL/2° w/ Eq.10	ODE	10	238.4	5.11	7.48
ϵ-prediction					
DiT-XL/2	ODE	50	238.6	2.26	4.29
DiT-XL/2 w/ Eq.10	ODE	50	230.8	2.81	6.41
DiT-XL/2 w/ Eq.12	ODE	50	239.9	2.21	4.41
DiT-Xl/2	ODE	10	158.3	12.38	11.22
DiT-XL/2 w/ Eq.10	ODE	10	51.8	45.99	26.95
DiT-XL/2 w/ Eq.12	ODE	10	184.7	7.99	9.91

Table 1. Results on v-prediction and ϵ-prediction diffusion models. For the experiments on REPA, we use the EMA model as the teacher for the experiments with 250 steps.

where $\boldsymbol{\epsilon}_\theta^{(1)}\left(x_s, s\right)$ is the first-order derivative w.r.t s. As above, we derive $\boldsymbol{\epsilon}_\theta^{(1)}\left(x_s, s\right)$ the same way between two different timesteps t and s, and we have:

$$\boldsymbol{\epsilon}_\theta \left(x_u, u\right) = (1 - \lambda)\boldsymbol{\epsilon}_\theta \left(x_s, s\right) + \lambda \boldsymbol{\epsilon}_\theta \left(x_t, t\right) \tag{12}$$

where $\lambda = (u - s) / (t - s)$. We get the same form as the explicit self step-distillation. The results with this equation are also shown in Table 1.

While the aforementioned method can approximate $\boldsymbol{\epsilon}_\theta(x_u, u)$ to some extent, it remains suboptimal. An alternative approach is the implicit form of interpolation, which approximates $\boldsymbol{\epsilon}_\theta(x_u, u)$ by interpolating on features rather than on the output. This one shares the same form with implicit self step-distillation. This idea is inspired by previous works [26, 37], which demonstrated that certain intermediate activations in timestep s can be reused in later timestep u to produce an approximated output for timestep u, with minimal effect on image quality. Consequently, we interpolate on the intermediate layers, and experimental results indicate that this method yields better performance than interpolation on predictions.

4. Experiment

In this section, we demonstrate the effectiveness of our proposed method. In Section 4.2, we present empirical evidence that both SSD and iSSD consistently enhance the performance of a pre-trained DiT across various image resolutions, model sizes, and sampling steps. Additionally, Section 4.3 provides the results of applying these methods to text-to-image generation. Section 4.4 highlights how iSSD can be applied to distilled models, yielding further performance improvements. Finally, Section 4.5 offers an in-depth analysis.

4.1. Experimental Setup

Models. For the conditional generation on ImageNet, we take all four DiT models: DiT-S/2, DiT-B/2, DiT-L/2, and

Method	NFEs	Train Epochs	IS ↑	FID ↓	sFID ↓	Prec. ↑	Rec. ↑	NFEs	Train Epochs	IS ↑	FID ↓	sFID ↓	Prec. ↑	Rec. ↑
	– 256×256 –							– 512×512 –						
DiT-XL/2	$250_{(ODE)}$	1400	243.4	2.14	4.56	0.81	0.61	$250_{(SDE)}$	600	240.8	3.04	5.02	0.84	0.54
SiT-XL/2	$250_{(SDE)}$	1400	277.5	2.06	4.49	0.83	0.59	$250_{(SDE)}$	600	252.2	2.62	4.18	0.84	0.57
DiT-XL/2	50	1400	238.6	2.26	4.29	0.80	0.60	50	600	204.1	3.28	4.50	0.83	0.55
+ SSD	50	1400 + 0.8	239.9	2.21	4.41	0.80	0.60	50	600 + 0.4	205.2	3.18	4.59	0.83	0.57
+ iSSD	50	1400 + 0.8	250.0	2.02	4.22	0.81	0.60	50	600 + 0.4	210.2	2.86	4.06	0.84	0.54
DiT-XL/2	20	1400	223.5	3.48	4.89	0.79	0.57	20	600	186.3	5.10	5.73	0.82	0.55
+ SSD	20	1400 + 0.8	228.6	2.82	4.63	0.79	0.60	20	600 + 0.4	194.7	4.01	5.33	0.83	0.56
+ iSSD	20	1400 + 0.8	227.5	2.67	4.85	0.79	0.60	20	600 + 0.4	200.8	3.69	4.41	0.84	0.54
DiT-XL/2	10	1400	158.3	12.38	11.22	0.67	0.53	10	600	128.4	15.19	11.58	0.74	0.49
+ SSD	10	1400 + 0.8	184.7	7.99	9.91	0.71	0.56	10	600 + 0.4	158.5	9.61	11.09	0.78	0.54
+ iSSD	10	1400 + 0.8	209.6	4.92	7.62	0.79	0.53	10	600 + 0.4	164.9	6.97	7.05	0.83	0.51

Table 2. Image quality on ImageNet 256×256 and 512×512, with DiT-XL/2 as the base model. Prec. and Rec. denotes Precision and Recall. We adopt CFG=1.5 in all the experiments for ImageNet.

Model	NFEs	Train Epochs	IS ↑	FID ↓	sFID ↓	Prec. ↑	Rec.↑
			– DiT-S/2 –				
DiT-S/2	250	400	59.4	26.51	7.31	0.59	0.56
DiT-S/2	50	400	53.5	29.21	5.89	0.57	0.58
+ iSSD	50	400 + 0.2	54.6	27.75	5.73	0.58	0.58
DiT-S/2	20	400	52.0	32.62	7.43	0.55	0.55
+ iSSD	20	400 + 0.2	52.5	30.62	8.23	0.56	0.55
DiT-S/2	10	400	40.1	49.50	18.33	0.43	0.48
+ iSSD	10	400 + 0.2	40.7	46.17	18.89	0.45	0.48
			– DiT-B/2 –				
DiT-B/2	250	200	119.6	10.12	5.39	0.73	0.55
DiT-B/2	50	200	101.8	12.84	4.81	0.71	0.56
+ iSSD	50	200 + 0.2	104.9	11.77	4.99	0.71	0.56
DiT-B/2	20	200	97.5	15.40	6.11	0.68	0.54
+ iSSD	20	200 + 0.2	100.6	13.84	8.63	0.71	0.52
DiT-B/2	10	200	72.1	28.99	15.54	0.56	0.48
+ iSSD	10	200 + 0.2	77.9	21.40	13.37	0.61	0.53
			– DiT-L/2 –				
DiT-L/2	250	200	196.3	3.73	4.62	0.82	0.54
DiT-L/2	50	200	168.8	4.77	4.41	0.78	0.55
+ iSSD	50	200 + 0.2	172.7	4.41	4.44	0.80	0.56
DiT-L/2	20	200	160.5	6.42	5.26	0.77	0.53
+ iSSD	20	200 + 0.2	172.3	4.92	4.87	0.80	0.53
DiT-L/2	10	200	118.5	16.01	12.63	0.67	0.47
+ iSSD	10	200 + 0.2	143.7	7.75	4.99	0.75	0.55

Table 3. Image quality on ImageNet 256×256 with DiT-S/2, DiT-B/2 and DiT-L/2

DiT-XL/2. For DiT-XL/2, we utilize the official pre-trained model, whereas for the other three models, we train them for 1 million to 2 million steps. For the text-to-image generation experiments, we employ Stable Diffusion v1.5 [44] as our base model.

Evaluation. We follow previous work [40] to use IS, FID, sFID, Precision and Recall as the evaluation metric. For text-to-image generation, we test the FID on the validation set of COCO2014 and COCO2017 [31].

Training and Sampling Details. For training, the teacher and student models are both initialized from the base model, with the teacher remaining fixed during training. We use DDIM [50] as our base ODE solver and only utilize N-step predictions during training if the fine-tuned model would be evaluated under N-step. For DiT, we set a learning rate of 5e-6, a batch size of 128. The training epochs for each experiment are listed in the tables. Exponential Moving Average (EMA) is not used due to the short training period. We follow the original DiT setup without weight decay, warm-up, or learning rate schedulers. Models were evaluated with 10, 20, and 50 numbers of function evaluations (NFEs), and we show that even the basic ODE algorithms outperformed many advanced samplers with our algorithms. More details, especially for text-to-image generation, are in the Appendix.

4.2. Conditional Generation

Results on DiT-XL. We show the results of DiT-XL/2 in Table 2. With less than 1 epoch for optimizing DiT-XL/2, our method achieves consistent performance improvement upon the base models. Notably, our method with 50 NFEs can exceed the performance of the base DiT-XL/2 sampled 250 times by a large margin (2.02 vs 2.14), both under the same ODE sampler. Also this improvement can be observed in the resolution of 512, where the base DiT achieves the FID of 3.04, while our method can improve it to 2.86. We also notice improvement compared with the 250-step SiT, with our method sampled with only 50 steps. Besides, we found the implicit SSD performs better than the explicit SSD in most cases, and sometimes the implicit SSD outperforms the explicit SSD by a large margin. We show the qualitative results on ImageNet in the Appendix.

Results on small-scale DiTs. We also apply our method under smaller DiT models, showing that the method can be generalized to models with different sizes. Since in the ex-

Figure 3. Visualization of generated images using various sampling algorithms with NFEs=10. From left to right, the algorithms shown are: PLMS, DDIM, DPM-Solver-2, and SSD.

Dataset	NFEs	DDIM	PLMS	DPM-Solver-2	Ours
COCO2014	10	12.78	17.71	15.23	10.94
	20	12.68	12.65	14.63	10.56
COCO2017	10	22.79	29.51	24.46	20.85
	20	22.43	22.14	24.02	20.69

Table 4. FID Comparison with different samplers on the validation set of COCO2014 and COCO2017

periment in DiT-XL/2, we found iSSD consistently outperforms SSD, we here test the performance of iSSD on these DiT models. From Table 3, with only 0.2 epoch for training, we observe a large improvement upon the base model. It would only take several minutes to finish the training, and thus, can be seen as a cheap way to improve the performance of the trained models.

4.3. Text-to-Image Generation

We apply SSD to Stable Diffusion v1.5. The experimental details are outlined in Appendix. Our method is compared against DDIM [50], DPM-Solver-2 [34] and PLMS [32] on COCO2014 and COCO2017, as shown in Table 4. Our approach demonstrates significant improvements in FID, outperforming other methods. Notably, our results with 10 NFEs surpass those achieved with 20 NFEs by competing samplers. Visualization results further highlight that images enhanced by our algorithm exhibit more detailed features.

4.4. Results on Distilled Models

In this section, we apply our method to a distilled model obtained through progressive distillation (PD). Notably, this implies that predictions at intermediate steps that are not explicitly part of the PD process are not trained, resulting in suboptimal performance at these steps. This experiment aims to demonstrate that our method is effective that distills from N steps to enhance performance across those N steps, since using the results from intermediate steps other than these N steps would harm the performance.

Model	NFEs	IS ↑	FID ↓	sFID ↓	Prec. ↑	Rec. ↑
DiT-XL/2	250	243.4	2.14	4.56	0.81	0.61
PD $64 \rightarrow 32$	32	239.8	2.24	4.33	0.80	0.60
+ iSSD	32	246.2	1.99	4.33	0.81	0.60
PD $20 \rightarrow 10$	10	209.9	4.19	5.24	0.76	0.57
+ iSSD	10	233.9	3.56	7.08	0.81	0.55
PD $20 \rightarrow 10 \rightarrow 5$	5	173.0	7.00	6.59	0.73	0.58
+ iSSD	5	186.7	5.41	6.48	0.76	0.56
PD $16 \rightarrow 8 \rightarrow 4$	4	152.6	8.75	7.90	0.70	0.59
+ iSSD	4	168.58	6.10	7.53	0.72	0.57

Table 5. Results of building iSSD upon diffusion models being distilled for few-step generation.

Model	IS ↑	FID ↓	sFID ↓	Prec.↑	Rec.↑
DiT-XL/2	13.4	107.75	94.57	0.11	0.36
PD [47]	158.3	7.84	7.11	0.70	0.60
Shortcut Model [9]	-	7.8	-	-	-
PD + Ours	168.6	6.11	7.53	0.72	0.57

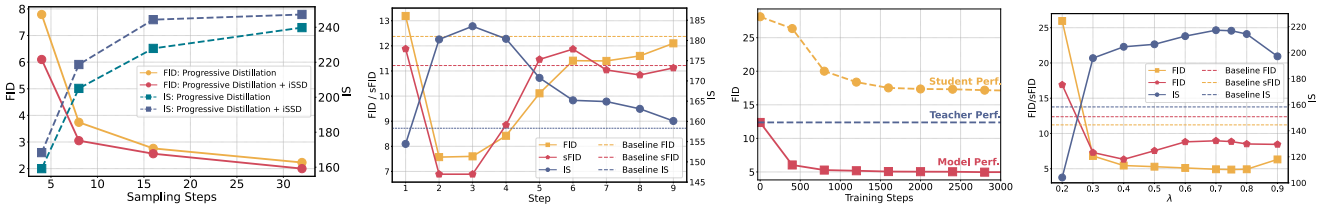
Table 6. Comparison with training-based method on ImageNet 256×256. We set NFEs=4 and use DiT-XL/2 for sampling.

Orthogonal to distillation methods. Table 5 presents the results for models distilled through two-stage and three-stage progressive distillation (PD), and Figure 4(a) illustrates the results on a chain of models distilled from progressive distillation. The results indicate that our method can significantly enhance performance. Specifically, when applied to a 32-step distilled model, our approach achieves an FID of 1.99 with minimal additional computational cost, representing a substantial improvement over the distilled model and even the pre-trained DiT-XL/2. However, when comparing our method to the model directly distilled from PD, the performance at 10-step NFE is slightly lower (4.19 vs. 4.92). We position our method as an auxiliary step that complements and further refines the performance of distilled models, providing an orthogonal improvement.

Comparison with training-based methods. We evaluated our model against flow-matching algorithms designed for training diffusion models to enable few-step generation. The experiments are performed using 4 steps to generate images, and the backbone for diffusion models is DiT-XL/2. The results, summarized in Table 6, indicate that our method surpasses the performance of competing algorithms. Specifically, when compared to the ShortCut Model [9], an enhanced variant of the flow matching model adapted for DiT, our approach achieves superior results (FID: 6.1 v.s. 7.8).

4.5. Analysis

Comparison with continual training. To demonstrate that the observed performance improvements are not merely



(a) (b) (c) (d)

Figure 4. Analysis of our proposed self-distillation framework includes: (a) iSSD consistently enhancing the quality of distilled DMs, (b) significant improvements in sample quality achieved by training only one step in the ODE trajectory, (c) evaluation of the performance of the teacher, student, and the final model, and (d) a substantial impact observed with variations in λ

Model	NFE	Train Epochs	IS↑	FID↓
DiT-XL/2	10	1400	158.3	12.38
DiT-XL/2$^\diamond$	10	1400 + 0.8	156.7	12.97
DiT-XL/2 + iSSD	10	1400 + 0.8	209.6	4.92
DiT-S$^\diamond$	250	400	27.6	51.55
DiT-S$^\diamond$	250	420	27.9	51.13
DiT-S$^\diamond$ + iSSD	250	400 + 2	28.7	48.38

Table 7. Comparison with Continual Training. The superscript $\diamond$ means we train these models from the official codebase.

Model	Data Source	IS	FID	sFID	Precision	Recall
DiT-XL/2	-	158.3	12.38	11.22	0.67	0.53
iSSD	Original Data	209.6	4.92	7.62	0.79	0.53
iSSD	Synthetic - 10 step	206.2	5.07	9.03	0.79	0.52
iSSD	Synthetic - 20 step	207.9	5.02	8.99	0.79	0.53

Table 8. Results on Synthetic Data on ImageNet 256×256. The experiments are based on DDIM with 10 sampling steps.

due to additional training iterations, we conducted experiments on DiT-XL/2 (officially pre-trained and only have the EMA model) and DiT-S/2 (has both the EMA and the base model). The results in Table 7 show that even with fewer training epochs, our method significantly outperforms models trained for 20 epochs on DiT-S/2. This confirms that the performance gains stem from the proposed self-distillation approach rather than extended training iterations.

Synthetic Data v.s. Real Data. An additional exploration involves assessing whether a model's performance can be improved when only the model itself is available, without any supplementary training data or external models. We conducted experiments under this setting, and the results are presented in Table 8. The findings indicate that, while real data remains the optimal choice for achieving the highest performance, the use of synthetic data can still result in substantial improvements in generation quality, albeit with a slight performance trade-off compared to using real data.

Distill upon only one step. In Figure 4(b), we show the results of applying iSSD to a single step. This process yields two models; here, our focus is on determining which step has the greatest impact on images. The results indicate that applying self-distillation in the later stages of denoising provides the most significant benefit to image quality.

Training dynamics for the teacher and the student in iSSD. Figure 4(c) presents the training dynamics observed during our experiments. It is crucial to note that the student model used in training differs from the one used for inference. The student model with fused activations starts with lower performance compared to the teacher model and remains inferior after training. In contrast, the unfused one, which maintains the same architecture and computation way for each layer, shows consistent improvement throughout training and eventually achieves significantly better performance than the teacher model.

Effect of gradient influential term λ. We show the effect of λ, as defined in Eq.5 and Eq.6. Though in Eq.12, we can directly calculate λ if we adopt the same intermediate step as in DPM-Solver, we also put this timestep scheduler in this hyper-parameter, and search the best λ for our approach. Figure 4(d) show for iSSD, with λ equals a small value, the model would get poor results. We show the λ used in each of our experiments in Appendix.

5. Conclusion

In this paper, we propose a novel self-distillation approach named Self Step-Distillation. We introduce two forms for Self Step-Distillation. Furthermore, we explain this approach by analyzing it through the lens of distillation with high-order samplers. Experimental results demonstrate that our method significantly outperforms the performance of the teacher model, achieving substantial improvements in both conditional generation and text-to-image generation. However, our method also has certain limitations. For instance, we still rely on suboptimal approaches to approximate the prediction of the intermediate step. Additionally, our method is relatively sensitive to hyperparameters.

Acknowledgement

This project is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award Number: MOE-T2EP20122-0006).

References

[1] 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*, pages 22563–22575, 2023. 1

[2] Kenneth Borup and Lars N Andersen. Even your teacher needs guidance: Ground-truth targets dampen regularization imposed by self-distillation. *Advances in Neural Information Processing Systems*, 34:5316–5327, 2021. 3

[3] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 19830–19843, 2023. 2

[4] Zigeng Chen, Xinyin Ma, Gongfan Fang, Zhenxiong Tan, and Xinchao Wang. Asyncdiff: Parallelizing diffusion models by asynchronous denoising. *arXiv preprint arXiv:2406.06911*, 2024. 3

[5] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 1

[6] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Genie: Higher-order denoising diffusion solvers. *Advances in Neural Information Processing Systems*, 35:30150–30166, 2022. 3

[7] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. 2

[8] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *Advances in neural information processing systems*, 36, 2024. 3

[9] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models, 2024. 7

[10] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International conference on machine learning*, pages 1607–1616. PMLR, 2018. 2, 3

[11] Sicheng Gao, Xuhui Liu, Bohan Zeng, Sheng Xu, Yanjing Li, Xiaoyan Luo, Jianzhuang Liu, Xiantong Zhen, and Baochang Zhang. Implicit diffusion models for continuous super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10021–10030, 2023. 2

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 1

[13] Yatharth Gupta, Vishnu V Jaddipal, Harish Prabhala, Sayak Paul, and Patrick Von Platen. Progressive knowledge distillation of stable diffusion xl using layer level loss. *arXiv preprint arXiv:2401.02677*, 2024. 2, 3

[14] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptqd: Accurate post-training quantization for diffusion models. *arXiv preprint arXiv:2305.10657*, 2023. 3

[15] Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1

[17] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 1

[18] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022. 1

[19] Tao Huang, Yuan Zhang, Mingkai Zheng, Shan You, Fei Wang, Chen Qian, and Chang Xu. Knowledge diffusion for distillation. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[20] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. 2, 3, 4

[21] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6007–6017, 2023. 2

[22] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. Bk-sdm: Architecturally compressed stable diffusion for efficient text-to-image generation. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*, 2023. 2, 3

[23] 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, 2024. 2

[24] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[25] Muyang Li, Tianle Cai, Jiaxin Cao, Qinsheng Zhang, Han Cai, Junjie Bai, Yangqing Jia, Ming-Yu Liu, Kai Li, and Song Han. Distrifusion: Distributed parallel inference for high-resolution diffusion models. *arXiv preprint arXiv:2402.19481*, 2024. 3

[26] Senmao Li, Taihang Hu, Joost van de Weijer, Fahad Shahbaz Khan, Tao Liu, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. Faster diffusion: Rethinking the role of the encoder for diffusion model inference, 2024. 5

[27] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022. 1

[28] Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17535–17545, 2023. 3

[29] Yize Li, Yihua Zhang, Sijia Liu, and Xue Lin. Pruning then reweighting: Towards data-efficient training of diffusion models. *arXiv preprint arXiv:2409.19128*, 2024. 3

[30] Shanchuan Lin, Anran Wang, and Xiao Yang. Sdxl-lightning: Progressive adversarial diffusion distillation. *arXiv preprint arXiv:2402.13929*, 2024. 3

[31] 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. 6

[32] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022. 3, 7

[33] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution, 2024. 1

[34] 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. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022. 3, 5, 7

[35] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022. 2

[36] 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. 2

[37] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. *arXiv preprint arXiv:2312.00858*, 2023. 3, 5

[38] 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. 2, 3

[39] Hossein Mobahi, Mehrdad Farajtabar, and Peter Bartlett. Self-distillation amplifies regularization in hilbert space. *Advances in Neural Information Processing Systems*, 33:3351–3361, 2020. 2, 3

[40] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 2, 6

[41] Minh Pham, Minsu Cho, Ameya Joshi, and Chinmay Hegde. Revisiting self-distillation. *arXiv preprint arXiv:2206.08491*, 2022. 3

[42] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 1

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

[44] 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. 1, 6

[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 4

[46] 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. *Advances in neural information processing systems*, 35:36479–36494, 2022. 1

[47] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 2, 7

[48] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pages 87–103. Springer, 2025. 3

[49] Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *arXiv preprint arXiv:2305.16317*, 2023. 3

[50] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2, 3, 6, 7, 1

[51] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 1

[52] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. 2

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4

[54] Fu-Yun Wang, Zhaoyang Huang, Alexander William Bergman, Dazhong Shen, Peng Gao, Michael Lingelbach, Keqiang Sun, Weikang Bian, Guanglu Song, Yu Liu, Hongsheng Li, and Xiaogang Wang. Phased consistency model, 2024. 2

[55] Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality. In *International Conference on Learning Representations*, 2022. 3

[56] Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, et al. Cache me if you can: Accelerating diffusion models through block caching. *arXiv preprint arXiv:2312.03209*, 2023. 3

[57] Chenglin Yang, Lingxi Xie, Siyuan Qiao, and Alan Yuille. Knowledge distillation in generations: More tolerant teachers educate better students. *arXiv preprint arXiv:1805.05551*, 2018. 3

[58] Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made slim. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22552–22562, 2023. 3

[59] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6613–6623, 2024. 2

[60] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024. 2

[61] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3713–3722, 2019. 2, 3

[62] Zhilu Zhang and Mert Sabuncu. Self-distillation as instance-specific label smoothing. *Advances in Neural Information Processing Systems*, 33:2184–2195, 2020. 2, 3

[63] 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. 3

[64] Zhenyu Zhou, Defang Chen, Can Wang, Chun Chen, and Siwei Lyu. Simple and fast distillation of diffusion models. *arXiv preprint arXiv:2409.19681*, 2024. 2

[65] Haowei Zhu, Dehua Tang, Ji Liu, Mingjie Lu, Jintu Zheng, Jinzhang Peng, Dong Li, Yu Wang, Fan Jiang, Lu Tian, et al. Dip-go: A diffusion pruner via few-step gradient optimization. *arXiv preprint arXiv:2410.16942*, 2024. 3