SlimFlow: Training Smaller One-Step Diffusion Models with Rectified Flow

Yuanzhi Zhu[1], Xingchao Liu[2], and Qiang Liu[2(✉)]

[1] ETH Zurich, 8092 Zurich, Switzerland
yuazhu@student.ethz.ch
[2] UT Austin, Austin, TX 78712, USA
xcliu@utexas.edu, lqiang@cs.utexas.edu

Abstract. Diffusion models excel in high-quality generation but suffer from slow inference due to iterative sampling. While recent methods have successfully transformed diffusion models into one-step generators, they neglect model size reduction, limiting their applicability in compute-constrained scenarios. This paper aims to develop small, efficient one-step diffusion models based on the powerful rectified flow framework, by exploring joint compression of inference steps and model size. The rectified flow framework trains one-step generative models using two operations, reflow and distillation. Compared with the original framework, squeezing the model size brings two new challenges: (1) the initialization mismatch between large teachers and small students during reflow; (2) the underperformance of naive distillation on small student models. To overcome these issues, we propose Annealing Reflow and Flow-Guided Distillation, which together comprise our SlimFlow framework. With our novel framework, we train a one-step diffusion model with an FID of 5.02 and 15.7M parameters, outperforming the previous state-of-the-art one-step diffusion model (FID = 6.47, 19.4M parameters) on CIFAR10. On ImageNet 64×64 and FFHQ 64×64, our method yields small one-step diffusion models that are comparable to larger models, showcasing the effectiveness of our method in creating compact, efficient one-step diffusion models.

Keywords: Diffusion Models · Flow-based Models · One-Step Generative Models · Efficient Models

1 Introduction

In recent years, diffusion models [14,56] have revolutionized the field of image generation [4,32,41,46], surpassing the quality achieved by traditional approaches such as Generative Adversarial Networks (GANs) [9,19], Normalizing Flows (NFs) [5,42] and Variational Autoencoders (VAEs) [22,47]. Its success

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-73007-8_20.

A. Leonardis et al. (Eds.): ECCV 2024, LNCS 15140, pp. 342–359, 2025.
https://doi.org/10.1007/978-3-031-73007-8_20

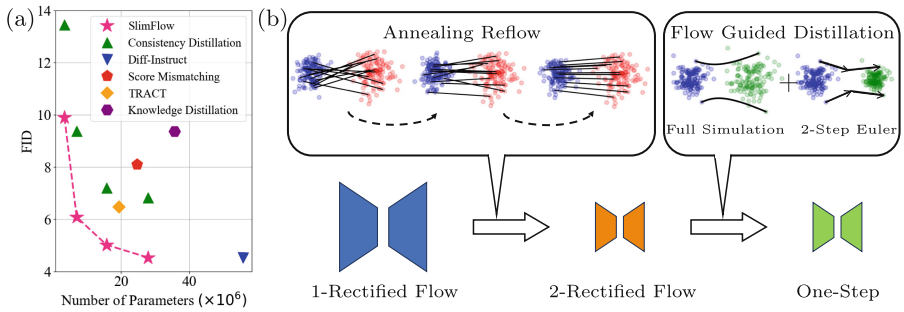

Fig. 1. (a) Comparison of different one-step diffusion models on the CIFAR10 dataset. (b) To get powerful one-step diffusion model, our SlimFlow framework designs two stages: **Annealing Reflow** provides a warm-start for the small 2-Rectified Flow model by gradually shifting from training with random pairs to teacher pairs; **Flow Guided Distillation** enhances the one-step small model by distillation from 2-Rectified Flow with both off-line generated data using precise ODE solver and online generated data using 2-step Euler solver.

even extends to other modalities, including audio [17,23], video [8,13,40] and 3D content generation [30,45,58,61].

Despite these advancements, diffusion models face significant challenges in terms of generation efficiency. Their iterative sampling process and substantial model size pose considerable obstacles to widespread adoption, particularly in resource-constrained environments, such as edge devices. To enhance the inference speed of diffusion models, recent research has focused on two primary strategies. The first strategy aims at reducing the number of inference steps, which can be achieved through either adoption of fast solvers [6,34,52,66,70] or distillation [10,29,35,49,53]. The second strategy focuses on lowering the computational cost within each inference step, by model structure modification [2,7,44,63], quantization [11,15,26,27,50,57], and caching [38,60]. Recently, these techniques have been applied to various large-scale diffusion models, e.g., Stable Diffusion [48], to significantly improve their inference speed and reduce their cost [33,36,39,62,65].

This paper focuses on advancing the rectified flow framework [29,31], which has demonstrated promising results in training few/one-step diffusion models [33]. While the framework has shown success in reducing inference steps, it has not addressed the challenge of model size reduction. We aim to enhance the rectified flow framework by simultaneously reducing both the number of inference steps and the network size of diffusion models. The rectified flow framework straightens the trajectories of pre-trained generative probability flows through a process called reflow, thereby decreasing the required number of inference steps. This procedure also refines the coupling between noise and data distributions. High-quality one-step generative models are then obtained by distilling from the straightened flow. Unlike typical rectified flow applications that maintain an

invariant model structure during the entire process, we aim to reduce the network size in reflow and distillation, targeting efficient one-step diffusion models. It introduces two key challenges: (1) The reflow operation typically initializes the student flow with the pre-trained teacher's weights to inherit knowledge and accelerate convergence. However, this strategy is inapplicable when the student network has a different structure. (2) The smaller student network suffers from reduced capacity, causing naive distillation to underperform. To address these challenges, we propose SlimFlow, comprising two stages: Annealing Reflow and Flow-Guided Distillation (Fig. 1). Annealing Reflow provides a warm-start initialization for the small student model by smoothly interpolating between training from scratch and reflow. Flow-Guided Distillation introduces a novel regularization that leverages guidance from the learned straighter student flow, resulting in better distilled one-step generators. SlimFlow achieves state-of-the-art Frechet Inception Distance (FID) among other one-step diffusion models with a limited number of parameters. Furthermore, when applied to ImageNet 64×64 and FFHQ 64×64, SlimFlow yields small one-step diffusion models that are comparable to larger models.

2 Background

2.1 Diffusion Models

Diffusion models define a forward diffusion process that maps data to noise by gradually perturbing the input data with Gaussian noise. Then, in the reverse process, they generate images by gradually removing Gaussian noise, with the intuition from non-equilibrium thermodynamics [51]. We denote the data $\mathbf{x}$ at t as $\mathbf{x}_t$. The forward process can be described by an Itô SDE [56]:

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

where $\mathbf{w}$ is the standard Wiener process, $\mathbf{f}(\cdot, t)$ is a vector-valued function called the drift coefficient, and $g(\cdot)$ is a scalar function called the diffusion coefficient.

For every diffusion process in Eq. (1), there exists a corresponding deterministic Probability Flow Ordinary Differential Equation (PF-ODE) which induces the same marginal density as Eq. (1):

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}_t, t) - \frac{1}{2}g^2(t)\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t), \tag{2}$$

where $p_t(\cdot)$ is the marginal probability density at time t. $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is called the score function, and can be modelled as $\mathbf{s}_\theta(\mathbf{x}, t)$ using a neural network θ. Usually, the network is trained by score matching [16,54,55]. Starting with samples from an initial distribution π_T such as a standard Gaussian distribution, we can generate data samples by simulating Eq. (2) from $t = T$ to $t = 0$. We call Eq. (2) with the approximated score function $\mathbf{s}_\theta(\mathbf{x}_t, t)$ the empirical PF-ODE, written as $\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}_t, t) - \frac{1}{2}g^2(t)\mathbf{s}_\theta(\mathbf{x}_t, t)$. It is worth noting that the deterministic PF-ODE gives a deterministic correspondence between the initial noise distribution and the generated data distribution.

2.2 Rectified Flows

Rectified flow [28,29,31] is an ODE-based generative modeling framework. Given the initial distribution π_T and the target data distribution π_0, rectified flow trains a velocity field parameterized by a neural network with the following loss function,

$$\mathcal{L}_{\text{rf}}(\theta) := \mathbb{E}_{\mathbf{x}_T\sim\pi_T,\mathbf{x}_0\sim\pi_0}\left[\int_0^T \left\|\mathbf{v}_\theta(\mathbf{x}_t,t)-(\mathbf{x}_T-\mathbf{x}_0)\right\|_2^2 \mathrm{d}t\right], \quad \text{where} \quad \mathbf{x}_t=(1-t/T)\mathbf{x}_0+t\mathbf{x}_T/T. \tag{3}$$

Without loss of generalization, T is usually set to 1. Based on the trained rectified flow, we can generate samples by simulating the following ODE from $t=1$ to $t=0$,

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t}=\mathbf{v}_\theta(\mathbf{x}_t,t). \tag{4}$$

PF-ODEs transformed from pre-trained diffusion models can be seen as special forms of rectified flows. For a detailed discussion of the mathematical relationship between them, we refer readers to [29,31]. In computer, Eq. (4) is approximated by off-the-shelf ODE solvers, e.g., the forward Euler solver,

$$\mathbf{x}_{t-\frac{1}{N}}=\mathbf{x}_t-\frac{1}{N}\mathbf{v}_\theta(\mathbf{x}_t,t), \quad \forall t\in\{1,2,\ldots,N\}/N. \tag{5}$$

Here, the ODE is solved in N steps with a step size of $1/N$. Large N leads to accurate but slow simulation, while small N gives fast but inaccurate simulation. Fortunately, straight probability flows with uniform speed enjoy one-step simulation with no numerical error because $\mathbf{x}_t=\mathbf{x}_1-(1-t)\mathbf{v}_\theta(\mathbf{x}_1,1)$.

Reflow. In the rectified flow framework, a special operation called reflow is designed to train straight probability flows,

$$\mathcal{L}_{\text{reflow}}(\phi) := \mathbb{E}_{\mathbf{x}_1\sim\pi_1}\left[\int_0^T \left\|\mathbf{v}_\phi(\mathbf{x}_t,t)-(\mathbf{x}_1-\hat{\mathbf{x}}_0)\right\|_2^2 \mathrm{d}t\right], \quad \text{where} \quad \mathbf{x}_t=(1-t)\hat{\mathbf{x}}_0+t\mathbf{x}_1 \ \text{ and } \ \hat{\mathbf{x}}_0=\mathrm{ODE}[\mathbf{v}_\theta](\mathbf{x}_1). \tag{6}$$

Compared with Eq. (3), $\hat{\mathbf{x}}_0$ is not a random sample from distribution π_0 that is independent with $\mathbf{x}_1$ anymore. Instead, $\hat{\mathbf{x}}_0=\mathrm{ODE}[\mathbf{v}_\phi](\mathbf{x}_1):=\mathbf{x}_1+\int_1^0\mathbf{v}_\theta(\mathbf{x}_t,t)\mathrm{d}t$ is induced from the pre-trained flow $\mathbf{v}_\theta$. After training, the new flow $\mathbf{v}_\phi$ has straighter trajectories and requires fewer inference steps for generation. A common practice in reflow is initializing $\mathbf{v}_\phi$ with the pre-trained $\mathbf{v}_\theta$ for faster convergence and higher performance [31]. Following previous works, we name $\mathbf{v}_\theta$ as 1-rectified flow and $\mathbf{v}_\phi$ as 2-rectified flow. The straightness of the learned flow $\mathbf{v}$ can be defined as:

$$S(\mathbf{v})=\int_0^1 \mathbb{E}\left[\|\mathbf{v}(\mathbf{x}_t,t)-(\mathbf{x}_1-\mathbf{x}_0)\|^2\right]\mathrm{d}t, \tag{7}$$

Distillation. Given a pre-trained probability flow, for example, $\mathbf{v}_\phi$, we can further enhance their one-step generation via distillation,

$$\mathcal{L}_{\text{distill}}(\phi') := \mathbb{E}_{\mathbf{x}_1 \sim \pi_1} \left[\mathbb{D}(\text{ODE}[\mathbf{v}_\phi](\mathbf{x}_1), \mathbf{x}_1 - \mathbf{v}_{\phi'}(\mathbf{x}_1, 1)) \right]. \tag{8}$$

In this equation, $\mathbb{D}$ is a discrepancy loss that measures the difference between two images, e.g., ℓ_2 loss or the LPIPS loss [67]. Through distillation, the distilled model $\mathbf{v}_{\phi'}$ can use one-step Euler discretization to approximate the result of the entire flow trajectory. Note that reflow can refine the deterministic mapping between the noise and the generated samples defined by the probability flow [31, 33]. Consequently, distillation from $\mathbf{v}_\phi$ gives better one-step generators than distillation from $\mathbf{v}_\theta$.

3 SlimFlow

In this section, we introduce the SlimFlow framework for learning small, efficient one-step generative models. SlimFlow enhances the rectified flow framework by improving both the reflow and distillation stages. For the reflow stage, we propose Annealing Reflow. This novel technique provides a warm-start initialization for the small student model. It smoothly transitions from training a 1-rectified flow to training a 2-rectified flow, accelerating the training convergence of the smaller network. For the distillation stage, we propose Flow-Guided Distillation, which leverages the pre-trained 2-rectified flow as an additional regularization during distillation to improve the performance of the resulting one-step model. By combining these two techniques, SlimFlow creates a robust framework for training efficient one-step diffusion models that outperform existing methods in both model size and generation quality.

3.1 Annealing Reflow

The reflow procedure in the rectified flow framework trains a new, straighter flow $\mathbf{v}_\phi$ from the pre-trained 1-rectified flow $\mathbf{v}_\theta$, as described in Eq. (6). Typically, when $\mathbf{v}_\phi$ and $\mathbf{v}_\theta$ have the same model structure, $\mathbf{v}_\phi$ is initialized with $\mathbf{v}_\theta$'s weights to accelerate training. However, our goal of creating smaller models cannot adopt this approach, as $\mathbf{v}_\phi$ and $\mathbf{v}_\theta$ have different structures. A naive solution would be to train a new 1-rectified flow using the smaller model with Eq. (3) and use this as initialization. However, this strategy is time-consuming and delays the process of obtaining an efficient 2-rectified flow with the small model.

To address this challenge, we propose Annealing Reflow, which provides an effective initialization for the small student model without significantly increasing training time. This method starts the training process with random pairs, as in Eq. (3), then gradually shifts to pairs generated from the teacher flow, as in Eq. (6).

Formally, given the teacher velocity field $\mathbf{v}_\theta$ defined by a large neural network θ, the objective of Annealing Reflow is defined as,

$$\begin{aligned}\mathcal{L}^k_{\text{a-reflow}}(\phi) &:= \mathbb{E}_{\mathbf{x}_1,\mathbf{x}'_1\sim\pi_1}\left[\int_0^T \left\|\mathbf{v}_\phi(\mathbf{x}_t^{\beta(k)},t) - \left(\mathbf{x}_1^{\beta(k)} - \hat{\mathbf{x}}_0\right)\right\|_2^2 \mathrm{d}t\right], \\ \text{where}\quad \mathbf{x}_t^{\beta(k)} &= (1-t)\hat{\mathbf{x}}_0 + t\mathbf{x}_1^{\beta(k)}, \\ \mathbf{x}_1^{\beta(k)} &= \left(\sqrt{1-\beta^2(k)}\mathbf{x}_1 + \beta(k)\mathbf{x}'_1\right), \\ \hat{\mathbf{x}}_0 &= \mathrm{ODE}[\mathbf{v}_\theta](\mathbf{x}_1) = \mathbf{x}_1 + \int_1^0 \mathbf{v}_\theta(\mathbf{x}_t,t)\mathrm{d}t.\end{aligned} \tag{9}$$

In Eq. (9), $\mathbf{x}_1$ and $\mathbf{x}'_1$ are independent random samples from the initial distribution π_1. k refers to the number of training iterations, and $\beta(k) : \mathbb{N} \to [0,1]$ is a function that maps k to a scalar between 0 and 1. The function $\beta(k)$ must satisfy two key conditions:

1. $\beta(0) = 1$: This initial condition reduces $\mathcal{L}^0_{\text{a-reflow}}$ to the rectified flow objective (Eq. (3)), with $\hat{\mathbf{x}}_0$ generated from the well-trained teacher model instead of being randomly sampled from π_0. Note that when $\mathbf{v}_\theta$ is well-trained, $\hat{\mathbf{x}}_0$ also follows the target data distribution π_0.
2. $\beta(+\infty) = 0$: This asymptotic condition ensures that $\mathcal{L}^{+\infty}_{\text{a-reflow}}$ converges to the reflow objective (Eq. (6)), guaranteeing that the small model will eventually become a valid 2-rectified flow.

By carefully choosing the schedule β, we create a smooth transition from training the small student model using random pairs to using pairs generated from the large teacher flow. This approach provides an appropriate initialization for the student model and directly outputs a small 2-rectified flow model, balancing efficiency and effectiveness in the training process.

Leveraging the Intrinsic Symmetry in Reflow Pairs. To perform reflow, a dataset of pair $(\mathbf{x}_1, \hat{\mathbf{x}}_0)$ is usually generated by the teacher flow before training the student model. We find that the hidden symmetry in the pairs can be exploited to create new pairs without simulating the teacher flow. Specifically, if the initial noise $\mathbf{x}_1$ is horizontally flipped (referred to as H-Flip), the corresponding image $\hat{\mathbf{x}}_0$ is also horizontally flipped (Fig. 2). As a result, for each generated pair $(\mathbf{x}_1, \hat{\mathbf{x}}_0)$, we can also add $(\text{H-Flip}(\mathbf{x}_1), \text{H-Flip}(\hat{\mathbf{x}}_0))$ as a valid pair to double the sample size of our reflow dataset.

3.2 Flow-Guided Distillation

Distillation is the other vital step in training one-step diffusion models with the rectified flow framework. To distill the student flow, naive distillation (Eq. (8)) requires simulating the entire student 2-rectified flow with off-the-shelf solvers, e.g., Runge-Kutta 45 (RK45) solver, to get the distillation target $\mathrm{ODE}[\mathbf{v}](\mathbf{x}_1)$. In most times, practitioners generate a dataset of M teacher samples in advance

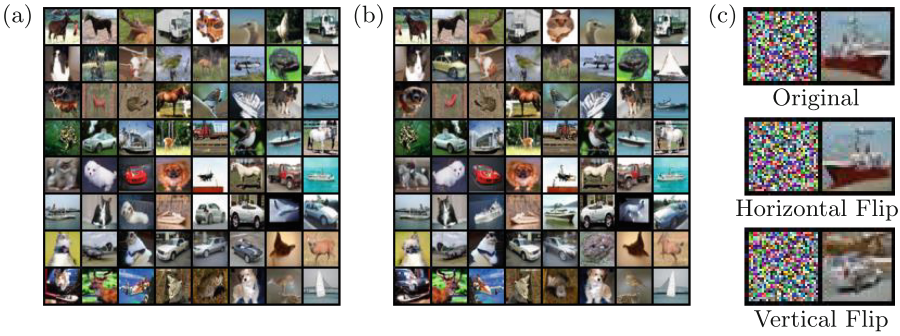

Fig. 2. (a) Generation from 1-Rectified Flow trained without data augmentation. (b) Generation from the 1-Rectified Flow model in (a) after applying horizontal flip to the same set of random noises in (a). (c) Horizontally flipping the noise results in horizontally flipped generated image, but vertical flip does not result in vertically flipped generated image.

to training the student model, denoted as $\mathcal{D}_{distill} = \left\{\left(\mathbf{x}_1^{(i)}, \text{ODE}[\mathbf{v}](\mathbf{x}_1^{(i)})\right)\right\}_{i=1}^{M}$. Such simulation takes tens of steps, leading to excessive time and storage for the distilled one-step model. On the one hand, for small models in our scenario, we want to use large M to reach satisfying one-step models due to their limited capacity. On the other hand, we also expect to avoid costly data generation in the distillation stage.

To balance this trade-off, we propose flow-guided distillation, where we complement the expensive direct distillation with an additional regularization. This regularization is based on few-step generation from the 2-rectified flow $\mathbf{v}_\phi$. When estimating the integration result of $\text{ODE}[\mathbf{v}_\phi](\mathbf{x}_1)$, we can adopt the forward Euler solver,

$$\mathbf{x}_{t_i} = \mathbf{x}_{t_{i+1}} + (t_i - t_{i+1})\mathbf{v}_\phi(\mathbf{x}_{t_{i+1}}, t_{i+1}), \quad \forall i \in \{0, 1, \ldots, N-1\}, \tag{10}$$

where the N time steps $\{t_i\}_{i=0}^N$ are defined by the user and $t_0 = 0, t_N = 1$. To find an appropriate supervision for one-step simulation without cumbersome precise simulation with advanced solvers, a feasible choice is two-step simulation, which is,

$$\hat{\mathbf{x}}_0 = \mathbf{x}_1 - (1-t)\mathbf{v}_\phi(\mathbf{x}_1, 1) - t\mathbf{v}_\phi(\mathbf{x}_t, t). \tag{11}$$

Here, t can be an intermediate time point between $(0, 1)$. Using this two-step approximation, we get another distillation loss,

$$\mathcal{L}_{\text{2-step}}(\phi') := \mathbb{E}_{\mathbf{x}_1 \sim \pi_1}\left[\int_0^1 \mathbb{D}(\mathbf{x}_1 - (1-t)\mathbf{v}_\phi(\mathbf{x}_1, 1) - t\mathbf{v}_\phi(\mathbf{x}_t, t), \mathbf{x}_1 - \mathbf{v}_{\phi'}(\mathbf{x}_1, 1))\mathrm{d}t\right]. \tag{12}$$

While this two-step generation is not as accurate as the precise generation $\text{ODE}[\mathbf{v}_\phi](\mathbf{x}_1)$ with Runge-Kutta solvers, it can serve as a powerful additional regularization in the distillation process to boost the performance of the student

Algorithm 1. Flow-Guided Distillation

Require: Pre-trained 2-rectified flow v_ϕ, dataset $\mathcal{D}_{distill}$ generated with v_ϕ.
1: Initialize the one-step student model $\mathbf{v}_{\phi'}$ with the weights of $\mathbf{v}_\phi$.
2: **repeat**
3: Randomly sample $(\mathbf{x}_1, \text{ODE}[\mathbf{v}_\phi](\mathbf{x}_1)) \sim \mathcal{D}_{distill}$.
4: Compute $\mathcal{L}_{\text{distill}}(\phi')$ with Eq. (8).
5: Randomly sample $\mathbf{x}_1 \sim \pi_1$.
6: Compute $\mathcal{L}_{\text{2-step}}(\phi')$ with Eq. (12).
7: Compute $\mathcal{L}_{\text{combined}}(\phi') = \mathcal{L}_{\text{distill}}(\phi') + \mathcal{L}_{\text{2-step}}(\phi')$.
8: Optimize ϕ' with an gradient-based optimizer using $\nabla_{\phi'}\mathcal{L}_{\text{combined}}$.
9: **until** $\mathcal{L}_{\text{combined}}$ converges.
10: **Return** one-step model $v_{\phi'}$.

one-step model. Because two-step generation is fast, we can compute this term online as extra supervision to the student one-step model without the need to increase the size of $\mathcal{D}_{distill}$. Our final distillation loss is,

$$\mathcal{L}_{\text{combined}}(\phi') = \mathcal{L}_{\text{distill}}(\phi') + \mathcal{L}_{\text{2-step}}(\phi'). \tag{13}$$

It is worth mentioning that we can use more simulation steps to improve the accuracy of the few-step regularization within the computational budget. In our experiments, using two-step generation as our regularization already gives satisfying improvement. We leave the discovery of other efficient regularization as our future work. The whole procedure of our distillation stage is captured in Algorithm 1.

4 Experiments

In this section, we provide empirical evaluation of SlimFlow and compare it with prior arts. The source code is available at https://github.com/yuanzhi-zhu/SlimFlow.

4.1 Experimental Setup

Datasets and Pre-trained Teacher Models. Our experiments are performed on the CIFAR10 [24] 32×32 and the FFHQ [19] 64×64 datasets to demonstrate the effectiveness of SlimFlow. Additionally, we evaluated our method's performance on the ImageNet [3] 64×64 dataset, focusing on conditional generation tasks. The pre-trained large teacher models are adopted from the official checkpoints from previous works, Rectified Flow [29] and EDM [18].

Implementation Details. For experiments on CIFAR10 32×32 and the FFHQ 64×64, we apply the U-Net architecture of NCSN++ proposed in [56]. For experiments on ImageNet 64×64, we adopt a different U-Net architecture proposed in [4]. In the CIFAR10 experiments, we executed two sets of experiments with

Table 1. Comparison on CIFAR10. '$*$' refers to reproduced results.

Category	Method	#Params	NFE (↓)	FID (↓)	MACs (↓)
Teacher Model	EDM [18]	55.7M	35	1.96	20.6G
	1-Rectified Flow [31]	61.8M	127	2.58	10.3G
Diffusion + Pruning	Proxy Pruning [25]	38.7M	100	4.21	
	Diff-Pruning [7]	27.5M	100	4.62	5.1G
	Diff-Pruning [7]	19.8M	100	5.29	3.4G
	Diff-Pruning [7]	14.3M	100	6.36	2.7G
Diffusion + Fast Samplers	AMED-Solver [70]	55.7M	5	7.14	20.6G
	GENIE [6]	61.8M	10	5.28	10.3G
	3-DEIS [66]	61.8M	10	4.17	10.3G
	DDIM [52]	35.7M	10	13.36	6.1G
	DPM-Solver-2 [34]	35.7M	10	5.94	6.1G
	DPM-Solver-Fast [34]	35.7M	10	4.70	6.1G
Diffusion + Distillation	DSNO [69]	65.8M	1	3.78	
	Progressive Distillation [49]	60.0M	1	9.12	
	Score Mismatching [64]†	24.7M	1	8.10	
	TRACT [1]	19.4M	1	6.47	
	Diff-Instruct [37]†	55.7M	1	4.53	20.6G
	TRACT [1]	55.7M	1	3.78	20.6G
	DMD [65]†	55.7M	1	3.77	20.6G
	Consistency Distillation (EDM teacher) [53]	55.7M	1	3.55	20.6G
	1-Rectified Flow (+distill) [29]	61.8M	1	6.18	10.3G
	2-Rectified Flow (+distill) [29]	61.8M	1	4.85	10.3G
	3-Rectified Flow (+distill) [29]	61.8M	1	5.21	10.3G
	Consistency Distillation (EDM teacher) [53]*	27.9M	1	6.83	6.6G
	SlimFlow (EDM teacher)	27.9M	1	4.53	6.6G
	Knowledge Distillation [35]	35.7M	1	9.36	6.1G
	Consistency Distillation (EDM teacher) [53]*	15.7M	1	7.21	3.7G
	SlimFlow (EDM teacher)	15.7M	1	5.02	3.7G
	SlimFlow (1-Rectified Flow teacher)	15.7M	1	5.81	3.7G

two different teacher models: one is the pre-trained 1-Rectitied Flow [31] and the other is the pre-trained EDM model [18]. For all the other experiments, we adopt only the pre-trained EDM model as the large teacher model, unless specifically noted otherwise. All the experiments are conducted on 4 NVIDIA 3090 GPUs. In Distillation, we found replacing $v_\phi(\mathbf{x}_1, 1)$ with $v_{\phi'}(\mathbf{x}_1, 1)$ in Eq. (12) leads to high empirical performance as $v_{\phi'}$ is a better one-step generator and speed up the training by saving one forward of the 2-rectified flow, so we keep that in our practice. More details can be found in the Appendix.

Evaluation Metrics. We use the Fréchet inception distance (FID) [12] to evaluate the quality of generated images. In our experiments, we calculate the FID by comparing 50,000 generated images with the training dataset using Clean-FID [43]. We also report the number of parameters (#Params), Multiply-Add Accumulation (MACs), and FLoating-point OPerations per second (FLOPs) as metrics to compare the computational efficiency of different models. It is important to note that in this paper, both MACs and FLOPs refer specifically to the computation required for a single forward inference pass through the deep neural network. We use Number of Function Evaluations (NFEs) to denote the number of inference steps.

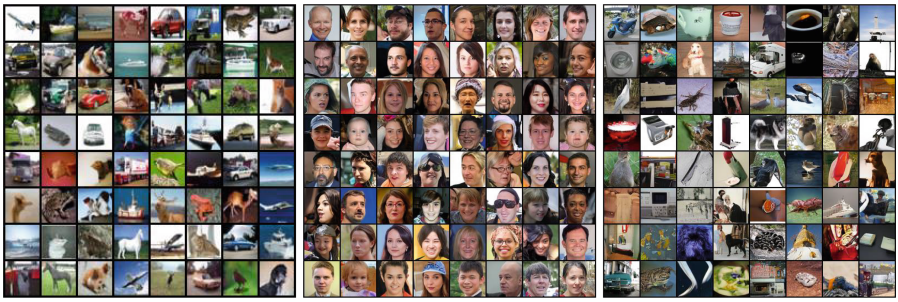

Fig. 3. Random generation from our best one-step small models on three different datasets. Left: CIFAR10 32 × 32 (#Params = 27.9M). Mid: FFHQ 64 × 64 (#Params = 27.9M). Right: ImageNet 64 × 64 (#Params = 80.7M).

Table 2. Comparison on FFHQ 64 × 64 and ImageNet 64 × 64.

Dataset	Method	#Params	NFE (↓)	FID (↓)	MACs (↓)	FLOPs (↓)
FFHQ 64 × 64	EDM [18]	55.7M	79	2.47	82.7G	167.9G
	DDIM [52]	55.7M	10	18.30	82.7G	167.9G
	AMED-Solver [70]	55.7M	5	12.54	82.7G	167.9G
	BOOT [10]	66.9M	1	9.00	25.3G	52.1G
	SlimFlow (EDM teacher)	27.9M	1	7.21	26.3G	53.8G
	SlimFlow (EDM teacher)	15.7M	1	7.70	14.8G	30.4G
ImageNet 64 × 64	EDM [18]	295.9M	79	2.37	103.4G	219.4G
	DDIM [52]	295.9M	10	16.72	103.4G	219.4G
	AMED-Solver [70]	295.9M	5	13.75	103.4G	219.4G
	DSNO [69]	329.2M	1	7.83		
	Progressive Distillation [49]	295.9M	1	15.39	103.4G	219.4G
	Diff-Instruct [37]	295.9M	1	5.57	103.4G	219.4G
	TRACT [1]	295.9M	1	7.43	103.4G	219.4G
	DMD [65]	295.9M	1	2.62	103.4G	219.4G
	Consistency Distillation [53]	295.9M	1	6.20	103.4G	219.4G
	Consistency Training [53]	295.9M	1	13.00	103.4G	219.4G
	BOOT [10]	226.5M	1	16.30	78.2G	157.4G
	SlimFlow (EDM teacher)	80.7M	1	12.34	31.0G	67.8G

4.2 Empirical Results

SlimFlow on CIFAR10. We report the results on CIFAR10 in Table 1 and Fig. 3. For all the SlimFlow models, we train them using Annealing Reflow with the data pairs generated from the large teacher model for 800,000 iterations, then distilling them to one-step generator with flow-guided distillation for 400,000 iterations. For the consistency distillation baselines, we train them for 1,200,000 iterations with the same EDM teacher to ensure a fair comparison. With only 27.9M parameters, one-step SlimFlow outperforms the 61.8M 2-Rectified Flow+Distill model (FID: 4.53 ↔ 4.85). With the less than 20M parameters, SlimFlow gives better FID (5.02, #Params = 15.7M) than TRACT (6.48, #Params = 19.4M) and Consistency Distillation (7.21,

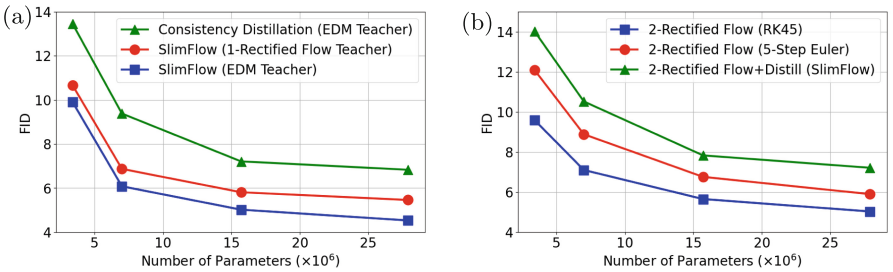

Fig. 4. (a) Comparison of models trained with different methods on CIFAR10. (b) Comparison between 2-rectified flow and the distilled one-step generator on CIFAR10.

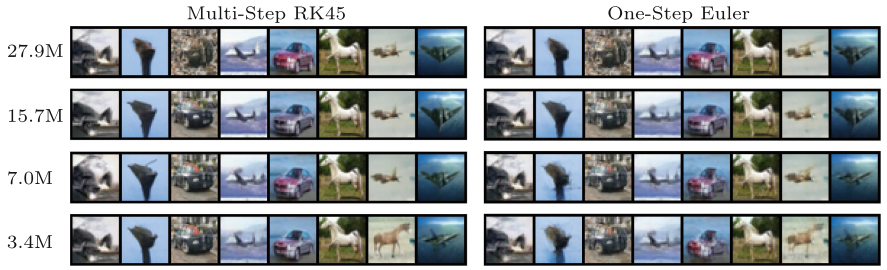

Fig. 5. CIFAR10 samples from 2-rectified flow models trained with Annealing Reflow. All images are generated with the same set of random noises.

#Params = 15.7M). Additional SlimFlow results with different parameters can be found in Fig. 4a for comparison with Consistency Distillation [53] and in Fig. 4b for comparison with 2-rectified flow.

SlimFlow on FFHQ and ImageNet. We report the results of SlimFlow on FFHQ 64 × 64 and ImageNet 64 × 64 in Table 2 and Fig. 3. For ImageNet 64 × 64, the models are trained in the conditional generation scenarios where class labels are provided. In FFHQ, our SlimFlow models surpasses BOOT with only 15.7M parameters. In ImageNet, SlimFlow obtains comparable performance as the 295.9M models trained with Consistency Training, BOOT and Progressive Distillation, using only 80.7M parameters. These results demonstrates the effectiveness of SlimFlow in training efficient one-step generative models.

Analysis of Annealing Reflow. We examine the straightening effect of Annealing Reflow. We measured the straightness in the Annealing Reflow stage of models with different sizes on both the CIFAR10 32×32 and the FFHQ 64×64 dataset. Here, straightness is defined as in Eq. (7) following [31,33]. In Fig. 6a, straightness decreases as $\beta(k)$ gradually approaches 0. In Fig. 6b, we observe that the straightness of the resulting 2-rectified flows decreases as their number of parameters increase. In Fig. 5, samples generated with four 2-rectified flows

Table 3. Ablation study on the reflow strategy of SlimFlow.

Annealing	H-Flip	FID (↓)
-	-	5.84
-	✓	5.06
✓	-	5.46
✓	✓	4.51

Table 4. Ablation study on the annealing strategy of SlimFlow. Recall that k is the number of training iterations. We set $K_{step} = 50,000$.

$\beta(k)$	FID (↓)
0	5.06
$\exp(-k/K_{step})$	4.78
$\cos(\pi \min(1, k/2K_{step})/2)$	4.79
$(1 + \cos(\pi \min(1, k/2K_{step})))/2$	4.70
$1 - \min(1, k/6K_{step})$	4.51

Table 5. Ablation study on the distillation stage of SlimFlow on CIFAR10.

Initialize from v_ϕ	Source of $\mathcal{D}_{distill}$	Loss $\mathbb{D}$	with $\mathcal{L}_{2\text{-step}}$	FID
-	1-rectified flow v_θ	ℓ_2	-	12.09
✓	1-rectified flow v_θ	ℓ_2	-	8.98
-	2-rectified flow v_ϕ	ℓ_2	-	9.19
✓	2-rectified flow v_ϕ	ℓ_2	-	7.90
✓	2-rectified flow v_ϕ	ℓ_2	✓	7.30
✓	2-rectified flow v_ϕ	LPIPS	-	6.43
✓	2-rectified flow v_ϕ	LPIPS	✓	5.81

with different samplers are presented. These results demonstrate the effectiveness of Annealing Reflow in learning straight generative flows.

4.3 Ablation Study

Annealing Reflow. We examine the design choices in Annealing Reflow. We train 2-rectified flow for 800,000 iterations and measure its FID with RK45 solver. In Table 3, we report the influence of the Annealing Reflow strategy and the effectiveness of exploiting the intrinsic symmetry of reflow. It can be observed that both the annealing strategy and the intrinsic symmetry improve the performance of 2-rectified flow, and their combination gives the best result. In Table 4, we analyze the schedule of $\beta(k)$. When $\beta(k) = 0$, it is equivalent to training 2-rectified flow with random initialization. All other schedules output lower FID than $\beta(k) = 0$, showing the usefulness of our annealing strategy. We adopt $\beta(k) = 1 - \min(1, k/6K_{step})$ as our default schedule in our experiments.

Flow-Guided Distillation. In Table 5, we analyze the influence of different choices in our distillation stage. We found that using our 2-step regularization boosts the FID the distilled one-step model. Using ℓ_2 loss, it improves the FID from 7.90 to 7.30. Using LPIPS loss, it improves the FID from 6.43 to 5.81. We use the model architecture with 15.7M parameters for all ablation studies.

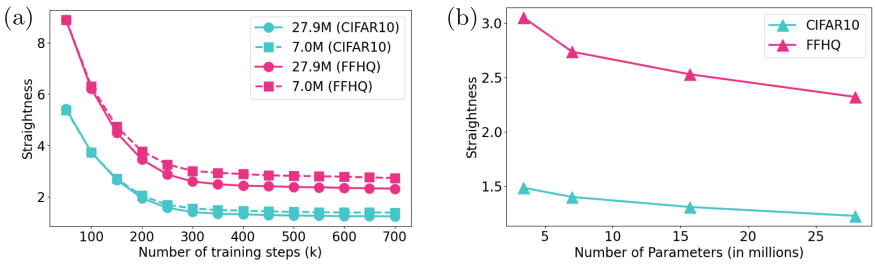

Fig. 6. (a) Straightness of 2-rectified flows with different sizes during Annealing Reflow. (b) Final straightness of 2-rectified flows with different numbers of parameters.

5 Related Work

Reducing the Inference Steps of Diffusion Models. Diffusion models generate new samples through an iterative process, where a noisy image is repeatedly denoised by a neural network. To accelerate this process, researchers have proposed various methods, which can be categorized into two main approaches.

The first category is training-free methods, which aim to reduce the number of inference steps for existing diffusion models by optimizing the sampling process. By reformulating diffusion models into PF-ODEs [56], numerous techniques are introduced to accelerate sampling while minimizing quality loss. Some of the notable examples in this area include DDIM [52], EDM [18], DEIS [66] and DPM-solver [34]. A recent advancement in this category is the AMED-Solver [70], which leverages the mean value theorem to minimize discretization error, achieving high-quality generation with even fewer function evaluations.

Beyond these fast ODE salvers, distillation-based methods aim to achieve few-step sampling, or even one-step sampling, by training a new student model with the pre-trained multi-step diffusion models as teachers. Progressive Distillation [49] proposed to repeatedly train a student network whose step size is twice as the step size of the teacher and set the student as a new teacher for the next round. BOOT [10] suggested distilling the knowledge of the teacher models in a data-free manner with the help of the signal-ODE. Distribution Matching Distillation [65] extended the idea of Variational Score Distillation [59] to train a one-step generator by alternatively updating the one-step generator and a fake data score function. Consistency models [36,53] are a new family of generative models that trains few-step diffusion models by applying consistency loss. Additionally, several methods have incorporated adversarial training to enhance the performance of one-step diffusion models [21,62,64].

Reducing the Size of Diffusion Models. The increasing demand for low-budget and on-device applications necessitates the development of compact diffusion models, as current state-of-the-art models like Stable Diffusion are typically very large. To address this challenge, researchers have explored various compression techniques, primarily focusing on network pruning and quantization methods.

Network pruning involves selectively removing weights from the network and subsequently fine-tuning the pruned model to maintain performance comparable to the pre-trained version. Diff-Pruning [7] utilizes Taylor expansion over pruned timesteps to identify non-contributory diffusion steps and important weights through informative gradients. BK-SDM [20] discovers that block pruning combined with feature distillation is an efficient and sufficient strategy for obtaining lightweight models.

Another line of work is model quantization, which aims to reduce the storage and computational requirements of diffusion models during deployment [11,15,26,27,50,57]. Q-diffusion [26] introduces timestep-aware calibration and split short-cut quantization, tailoring post-training quantization (PTQ) methods specifically for diffusion models. EfficientDM [11] proposes a quantization-aware variant of the low-rank adapter (QALoRA), achieving Quantization-Aware Training (QAT)-level performance with PTQ-like efficiency.

In summary, SlimFlow advances the state of the art by addressing the dual challenge of minimizing both inference steps and neural network size, with the ultimate goal of developing the most efficient diffusion models. While our approach shares some similarities with MobileDiffusion [68], which also explores efficient structure design and acceleration of diffusion models, SlimFlow distinguishes itself in several key aspects: (1) SlimFlow is built upon the rectified flow framework, which provides more stable training dynamics than the GAN-based training used in MobileDiffusion; (2) Unlike MobileDiffusion's focus on fine-grained specific network structure optimization, SlimFlow offers a general framework applicable to a wide range of efficient network architectures. This flexibility allows our approach to leverage advancements in efficient network design across the field, including MobileDiffusion's efficient text-to-image network.

6 Limitations and Future Works

While our approach demonstrates advancements in efficient one-step diffusion models, we acknowledge several limitations and areas for future research. The quality of our one-step model is inherently bounded by the capabilities of the teacher models, specifically the quality of synthetic data pairs they generate. This limitation suggests a direct relationship between teacher model performance and the potential of our approach. In the future, we will leverage more advanced teacher models and real datasets. Besides, we plan to extend our method to more network architectures, e.g., transformers and pruned networks. Finally, given sufficient computational resources, we aim to apply our approach to more complex diffusion models, such as Stable Diffusion.

7 Conclusions

In this paper, we introduced SlimFlow, an innovative approach to developing efficient one-step diffusion models. Our method can significantly reduce model

complexity while preserving the quality of one-step image generation, as evidenced by our results on CIFAR-10, FFHQ, and ImageNet datasets. This work paves the way for faster and more resource-efficient generative modeling, broadening the potential for real-world applications of diffusion models.

Acknowledgements. We would like to thank Zewen Shen for his insightful discussion during this work.

References

1. Berthelot, D., et al.: Tract: denoising diffusion models with transitive closure time-distillation. arXiv preprint arXiv:2303.04248 (2023)
2. Crowson, K., Baumann, S.A., Birch, A., Abraham, T.M., Kaplan, D.Z., Shippole, E.: Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers. arXiv preprint arXiv:2401.11605 (2024)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)
4. Dhariwal, P., Nichol, A.: Diffusion models beat GANs on image synthesis. In: Advances in Neural Information Processing Systems, vol. 34, pp. 8780–8794 (2021)
5. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real NVP. arXiv preprint arXiv:1605.08803 (2016)
6. Dockhorn, T., Vahdat, A., Kreis, K.: Genie: higher-order denoising diffusion solvers. In: Advances in Neural Information Processing Systems, vol. 35, pp. 30150–30166 (2022)
7. Fang, G., Ma, X., Wang, X.: Structural pruning for diffusion models. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
8. Geyer, M., Bar-Tal, O., Bagon, S., Dekel, T.: TokenFlow: consistent diffusion features for consistent video editing. arXiv preprint arXiv:2307.10373 (2023)
9. Goodfellow, I., et al.: Generative adversarial networks. Commun. ACM **63**(11), 139–144 (2020)
10. Gu, J., Zhai, S., Zhang, Y., Liu, L., Susskind, J.: Boot: data-free distillation of denoising diffusion models with bootstrapping. arXiv preprint arXiv:2306.05544 (2023)
11. He, Y., Liu, J., Wu, W., Zhou, H., Zhuang, B.: EfficientDM: efficient quantization-aware fine-tuning of low-bit diffusion models. arXiv preprint arXiv:2310.03270 (2023)
12. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
13. Ho, J., et al.: Imagen video: high definition video generation with diffusion models. arXiv preprint arXiv:2210.02303 (2022)
14. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems, vol. 33, pp. 6840–6851 (2020)
15. Huang, Y., Gong, R., Liu, J., Chen, T., Liu, X.: TFMQ-DM: temporal feature maintenance quantization for diffusion models. arXiv preprint arXiv:2311.16503 (2023)
16. Hyvärinen, A., Dayan, P.: Estimation of non-normalized statistical models by score matching. J. Mach. Learn. Res. **6**(4) (2005)

17. Jeong, M., Kim, H., Cheon, S.J., Choi, B.J., Kim, N.S.: Diff-TTS: a denoising diffusion model for text-to-speech. arXiv preprint arXiv:2104.01409 (2021)
18. Karras, T., Aittala, M., Aila, T., Laine, S.: Elucidating the design space of diffusion-based generative models. arXiv preprint arXiv:2206.00364 (2022)
19. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4401–4410 (2019)
20. Kim, B.K., Song, H.K., Castells, T., Choi, S.: BK-SDM: a lightweight, fast, and cheap version of stable diffusion. arXiv preprint arXiv:2305.15798 (2023)
21. Kim, D., et al.: Consistency trajectory models: learning probability flow ode trajectory of diffusion. arXiv preprint arXiv:2310.02279 (2023)
22. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
23. Kong, Z., Ping, W., Huang, J., Zhao, K., Catanzaro, B.: DiffWave: a versatile diffusion model for audio synthesis. arXiv preprint arXiv:2009.09761 (2020)
24. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
25. Li, W., et al.: Not all steps are equal: efficient generation with progressive diffusion models. arXiv preprint arXiv:2312.13307 (2023)
26. Li, X., et al.: Q-diffusion: quantizing diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 17535–17545 (2023)
27. Li, Y., Xu, S., Cao, X., Sun, X., Zhang, B.: Q-DM: an efficient low-bit quantized diffusion model. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
28. Lipman, Y., Chen, R.T., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling. arXiv preprint arXiv:2210.02747 (2022)
29. Liu, Q.: Rectified flow: a marginal preserving approach to optimal transport. arXiv preprint arXiv:2209.14577 (2022)
30. Liu, R., Wu, R., Van Hoorick, B., Tokmakov, P., Zakharov, S., Vondrick, C.: Zero-1-to-3: Zero-shot one image to 3D object. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9298–9309 (2023)
31. Liu, X., Gong, C., Liu, Q.: Flow straight and fast: learning to generate and transfer data with rectified flow. arXiv preprint arXiv:2209.03003 (2022)
32. Liu, X., Wu, L., Ye, M., Liu, Q.: Let us build bridges: understanding and extending diffusion generative models. arXiv preprint arXiv:2208.14699 (2022)
33. Liu, X., Zhang, X., Ma, J., Peng, J., Liu, Q.: Instaflow: one step is enough for high-quality diffusion-based text-to-image generation. arXiv preprint arXiv:2309.06380 (2023)
34. Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: DPM-solver: a fast ode solver for diffusion probabilistic model sampling in around 10 steps. In: Advances in Neural Information Processing Systems, vol. 35, pp. 5775–5787 (2022)
35. Luhman, E., Luhman, T.: Knowledge distillation in iterative generative models for improved sampling speed. arXiv preprint arXiv:2101.02388 (2021)
36. Luo, S., Tan, Y., Huang, L., Li, J., Zhao, H.: Latent consistency models: synthesizing high-resolution images with few-step inference. arXiv preprint arXiv:2310.04378 (2023)
37. Luo, W., Hu, T., Zhang, S., Sun, J., Li, Z., Zhang, Z.: Diff-instruct: a universal approach for transferring knowledge from pre-trained diffusion models. arXiv preprint arXiv:2305.18455 (2023)
38. Ma, X., Fang, G., Wang, X.: DeepCache: accelerating diffusion models for free. arXiv preprint arXiv:2312.00858 (2023)

39. Meng, C., et al.: On distillation of guided diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14297–14306 (2023)
40. Molad, E., et al.: Dreamix: video diffusion models are general video editors. arXiv preprint arXiv:2302.01329 (2023)
41. Nichol, A.Q., Dhariwal, P.: Improved denoising diffusion probabilistic models. In: International Conference on Machine Learning, pp. 8162–8171. PMLR (2021)
42. Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. J. Mach. Learn. Res. **22**(57), 1–64 (2021)
43. Parmar, G., Zhang, R., Zhu, J.Y.: On aliased resizing and surprising subtleties in GAN evaluation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11410–11420 (2022)
44. Pernias, P., Rampas, D., Richter, M.L., Pal, C., Aubreville, M.: Würstchen: an efficient architecture for large-scale text-to-image diffusion models. In: The Twelfth International Conference on Learning Representations (2023)
45. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: DreamFusion: text-to-3D using 2D diffusion. arXiv preprint arXiv:2209.14988 (2022)
46. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125 (2022)
47. Razavi, A., Van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with VQ-VAE-2. In: Advances in neural information processing systems, vol. 32 (2019)
48. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10684–10695 (2022)
49. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. arXiv preprint arXiv:2202.00512 (2022)
50. Shang, Y., Yuan, Z., Xie, B., Wu, B., Yan, Y.: Post-training quantization on diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1972–1981 (2023)
51. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International Conference on Machine Learning, pp. 2256–2265. PMLR (2015)
52. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)
53. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models. arXiv preprint arXiv:2303.01469 (2023)
54. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
55. Song, Y., Garg, S., Shi, J., Ermon, S.: Sliced score matching: a scalable approach to density and score estimation. In: Uncertainty in Artificial Intelligence, pp. 574–584. PMLR (2020)
56. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456 (2020)
57. Wang, C., Wang, Z., Xu, X., Tang, Y., Zhou, J., Lu, J.: Towards accurate data-free quantization for diffusion models. arXiv preprint arXiv:2305.18723 (2023)

58. Wang, H., Du, X., Li, J., Yeh, R.A., Shakhnarovich, G.: Score Jacobian chaining: lifting pretrained 2D diffusion models for 3D generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12619–12629 (2023)
59. Wang, Z., et al.: ProlificDreamer: high-fidelity and diverse text-to-3d generation with variational score distillation. arXiv preprint arXiv:2305.16213 (2023)
60. Wimbauer, F., et al.: Cache me if you can: accelerating diffusion models through block caching. arXiv preprint arXiv:2312.03209 (2023)
61. Wu, L., Gong, C., Liu, X., Ye, M., Liu, Q.: Diffusion-based molecule generation with informative prior bridges. In: Advances in Neural Information Processing Systems, vol. 35, pp. 36533–36545 (2022)
62. Xu, Y., Zhao, Y., Xiao, Z., Hou, T.: UFOGen: you forward once large scale text-to-image generation via diffusion GANs. arXiv preprint arXiv:2311.09257 (2023)
63. Yang, X., Zhou, D., Feng, J., Wang, X.: Diffusion probabilistic model made slim. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22552–22562 (2023)
64. Ye, S., Liu, F.: Score mismatching for generative modeling. arXiv preprint arXiv:2309.11043 (2023)
65. Yin, T., et al.: One-step diffusion with distribution matching distillation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6613–6623 (2024)
66. Zhang, Q., Chen, Y.: Fast sampling of diffusion models with exponential integrator. arXiv preprint arXiv:2204.13902 (2022)
67. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
68. Zhao, Y., Xu, Y., Xiao, Z., Hou, T.: MobileDiffusion: subsecond text-to-image generation on mobile devices. arXiv preprint arXiv:2311.16567 (2023)
69. Zheng, H., Nie, W., Vahdat, A., Azizzadenesheli, K., Anandkumar, A.: Fast sampling of diffusion models via operator learning. In: International Conference on Machine Learning, pp. 42390–42402. PMLR (2023)
70. Zhou, Z., Chen, D., Wang, C., Chen, C.: Fast ode-based sampling for diffusion models in around 5 steps. arXiv preprint arXiv:2312.00094 (2023)

Appendix

A Results of Annealing Reflow with Different Fixed β

We present the sampled images and the corresponding FID over 5k images of Annealing Reflow training of different fixed β using the same data pairs in Fig. 7. We test the results with four different β values: $\{0, 0.1, 0.3, 0.5\}$ and found that the RK45 samples are similar in four different models but the quality of one-step generated samples is much worse when β is large. This observation validates the marginal preserving property of the Annealing Reflow training.

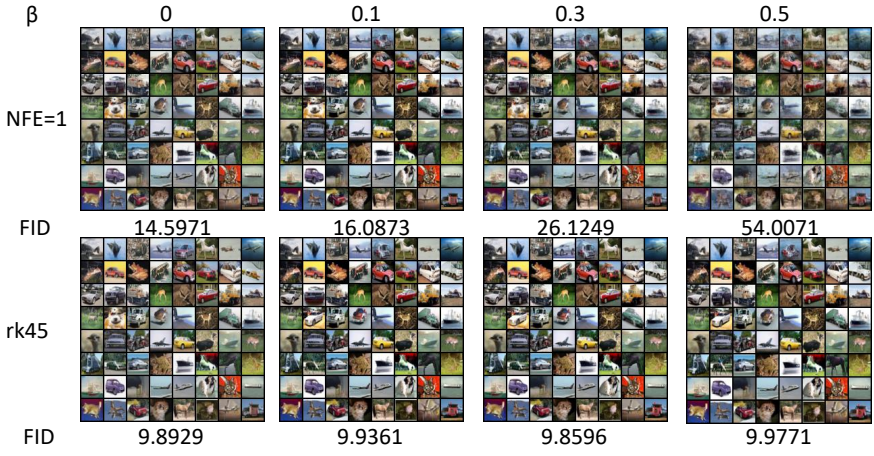

Fig. 7: Result on models with 15M parameters trained with different fixed β values. The FIDs are calculated with only 5k generated samples.

B More Experimental Details

In Tab. 6, we list all the architecture choices and related hyper-parameters in our experiments on CIFAR10 and FFHQ. In Tab. 7, we list all the architecture choices and related hyper-parameters in our experiments on ImageNet. The training of all the networks is smoothed by EMA with a ratio of 0.999999. Adam optimizer is adopted with a learning rate of $2e-4$ and the dropout rate is set to 0.15, following [29]. For the Annealing Reflow training, we use ℓ_2 loss with a uniform loss weight; for the distillation, we switch to the LPIPS loss. Most of the ablation experiments are conducted with configuration D using the data pairs from the 1-rectified flow teacher.

For the experiment on ImageNet, we also trained a model with configuration J in Tab. 7, which has almost half the parameters but similar MACs to configuration I. We get an FID of 8.86 from the 2-rectified flow with RK45 sampler

	A	B	C	D†	E	F	G
#Blocks	4	4	2	2	2	1	1
Base Channels	128	128	128	96	64	64	64
Channel Multiplier	(1, 2, 2, 2)	(2, 2, 2)	(1, 2, 2)	(1, 2, 2)	(1, 2, 2)	(1, 1, 2)	(1, 1)
Attention resolutions	(16,)	(16,)	(16,)	(16,)	(16,)	(16,)	(16,)
Batch Size	-	-	128	128	128	128	128
Batch Size (FFHQ distillation)	-	-	64	64	128	128	-
#Paras	61.8M	55.7M	27.9M	15.7M	7.0M	3.4M	1.2M
MACs	10.3G	20.7G	6.6G	3.7G	1.6G	0.7G	0.5G
FLOPs	22.0G	42.7G	13.9G	7.9G	3.6G	1.5G	1.2G

Table 6: Architecture configurations that are used in this work for CIFAR10 and FFHQ. † represents the default configuration for ablations. MACs and FLOPs are calculated with input shape $(1, 3, 32, 32)$.

		H	I	J
Architecture Configuration	#Blocks	3	2	2
	Base Channels	192	128	128
	Channel Multiplier	(1, 2, 3, 4)	(1, 2, 2, 4)	(1, 2, 2, 2)
	Attention resolutions	(32,16,8)	(32, 16)	(32, 16)
Training	Batch Size	-	96	96
	Batch Size (distillation)	-	64	64
Model Size	#Paras	259.9M	80.7M	44.7M
	MACs	103.4G	31.0G	28.1G
	FLOPs	219.4G	67.8G	61.9G

Table 7: Architecture configurations that are used in this work for the ImageNet dataset. MACs and FLOPs are calculated with input shape $(1, 3, 64, 64)$.

	Channel Multiplier	#Paras	MACs	FLOPs	FID (2-rectified flows)	FID (distilled flows)
I	(1, 2, 2, 4)	80.7M	31.0G	67.8G	8.89	12.34
J	(1, 2, 2, 2)	44.7M	28.1G	61.9G	8.86	12.52

Table 8: Comparison between two ImageNet experiments in this work.

(NFE≈40) and a final FID of 12.52 on the final one-step flow. The comparison is listed in Tab. 8.

For the Annealing Reflow training, we use 50k data pairs from the 1-rectified flow, 100k data pairs from EDM on CIFAR10 dataset, 200k data pairs from EDM on FFHQ dataset, and 400k data pairs from EDM on the ImageNet dataset. For distillation, we always simulate 500k data pairs from the 2-rectified flows.

For distillation, the loss after replacing the 2-rectified flow with the one-step model as mentioned in Sec. 4.1 is:

$$\mathcal{L}'_{\text{2-step}}(\phi') := \mathbb{E}_{\mathbf{x}_1 \sim \pi_1}\left[\int_0^1 \mathbb{D}(\mathbf{x}_1 - (1-t)\mathbf{v}_{\phi'}(\mathbf{x}_1, 1) - t\mathbf{v}_{\phi}(\mathbf{x}_t, t), \mathbf{x}_1 - \mathbf{v}_{\phi'}(\mathbf{x}_1, 1))\mathrm{d}t\right]. \tag{14}$$

where a stop-gradient operation is added to the first $\mathbf{v}_{\phi'}$. This will help the convergence of the training in practice and save one forward step of the 2-rectified flow.

All of the reference statistics for computing FID are from EDM [18]. All of the straightness is calculated using 100 Euler steps and averaging over 256 images, with the following Eq. (7).

C Additional Samples from SlimFlow

In this section, we provide some additional samples from our one-step models.

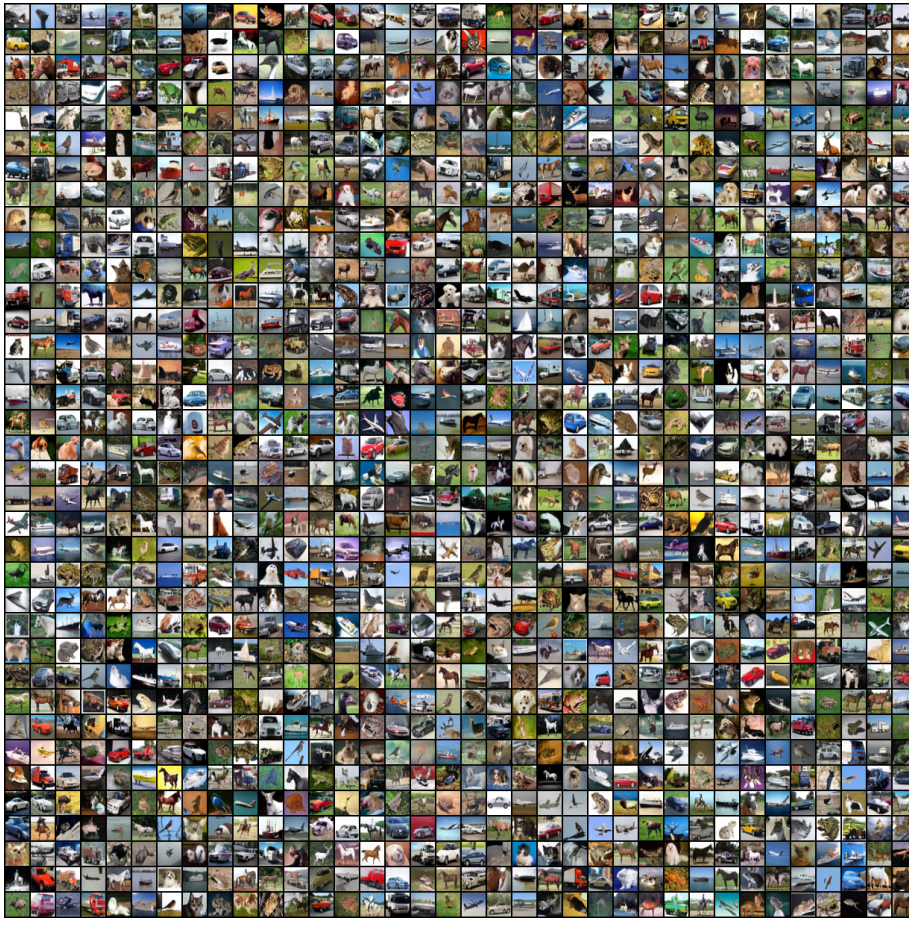

Fig. 8: Uncurated samples from unconditional CIFAR-10 32×32 using SlimFlow with single step generation (FID=4.53).

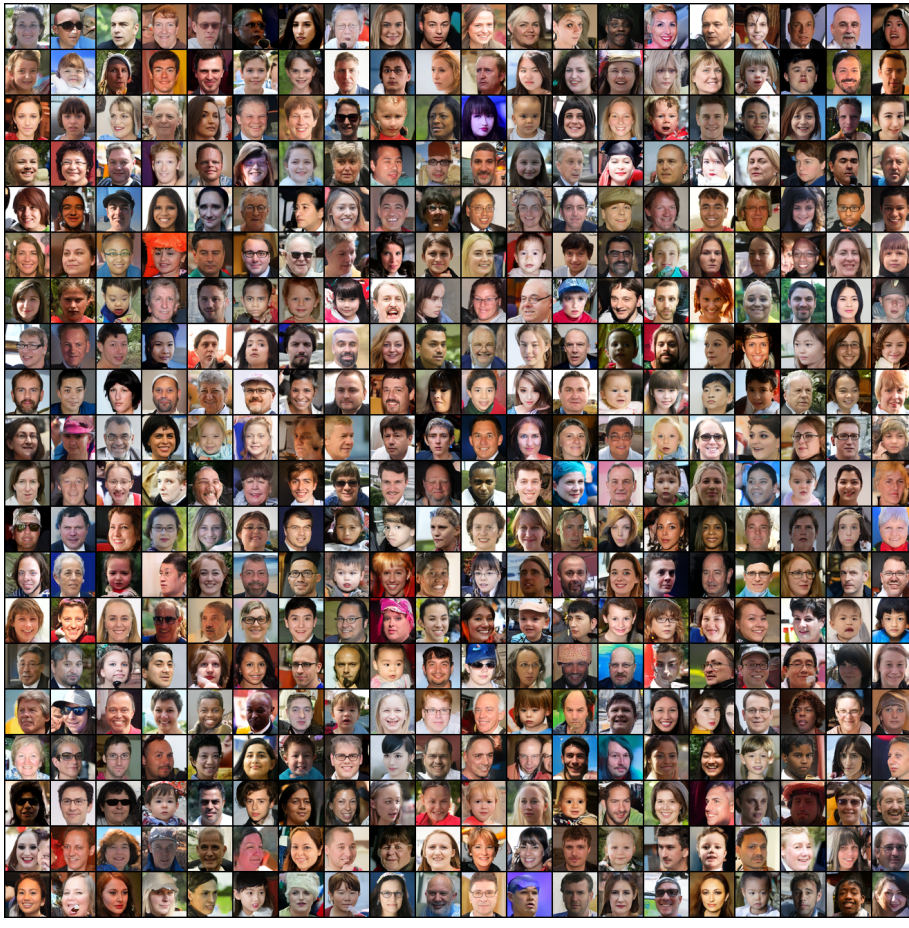

Fig. 9: Uncurated samples from unconditional FFHQ 64×64 using SlimFlow with single step generation (FID=7.21).

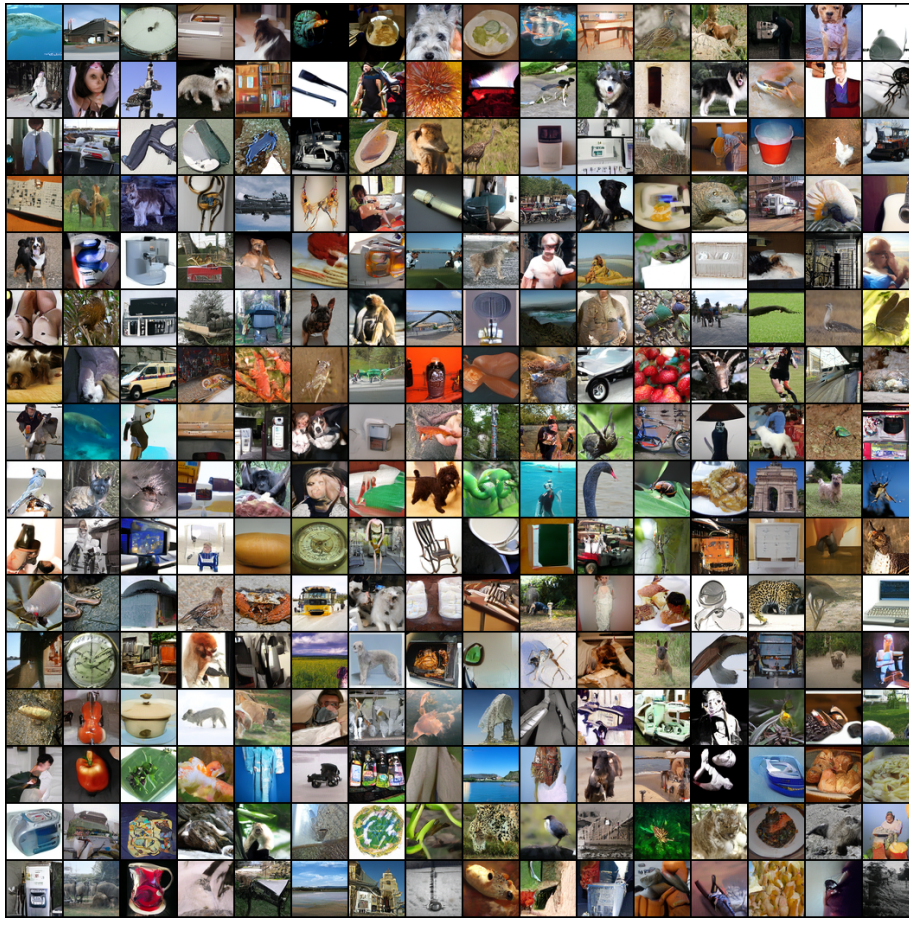

Fig. 10: Uncurated samples with random class labels from conditional ImageNet 64×64 using SlimFlow with single step generation (FID=12.34).

Fig. 11: Uncurated samples with three given classes from conditional ImageNet 64×64 using SlimFlow with single step generation (FID=12.34).