Improving the Training of Rectified Flows

Sangyun Lee
Carnegie Mellon University
sangyunl@andrew.cmu.edu

Zinan Lin
Microsoft Research
zinanlin@microsoft.com

Giulia Fanti
Carnegie Mellon University
gfanti@andrew.cmu.edu

Abstract

Diffusion models have shown great promise for image and video generation, but sampling from state-of-the-art models requires expensive numerical integration of a generative ODE. One approach for tackling this problem is rectified flows, which iteratively learn smooth ODE paths that are less susceptible to truncation error. However, rectified flows still require a relatively large number of function evaluations (NFEs). In this work, we propose improved techniques for training rectified flows, allowing them to compete with *knowledge distillation* methods even in the low NFE setting. Our main insight is that under realistic settings, a single iteration of the Reflow algorithm for training rectified flows is sufficient to learn nearly straight trajectories; hence, the current practice of using multiple Reflow iterations is unnecessary. We thus propose techniques to improve one-round training of rectified flows, including a U-shaped timestep distribution and LPIPS-Huber premetric. With these techniques, we improve the FID of the previous 2-rectified flow by up to 75% in the 1 NFE setting on CIFAR-10. On ImageNet 64×64, our improved rectified flow outperforms the state-of-the-art distillation methods such as consistency distillation and progressive distillation in both one-step and two-step settings and rivals the performance of improved consistency training (iCT) in FID. Code is available at https://github.com/sangyun884/rfpp.

1 Introduction

Diffusion models [Sohl-Dickstein et al., 2015, Ho et al., 2020, Song and Ermon, 2019, Song et al., 2020b] have shown great promise in image [Ramesh et al., 2022] and video [Ho et al., 2022] generation. They generate data by simulating a stochastic denoising process where noise is gradually transformed into data. To sample efficiently from diffusion models, the denoising process is typically converted into a counterpart of Ordinary Differential Equations (ODEs) [Song et al., 2020b] called probability flow ODEs (PF-ODEs).

Despite the success of diffusion models using PF-ODEs, drawing high-quality samples requires numerical integration of the PF-ODE with small step sizes, which is computationally expensive. Today, two prominent classes of approaches for tackling this issue are: (1) knowledge distillation (e.g., consistency distillation [Song et al., 2023], progressive distillation [Salimans and Ho, 2022]) and (2) simulation-free flow models (e.g., rectified flows [Liu et al., 2022], flow matching [Lipman et al., 2022]).

In knowledge distllation-based methods [Luhman and Luhman, 2021, Salimans and Ho, 2022, Song et al., 2023, Zheng et al., 2022] a student model is trained to directly predict the solution of the PF-ODE. These models are currently state-of-the-art in the low number of function evaluations (NFEs) regime (e.g. 1-4).

Another promising direction is simulation-free flow models such as rectified flows [Liu et al., 2022, Liu, 2022], a generative model that learns a transport map between two distributions defined via neural ODEs. Diffusion models with PF-ODEs are a special case. Rectified flows can learn smooth

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

arXiv:2405.20320v2 [cs.CV] 8 Oct 2024

ODE trajectories that are less susceptible to truncation error, which allows for high-quality samples with fewer NFEs than diffusion models. They have been shown to outperform diffusion models in the moderate to high NFE regime [Lipman et al., 2022, Liu et al., 2022, Esser et al., 2024], but they still require a relatively large number of NFEs compared to distillation methods.

Compared to knowledge distillation methods [Luhman and Luhman, 2021, Salimans and Ho, 2022, Song et al., 2023, Zheng et al., 2022] rectified flows have several advantages. First, they can be generalized to map two arbitrary distributions to one another, while distillation methods are limited to a Gaussian noise distribution. Also, as a neural ODE, rectified flows naturally support *inversion* from data to noise, which has many applications including image editing [Hertz et al., 2022, Kim et al., 2022, Wallace et al., 2023, Couairon et al., 2022, Mokady et al., 2023, Su et al., 2022, Hong et al., 2023] and watermarking [Wen et al., 2023]. Further, the likelihood of rectified flow models can be evaluated using the instantaneous change of variable formula [Chen et al., 2018], whereas this is not possible with knowledge distillation-based methods. In addition, rectified flows can flexibly adjust the balance between the sample quality and computational cost by altering NFEs, whereas distillation methods either do not support multi-step sampling or do not necessarily perform better with more NFEs (e.g. > 4) [Kim et al., 2023].

Given the qualitative advantages of rectified flows, a natural question is, **can rectified flows compete with distillation-based methods such as consistency models [Song et al., 2023] in the low NFE setting?** Today, the state-of-the-art techniques for training rectified flows use the *Reflow* algorithm to improve low NFE performance [Liu et al., 2022, 2023]. Reflow is a recursive training algorithm where the rectified flow is trained on data-noise pairs generated by the generative ODE of the previous stage model. In current implementations of Reflow, to obtain a reasonable one-step generative performance, Reflow should be applied at least twice, followed by an optional distillation stage to further boost performance [Liu et al., 2022, 2023]. Each training stage requires generating a large number of data-noise pairs and training the model until convergence, which is computationally expensive and leads to error accumulation across rounds. Even with these efforts, the generative performance of rectified flow still lags behind the distillation methods such as consistency models [Song et al., 2023].

We show that **rectified flows can indeed be competitive with the distillation methods in the low NFE setting by applying Reflow with our proposed training techniques**. Our techniques are based on the observation that under realistic settings, the linear interpolation trajectories of the pre-trained rectified flow rarely intersect with each other. This provides several insights: 1) applying Reflow once is sufficient to obtain straight-line generative ODE in the optima, 2) the training loss of 2-rectified flow has zero lower bound, and 3) other loss functions than the squared ℓ_2 distance can be used during training. Based upon this finding, we propose several training techniques to improve Reflow, including: (1) a U-shaped timestep distribution, (2) an LPIPS-huber premetric, which we find to be critical for the few-step generative performance. After being initialized with pre-trained diffusion models such as EDM [Karras et al., 2022], our method only requires one training stage without additional Reflow or distillation stages, unlike previous works [Liu et al., 2022, 2023].

Our evaluation shows that on several datasets (CIFAR-10 [Krizhevsky et al., 2009], ImageNet 64×64 [Deng et al., 2009]), our improved rectified flow outperforms the state-of-the-art distillation methods such as consistency distillation (CD) [Song et al., 2023] and progressive distillation (PD) [Salimans and Ho, 2022] in both one-step and two-step settings, and it rivals the performance of the improved consistency training (iCT) [Song et al., 2023] in terms of the Frechet Inception Distance [Heusel et al., 2017] (FID). Our training techniques **reduce the FID of the previous 2-rectified flow [Liu et al., 2022] by about 75**% ($12.21 \rightarrow 3.07$) on CIFAR-10. Ablations on three datasets show that the proposed techniques give a consistent and sizeable gain. We also showcase the qualitative advantages of rectified flow such as few-step inversion, and its application to interpolation and image-to-image translation.

2 Background

2.1 Rectified Flow

Rectified flow (see also flow matching [Lipman et al., 2022] and stochastic interpolant [Albergo and Vanden-Eijnden, 2022]) is a generative model that smoothly transitions between two distributions $p_{\mathbf{x}}$ and $p_{\mathbf{z}}$ by solving ODEs [Liu et al., 2022]. For $\mathbf{x} \sim p_{\mathbf{x}}$ and $\mathbf{z} \sim p_{\mathbf{z}}$, we define the interpolation between $\mathbf{x}$ and $\mathbf{z}$ as $\mathbf{x}_t = (1 - t)\mathbf{x} + t\mathbf{z}$ for $t \in [0, 1]$. Liu et al. [2022] showed that for $\mathbf{z}_0 \sim p_{\mathbf{x}}$, the

2

(a) Linear interpolation induced by $p_{\mathbf{xz}}^0$ (b) Generative process of 1-rectified flow (c) Linear interpolation induced by $p_{\mathbf{xz}}^1$ (d) Generative process of 2-rectified flow
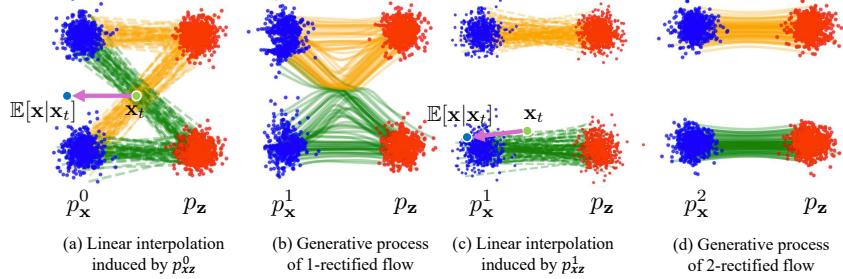
Figure 1: Rectified flow process (figure modified from Liu et al. [2022]). Rectified flow rewires trajectories so there are no intersecting trajectories $(a) \to (b)$. Then, we take noise samples from $p_{\mathbf{z}}$ and their generated samples from $p_{\mathbf{x}}^1$, and linearly interpolate them (c). In Reflow, rectified flow is applied again $(c) \to (d)$ to straighten flows. This procedure is repeated recursively.

following ODE yields the same marginal distribution as $\mathbf{x}_t$ for any t:

$$\frac{d\mathbf{z}_t}{dt} = \mathbf{v}_t(\mathbf{z}_t) := \frac{1}{t}(\mathbf{z}_t - \mathbb{E}[\mathbf{x}|\mathbf{x}_t = \mathbf{z}_t]). \tag{1}$$

Since $\mathbf{x}_1 = \mathbf{z}$, Eq. (1) transports $p_{\mathbf{x}}$ to $p_{\mathbf{z}}$. We can also transport $p_{\mathbf{z}}$ to $p_{\mathbf{x}}$ by drawing $\mathbf{z}_1$ from $p_{\mathbf{z}}$ and solving the ODE backwards from $t = 1$ to $t = 0$. During training, we estimate the conditional expectation $\mathbb{E}[\mathbf{x}|\mathbf{x}_t = \mathbf{z}_t]$ with a vector-valued neural network $\mathbf{x}_{\boldsymbol{\theta}}$ trained on the squared ℓ_2 loss:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x},\mathbf{z}\sim p_{\mathbf{xz}}} \mathbb{E}_{t\sim p_t}[\omega(t)||\mathbf{x} - \mathbf{x}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)||_2^2], \tag{2}$$

where $p_{\mathbf{xz}}$ is the joint distribution of $\mathbf{x}$ and $\mathbf{z}$, $\mathbf{x}_{\boldsymbol{\theta}}$ is parameterized by $\boldsymbol{\theta}$, and $\omega(t)$ is a weighting function. p_t is chosen to be the uniform distribution on $[0, 1]$ in Liu et al. [2022, 2023]. In the optimum of Eq. (2), $\mathbf{x}_{\boldsymbol{\theta}}$ becomes the conditional expectation as it is a minimum mean squared error (MMSE) estimator, which is then plugged into the ODE (1) to generate samples. Instead of predicting the conditional expectation directly, Liu et al. [2022] choose to parameterize the velocity $\mathbf{v}_t$ with a neural network $\mathbf{v}_{\boldsymbol{\theta}}$ and train it on

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x},\mathbf{z}\sim p_{\mathbf{xz}}} \mathbb{E}_{t\sim p_t}[||(\mathbf{z} - \mathbf{x}) - \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)||_2^2], \tag{3}$$

which is equivalent to Eq. (2) with $\omega(t) = \frac{1}{t^2}$. See Appendix. A.

In this paper, we consider the Gaussian marginal case, i.e., $p_{\mathbf{z}} = \mathcal{N}(\mathbf{0}, \mathbf{I})$. In this case, if we define $\mathbf{x}$ and $\mathbf{z}$ as independent random variables (i.e., $p_{\mathbf{xz}}(\mathbf{x}, \mathbf{z}) = p_{\mathbf{x}}(\mathbf{x})p_{\mathbf{z}}(\mathbf{z})$) and use a specific nonlinear interpolation instead of the linear interpolation for $\mathbf{x}_t$, Eq. (2) becomes the weighted denoising objective of the diffusion model [Vincent, 2011], and Eq. (1) becomes the probability flow ODE (PF-ODE) [Song et al., 2020b].

2.2 Reflow

Algorithm 1 Reflow Procedure

1: **First iteration:**
2: $\quad \boldsymbol{\theta}_1 = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x},\mathbf{z}\sim p_{\mathbf{xz}}^0} \mathbb{E}_{t\sim p_t}\left[\omega(t)\|\mathbf{x} - \mathbf{x}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\|_2^2\right]$ $\qquad\qquad$ ▷ Train 1-rectified flow
3: **for** $k = 1$ to $K - 1$ **do**
4: $\quad T^k(\mathbf{z}) = \mathbf{z} + \int_1^0 \frac{1}{t}(\mathbf{z}_t - \mathbf{x}_{\boldsymbol{\theta}_k}(\mathbf{z}_t, t))\, dt$ with $\mathbf{z}_1 = \mathbf{z}$
5: $\quad p_{\mathbf{xz}}^k(\mathbf{x}, \mathbf{z}) = p_{\mathbf{z}}(\mathbf{z})\delta\left(\mathbf{x} - T^k(\mathbf{z})\right)$ $\qquad\qquad$ ▷ Generate synthetic pairs for next coupling
6: $\quad \boldsymbol{\theta}_{k+1} = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x},\mathbf{z}\sim p_{\mathbf{xz}}^k} \mathbb{E}_{t\sim p_t}\left[\omega(t)\|\mathbf{x} - \mathbf{x}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\|_2^2\right]$ $\quad$ ▷ Train $(k+1)$-rectified flow
7: **end for**

The independent coupling $p_{\mathbf{xz}}(\mathbf{x}, \mathbf{z}) = p_{\mathbf{x}}(\mathbf{x})p_{\mathbf{z}}(\mathbf{z})$ is known to lead to curved ODE trajectories, which require a large number of function evaluations (NFE) to generate high-quality samples [Pooladian et al., 2023, Lee et al., 2023]. Reflow [Liu et al., 2022] is a recursive training algorithm to find a better coupling that yields straighter ODE trajectories. Starting from the independent coupling $p_{\mathbf{xz}}^0(\mathbf{x}, \mathbf{z}) = p_{\mathbf{x}}(\mathbf{x})p_{\mathbf{z}}(\mathbf{z})$, the Reflow algorithm generates $p_{\mathbf{xz}}^{k+1}(\mathbf{x}, \mathbf{z})$ from $p_{\mathbf{xz}}^k(\mathbf{x}, \mathbf{z})$ by first

3
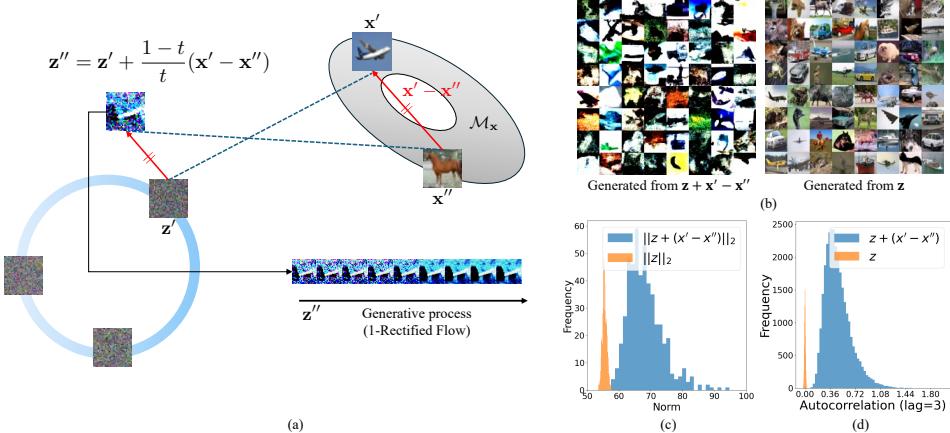
Figure 2: An illustration of the intuition in Sec. 3. (a) If two linear interpolation trajectories intersect, $\mathbf{z}'' - \mathbf{z}'$ is parallel to $\mathbf{x}' - \mathbf{x}''$. This generally maps $\mathbf{z}''$ to an atypical (e.g., one with high autocorrelation or a norm that is too large to be on a Gaussian annulus) realization of Gaussian noise, so the 1-rectified flow cannot reliably map $\mathbf{z}''$ to $\mathbf{x}''$ on $\mathcal{M}_\mathbf{x}$. (b) Generated samples from the pre-trained 1-rectified flow starting from $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ (*right*), which is the standard setting, and $\mathbf{z}'' = \mathbf{z} + (\mathbf{x}' - \mathbf{x}'')$, where $\mathbf{x}', \mathbf{x}''$ are sampled from 1-rectified flow trained on CIFAR-10 (*left*). Qualitatively, we see that the left samples have very low quality. (c) Empirically, we show the ℓ_2 norm of $\mathbf{z}'' = \mathbf{z} + (\mathbf{x}' - \mathbf{x}'')$ compared to z', which is sampled from the standard Gaussian. $\mathbf{z}''$ generally lands outside the annulus of typical Gaussian noise. (d) $\mathbf{z} + (\mathbf{x}' - \mathbf{x}'')$ has high autocorrelation while the autocorrelation of Gaussian noise is nearly zero in high-dimensional space.

generating synthetic $(\mathbf{x}, \mathbf{z})$ pairs from $p_{\mathbf{xz}}^k$, then training rectified flow on the generated synthetic pairs (Figure 1$(b) - (d)$). We call the vector field resulting from the k-th iteration of this procedure k-rectified flow. Pseudocode for Reflow is provided in Algorithm. 1.

Convergence: Liu et al. [2022] show that Reflow trajectories are straight in the limit as $K \rightarrow \infty$. Hence, to achieve perfectly straight ODE paths that allow for accurate one-step generation, Reflow may need to be applied many times until equilibrium, with each training stage requiring many data-noise pairs, training the model until convergence, and a degradation in generated sample quality. **Prior work has empirically found that Reflow should be applied at least twice** (i.e. 3-rectified flow) for reasonably good one-step generative performance [Liu et al., 2022, 2023]. This has been a major downside for rectified flows compared to knowledge distillation methods, which typically require only one distillation stage [Luhman and Luhman, 2021, Song et al., 2023, Zheng et al., 2022].

3   Applying Reflow Once is Sufficient

In this section, we argue that under practical settings, the trajectory curvature of the optimal 2-rectified flow is actually close to zero. Hence, prior empirical results requiring more rounds of Reflow may be the result of suboptimal training techniques, and we should focus on improving those training techniques rather than stacking additional Reflow stages.

First, note that the curvature of the optimal 2-rectified flow is zero if and only if the linear interpolation trajectories of 1-rectified flow-generated pairs do not intersect, or equivalently, $\mathbb{E}[\mathbf{x}|\mathbf{x}_t = (1-t)\mathbf{x}' + t\mathbf{z}'] = \mathbf{x}'$ for all pairs $(\mathbf{x}', \mathbf{z}')$ [Liu et al., 2022].

To begin with, consider the manifold $\mathcal{M}_\mathbf{x}$ of the synthetic distribution $p^1(\mathbf{x}) = \int p^1(\mathbf{x}, \mathbf{z})d\mathbf{z}$. Consider two points $\mathbf{x}'$ and $\mathbf{x}''$ from the manifold, and two noises $\mathbf{z}'$ and $\mathbf{z}''$ that are mapped to $\mathbf{x}'$ and $\mathbf{x}''$ by 1-rectified flow. Here, we say two pairs $(\mathbf{x}', \mathbf{z}')$ and $(\mathbf{x}'', \mathbf{z}'')$ intersect if $\exists t \in [0, 1]$ s.t. $(1-t)\mathbf{x}' + t\mathbf{z}' = (1-t)\mathbf{x}'' + t\mathbf{z}''$. For example, in Figure 2(a), we observe that the two trajectories intersect at an intermediate t.

For an intersection to exist at t it must hold that 1) 1-rectified flow maps $\mathbf{z}''$ to $\mathbf{x}''$, and 2) $\mathbf{z}'' = \mathbf{z}' + \frac{1-t}{t}(\mathbf{x}' - \mathbf{x}'')$ by basic geometry. However, note that for realistic data distributions and if

4

1-rectified flow is sufficiently well-trained, $\mathbf{z}'' = \mathbf{z}' + \frac{1-t}{t}(\mathbf{x}' - \mathbf{x}'')$ is not a common noise realization (e.g., it is likely to have nonzero autocorrelation or a norm that is too large to be on a Gaussian annulus), as shown visually in Figure 2(a). As 1-rectified flow is almost entirely trained on common Gaussian noise inputs, it cannot generally map an atypical $\mathbf{z}''$ to $\mathcal{M}_{\mathbf{x}}$. Figure 2(c) shows qualitatively that if we draw values of $\mathbf{z}''$ by first drawing $\mathbf{z}' \sim \mathcal{N}(0, I)$ and then adding $(\mathbf{x}' - \mathbf{x}'')$ for independent draws of $\mathbf{x}', \mathbf{x}''$, the $\mathbf{z}''$ vectors fall outside the annulus of typical standard Gaussian noise. Similarly, Figure 2(d) shows that the constructed noise vectors $\mathbf{z}''$ have higher autocorrelation than expected. As a result, Figure 2(b) visually shows that the generated samples have little overlap with the expected samples from typical draws of $\mathbf{z}'$.

This suggests empirically that when training 2-rectified flow, intersections are rare (i.e. $\mathbb{E}[\mathbf{x}|\mathbf{x}_t = (1-t)\mathbf{x}' + t\mathbf{z}'] \approx \mathbf{x}'$), which in turn implies that the optimal 2-rectified flow trajectories are nearly straight. Hence, additional rounds of Reflow are unnecessary, while also degrading sample quality. This intuition allows us to focus on better training techniques for 2-rectified flow rather than training 3- or 4-rectified flow. It also leads us to several improved techniques, discussed in Sec. 4.

Edge cases: Note that if $||\mathbf{x}' - \mathbf{x}''||_2$ is small, 1-rectified flow could map $\mathbf{z}''$ to some point on $\mathcal{M}_{\mathbf{x}}$. However, it does not alter the conclusion because the average of $\mathbf{x}'$ and $\mathbf{x}''$ is close to $\mathbf{x}'$ anyway, so $\mathbb{E}[\mathbf{x}|\mathbf{x}_t = (1-t)\mathbf{x}' + t\mathbf{z}'] \approx \mathbf{x}'$. Similarly, if t is close to 1, $\frac{1-t}{t}(\mathbf{x}' - \mathbf{x}'') \approx \mathbf{0}$, so 1-rectified flow can map $\mathbf{z}''$ to $\mathcal{M}_{\mathbf{x}}$. If the 1-rectified flow is L-Lipschitz, $||\mathbf{x}' - \mathbf{x}''||_2 \leq L||\mathbf{z}' - \mathbf{z}''||_2$. Therefore, the expectation $\mathbb{E}[\mathbf{x}|\mathbf{x}_t]$ again will not deviate much from $\mathbf{x}'$.

4 Improved Training Techniques for Reflow

The observation in Sec. 3 suggests that the optimal 2-rectified flow is nearly straight. Therefore, if the one-step generative performance of the 2-rectified flow model is not as good as expected, it is likely due to suboptimal training. In this section, we show that the few-step generative performance of the 2-rectified flow can be significantly improved by applying several new training techniques.

4.1 Timestep distribution

As in diffusion models, rectified flows are trained on randomly sampled timesteps t, and the distribution from which t is sampled is an important design choice. Ideally, we want to focus the training effort on timesteps that are more challenging rather than wasting computational resources on easy tasks. One common approach is to focus on the tasks where the training loss is high [Shrivastava et al., 2016]. However, the training error of rectified flows is not a reliable measure of difficulty because different timesteps have different non-zero lower bounds. To understand this, let us decompose the training error into two terms:

$$\mathcal{L}(\boldsymbol{\theta}, t) := \frac{1}{t^2}\mathbb{E}[||\mathbf{x} - \mathbf{x}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)||_2^2] = \underbrace{\frac{1}{t^2}\mathbb{E}[||\mathbf{x} - \mathbb{E}[\mathbf{x}|\mathbf{x}_t]||_2^2]}_{\text{Lower bound}} + \bar{\mathcal{L}}(\boldsymbol{\theta}, t). \tag{4}$$

The first term does not depend on $\boldsymbol{\theta}$ and thus cannot be reduced. The second term represents the actual minimizable training error, but its value cannot be directly observed because the first term is usually unknown. Fortunately, because of the finding in Sec. 3, we expect that the first term is nearly zero when training 2-rectified flow, so we can use $\bar{\mathcal{L}}(\boldsymbol{\theta}, t)$ for designing the timestep distribution.

Figure 3 shows that the training loss of 2-rectified flow is large at each end of the interval $t \in [0, 1]$ and small in the middle. We thus propose to use a U-shaped timestep distribution for p_t. Specifically, we define $p_t(u) \propto \exp(au) + \exp(-au)$ on $u \in [0, 1]$. We find that $a = 4$ works well in practice (Table 1). Compared to the uniform timestep distribution (config B), the U-shaped distribution (config C) improves
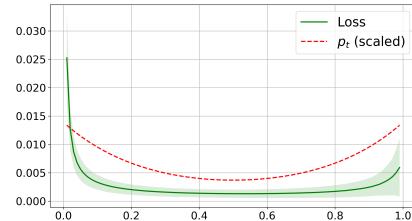
Figure 3: Training loss of the vanilla 2-rectified flow on CIFAR-10 measured on $5,000$ samples after $200,000$ iterations. The shaded area represents the 1 standard deviation of the loss. The dashed curve is our U-shaped timestep distribution, scaled by a constant factor for visualization.

5

Table 1: Effects of the improved training techniques. The baseline (config A) is the 2-rectified flow with the uniform timestep distribution and the squared ℓ_2 metric [Liu et al., 2022]. Config B is the improved baseline with EDM initialization (Sec. 4.3) and increased batch size ($128 \rightarrow 512$ on CIFAR-10). FID (the lower the better) is computed using $50,000$ synthetic samples and the entire training set. We train the models for $800,000$ iterations on CIFAR-10 and $1,000,000$ iterations on AFHQ and FFHQ and report the best FID for each setting.

	CIFAR-10		AFHQ 64×64		FFHQ 64×64	
Base [Liu et al., 2022] (A)	12.21	-	-	-	-	-
(A) + EDM init + larger batch size (B)	7.14	3.61	12.39	4.16	8.84	4.79
(B) + Our p_t (C)	5.17	3.37	9.03	3.61	6.81	4.66
(C) + Huber (D)	5.24	3.34	8.20	3.55	7.06	4.79
(C) + LPIPS-Huber (E)	3.42	2.95	4.13	3.15	**5.21**	**4.26**
(C) + LPIPS-Huber-$\frac{1}{t}$ (F)	3.38	2.76	**4.11**	**3.12**	5.65	4.41
(F) + Incorporating real data (G)	**3.07**	**2.40**	-	-	-	-
NFE	1	2	1	2	1	2

the FID of 2-rectified flow from 7.14 to 5.17 (a 28% improvement) on CIFAR-10, 12.39 to 9.03 (27%) on AFHQ, and 8.84 to 6.81 (23%) on FFHQ in the one-step setting.

For 1-rectified flow training, p_t was chosen to be the uniform distribution [Liu et al., 2022, 2023] or logit-normal distribution [Esser et al., 2024] which puts more emphasis on the middle of the interval. When training 1-rectified flow, a model learns to simply output the dataset average when $t = 1$ and the noise average (i.e., zero) when $t = 0$. The meaningful part of the training thus happens in the middle of the interval. In contrast, from Eq. (3) we can see that 2-rectified flow learns to directly predict the data from the noise at $t = 1$ and the noise from the data at $t = 0$, which are nontrivial tasks. Therefore, the U-shaped timestep distribution is more suitable for 2-rectified flow.

4.2 Loss function

Previously, the squared ℓ_2 distance was used as the training metric for rectified flow to obtain the MMSE estimator $\mathbb{E}[\mathbf{x}|\mathbf{x}_t]$. However, as we have shown in Sec. 3 that $\mathbb{E}[\mathbf{x}|\mathbf{x}_t = (1-t)\mathbf{x}' + t\mathbf{z}'] \approx \mathbf{x}'$, we can generalize Eq. (2) or equivalently Eq. (3) to any premetric m (i.e. $m(\mathbf{a}, \mathbf{b}) = 0 \Leftrightarrow \mathbf{a} = \mathbf{b}$):

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}, \mathbf{z} \sim p_{\mathbf{xz}}} \mathbb{E}_{t \sim p_t} [m(\mathbf{z} - \mathbf{x}, \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t))], \tag{5}$$

Note that **without the intuition in Sec. 3, only the squared ℓ_2 distance would have been a valid premetric**, as any other premetric makes the model deviate from the intended optimum (the posterior expectation $\mathbb{E}[\mathbf{x}|\mathbf{x}_t]$). Although the choice of m does not affect the optimum, it does affect the training dynamics and thus the obtained model. Other than the squared ℓ_2 distance, we consider the following premetrics:

- Pseudo-Huber [Charbonnier et al., 1997, Song and Dhariwal, 2023]: $m_{\text{hub}}(\mathbf{z} - \mathbf{x}, \mathbf{v}_{\boldsymbol{\theta}}((\mathbf{x}_t, t)) = \sqrt{||\mathbf{z} - \mathbf{x} - \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)||_2^2 + c^2} - c$, where $c = 0.00054d$ with d being data dimensionality.
- LPIPS-Huber: $m_{\text{lp-hub}}(\mathbf{z} - \mathbf{x}, \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)) = (1-t)m_{\text{hub}}(\mathbf{z} - \mathbf{x}, \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)) + \text{LPIPS}(\mathbf{x}, \mathbf{x}_t - t \cdot \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t))$, where $\text{LPIPS}(\cdot, \cdot)$ is the learned perceptual image patch similarity [Zhang et al., 2018].
- LPIPS-Huber-$\frac{1}{t}$: $m_{\text{lp-hub-}\frac{1}{t}}(\mathbf{z} - \mathbf{x}, \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)) = (1-t)m_{\text{hub}}(\mathbf{z} - \mathbf{x}, \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)) + \frac{1}{t}\text{LPIPS}(\mathbf{x}, \mathbf{x}_t - t \cdot \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t))$,

The Pseudo-Huber loss is less sensitive to the outliers than the squared ℓ_2 loss, which can potentially reduce the gradient variance [Song and Dhariwal, 2023] and make training easier. In our initial experiments, we found that the Pseudo-Huber loss tends to work better than the squared ℓ_2 loss with a small batch size (e.g. 128 on CIFAR-10). When the batch size is sufficiently large, it performs on par with the squared ℓ_2 loss on CIFAR-10 and FFHQ-64 and outperforms it on AFHQ-64, as shown in Table 1. As it is less sensitive to the batch size, we choose to use the Pseudo-Huber loss in the following experiments.

We also explore the LPIPS, which forces the model to focus on reducing the perceptual distance between the generated data and the ground truth. Since LPIPS is not a premetric as two different

6

Table 2: The converted time and scale for the variance preserving (VP) and variance exploding (VE) diffusion models. Here, $\alpha(t) = \exp(-\frac{1}{2}\int_0^t (19.9s + 0.1)ds)$ following Song et al. [2020b], and the perturbation kernel of the VE diffusion is $\mathcal{N}(\mathbf{x}, t^2\mathbf{I})$ following Karras et al. [2022].

t_{VP}	t_{VE}	s_{VP}	s_{VE}
$\frac{1}{9.95}\left(-0.05 + \sqrt{0.0025 - 19.9 \cdot \ln \frac{1-t}{\sqrt{(1-t)^2 + t^2}}}\right)$	$\frac{t}{1-t}$	$\frac{\alpha(t_{\text{VP}})}{1-t}$	$\frac{1}{1-t}$

points could have zero LPIPS if they are perceptually similar, we use it in combination with the Pseudo-Huber loss with the weighting $1 - t$, thereby relying more on LPIPS when t is close to 1 where the task is more challenging. Note that in $m_{\text{lp-hub}}$, the gradient vanishes when t is close to zero. To compsensate, we experiment with $m_{\text{lp-hub-}\frac{1}{t}}$ where we multiply LPIPS by $\frac{1}{t}$. Compared to config D, the LPIPS-Huber loss improves the FID of 2-rectified flow from 5.24 to 3.38 (a 35% improvement) on CIFAR-10, 8.20 to 4.11 (50%) on AFHQ, and 7.06 to 5.21 (26%) on FFHQ in the one-step setting, as seen in Table 1.

4.3 Initialization with pre-trained diffusion models

Training 1-rectified flow from scratch is computationally expensive. Recently, Pokle et al. [2023] showed that pre-trained diffusion models can be used to approximate $\mathbb{E}[\mathbf{x}|\mathbf{x}_t = \mathbf{z}_t]$ in Eq. (1) by adjusting the signal-to-noise ratio. The following proposition is the special cases of Lemma 2 of Pokle et al. [2023] restated with extended proof and a minor fix. We provide the constants and proof in Appendix. C.1.

Proposition 1 *Let $p^{RE}(\mathbf{x}|\mathbf{x}_t, t)$ be the posterior distribution of the perturbation kernel $\mathcal{N}((1 - t)\mathbf{x}, t^2\mathbf{I})$. Also, let $p^{VP}(\mathbf{x}|\mathbf{x}_t, t)$ and $p^{VE}(\mathbf{x}|\mathbf{x}_t, t)$ be the posterior distributions of $\mathcal{N}(\alpha(t)\mathbf{x}, (1 - \alpha(t))^2\mathbf{I})$ and $\mathcal{N}(\mathbf{x}, t^2\mathbf{I})$, each. Then,*

$$\int p^{RE}(\mathbf{x}|\mathbf{x}_t = \mathbf{z}_t, t)\mathbf{x}\, d\mathbf{x} = \int p^{VP}(\mathbf{x}|\mathbf{x}_t = s_{VP}\mathbf{z}_t, t_{VP})\mathbf{x}\, d\mathbf{x} = \int p^{VE}(\mathbf{x}|\mathbf{x}_t = s_{VP}\mathbf{z}_t, t_{VE})\mathbf{x}\, d\mathbf{x},$$

(6)

where s_{VP} and s_{VE} are the scaling factors and t_{VP} and t_{VE} are the converted times for the VP and VE diffusion models, respectively.

We have explicitly computed the time and scale conversion factors for the VP and VE diffusion models in Table 2. See Appendix C for derivation.

Proposition 1 allows us to initialize the Reflow with the pre-trained diffusion models such as EDM [Karras et al., 2022] or DDPM [Ho et al., 2020] and use Table 2 to adjust the time and scaling factors.

Starting from the vanilla 2-rectified flow setup [Liu et al., 2022] (config A), we initialize 1-rectified flow with the pre-trained EDM (VE). We also increase the batch size from 128 to 512 on CIFAR-10 compared to Liu et al. [2022]. Overall, these improve the FID of 2-rectified flow from 12.21 to 7.14 (a 42% improvement) in the one-step setting on CIFAR-10 (config B).

4.4 Incorporating real data

Training 2-rectified flow does not require real data (i.e., it can be data-free), but we can use real data if it is available. To see the effects of incorporating real data, we integrate the generative ODE of 1-rectified flow backward from $t = 0$ to $t = 1$ using an NFE of 128 to collect $50,000$ pairs of (real data, synthetic noise) on CIFAR-10. For quick validation, we take the pre-trained 2-rectified flow model (config F) and fine-tune it using the (real data, synthetic noise) pairs for $5,000$ iterations with a learning rate of 1e-5. This improves the FID of 2-rectified flow from 3.38 to 3.07 in the one-step setting on CIFAR-10 (config G).

In this fine-tuning setting, we also explored using (synthetic data, real noise) pair with a probability of p, but we found that not incorporating (synthetic data, real noise) pairs at all (i.e., $p = 0$) performs

7

Table 3: Unconditional generation on CIFAR-10.

METHOD	NFE (↓)	FID (↓)	IS (↑)
Diffusion models			
Score SDE [Song et al., 2020b]	2000	2.38	9.83
DDPM [Ho et al., 2020]	1000	3.17	9.46
LSGM [Vahdat et al., 2021]	147	2.10	
EDM [Karras et al., 2022]	35	1.97	
Distilled diffusion models			
Knowledge Distillation [Luhman and Luhman, 2021]	1	9.36	
DFNO (LPIPS) [Zheng et al., 2022]	1	3.78	
TRACT [Berthelot et al., 2023]	1	3.78	
	2	3.32	
PD [Salimans and Ho, 2022]	1	9.12	
	2	4.51	
Score distillation			
Diff-Instruct [Luo et al., 2024]	1	4.53	9.89
DMD [Yin et al., 2023]	1	3.77	
GANs			
BigGAN [Brock et al., 2018]	1	14.7	9.22
StyleGAN2 [Karras et al., 2020b]	1	8.32	9.21
StyleGAN2-ADA [Karras et al., 2020a]	1	2.92	9.83
Consistency models			
CD (LPIPS) [Song et al., 2023]	1	3.55	9.48
	2	2.93	9.75
CT (LPIPS) [Song et al., 2023]	1	8.70	8.49
	2	5.83	8.85
iCT [Song and Dhariwal, 2023]	1	2.83	9.54
	2	2.46	9.80
iCT-deep [Song and Dhariwal, 2023]	1	2.51	9.76
	2	2.24	9.89
CTM [Kim et al., 2023]	1	5.19	
CTM [Kim et al., 2023] + GAN	1	**1.98**	
Rectified flows			
1-rectified flow (+distill) [Liu et al., 2022]	1	6.18	9.08
2-rectified flow [Liu et al., 2022]	1	12.21	8.08
	110	3.36	9.24
+distill [Liu et al., 2022]	1	4.85	9.01
3-rectified flow [Liu et al., 2022]	1	8.15	8.47
	104	3.96	9.01
+Distill [Liu et al., 2022]	1	5.21	8.79
2-rectified flow++ (ours)	1	3.07	
	2	2.40	

Table 4: Class-conditional generation on ImageNet 64×64.

METHOD	NFE (↓)	FID (↓)	Prec. (↑)	Rec. (↑)
Diffusion models				
DDIM [Song et al., 2020a]	50	13.7	0.65	0.56
	10	18.3	0.60	0.49
DPM solver [Lu et al., 2022]	10	7.93		
	20	3.42		
DEIS [Zhang and Chen, 2022]	10	6.65		
	20	3.10		
DDPM [Ho et al., 2020]	250	11.0	0.67	0.58
iDDPM [Nichol and Dhariwal, 2021]	250	2.92	0.74	0.62
ADM [Dhariwal and Nichol, 2021]	250	2.07	0.74	0.63
EDM [Karras et al., 2022]	79	2.30		
Distilled diffusion models				
DFNO (LPIPS) [Zheng et al., 2022]	1	7.83		0.61
TRACT [Berthelot et al., 2023]	1	7.43		
	2	4.97		
BOOT [Gu et al., 2023]	1	16.3	0.68	0.36
PD [Salimans and Ho, 2022]	1	15.39	0.59	0.62
	2	8.95	0.63	0.65
	4	6.77	0.66	0.65
Score distillation				
Diff-Instruct [Luo et al., 2024]	1	5.57		
DMD [Yin et al., 2023]	1	2.62		
GANs				
BigGAN-deep [Brock et al., 2018]	1	4.06	0.79	0.48
Consistency models				
CD (LPIPS) [Song et al., 2023]	1	6.20	0.68	0.63
	2	4.70	0.69	0.64
	3	4.32	0.70	0.64
CT (LPIPS) [Song et al., 2023]	1	13.0	0.71	0.47
	2	11.1	0.69	0.56
iCT [Song and Dhariwal, 2023]	1	4.02	0.70	0.63
	2	3.20	0.73	0.63
iCT-deep [Song and Dhariwal, 2023]	1	3.25	0.72	0.63
	2	2.77	0.74	0.62
CTM + GAN [Kim et al., 2023]	1	**1.92**		0.57
	2	**1.73**		0.57
Rectified flows				
2-rectified flow++ (ours)	1	4.31		
	2	3.64		

The red rows correspond to the top-5 baselines for the 1-NFE setting, and the blue rows correspond to the top 5 baselines for the 2-NFE setting. The lowest FID scores for 1-NFE and 2-NFE are **boldfaced**.

the best. We expect that training from scratch will further improve the performance with different values of p, and leave it to future work. A similar idea is also explored in Anonymous [2024].

5 Experiments

We call these combined improvements to Reflow training *2-rectified flow++* and evaluate it on four datasets: CIFAR-10 Krizhevsky et al. [2009], AFHQ Choi et al. [2020], FFHQ Karras et al. [2019], and ImageNet Deng et al. [2009]. We compare our improved Reflow to up to 20 recent baselines, in the families of diffusion models, distilled diffusion models, score distillation, GANs, consistency models, and rectified flows. The details of our experimental setup are included in Appendix E.

5.1 Unconditional and class-conditional image generation

In Tables 3 and 4, we compare 2-rectified flow++ with the state-of-the-art methods on CIFAR-10 and ImageNet 64×64. We observe two main messages:

On both datasets, 2-rectified flow++ (ours) outperforms or is competitive with SOTA baselines in the 1-2 NFE regime. On CIFAR-10 (Table 3), our 2-rectified flow achieves an FID of 3.07 in one step, surpassing existing distillation methods such as consistency distillation (CD) [Song et al., 2023], progressive distillation (PD) [Salimans and Ho, 2022], diffusion model sampling with neural operator (DSNO) [Zheng et al., 2022], and TRAnsitive Closure Time-distillation (TRACT) [Berthelot et al., 2023]. On ImageNet 64×64 (Table 4), our model surpasses the distillation methods such as CD, PD, DFNO, TRACT, and BOOT in one-step generation. We also close the gap with iCT (4.01 vs

8

(a) CIFAR-10	(b) AFHQ 64×64	(c) FFHQ 64×64
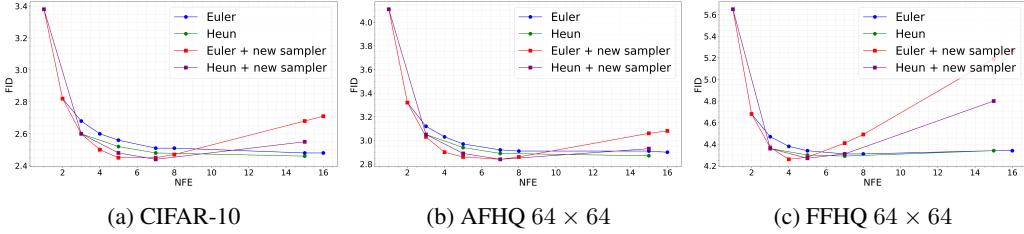
Figure 4: Effects of ODE Solver and new update rule.

4.31), the state-of-the-art consistency model, even with half the batch size. Note that on ImageNet, our model does not use real data during training, while consistency training (CT) requires real data. We believe we could further reduce the gap by using config G in Tab. 1. Uncurated samples of our model are provided in Appendix. G.

2-rectified flow++ reduces the FID of 2-rectified flows by up to 75%. Compared to vanilla rectified flows [Liu et al., 2022], our one-step FID on CIFAR-10 is lower than that of the previous 2-rectified flow by 9.14 (a reduction of 75%), and of the 3-rectified flow by 5.08 (see also Table 1 for ablations on other datasets). In addition, it outperforms the previous 2-rectified flow with 110 NFEs using only one step and also surpasses 2-rectified flow + distillation, which requires an additional distillation stage.

5.2 Reflow can be computationally more efficient than other distillation methods

At first glance, Reflow seems computationally expensive compared to CD and CT as it requires generating synthetic pairs before training. However, CD requires 4 (1 for student, 1 for teacher, and 2 for Heun's solver) forward passes for each training iteration, and CT requires 2 (1 for student and 1 for teacher) forward passes, while Reflow requires only 1. For example, in our ImageNet experiment setting, the total number of forward passes for Reflow is 395M + 1433.6M = 1828.6M (395M for generating pairs

Table 6: Comparison of the number of forward passes. Reflow uses 395M forward passes for generating pairs and $1,433.6$M for training.

Method	Per iteration	Total	Rel. total cost
Reflow	1	1828.6M	$\times 1$
CD	4	5734.4M	$\times 3.1$
CT	2	2867.2M	$\times 1.5$

and 1,433.6M for training), while the total numbers of forward passes for CD and CT would be $1,433.6 \cdot 4 = 5,734.4$M and $1,433.6 \cdot 2 = 2,867.2$M under the same setting. See Table 6 for the comparison. Moreover, generating pairs is a one-time cost since we can reuse the pairs for multiple training runs.

In terms of the storage cost, the synthetic images for ImageNet 64×64 require 42 GB. For noise, we only store the states of the random number generator, which is negligible.

While these results should be further validated for larger datasets, our results suggest that the fact that Reflow requires generating synthetic pairs does not necessarily make it less computationally efficient than other distillation methods.

5.3 Effects of samplers

Unlike distillation methods, rectified flow is a neural ODE, and its outputs approach the true solution of the ODE as NFE increases (i.e., precision grows). Figure 4 shows that with the standard Euler solver, FID decreases as NFE increases on all datasets. Moreover, Heun's second-order solver further improves the trade-off curve between FID and NFE. This suggests that there may be further room for improvement by using more advanced samplers. We provide some preliminary ideas towards this goal in Appendix D.

9

| (a) Reconstruction loss | (b) Distribution of $\|\mathbf{z}\|_2^2$ | (c) Inversion/reconstruction results |
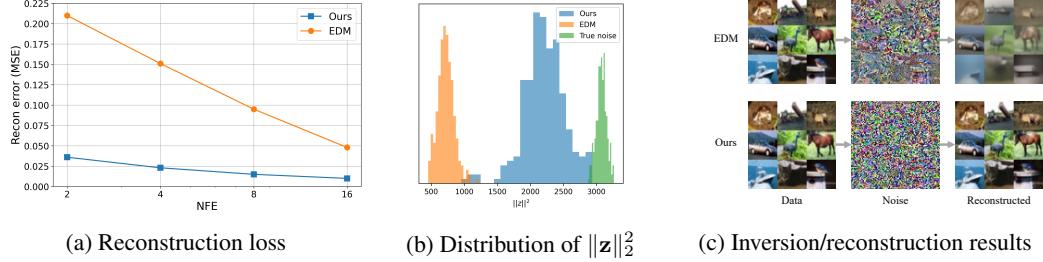
Figure 5: Inversion results on CIFAR-10. (a) Reconstruction error between real and reconstructed data is measured by the mean squared error (MSE), where the x-axis represents NFEs used for inversion and reconstruction (e.g. 2 means 2 for inversion and 2 for reconstruction). (b) Distribution of $\|\mathbf{z}\|_2^2$ of the inverted noises as a proxy for Gaussianity (NFE = 8). The green histogram represents the distribution of true noise, which is Chi-squared with $3 \times 32 \times 32 = 3072$ degrees of freedom. (c) Inversion and reconstruction results using (8 + 8) NFEs. With only 8 NFEs, EDM fails to produce realistic noise, and also the reconstructed samples are blurry.

| (a) Interpolation | (b) Image translation |

Figure 6: Applications of few-step inversion. (a) Interpolation between two real images. (b) Image-to-image translation. The total NFEs used are 6 (4 for inversion and 2 for generation).

5.4 Inversion

Unlike distillation methods, rectified flows are neural ODEs, thus they allow for *inversion* from data to noise by simply integrating the ODE in the backward direction. In diffusion models, inversion has been used for various applications such as image editing [Hertz et al., 2022, Kim et al., 2022, Wallace et al., 2023, Su et al., 2022, Hong et al., 2023] and watermarking [Wen et al., 2023], but it usually requires many NFEs. Figure 5 (a) demonstrates that our 2-rectified flow++ achieves significantly lower reconstruction error than EDM. Notably, the reconstruction error of 2-rectified flow++ with only 2 NFEs is lower than that of EDM with 16 NFEs. In (b), we compare the quality of the inverted noise, where we find that the noise vectors of 2-rectified flow are more Gaussian-like than those of EDM, in the sense that their norm is closer to that of typical Gaussian noise. These are also shown visually in (c). In Figure 6, we show two applications of inversion: interpolating between two real images (a) and image-to-image translation (b). Notably, the total NFE used is only 6 (4 for inversion and 2 for generation), which is significantly lower than what is typically required in diffusion models (≥ 100) [Hong et al., 2023].

6 Conclusion

In this work, we propose several improved training techniques for rectified flows, including the U-shaped timestep distribution and LPIPS-Huber loss. We show that by combining these improvements, 2-rectified flows++ outperforms the state-of-the-art distillation methods in the 1-2 NFE regime on CIFAR-10 and ImageNet 64×64 and closes the gap with iCT, the state-of-the-art consistency model. 2-rectified flows++ have limitations though—they still do not outperform the best consistency models (iCT), and their training is slower (by about 15% per iteration on ImageNet) than previous rectified flows because of the LPIPS loss. Despite these shortcomings, the training techniques we propose can easily and significantly boost the performance of rectified flows in the low NFE setting, without harming performance at the higher NFE setting.

10

Acknowledgments

This work was made possible in part by the National Science Foundation under grant CCF-2338772, CNS-2325477, as well as generous support from Google, the Sloan Foundation, Intel, and Bosch. This work used Bridges-2 GPU at the Pittsburgh Supercomputing Center through allocation CIS240037 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants 2138259, 2138286, 2138307, 2137603, and 2138296 Boerner et al. [2023].

References

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Anonymous. Balanced conic rectified flow. In *Submitted to The Thirteenth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=ctSjIlYN74`. under review.

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

Timothy J Boerner, Stephen Deems, Thomas R Furlani, Shelley L Knuth, and John Towns. Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing*, pages 173–176. 2023.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing*, 6(2): 298–311, 1997.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8188–8197, 2020.

Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. *arXiv preprint arXiv:2210.11427*, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Joshua M Susskind. Boot: Data-free distillation of denoising diffusion models with bootstrapping. In *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*, 2023.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

11

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

Seongmin Hong, Kyeonghyun Lee, Suh Yoon Jeon, Hyewon Bae, and Se Young Chun. On exact inversion of dpm-solvers. *arXiv preprint arXiv:2311.18387*, 2023.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020a.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020b.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.

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

Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2426–2435, 2022.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Toronto, ON, Canada*, 2009.

Sangyun Lee, Beomsu Kim, and Jong Chul Ye. Minimizing trajectory curvature of ode-based generative models. *arXiv preprint arXiv:2301.12003*, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

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

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. *arXiv preprint arXiv:2309.06380*, 2023.

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. *arXiv preprint arXiv:2206.00927*, 2022.

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

Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diffinstruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

12

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6038–6047, 2023.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

Ashwini Pokle, Matthew J Muckley, Ricky TQ Chen, and Brian Karrer. Training-free linear image inversion via flows. *arXiv preprint arXiv:2310.04432*, 2023.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky Chen. Multisample flow matching: Straightening flows with minibatch couplings. *arXiv preprint arXiv:2304.14772*, 2023.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

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

Neta Shaul, Juan Perez, Ricky TQ Chen, Ali Thabet, Albert Pumarola, and Yaron Lipman. Bespoke solvers for generative flow models. *arXiv preprint arXiv:2310.19075*, 2023.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

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

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

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*, 2020b.

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

Xuan Su, Jiaming Song, Chenlin Meng, and Stefano Ermon. Dual diffusion implicit bridges for image-to-image translation. *arXiv preprint arXiv:2203.08382*, 2022.

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

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22532–22541, 2023.

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*, 2021.

13

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *arXiv preprint arXiv:2305.20030*, 2023.

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

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

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

Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning. *arXiv preprint arXiv:2211.13449*, 2022.

14

(a) Reconstruction loss   (b) Distribution of $\|\mathbf{z}\|_2^2$
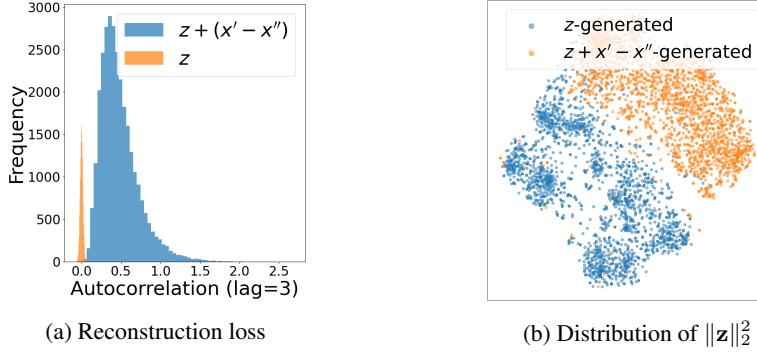
Figure 7: (a) Autocorrelation plot on CIFAR-10 analogous to Figure 2(d). (b) t-SNE visualization of the inception-v3 features of the samples in Figure 2(b). They show negligible overlap.

A Equivalence of v-parameterization and x-parameterization

Given $\mathbf{x}_t = (1-t)\mathbf{x} + t\mathbf{z}$ and $\mathbf{z} - \mathbf{x} = (\mathbf{x}_t - \mathbf{x})/t$, the equivalence of the x-parameterization (Eq. (2)) and the v-parameterization (Eq. (3)) can be shown in the following way. The result is borrowed from the Appendix of Lee et al. [2023], and we provide here for completeness.

$$\int_0^1 \mathbb{E}[\|(\mathbf{z}-\mathbf{x}) - \mathbf{v}_\theta(\mathbf{x}_t, t)\|_2^2]\, dt = \int_0^1 \mathbb{E}[\|(\mathbf{x}_t - \mathbf{x})/t - \mathbf{v}_\theta(\mathbf{x}_t, t)\|_2^2]\, dt \tag{7}$$

$$= \int_0^1 \mathbb{E}[\|(\mathbf{x}_t - \mathbf{x})/t - (\mathbf{x}_t - \mathbf{x}_\theta(\mathbf{x}_t, t))/t\|_2^2]\, dt \tag{8}$$

$$= \int_0^1 \mathbb{E}[\frac{1}{t^2}\|\mathbf{x} - \mathbf{x}_\theta(\mathbf{x}_t, t))\|_2^2]\, dt. \tag{9}$$

This is equivalent to Eq. (2) with $\omega(t) = 1/t^2$.

B Additional Details for Figure 2

In this section, we provide additional results details for Figure 2. Figure 7(a) shows the autocorrelation histogram on CIFAR-10 while Figure 2(d) is on FFHQ-64. Figure 7(b) shows that the inception features of the samples in Figure 2(b) rarely overlap with each other.

For the autocorrelation plots, we use 30,000 pairs of $(\mathbf{x}', \mathbf{x}'')$ and randomly sample $\mathbf{z}$ from the standard Gaussian distribution. For a d-dimensional vector $\mathbf{u}$, we define the autocorrelation as:

$$R_\mathbf{u}(l) = \frac{1}{d-l}\sum_{k=1}^{d-l} \mathbf{u}_k \mathbf{u}_{k+l}, \tag{10}$$

where $\mathbf{u}_k$ is the k-th element of a vector $\mathbf{u}$ and $l > 0$ represents the lag.

C Initialization Results

C.1 Proof of Proposition 1

Consider two perturbation kernels $p_t(\mathbf{x}_t|\mathbf{x}) = \mathcal{N}(s(t)\mathbf{x}, \sigma(t)^2\mathbf{I})$ and $p'_t(\mathbf{x}_t|\mathbf{x}) = \mathcal{N}(s'(t)\mathbf{x}, \sigma'(t)^2\mathbf{I})$:

$$p_t(\mathbf{x}_t|\mathbf{x}) = \frac{1}{(2\pi\sigma(t)^2)^{d/2}} \exp(-\frac{1}{2\sigma(t)^2}\|\mathbf{x}_t - s(t)\mathbf{x}\|_2^2) \tag{11}$$

$$p'_t(\mathbf{x}_t|\mathbf{x}) = \frac{1}{(2\pi\sigma'(t)^2)^{d/2}} \exp(-\frac{1}{2\sigma'(t)^2}\|\mathbf{x}_t - s'(t)\mathbf{x}\|_2^2). \tag{12}$$

15

Let $t'(t)$ be such that $\frac{s(t)}{\sigma(t)} = \frac{s'(t')}{\sigma'(t')}$. We will show that $p_t(\mathbf{x}|\mathbf{x}_t) = p'_{t'}(\mathbf{x}|\frac{s'(t')}{s(t)}\mathbf{x}_t)$. We start by showing:

$$p'_{t'}(\frac{s'(t')}{s(t)}\mathbf{x}_t|\mathbf{x}) = \frac{1}{(2\pi\sigma'(t')^2)^{d/2}} \exp(-\frac{1}{2\sigma'(t')^2}||\frac{s'(t')}{s(t)}\mathbf{x}_t - s'(t')\mathbf{x}||_2^2) \tag{13}$$

$$= \frac{1}{(2\pi\sigma'(t')^2)^{d/2}} \exp(-\frac{1}{2\sigma'(t')^2}||\frac{s'(t')}{s(t)}(\mathbf{x}_t - s(t)\mathbf{x})||_2^2) \tag{14}$$

$$= \frac{1}{(2\pi\sigma'(t')^2)^{d/2}} \exp(-\frac{1}{2\sigma'(t')^2}\frac{s'(t')^2}{s(t)^2}||\mathbf{x}_t - s(t)\mathbf{x}||_2^2) \tag{15}$$

$$= \frac{1}{(2\pi\sigma'(t')^2)^{d/2}} \exp(-\frac{1}{2\sigma(t)^2}\frac{s(t)^2}{s(t)^2}||\mathbf{x}_t - s(t)\mathbf{x}||_2^2) \tag{16}$$

$$= \frac{1}{(2\pi\sigma'(t')^2)^{d/2}} \exp(-\frac{1}{2\sigma(t)^2}||\mathbf{x}_t - s(t)\mathbf{x}||_2^2) \tag{17}$$

$$= \frac{1}{(2\pi\sigma'(t')^2)^{d/2}} \frac{(2\pi\sigma(t)^2)^{d/2}}{(2\pi\sigma(t)^2)^{d/2}} \exp(-\frac{1}{2\sigma(t)^2}||\mathbf{x}_t - s(t)\mathbf{x}||_2^2) \tag{18}$$

$$= \frac{(2\pi\sigma(t)^2)^{d/2}}{(2\pi\sigma'(t')^2)^{d/2}} p_t(\mathbf{x}_t|\mathbf{x}) \tag{19}$$

$$= \left(\frac{\sigma(t)}{\sigma'(t')}\right)^d p_t(\mathbf{x}_t|\mathbf{x}). \tag{20}$$

Here, Eq. (20) says that $p_t(\mathbf{x}_t|\mathbf{x}) \propto p'_{t'}(\frac{s'(t')}{s(t)}\mathbf{x}_t|\mathbf{x})$ (but not equal), which is a minor fix from the original proof [Pokle et al., 2023]. Then, we have

$$p_t(\mathbf{x}|\mathbf{x}_t) = \frac{1}{p_t(\mathbf{x}_t)}p_\mathbf{x}(\mathbf{x})p_t(\mathbf{x}_t|\mathbf{x}) \tag{21}$$

$$p'_{t'}(\mathbf{x}|\frac{s'(t')}{s(t)}\mathbf{x}_t) = \left(\frac{\sigma(t)}{\sigma'(t')}\right)^d \frac{1}{p'_{t'}(\frac{s'(t')}{s(t)}\mathbf{x}_t)}p_\mathbf{x}(\mathbf{x})p_t(\mathbf{x}_t|\mathbf{x}) \tag{22}$$

Since $p'_{t'}(\mathbf{x}|\frac{s'(t')}{s(t)}\mathbf{x}_t)$ should be integrated to one, $\left(\frac{\sigma(t)}{\sigma'(t')}\right)^d \frac{1}{p'_{t'}(\frac{s'(t')}{s(t)}\mathbf{x}_t)} = \int p_\mathbf{x}(\mathbf{x})p_t(\mathbf{x}_t|\mathbf{x})d\mathbf{x}$ and thus two densities are equal. As the posterior densities are the same, their expectations are also the same.

In our case, $s(t) = 1 - t$, $\sigma(t) = t$, and $p'_t(\mathbf{x}_t|\mathbf{x})$ is the perturbation kernel of either the VP or VE diffusion model. Now we have to find t' such that $\frac{1-t}{t} = \frac{s'(t')}{\sigma'(t')}$.

C.2 Perturbation Kernel Instantiations

We next provide the values of the converted time and scale of the variance preserving (VP) and variance exploding (VE) diffusion models.

VE diffusion model Karras et al. [2022] defines the perturbation kernel of the VE diffusion model as $p'_t(\mathbf{x}_t|\mathbf{x}) = \mathcal{N}(\mathbf{x}, t^2\mathbf{I})$. Then, t' satisfies $\frac{1-t}{t} = \frac{s'(t')}{\sigma'(t')} = \frac{1}{t'}$, so $t' = \frac{t}{1-t}$, and $\frac{s'(t')}{s(t)} = \frac{1}{1-t}$, which correspond to t_{VE} and s_{VE} in Table 2.

VP diffusion model Song et al. [2020b] defines the perturbation kernel of the VP diffusion model as $p'_t(\mathbf{x}_t|\mathbf{x}) = \mathcal{N}(\alpha(t)\mathbf{x}, (1 - \alpha(t)^2)\mathbf{I})$, where $\alpha(t) := \exp(-\frac{1}{2}\int_0^t(19.9s + 0.1)ds)$ defined on $t \in [0, 1]$. Then, t' satisfies $\frac{1-t}{t} = \frac{s'(t')}{\sigma'(t')} = \frac{\alpha(t')}{\sqrt{1-\alpha(t')^2}}$. From here, we have

$$\frac{(1-t)^2}{t^2} = \frac{\alpha(t')^2}{1 - \alpha(t')^2} \tag{23}$$

$$\alpha(t') = \sqrt{\frac{(1-t)^2}{t^2 + (1-t)^2}}, \tag{24}$$

16

where we used the fact that $\alpha(t) > 0$. Since $\alpha(t)$ is a monotonically decreasing function for $t \geq 0$, we can use its inverse α^{-1} to find $t' = \alpha^{-1}(\sqrt{\frac{(1-t)^2}{t^2+(1-t)^2}})$.

$$y = \alpha(t) = \exp(-\frac{1}{2}\int_0^t (19.9s + 0.1)ds) = \exp(-\frac{19.9}{4}t^2 - 0.05t) \tag{25}$$

$$\ln y = -\frac{19.9}{4}t^2 - 0.05t \tag{26}$$

$$\frac{19.9}{4}t^2 + 0.05t + \ln y = 0 \tag{27}$$

Applying the quadratic formula, we have

$$t = \frac{-0.05 \pm \sqrt{0.05^2 - 4 \cdot \frac{19.9}{4}\ln y}}{2 \cdot \frac{19.9}{4}} = \frac{-0.05 \pm \sqrt{0.0025 - 19.9\ln y}}{9.95}. \tag{28}$$

Since $y = \alpha(t)$ is monotonically decreasing, we can choose the positive root:

$$\alpha^{-1}(y) = \frac{-0.05 + \sqrt{0.0025 - 19.9\ln y}}{9.95}. \tag{29}$$

Now, we arrive at

$$t' = \alpha^{-1}(\sqrt{\frac{(1-t)^2}{t^2+(1-t)^2}}) = \frac{-0.05 + \sqrt{0.0025 - 19.9\ln\sqrt{\frac{(1-t)^2}{t^2+(1-t)^2}}}}{9.95}, \tag{30}$$

which corresponds to t_{VP} in Table 2. Also, we have $\frac{s'(t')}{s(t)} = \frac{\alpha(t')}{1-t}$, which is s_{VP} in Table 2.

D   New Update Rule

In the standard Euler solver, the update rule is $\mathbf{z}_{t-\Delta t} := \mathbf{z}_t - \mathbf{v}(\mathbf{z}_t, t)\Delta t$. Alternatively, as $\mathbf{x}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$ of our model generates pretty good samples, we can instead use the linear interpolation between $\mathbf{x}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$ and $\mathbf{z}_1$ to get the next step: $\mathbf{z}_{t-\Delta t} := (1 - (t - \Delta t))\mathbf{x}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) + (t - \Delta t)\mathbf{z}_1$. Note that when NFE < 3, the two update rules are equivalent and do not affect our results in Section 5.1. Fig. 4 shows that when applied to existing solvers, the new update rule improves the sampling efficiency up to $4\times$, achieving the best FID with ≤ 5 NFEs.

Algorithm 2 shows the pseudocode for generating samples using the new update rule. Unlike the standard Euler update rule which only depends on the current state $\mathbf{z}_t$, our new update rule utilizes the previous state (i.e., $\mathbf{z}_1$) to generate the next state $\mathbf{z}_{t-\Delta t}$ and thus can be viewed as a form of history-dependent samplers. Obviously, incorporating the initial state only would not be the best choice. We believe that the result can be further improved, especially by using learning-based solvers [Watson et al., 2021, Shaul et al., 2023]; and leave such exploration to future work.

E   Experimental Details

Before training 2-rectified flow, we generate data-noise pairs following the sampling regime of EDM [Karras et al., 2022]. For CIFAR-10, we generate 1M pairs using 35 NFEs. For AFHQ, FFHQ, and ImageNet, we generate 5M pairs using 79 NFEs. We use Heun's second-order solver for all cases. In Table 3, we report the result of config G in Table 1. In ImageNet, we use the batch size of 2048 and train the models for 700,000 iterations using mixed-precision training [Micikevicius et al., 2017] with the dynamic loss scaling. We use config E in Table 1 for ImageNet.

We provide training configurations in Table 7. For all datasets, we use Adam optimizer. We use the exponential moving average (EMA) with 0.9999 decay rate for all datasets.

On ImageNet, the training takes roughly 9 days with 64 NVIDIA V100 GPUs. On CIFAR-10 and FFHQ/AFHQ, it takes roughly 4 days with 16 and 8 V100 GPUs, respectively. For all cases, we use

17

Algorithm 2 Generate

```
def generate(z1, label, model, time_schedule, N, solver, sampler, device):
    """
    z1: initial noise
    label: class label
    model: v_theta
    time_schedule: time schedule, e.g., [0.99999, 0.5, 0] for 2 steps
    N: NFE
    solver: 'euler' or 'heun'
    sampler: 'default' or 'new'
    """

    z = z1.clone()
    cnt = 0
    for i in range(len(time_schedule[:-1])):
        t = torch.ones((z.shape[0]), device=device) * time_schedule[i]
        t_next = torch.ones((z.shape[0]), device=device) * time_schedule[i+1]
        dt = t_next[0] - t[0]
        vt = model(z, t, label)
        x0hat = z - vt * t.view(-1,1,1,1)
        if solver == 'heun' and cnt < N - 1: # Heun correction
            if sampler == 'default':
                z_next = z.detach().clone() + vt * dt
            elif sampler == 'new':
                z_next = (1 - t_next.view(-1,1,1,1)) * x0hat + t_next.view(-1,1,1,1) * z1
            vt_next = model(z_next, t_next, label)
            vt = (vt + vt_next) / 2
            x0hat = z - vt * t.view(-1,1,1,1)
        if sampler == 'default':
            z = z.detach().clone() + vt * dt
        elif sampler == 'new':
            z = (1 - t_next.view(-1,1,1,1)) * x0hat + t_next.view(-1,1,1,1) * z1
        cnt += 1

    return z
```

Table 7: Training configurations for each dataset. We linearly ramp up learning rates for all datasets.

Datasets	Batch size	Dropout	Learning rate	Warm up iter.
CIFAR-10	512	0.13	2e-4	5000
FFHQ / AFHQ	256	0.25	2e-4	5000
ImageNet	2048	0.10	1e-4	2500

the NVIDIA DGX-2 cluster. To prevent zero-division error with EDM initialization, we sample t from $[0.00001, 0.99999]$ in practice. For a two-step generation, we evaluate $\mathbf{v}_{\boldsymbol{\theta}}$ at $t = 0.99999$ and $t = 0.8$. For other NFEs, we uniformly divide the interval $[0.00001, 0.99999]$.

License The following are licenses for each dataset we use:

- CIFAR-10: Unknown
- FFHQ: CC BY-NC-SA 4.0
- AFHQ: CC BY-NC 4.0
- ImageNet: Custom (research, non-commercial)

F   Broader Impacts

This paper proposes an advanced algorithm to generate realistic data at high speed, which could have both positive and negative impacts. For example, it could be used for generating malicious or misleading content. Therefore, such technology should be deployed and used responsibly and with caution. We believe that our work is not expected to have any more potential negative impact than other work in the field of generative modeling.

18

G Uncurated Synthetic Samples

We provide uncurated synthetic samples from our 2-rectified flow++ on CIFAR-10, AFHQ, and ImageNet in Figures 8, 9, 10, 11, 20, 21, 16, 17, 18, 19, 12, 13, 14, and 15. We use our new sampler (Sec. 5.3) to generate these images.

19

Figure 8: Synthetic samples from 2-rectified flow++ on CIFAR-10 with NFE = 1 (FID=3.38).
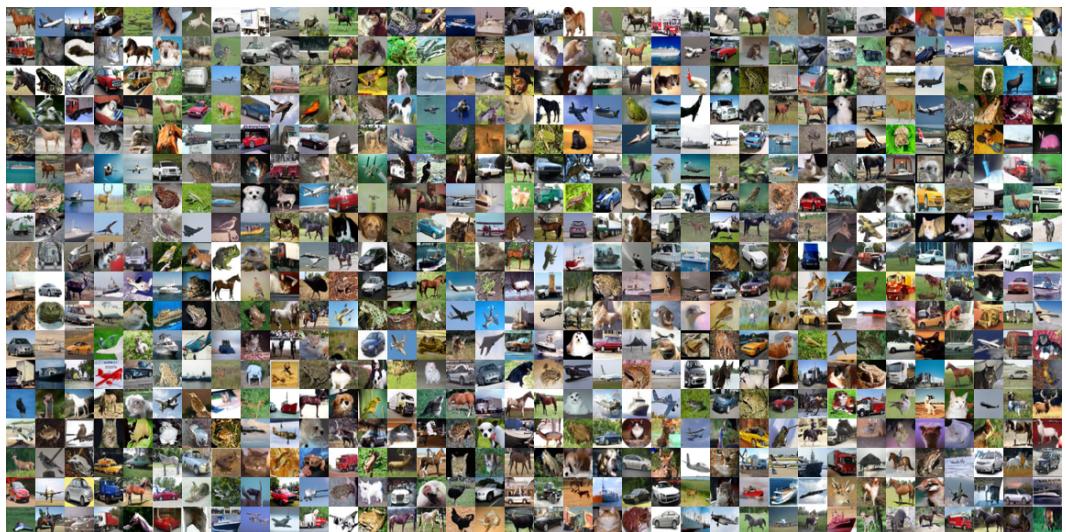
Figure 9: Synthetic samples from 2-rectified flow++ on CIFAR-10 with NFE = 2 (FID=2.76).

20

Figure 10: Synthetic samples from 2-rectified flow++ on CIFAR-10 with NFE = 4 (FID=2.50).
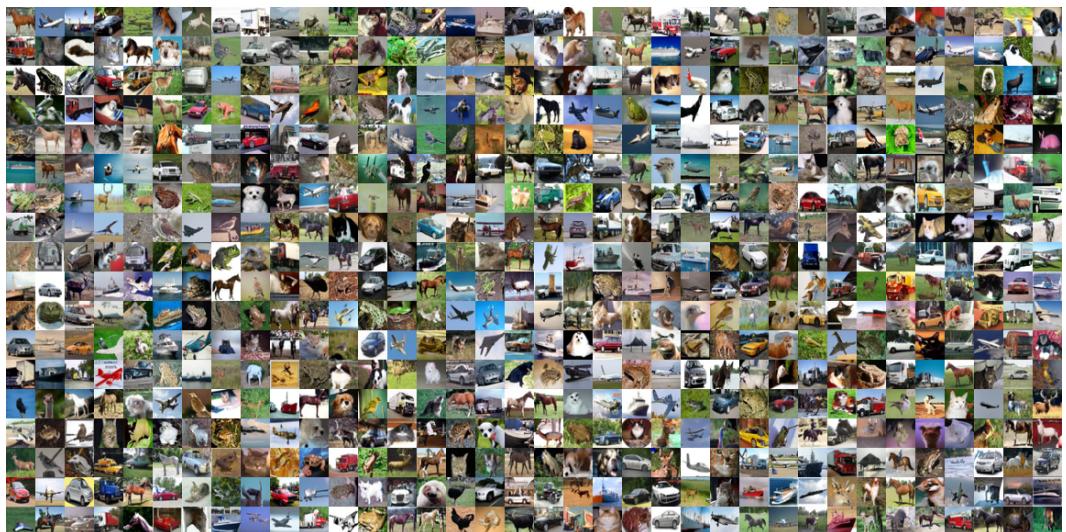
Figure 11: Synthetic samples from 2-rectified flow++ on CIFAR-10 with NFE = 5 (FID=2.45).

21
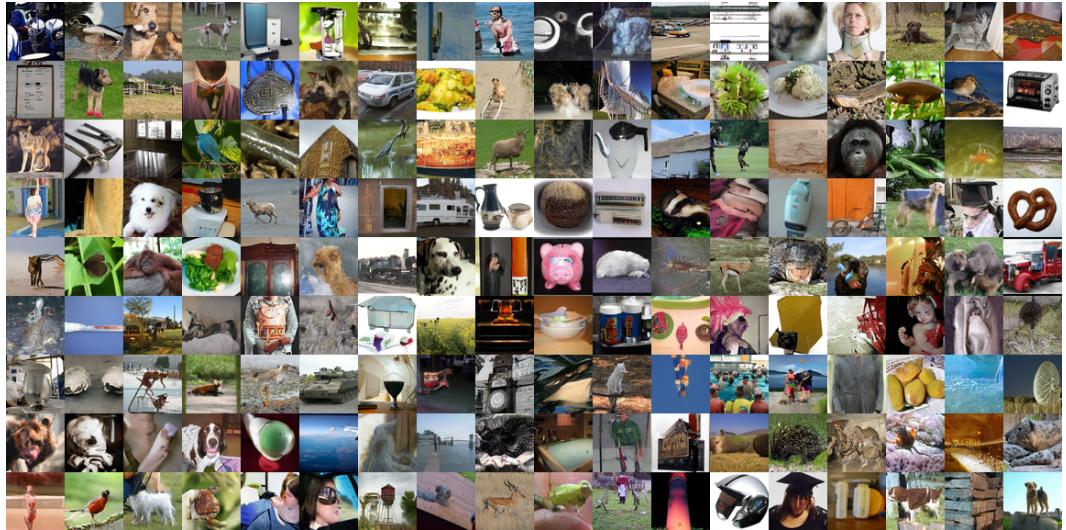
Figure 12: Synthetic samples from 2-rectified flow++ on ImageNet 64×64 with NFE = 1 (FID=4.31).
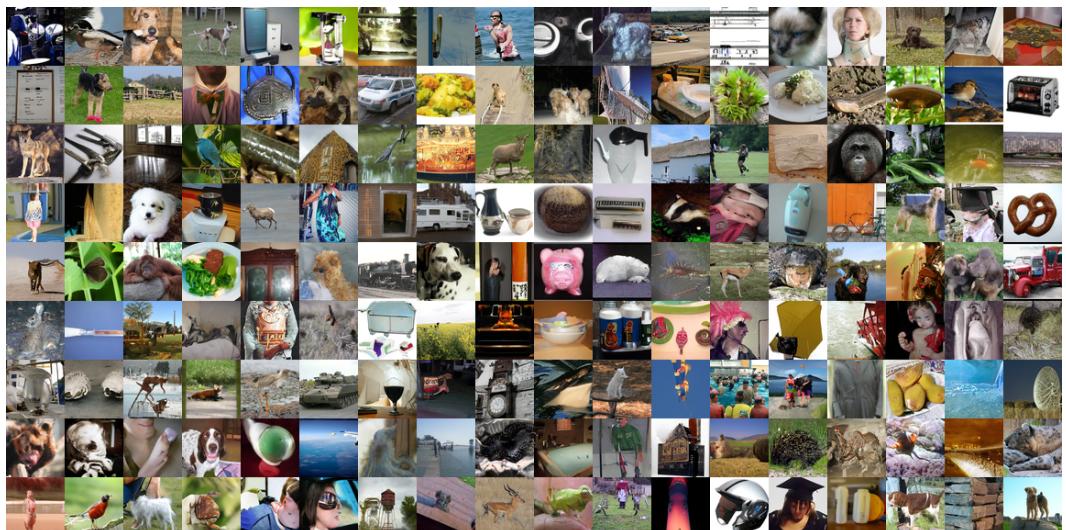
Figure 13: Synthetic samples from 2-rectified flow++ on ImageNet 64×64 with NFE = 2 (FID=3.64).

22

Figure 14: Synthetic samples from 2-rectified flow++ on ImageNet 64×64 with NFE = 4 (FID=3.44).
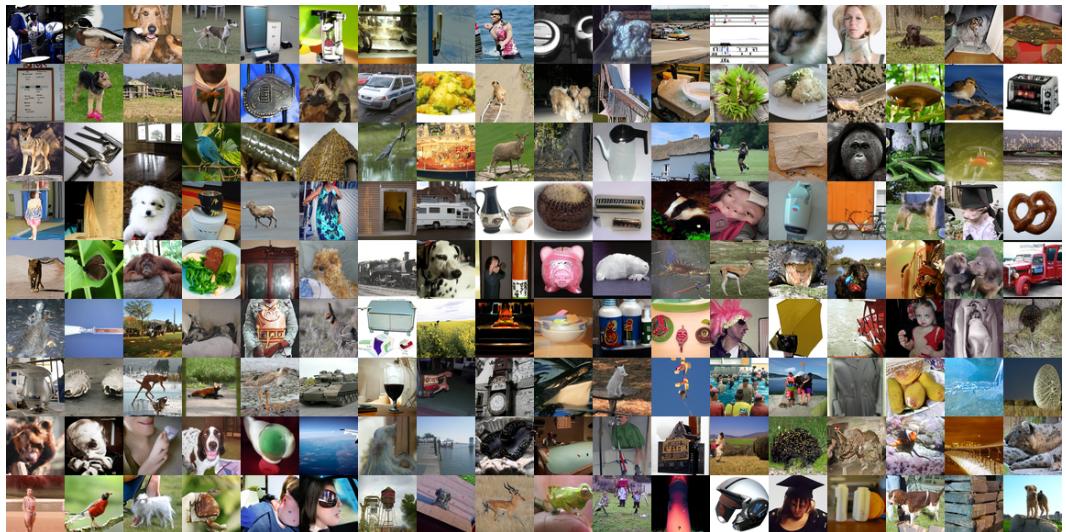
Figure 15: Synthetic samples from 2-rectified flow++ on ImageNet 64×64 with NFE = 8 (FID=3.32).

23

Figure 16: Synthetic samples from 2-rectified flow++ on AFHQ 64×64 with NFE = 1 (FID=4.11).

Figure 17: Synthetic samples from 2-rectified flow++ on AFHQ 64×64 with NFE = 2 (FID=3.12).

24

Figure 18: Synthetic samples from 2-rectified flow++ on AFHQ 64×64 with NFE = 4 (FID=2.90).

Figure 19: Synthetic samples from 2-rectified flow++ on AFHQ 64×64 with NFE = 5 (FID=2.86).

25

Figure 20: Synthetic samples from 2-rectified flow++ on FFHQ 64×64 with NFE = 1 (FID=5.21).
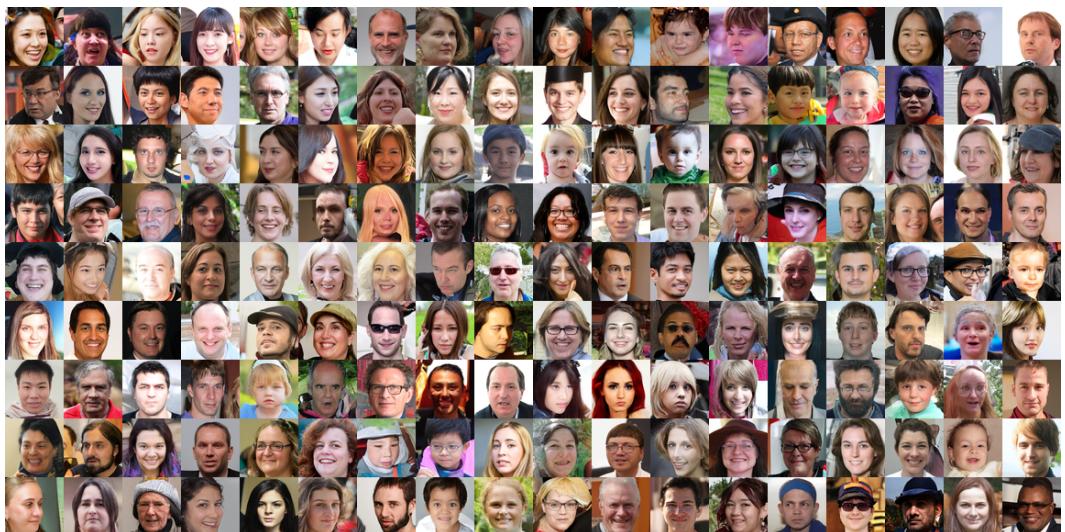
Figure 21: Synthetic samples from 2-rectified flow++ on FFHQ 64×64 with NFE = 2 (FID=4.26).

26

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction reflect the paper's contributions and scope such as improved empirical performance, which is backed up by our experiments in Sec. 5.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitations are discussed in the conclusion, such as the increased training time of our method.

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

   Justification: Our only theoretical result is Proposition 1, for which we provide the proof in Appendix C.1. The argument in Sec. 3 is intuitive, and we do not prove it theoretically.

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: We provide our experimental details in Sec. 5 and App. E. For the new update algorithm, we provide full pseudocode in Sec. D, and we will release code publicly as soon as we obtain approval to do so.

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

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

   Answer: [No]

   Justification: The datasets we used are publicly available. We will release the code publicly as soon as we obtain internal approval.

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: We provide these details in Sec. E.

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: The shaded area in Figure 3 indicates the standard deviation. We only compute FID once due to cost constraints.

8. **Experiments Compute Resources**

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

27

Answer: [Yes]

Justification: We provide details in Sec. E.

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: The paper conforms with the NeurIPS Code of Ethics.

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: We provide an impact statement in Sec. F. Effectively, as with other work on generative models, they can be used for both beneficial and harmful purposes. Our work, being focused on the mechanics of these models, does not introduce new risks, nor does it mitigate existing ones.

11. **Safeguards**

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

    Answer: [NA]

    Justification: We believe that our work is not expected to have any more potential risks than other work in this field. We have discussed some of these considerations in Sec. F.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: For the datasets we use, we cite the original papers and describe the license information in Sec. E.

13. **New Assets**

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

    Answer: [Yes]

    Justification: We provide necessary information to reproduce our new models in Sec. E.

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

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

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

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

    Answer: [NA]

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

28