Analyzing and Mitigating Model Collapse in Reflow Methods

Huminhao Zhu[1], Fangyikang Wang[2], Tianyu Ding[3], Qing Qu[4], Zhihui Zhu[1]
[1]Ohio State University, [2]MBZUAI, [3]Microsoft, [4]University of Michigan
zhu.4228@osu.edu, fangyikang.wang@mbzuai.ac.ae, tianyuding@microsoft.com,
qingqu@umich.edu, zhu.3440@osu.edu

Generative models increasingly encounter synthetic data produced by earlier model snapshots, either unintentionally through data contamination or deliberately through self-training procedures such as Reflow. In rectified flow and related diffusion/flow systems, Reflow retrains on model-generated samples to straighten trajectories and accelerate sampling, but repeated self-training can degrade sample quality and diversity. We provide a mechanistic analysis of this failure mode and a principled mitigation strategy. Using a linear denoising autoencoder (DAE) as a tractable surrogate for Reflow-style recursion, we show that under purely synthetic recursive training the end-to-end linear map contracts: its operator norm decays to zero at a geometric rate, reflecting a progressive loss of representational power. We further prove that augmenting each Reflow round with a fixed fraction of real data prevents this degeneration by keeping the operator norm bounded away from zero. Finally, we validate that the qualitative trends implied by the theory are observable in practical Reflow pipelines on toy settings and image benchmarks, and we show that simple real-data–augmented Reflow schemes preserve Reflow's sampling-speed benefits while maintaining image quality.

1. Introduction

Generative modeling aims to produce synthetic data that is indistinguishable from real-world distributions. Recent advances in diffusion models, flow matching, and rectified flows have enabled high-fidelity generation across images, audio, and language [1–8]. At the same time, the proliferation of synthetic content has created a new training regime: models are increasingly exposed to their own outputs, either unintentionally through contaminated datasets [9–11] or deliberately through iterative self-training procedures. A central risk in such regimes is *model collapse* (*MC*), where repeated training on self-generated data progressively degrades generation quality and diversity [9, 12, 13].

Most recent work on MC falls into three categories. A first line consists of empirical studies documenting degradation under synthetic retraining [12–17]. A second line analyzes simplified regression settings [18–21], and a third investigates stability properties under maximum-likelihood training objectives [22]. As emphasized by Schaeffer et al. [23], the literature still lacks (i) a clear, operational definition of MC that captures realistic generative training loops, and (ii) an analysis of regimes where models are trained not only on synthetic data, but on mixtures of real and generated samples, as is common in practice.

A particularly compelling testbed for these questions is the *Reflow* procedure used by rectified flow (RF) models [7, 24]. RF replaces stochastic diffusion trajectories with an explicit flow, and Reflow iteratively retrains the model on its own outputs to straighten this flow, yielding substantial sampling-speed improvements [7, 8, 25]. In this work, we use the term *Reflow* more broadly to denote such iterative retraining of a generative model on its self-generated samples, possibly mixed with real data. Empirically, Reflow is known to sometimes degrade sample quality, especially when applied for multiple rounds [7]. As a result, most practical implementations only recommend performing a single Reflow step [26–28], which stands in tension with Reflow's convergence guarantees [24]. Several heuristics have been proposed to partially mitigate this degradation [29–31]. However,

Third Conference on Parsimony and Learning (CPAL 2026).

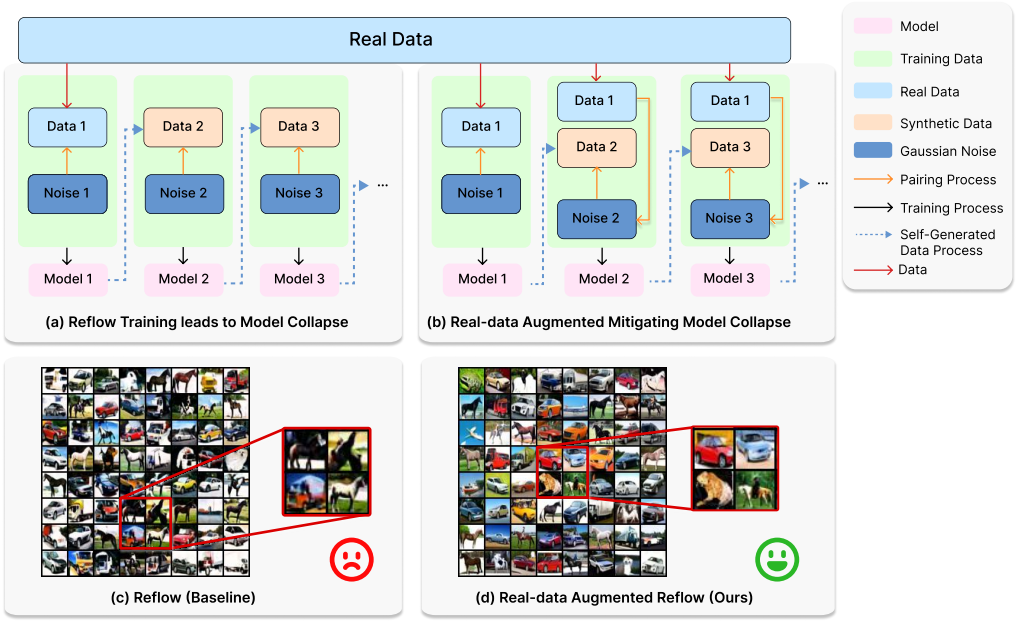


Figure 1: **Reflow collapse and mitigation via real-data augmentation.** (a) Reflow uses the same self-training loop that drives model collapse. It adds image and noise pairs as learning targets, but the model still relies entirely on its own synthetic outputs. No extra prior knowledge, such as real data, is included during training. (b) Illustrates RA Reflow training by adding real data pairs with each training iteration. (c/d) The Rectified Flow models were trained in both modes after 8 iterations. The baseline lacks color, and the images are blurry and mixed. With the real-data augmentation method, the images maintain their generation quality.

these issues are often attributed to vague "error accumulation", and we still lack a principled understanding of when and why Reflow collapses, how bad it will be, and how to design more stable Reflow procedures.

Rectified flow and model collapse thus share a common setting, namely retraining on self-generated data, but suggest apparently conflicting narratives: rectified flow delivers more efficient samplers, while MC highlights the failure modes of self-training. This raises two central questions that motivate our work:

Does Reflow methods suffer from model collapse?
If so, how can we mitigate collapse while retaining the benefits of Reflow?

To address these questions, we take a theoretical approach. An end-to-end analysis of Reflow for modern large-scale flow and diffusion models is analytically challenging. We therefore study a tractable *linear Reflow surrogate*: a two-layer linear denoising autoencoder (DAE) trained under a self-consuming Reflow loop. This surrogate isolates the effect of recursive self-training while admitting exact analysis. In this framework, we show that purely synthetic Reflow induces a systematic contraction of the learned map, and that injecting real data into each round prevents this degeneration by maintaining a strictly positive operator-norm lower bound. We then use experiments on rectified flow models to test whether the qualitative trends predicted by the surrogate persist in practice and to extract practical guidance.

Contributions. The contributions of this paper are summarized as follows:

- **Theory.** We introduce a linear DAE Reflow surrogate and establish a concrete collapse mechanism: under self-consuming Reflow with only synthetic data, the operator norm of the end-to-end

linear map decays geometrically to zero, indicating progressive loss of representational power. We further prove that augmenting each Reflow round with a modest amount of real data prevents this degeneration by guaranteeing a uniform operator-norm lower bound. We discuss how these results connect back to Reflow in rectified flow models.

- **Empirical illustration.** We study Reflow in both toy settings and image benchmarks. Consistent with the theory, repeated Reflow rounds degrade fidelity and coverage, while real-data mixing stabilizes training and avoids degenerate behavior.
- **Practical implications.** Guided by the analysis, we evaluate simple real-data mixing schemes for Reflow, including an offline variant (RA) and two online variants (ORA and ORAS) that avoid storing image–noise pairs. Across CIFAR-10, ImageNet, and CelebA-HQ, these schemes preserve the sampling-speed benefits of Reflow while maintaining or slightly improving image quality, yielding lightweight, theory-informed recipes for using synthetic data in practice.

2. Preliminaries

In this section, we briefly review rectified flow and its Reflow procedure, which serve as the practical testbed in our experiments, and recall the connection between denoising autoencoders and diffusion or flow models. All new definitions and the precise Reflow setup that we analyze theoretically will be introduced in Section 3.

Notation. We denote by $\mathbb{R}^d$ the d-dimensional Euclidean space. Vectors are written in bold lower-case, for example $\boldsymbol{x} \in \mathbb{R}^d$, and matrices in bold upper-case, for example $X \in \mathbb{R}^{d\times n}$. For a matrix $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$, the columns $\boldsymbol{x}_i$ are individual data samples. For a vector $\boldsymbol{x}$, $\|\boldsymbol{x}\|$ denotes the Euclidean norm. For a matrix A, $\|A\|$ denotes the operator (spectral) norm, that is, the largest singular value, and $\|A\|_{\mathrm{F}}$ denotes the Frobenius norm. We write $\mathcal{N}(0, \Sigma)$ for a Gaussian distribution with mean zero and covariance Σ.

2.1. Rectified Flow and Reflow

Rectified Flow (RF) [7, 24, 26, 28] builds on flow and diffusion models by explicitly learning a time-dependent vector field that transports Gaussian noise to data. Compared with conventional diffusion samplers, RF aims to straighten probability flow trajectories, which enables sampling with a much smaller number of function evaluations (NFEs). In practice, RF is trained with a regression loss that asks a neural network to predict the velocity along simple interpolants between noise and data; we refer the reader to Lipman et al. [6], Liu et al. [7], Esser et al. [8], Albergo and Vanden-Eijnden [25] for full details.

The *Reflow* algorithm [7] further improves sampling efficiency by recursively refining the coupling between noise and data through self-generated pairs. Starting from an RF model trained on real data, Reflow repeatedly:

1. uses the current RF model to generate new noise–image pairs by integrating its learned ODE
2. retrains a new RF model on these pairs with the same regression loss.

Each Reflow round aims to produce straighter trajectories and thus reduce the NFEs required during sampling. We refer to the RF model after k rounds of Reflow as the k-*Reflow* model. Figure 3 illustrates this process in a two-dimensional multi-Gaussian example. Panel (A) shows how trajectories are progressively straightened by recursively retraining on self-generated pairs. In this work, we use the term *Reflow* more broadly to denote such iterative retraining of a generative model on its own synthetic samples, possibly mixed with real data. Rectified Flow is an important instance of this general Reflow paradigm and provides the main empirical testbed for our study of model collapse.

2.2. Denoising Autoencoders and Diffusion Models

Denoising autoencoders (DAEs) provide a convenient bridge between reconstruction-based training and score-based generative modeling. Given data $\boldsymbol{x} \in \mathbb{R}^d$ and Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, a DAE

$f_\theta : \mathbb{R}^d \to \mathbb{R}^d$ is trained to reconstruct clean inputs from noisy observations by minimizing

$$\mathcal{L}_{\mathrm{DAE}} = \mathbb{E}_{\boldsymbol{x}\sim p_1,\ \epsilon\sim\mathcal{N}(0,\sigma^2 I)} \big\| f_\theta(\boldsymbol{x}+\epsilon) - \boldsymbol{x} \big\|_2^2. \tag{1}$$

Under mild conditions, the residual between the output and the noisy input approximates a scaled score function [32]:

$$f_\theta(\boldsymbol{x}+\epsilon) - \boldsymbol{x} \approx \sigma^2 \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}+\epsilon), \tag{2}$$

so training a DAE implicitly performs score matching on a smoothed version of the data distribution. This observation underlies the close relationship between DAEs, diffusion models, and flow-based formulations [33, 34].

Recent work has further shown that, under appropriate parameterizations and affine reparameterizations of time, the training objectives of diffusion models and rectified flows can be unified [8, 35], and that flow matching losses and denoising losses can be interchanged or combined in practice [31]. Consequently, analyzing model collapse in a DAE formulation is informative for understanding collapse in rectified flow and related flow or diffusion systems. In Section 3, we make this connection precise in a simplified linear DAE Reflow model that is amenable to exact analysis.

3. Theoretical Analysis of Model Collapse in Reflow

Having introduced rectified flow, Reflow, and their connection to denoising autoencoders (DAEs) in Section 2, we now turn to a simplified model that is amenable to exact analysis. Our goal is to formalize how self-consuming Reflow affects the capacity of a model and to understand how incorporating real data changes this behavior. We focus on a linear DAE Reflow surrogate that captures the recursive self-training mechanism of Reflow while remaining analytically tractable.

3.1. Problem Setup: Linear DAE Reflow

We follow the setup of Pretorius et al. [36] and consider a two-layer linear DAE

$$f_\theta(\boldsymbol{x}) = \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{x}, \tag{3}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d'\times d}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{d\times d'}$ are weight matrices and $\theta = (\boldsymbol{W}_1, \boldsymbol{W}_2)$. The DAE is trained to reconstruct clean samples $\boldsymbol{x} \sim p_1$ from noisy observations $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{z}$, where $\boldsymbol{z} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$, by minimizing the population loss

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{x}\sim p_1,\ \boldsymbol{z}\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})} \big\| f_\theta(\boldsymbol{x}+\boldsymbol{z}) - \boldsymbol{x} \big\|_2^2. \tag{4}$$

Given a finite training set $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$, we learn the DAE by solving the empirical objective

$$\theta^\star(\boldsymbol{X}) := \arg\min_\theta \sum_{i=1}^n \mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})} \big\| f_\theta(\boldsymbol{x}_i+\boldsymbol{z}) - \boldsymbol{x}_i \big\|_2^2, \tag{5}$$

and we write $\theta^\star(\boldsymbol{X}) = (\boldsymbol{W}_1^\star(\boldsymbol{X}), \boldsymbol{W}_2^\star(\boldsymbol{X}))$ to emphasize the dependence on the data.

We assume that the data lie in a low-dimensional subspace. Concretely, we let

$$\boldsymbol{x}_i = \boldsymbol{U}^\star \boldsymbol{U}^{\star\top} \boldsymbol{a}_i, \quad \boldsymbol{a}_i \sim \mathcal{N}(0, \boldsymbol{I}), \quad \boldsymbol{U}^\star \in \mathbb{R}^{d\times r},$$

where the columns of $\boldsymbol{U}^\star$ form an orthonormal basis for the data subspace. Starting from the initial dataset $\boldsymbol{X}_1 = \boldsymbol{X}$, we define a Reflow procedure that repeatedly retrains the DAE on its own outputs.

Definition 3.1 (Self-consuming Reflow for a linear DAE)**.** Starting from $\boldsymbol{X}_1 = \boldsymbol{X}$, the j-th Reflow round performs:

1. **Fit DAE.** Set $(\boldsymbol{W}_2^j, \boldsymbol{W}_1^j) = \theta^\star(\boldsymbol{X}_j)$ by solving equation 5 on $\boldsymbol{X}_j$.
2. **Generate synthetic data.** Form the next training set

$$\boldsymbol{X}_{j+1} = \boldsymbol{W}_2^j \boldsymbol{W}_1^j (\boldsymbol{X}_j + \boldsymbol{E}_j), \tag{6}$$

where each column of the noise matrix $\boldsymbol{E}_j$ is sampled independently from $\mathcal{N}(0, \frac{\hat{\sigma}^2}{n}\boldsymbol{I})$.

The sequence $\{(\boldsymbol{X}_j, \boldsymbol{W}_2^j, \boldsymbol{W}_1^j)\}_{j\geq 1}$ is called the self-consuming Reflow trajectory of the linear DAE.

Intuitively, the expressive capacity of the linear DAE is controlled by the end-to-end map $\boldsymbol{W}_2^j \boldsymbol{W}_1^j$. If this map shrinks toward zero, then the DAE can no longer faithfully reconstruct variations in the data subspace, and almost all inputs are mapped close to a constant. In the following, we show that self-consuming Reflow drives the operator norm $\|\boldsymbol{W}_2^j \boldsymbol{W}_1^j\|$ to zero at a geometric rate, and we interpret this behavior as model collapse in the linear Reflow setting.

3.2. Model Collapse under Self-consuming Reflow

The next result characterizes the behavior of the linear DAE under the self-consuming Reflow procedure in Definition 3.1.

Theorem 3.2 (Collapse of linear DAE Reflow)**.** *In the self-consuming Reflow process of Definition 3.1, suppose that the variance of the added noise is not too large, that is, $\hat{\sigma} \leq C\sigma$ for some universal constant C. Then, with probability at least $1 - 2je^{-n}$, the operator norm of the end-to-end linear map $\boldsymbol{W}_2^j \boldsymbol{W}_1^j$ decays at a geometric rate:*

$$\left\|\boldsymbol{W}_2^j \boldsymbol{W}_1^j\right\|^2 \leq \frac{\|\boldsymbol{X}\|^2}{\sigma^2} \left(\frac{\|\boldsymbol{X}\|^2}{\|\boldsymbol{X}\|^2 + \sigma^2}\right)^{j-1} \quad \text{for all } j \geq 1, \tag{7}$$

Remark 3.3 (Why norm decay indicates collapse). For the linear DAE, the end-to-end map after round j is $F_j(\boldsymbol{x}) = \boldsymbol{W}_2^j \boldsymbol{W}_1^j \boldsymbol{x}$. If $\|\boldsymbol{W}_2^j \boldsymbol{W}_1^j\| \to 0$, then $\|F_j(\boldsymbol{x})\| \leq \|\boldsymbol{W}_2^j \boldsymbol{W}_1^j\| \, \|\boldsymbol{x}\| \to 0$ for all $\boldsymbol{x}$. Equivalently, $\|\boldsymbol{W}_2^j \boldsymbol{W}_1^j\|$ upper bounds the Lipschitz constant of F_j; as it vanishes, F_j becomes insensitive to input variations, reflecting a loss of expressiveness in this surrogate.

Theorem 3.2 therefore quantifies a concrete collapse mechanism for self-consuming Reflow: the learned denoising map contracts geometrically across rounds and approaches a degenerate mapping. We interpret this contraction-driven degeneration as a collapse mode in the linear DAE Reflow surrogate. Figure 2 visualizes the corresponding distributional contraction in a four-dimensional Gaussian example. While related collapse behaviors have been reported empirically for diffusion and flow models [9, 22], our theorem provides an explicit rate in a tractable Reflow surrogate.

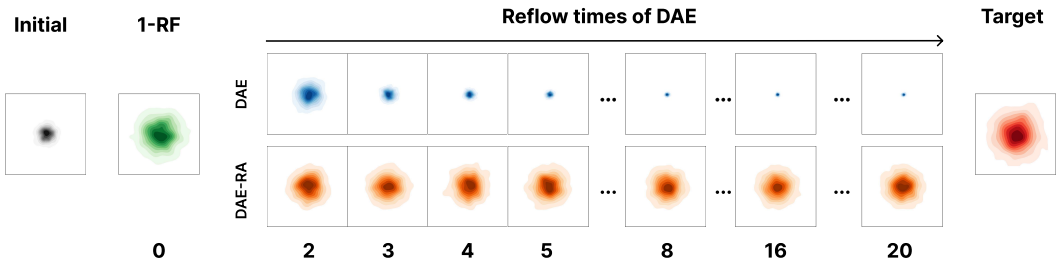


Figure 2: **Reflow Process of the DAE on a 4-D Gaussian Distribution.** The figure visualizes a slice of the distribution along dimensions 0 and 1. Both kernel density estimation plots and sample points are shown for the initial and target distributions.

3.3. Mitigating Collapse by Incorporating Real Data

Empirically, it has been observed that mixing real data into self-training can stabilize models and improve robustness [9, 20, 22]. Existing theoretical analyses, however, mostly focus on regression settings or maximum-likelihood objectives and do not explicitly model Reflow-type training loops [18–21]. In our linear DAE Reflow model, we can make this effect precise.

We modify the self-consuming Reflow in Definition 3.1 by augmenting the synthetic data with real samples at each round. Concretely, at round j we form an augmented dataset

$$\hat{\boldsymbol{X}}_j = [\boldsymbol{X}_j \quad \boldsymbol{X}], \tag{8}$$

and we fit the DAE on $\hat{\boldsymbol{X}}_j$ instead of $\boldsymbol{X}_j$, that is, we set $(\boldsymbol{W}_2^j, \boldsymbol{W}_1^j) = \theta^\star(\hat{\boldsymbol{X}}_j)$ in step 1 of Definition 3.1. The next result shows that this simple modification prevents collapse by maintaining a non-trivial lower bound on the operator norm.

Proposition 3.4 (Mitigating collapse with real data)**.** *In the Reflow process of Definition 3.1 with the modification described above, suppose that the variance of the added noise is not too large, that is, $\hat{\sigma} \le C\sigma$ for some universal constant C. Then, with probability at least $1 - 2je^{-n}$, the operator norm of the end-to-end map $\boldsymbol{W}_2^j \boldsymbol{W}_1^j$ is bounded away from zero:*

$$\left\|\boldsymbol{W}_2^j \boldsymbol{W}_1^j\right\|^2 \ge \frac{\|\boldsymbol{X}\|^2}{2\|\boldsymbol{X}\|^2 + \sigma^2} \quad \textit{for all } j \ge 1. \tag{9}$$

Compared with Theorem 3.2, Proposition 3.4 shows that augmenting each Reflow round with real data prevents geometric decay of the operator norm and keeps the DAE in a non-degenerate regime. In particular, the model retains a fixed amount of capacity to represent the data subspace and no longer exhibits the collapse behavior described above. Together, Theorem 3.2 and Proposition 3.4 formally capture a phenomenon that has been widely reported in practice: purely synthetic self-training tends to degenerate, while a modest but consistent supply of real data can stabilize Reflow and prevent model collapse.

In Section 4, we empirically illustrate these theoretical predictions on both synthetic toy problems and rectified flow models, and we use the insights from Proposition 3.4 to design real-data–augmented Reflow variants for practical generative modeling.

4. Experiments on Model Collapse and Its Mitigation in Reflow

We now empirically examine the main predictions of our analysis. The linear DAE theory in Section 3 makes two claims: (i) purely self-consuming Reflow induces a geometric loss of capacity, and (ii) mixing in a amount of real data prevents this collapse. Our experiments address three research questions (RQ):

- **RQ1:** Do linear and shallow DAEs exhibit the collapse and mitigation behaviors predicted by Theorems 3.2 and 3.4, and do these dynamics persist for a more RF-like, t-conditioned DAE?
- **RQ2:** On real image benchmarks, do Rectified Flow models trained with vanilla Reflow show empirical signs of model collapse, and can simple real-data mixing stabilize training over many Reflow iterations?
- **RQ3:** When applied on top of a strong RF baseline, does our real-data mixing scheme preserve?

4.1. A simple real-data mixing scheme for Rectified Flow

The linear DAE analysis in Section 3 suggests that Reflow remains stable as long as each self-training round receives a non-negligible amount of real data. We now instantiate a minimal real-data mixing scheme for practical Rectified Flow models that follows this principle.

A naive way to add real data would be to sample a real image $\boldsymbol{x}$ and couple it with an independent Gaussian noise $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, then treat $(\boldsymbol{z}, \boldsymbol{x})$ as a training pair. As illustrated in Figure 3B, this breaks the model-induced coupling between noise and data and introduces additional curvature into the learned transport, counteracting the straightening effect of Reflow. Instead, we construct *consistent* image–noise pairs by inverting the learned flow, so that the two endpoints of each pair are linked by the current model dynamics. Given a trained RF vector field v_θ, we build two types of pairs: (i)

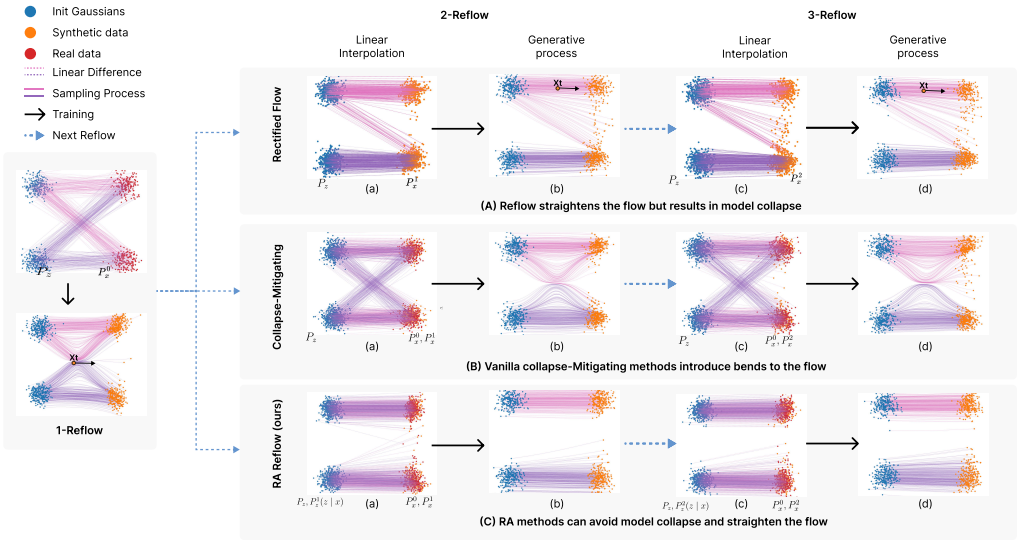
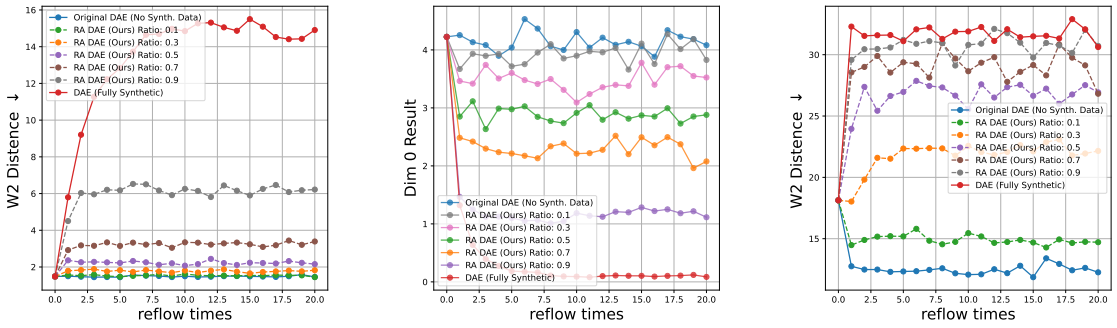


Figure 3: **2D multi-Gaussian experiment demonstration.** (A) Rectified Flow rewires trajectories to eliminate intersecting paths, transforming from (a) to (b). We then take noise samples from the distribution p^z and their corresponding generated samples from the synthetic distribution p_1^x to construct noise-target sample pairs (blue to orange) and linearly interpolate them at point (c). In Reflow, Rectified Flow is applied again from (c) to (c) to straighten the flows. This procedure is repeated recursively. (B) Since iterative training on self-generated data can cause MC, we can incorporate real data (shown in red) during training to prevent collapse. (C) However, adding real data introduces additional bends to the Rectified Flow because the pairs of real data and initial Gaussian samples are not pre-paired. Our method employs reverse sampling generated real-noise pairs (red to blue) to avoid MC while simultaneously straightening the flow.

synthetic pairs $(\boldsymbol{z}, \hat{\boldsymbol{x}})$ by sampling $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and solving the forward ODE $\hat{\boldsymbol{x}} = \mathrm{ODE}_{v_\theta}(0, 1, \boldsymbol{z})$, and (ii) *real reverse pairs* $(\hat{\boldsymbol{z}}, \boldsymbol{x})$ by taking real images $\boldsymbol{x}$ from the dataset and approximately inverting the model with the backward ODE $\hat{\boldsymbol{z}} = \mathrm{ODE}_{v_\theta}(1, 0, \boldsymbol{x})$. This construction (Figure 3C) preserves forward–backward consistency in the sense that each pair shares an endpoint with a trajectory generated (or inverted) by the same current field v_θ, avoiding the spurious coupling introduced by independent $(\boldsymbol{z}, \boldsymbol{x})$ pairing.

RA Reflow. At each Reflow round, we train on a mixture of pair datasets formed by the union of synthetic pairs and real reverse pairs. Concretely, each minibatch is assembled by drawing a fraction λ of samples from the synthetic-pair set and a fraction $1 - \lambda$ from the real-reverse-pair set, and we minimize the standard RF loss over this minibatch. Equivalently, the training objective takes expectation over a mixture distribution on pairs, $\lambda\mathcal{D}_{\mathrm{syn}} + (1-\lambda)\mathcal{D}_{\mathrm{rev}}$, where $\mathcal{D}_{\mathrm{syn}}$ and $\mathcal{D}_{\mathrm{rev}}$ denote the empirical distributions of synthetic and real-reverse pairs, respectively. Importantly, our mixing is set-level (sampling from a mixture over pairs) rather than sample-level convex interpolation between endpoints. We refer to this offline set-mixing scheme as Real-data Augmented Reflow (RA Reflow). Unless noted otherwise, we use $\lambda \approx 0.5$ in our image experiments. Further details on pseudocode and hyperparameters are given in Appendix C.

ORA and ORAS Reflow. For large-scale RF models, storing all image–noise pairs is impractical. To address this, we also consider online variants that regenerate pairs on the fly. In *Online Real-data Augmented Reflow* (ORA Reflow), synthetic and real reverse pairs are periodically refreshed during training rather than stored for the entire Reflow round. To further improve diversity, *Online Real-data Augmented Stochastic Reflow* (ORAS Reflow) replaces the backward ODE with a backward SDE and adds a small amount of reverse-time noise when constructing real reverse pairs.



(a) W2 distance (b) Dim 0 (c) W2 distance

Figure 4: **Reflow experiments with DAEs on synthetic Gaussian data.** The first two panels show results on a 4D Gaussian using a standard DAE; the third panel shows results on a 10D Gaussian using a t-conditioned DAE that approximates Rectified Flow behavior.

4.2. Experimental setup

Models. For RQ1, we use (i) a two-layer linear DAE and (ii) a shallow t-conditioned DAE with SiLU that maps a Gaussian prior to a low-dimensional Gaussian target, providing an RF-like surrogate. For RQ2 and RQ3, we consider three RF families:(i)**U-Net RF.** A U-Net Rectified Flow model on CIFAR-10, with and without improvements such as EDM initialization and perceptual losses, following Lee et al. [30], Tong et al. [37]. (ii) **Latent RF.** A Latent Flow Matching model with a DiT backbone on CelebA-HQ, using the publicly available LFM configuration [38]. All models within the same setting share the same backbone, training schedule and random seed; method-specific hyperparameters are tuned on a held-out validation split. Detailed architectures and optimization settings are reported in Appendix D.

Datasets and metrics. We use synthetic Gaussian tasks (Section 4.3), CIFAR-10 (32×32) [39], ImageNet 64×64 [40], and CelebA-HQ 256×256 [41]. Distributional fidelity is measured with Fréchet Inception Distance (**FID**) and the 2-Wasserstein distance (**W2**); lower is better for both. For image models we also report Precision and Recall (**Prec., Rec.**) to capture fidelity-diversity tradeoffs. Sampling cost is measured by the number of function evaluations (**NFE**) per sample.

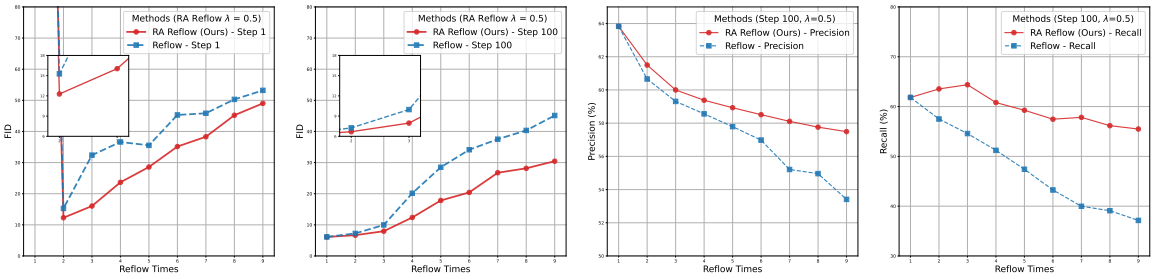
4.3. RQ1: Linear and t-conditioned DAEs

For RQ1, we map an initial Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$ to a 4-dimensional target $\mathcal{N}(\mathbf{0}, 5\mathbf{I})$ with a standard DAE, and to a 10-dimensional target $\mathcal{N}(\mathbf{0}, 2\mathbf{I})$ with a t-conditioned DAE. Figure 2 visualizes the evolution of the learned distributions, and Figure 4 summarizes W2 and variance statistics across Reflow rounds.

Obs. 1 (collapse under pure self-training). Under purely synthetic recursion, DAE Reflow progressively contracts the generated distribution: variance shrinks and the effective map loses rank, driving the output toward a point mass. The t-conditioned DAE exhibits the same qualitative failure mode, with repeated Reflow drifting toward the prior and the effective update degenerating (see Appendix B.4).

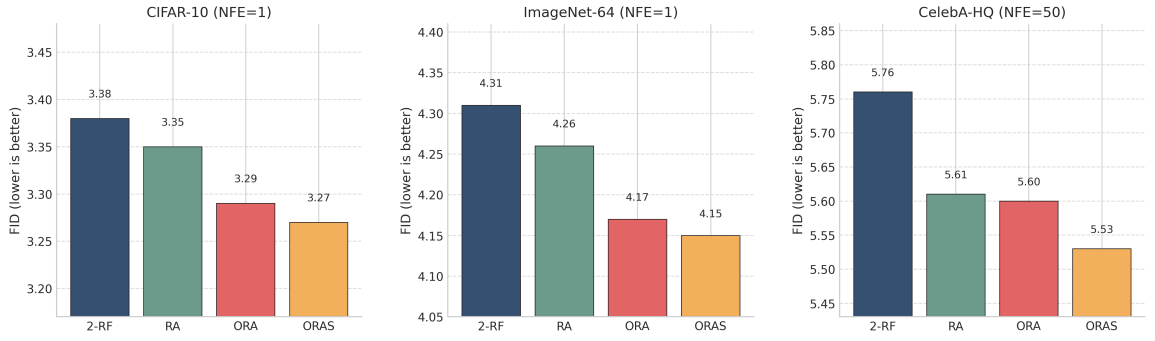
Obs. 2 (real data prevents degeneration). Mixing real samples each round stabilizes the principal components and yields uniformly lower W2 than the fully synthetic recursion, consistent with the operator-norm lower-bound behavior in Proposition 3.4.

Takeaway. Across both linear and t-conditioned DAEs, repeated Reflow collapses under purely synthetic recursion, while a constant real-data fraction stabilizes training, consistent with Theorem 3.2 and Proposition 3.4.



(a) FID, NFE = 1 (b) FID, NFE = 100 (c) Precision, NFE = 100 (d) Recall, NFE =100

Figure 5: **CIFAR-10 Reflow: vanilla RF versus real-data mixing.** Half-scale U-Net RF with mixing ratio $\lambda = 0.5$ and refresh parameter $\alpha = 8$. Vanilla Reflow degrades over iterations, whereas the real-data mixing scheme stabilizes FID and preserves diversity.



(a) CIFAR-10, NFE = 1 (b) ImageNet 64, NFE = 1 (c) CelebA-HQ, NFE = 50

Figure 6: **Summary of experiments on three image datasets.** Each panel reports FID (lower is better) for a two-round RF baseline and its real-data mixed variants.

4.4. RQ2: Reflow behavior on image data

We next examine whether similar patterns appear in RF models on real images. We consider (i) a half-scale U-Net RF on CIFAR-10 with multiple Reflow rounds (Fig. 5), and (ii) a DiT-based latent flow model on CelebA-HQ (Fig. 6c). In both cases, we compare vanilla Reflow with the real-data mixing scheme introduced in Section 4.1.

Obs. 4 (CIFAR-10: collapse versus stabilization). On CIFAR-10, vanilla Reflow degrades steadily over rounds: one-step FID rises rapidly and Precision/Recall deteriorate, indicating simultaneous loss of fidelity and coverage. In contrast, under the same backbone and training budget, real-data mixing keeps FID and diversity metrics stable across Reflow rounds, with Recall remaining non-collapsed.

Obs. 6 (CelebA-HQ). On CelebA-HQ, the same mixing principle yields a small but consistent FID improvement across sampling budgets.

Takeaway. The collapse and mitigation patterns from the DAE surrogate persist in RF on images: a modest but persistent supply of real data is sufficient to halt the degradation induced by repeated Reflow.

4.5. RQ3: Strong RF baselines with online variants

We finally evaluate the same mixing principle on top of a strong RF configuration that follows RF++ [30]. A compact summary on CIFAR-10 and ImageNet 64×64 is shown in Fig. 6a and b.

Obs. 7 (mixing transfers to strong baselines). Real-data mixing maintains or slightly improves FID relative to the two-round RF++ baseline across datasets. The offline variant RA matches or improves

FID with fewer image-noise pairs, while the online variants ORA and ORAS achieve the best FID in each setting.

5. Conclusion

We studied model collapse in Reflow through a tractable linear DAE surrogate. The analysis identifies a simple mechanism: purely synthetic self-training leads to geometric contraction of the learned end-to-end map, with the operator norm decaying across rounds and reducing representational capacity. This perspective also explains why collapse can emerge even when the per-round training loss keeps decreasing.

We also prove a constructive remedy. Injecting a non-negligible fraction of real-data-anchored pairs at each round prevents contraction by keeping the operator norm bounded away from zero. Guided by these insights, we instantiated minimal real-data mixing schemes for practical Rectified Flow models and verified the predicted collapse and mitigation on Gaussian tasks and image benchmarks, including strong settings. Overall, our results suggest a practical design principle for recursive Reflow pipelines: use real data as an anchor during training, rather than only as a post-hoc refinement.

Future work includes extending the theory beyond linear surrogates, accounting for imperfect inversion and solver discretization, and developing adaptive schedules that allocate real pairs more selectively while preserving the low-NFE benefits of Reflow. We also plan to study how stabilizing instantaneous-field learning interacts with downstream low-NFE objectives such as one-step generators.

Acknowledgments

HZ and ZZ acknowledge support from NSF grants IIS-2402952 and ECCS-2409701.

References

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

[2] 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.

[3] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

[4] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[5] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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

[7] 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.

[8] 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. In *Forty-first International Conference on Machine Learning*, 2024.

[9] Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard G. Baraniuk. Self-consuming generative models go mad. *arXiv preprint arxiv:2307.01850*, 2023.

[10] Andrew J Peterson. Ai and the problem of knowledge collapse. *AI & SOCIETY*, pages 1–21, 2025.

[11] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024.

[12] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arxiv:2305.17493*, 2023.

[13] Martin Briesch, Dominik Sobania, and Franz Rothlauf. Large language models suffer from their own output: An analysis of the self-consuming training loop, 2023.

[14] Ryuichiro Hataya, Han Bao, and Hiromi Arai. Will large-scale generative models corrupt future datasets? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20555–20565, October 2023.

[15] Max F. Burg, Florian Wenzel, Dominik Zietlow, Max Horn, Osama Makansi, Francesco Locatello, and Chris Russell. Image retrieval outperforms diffusion models on data augmentation, 2023.

[16] Matyas Bohacek and Hany Farid. Nepotistically trained generative-ai models collapse, 2023.

[17] Sina Alemohammad, Ahmed Imtiaz Humayun, Shruti Agarwal, John Collomosse, and Richard Baraniuk. Self-improving diffusion models with synthetic data, 2024. URL `https://arxiv.org/abs/2408.16333`.

[18] Shi Fu, Sen Zhang, Yingjie Wang, Xinmei Tian, and Dacheng Tao. Towards theoretical understandings of self-consuming generative models. *arXiv preprint arXiv:2402.11778*, 2024.

[19] Elvis Dohmatob, Yunzhen Feng, and Julia Kempe. Model collapse demystified: The case of regression, 2024.

[20] Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, et al. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. *arXiv preprint arXiv:2404.01413*, 2024.

[21] Elvis Dohmatob, Yunzhen Feng, Arjun Subramonian, and Julia Kempe. Strong model collapse. *arXiv preprint arXiv:2410.04840*, 2024.

[22] Quentin Bertrand, Avishek Joey Bose, Alexandre Duplessis, Marco Jiralerspong, and Gauthier Gidel. On the stability of iterative retraining of generative models on their own data. *arXiv preprint arxiv:2310.00429*, 2023.

[23] Rylan Schaeffer, Joshua Kazdan, Alvan Caleb Arulandu, and Sanmi Koyejo. Position: Model collapse does not mean what you think. *arXiv preprint arXiv:2503.03150*, 2025.

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

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

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

[27] Yuanzhi Zhu, Xingchao Liu, and Qiang Liu. Slimflow: Training smaller one-step diffusion models with rectified flow. In *European Conference on Computer Vision*, pages 342–359. Springer, 2024.

[28] Fu-Yun Wang, Ling Yang, Zhaoyang Huang, Mengdi Wang, and Hongsheng Li. Rectified diffusion: Straightness is not your need in rectified flow. *arXiv preprint arXiv:2410.07303*, 2024.

[29] Kim Shin Seong, Mingi Kwon, Jaeseok Jeong, and Youngjung Uh. Balanced conic rectified flow.

[30] Sangyun Lee, Zinan Lin, and Giulia Fanti. Improving the Training of Rectified Flows, May 2024. URL `http://arxiv.org/abs/2405.20320`. arXiv:2405.20320 [cs].

[31] Beomsu Kim, Yu-Guan Hsieh, Michal Klein, Marco Cuturi, Jong Chul Ye, Bahjat Kawar, and James Thornton. Simple reflow: Improved techniques for fast flow models. *arXiv preprint arXiv:2410.07815*, 2024.

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

[33] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

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

[35] Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin Patrick Murphy, and Tim Salimans. Diffusion models and gaussian flow matching: Two sides of the same coin. In *The Fourth Blogpost Track at ICLR 2025*.

[36] Arnu Pretorius, Steve Kroon, and Herman Kamper. Learning dynamics of linear denoising autoencoders. In *International Conference on Machine Learning*, pages 4141–4150. PMLR, 2018.

[37] Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

[38] Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

[39] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[40] 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.

[41] Tero Karras. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[42] Gonzalo Martínez, Lauren Watson, Pedro Reviriego, José Alberto Hernández, Marc Juarez, and Rik Sarkar. Combining generative artificial intelligence (ai) and the internet: Heading towards evolution or degradation? *arXiv preprint arxiv: 2303.01255*, 2023.

[43] Gonzalo Martínez, Lauren Watson, Pedro Reviriego, José Alberto Hernández, Marc Juarez, and Rik Sarkar. Towards understanding the interplay of generative artificial intelligence and the internet. *arXiv preprint arxiv: 2306.06130*, 2023.

[44] Yanzhu Guo, Guokan Shang, Michalis Vazirgiannis, and Chloé Clavel. The curious decline of linguistic diversity: Training language models on synthetic text, 2023.

[45] Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34:17695–17709, 2021.

[46] Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion schrödinger bridge matching. *Advances in Neural Information Processing Systems*, 36:62183–62223, 2023.

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

[48] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020.

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

[50] Shaobin Zhuang, Zhipeng Huang, Binxin Yang, Ying Zhang, Fangyikang Wang, Canmiao Fu, Chong Sun, Zheng-Jun Zha, Chen Li, and Yali Wang. Get in video: Add anything you want to the video. *arXiv preprint arXiv:2503.06268*, 2025.

[51] Zhichao Chen, Haoxuan Li, Fangyikang Wang, Odin Zhang, Hu Xu, Xiaoyu Jiang, Zhihuan Song, and Hao Wang. Rethinking the diffusion models for missing data imputation: A gradient flow perspective. *Advances in Neural Information Processing Systems*, 37:112050–112103, 2024.

[52] Fangyikang Wang, Huminhao Zhu, Chao Zhang, Hanbin Zhao, and Hui Qian. Gad-pvi: A general accelerated dynamic-weight particle-based variational inference framework. In *AAAI*, 2024.

[53] Jiahang Tu, Wei Ji, Hanbin Zhao, Chao Zhang, Roger Zimmermann, and Hui Qian. Driveditfit: Fine-tuning diffusion transformers for autonomous driving data generation. *ACM Transactions on Multimedia Computing, Communications and Applications*, 21(3):1–29, 2025.

[54] Jiahang Tu, Hao Fu, Fengyu Yang, Hanbin Zhao, Chao Zhang, and Hui Qian. Texttoucher: Fine-grained text-to-touch generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 7455–7463, 2025.

[55] Qian Feng, JiaHang Tu, Mintong Kang, Hanbin Zhao, Chao Zhang, and Hui Qian. Fg-oriu: Towards better forgetting via feature-gradient orthogonality for incremental unlearning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1957–1967, 2025.

[56] Eric Wang, Hao, Licheng Pan, Yuan Lu, Zhixuan Chu, Xiaoxi Li, Shuting He, Zhichao Chen, Haoxuan Li, Qingsong Wen, and Zhouchen Lin. Distdf: Time-series forecasting needs joint-distribution wasserstein alignment. In *Proc. Int. Conf. Learn. Represent.*, pages 1–24, 2026.

[57] Fangyikang Wang, Hubery Yin, Yue-Jiang Dong, Huminhao Zhu, Hanbin Zhao, Hui Qian, Chen Li, et al. Belm: Bidirectional explicit linear multi-step sampler for exact inversion in diffusion models. *Advances in Neural Information Processing Systems*, 37:46118–46159, 2024.

[58] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

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

[60] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6613–6623, 2024.

[61] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

[62] Huijie Zhang, Aliaksandr Siarohin, Willi Menapace, Michael Vasilkovsky, Sergey Tulyakov, Qing Qu, and Ivan Skorokhodov. Alphaflow: Understanding and improving meanflow models. *arXiv preprint arXiv:2510.20771*, 2025.

[63] Fangyikang Wang, Hubery Yin, Lei Qian, Yinan Li, Shaobin Zhuang, Huminhao Zhu, Yilin Zhang, Yanlong Tang, Chao Zhang, Hanbin Zhao, et al. Unleashing high-quality image generation in diffusion sampling using second-order levenberg-marquardt-langevin. *arXiv preprint arXiv:2505.24222*, 2025.

[64] Fangyikang Wang, Hubery Yin, Shaobin Zhuang, Huminhao Zhu, Yinan Li, Lei Qian, Chao Zhang, Hanbin Zhao, Hui Qian, and Chen Li. Efficiently access diffusion fisher: Within the outer product span space. *arXiv preprint arXiv:2505.23264*, 2025.

[65] Xinxi Zhang, Shiwei Tan, Quang Nguyen, Quan Dao, Ligong Han, Xiaoxiao He, Tunyu Zhang, Alen Mrdovic, and Dimitris Metaxas. Flow straighter and faster: Efficient one-step generative modeling via meanflow on rectified trajectories. *arXiv preprint arXiv:2511.23342*, 2025.

[66] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252. PMLR, 2023.

[67] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[68] 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.

[69] Huijie Zhang, Jinfan Zhou, Yifu Lu, Minzhe Guo, Peng Wang, Liyue Shen, and Qing Qu. The emergence of reproducibility and consistency in diffusion models. In *Forty-first International Conference on Machine Learning*, 2023.

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

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

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

A. Related Work

A.1. Model Collapse in Generative Models

Model collapse has recently attracted considerable attention. Empirical signs of collapse have been reported for both language and vision systems [13, 14, 16, 42–44]. Theoretical accounts attribute the phenomenon to sampling bias and error amplification [12, 19]. Several papers show that adding real data can prevent model collapse [9, 18, 20, 22]. Bertrand et al. [22] shows that adding real data can mathematically stabilise maximum-likelihood training, which is relevant to the diffusion models, but they do not formalise MC in that setting instead, they reprise the Gaussian toy example of Shumailov et al. [12] to show the collapse. Their guarantee therefore delivers conflicting messages with the negative result of Dohmatob et al. [21], who prove that even a tiny synthetic fraction can still provoke collapse. Schaeffer et al. [23] analyzes this contradiction and traces it to the papers' incompatible collapse definitions. We try to make progress toward this gap by giving a formal definition of MC for Reflow, providing the first DAE-based analysis, and proposing an efficient remedy that preserves Rectified Flow's sampling advantages.

A.2. Real-data augmentation for Rectified Flow

The idea of coupling synthetic trajectories with real samples predates Reflow and can be traced to Schrödinger-bridge methods [45, 46]. In particular, iterative Markovian fitting (IMF) [46] formally interpolates two marginals with a Brownian bridge; when the bridge noise is set to zero, the path becomes linear and coincides with Reflow. Our study, however, adopts a different goal and viewpoint: we analyse how real samples counteract *model collapse*. Within the Rectified Flow literature, several heuristic schemes also inject real data to boost post-Reflow quality. Lee et al. [30] performs an additional image-alignment stage with real data after Reflow has converged. Concurrent with an early version of this paper, independent works [29, 31] employed real images to a similar effect. Balanced Conic Rectified Flow [29] projects onto real data in a manner reminiscent of IMF, but does not supply theoretical insight, while Kim et al. [31] propose a related correction aimed specifically at compensating for EDM initialisation, where the unavailability of $T{=}1$ real samples forces the use of $T{=}0.999$ surrogates and biases Reflow. Distinct from these efforts, we show that real data mitigate *model collapse* itself—a failure mode previously unaddressed in the Reflow setting—and we introduce principled variants that preserve sampling efficiency while guaranteeing stability. This stabilization of the instantaneous field is also compatible with downstream low-NFE objectives (e.g., one-step mean-velocity modeling), which can benefit from a higher-quality and more stable continuous-time field.

A.3. Low-NFE and one-step generative modeling

Modern generative modeling has increasingly shifted toward diffusion and flow-based formulations, which offer stable training and strong sample quality [33, 34, 47–56]. Recent progress in diffusion and flow-based generative modeling has increasingly emphasized low-NFE sampling, ranging from few-step solvers to one-step generators. [57–64] Among one-step approaches, Geng et al. [61] propose MeanFlow, a self-contained framework that targets average (time-aggregated) velocity fields for one-step generation, in contrast to flow-matching and Rectified Flow methods that learn the instantaneous velocity field $v_\theta(x, t)$ used in continuous-time ODE/SDE sampling. MeanFlow derives an identity linking average and instantaneous velocities and trains a one-step model without relying on a pre-trained teacher or explicit distillation, achieving strong empirical performance in the 1-NFE regime.

This line of work is complementary to ours. Our focus is on the stability of learning the instantaneous field under recursive self-training (Reflow) and on how real-data anchoring mitigates model collapse in this field-learning process. MeanFlow, by contrast, primarily addresses a downstream objective: given an (approximately) learned continuous-time dynamics, how to obtain an accurate one-step generator. Notably, very recent work has begun to combine these ideas by applying MeanFlow-style

Methods	Variant	Eff. Sampling	Collapse Mitigating	Theoretical basis
Diffusion/ Flow Model	**RA Reflow (Ours)**	✓	✓	✓
	DDPM [47]	✗	✗	✓
	FM [6]	✓(Weak)	✗	✓
	OTCFM [37]	✓(Weak)	✗	✓
	RF [7]	✓	✗	✓
Distillation	CD/CT [66]	✓(1-step)	Unknown	✓
Collapse Mitigating	MAD [9]	✗	✓	✗
	Stability [22]	✗	✓	✓
	MCI [20]	N/A	✓	✓
	MCD [19]	N/A	✓	✓
Improved RF	Bonic [29]	✓	Unknow	✗
	RF++ [30]	✓	✗	✗
	SimpleReflow [31]	✓	Unknow	✗

Table 1: Comparison of methods regarding efficient sampling and model collapse mitigating. ✓ and ✗ indicate feature presence or absence; "Weak" denotes limited capability, and "Unknown" or "N/A" means insufficient information or not applicable.

objectives on rectified trajectories, reporting improved one-step generation and suggesting a synergy between trajectory rectification and mean-velocity learning [65]. These results further motivate our study: stabilizing the instantaneous-field learning stage can be beneficial not only for multi-step sampling, but also for subsequent one-step compression.

B. Proofs and Formulations

B.1. Proof of Theorem 3.2

Proof of Theorem 3.2. We can first expand the training loss in equation **??** as follows:

$$\begin{aligned}\mathcal{L}(\theta) &= \mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,\sigma^2 I)}\left[\|\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}_i-\boldsymbol{x}_i\|^2 - 2\langle \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}_i-\boldsymbol{x}_i, \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{z}\rangle + \|\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{z}\|_2^2\right], \\ &= \sum_{i=1}^{n}\|\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}_i-\boldsymbol{x}_i\|^2 + \sigma^2\|\boldsymbol{W}_2\boldsymbol{W}_1\|_F^2. \end{aligned} \tag{10}$$

We denote by $\boldsymbol{\Phi} = \boldsymbol{W}_2\boldsymbol{W}_1$ to simplify the following analysis. The induced ℓ_2 regularization in equation 10 suggests that DAE performs denoising by learning a low-dimensional model. The optimal solution for equation 10, written in terms of $\boldsymbol{\Phi}$, is simply given by $(\boldsymbol{X}\boldsymbol{X}^\top)(\boldsymbol{X}\boldsymbol{X}^\top+\sigma^2 I)^{-1}$. When $\sigma\to 0$, the solution converges to PCA. Plugging this into the process of recursively learning DAE from generational data, we have $\boldsymbol{\Phi}_j = \boldsymbol{W}_2^j\boldsymbol{W}_1^j = (\boldsymbol{X}_j\boldsymbol{X}_j^\top)(\boldsymbol{X}_j\boldsymbol{X}_j^\top+\sigma^2 I)^{-1}$.

Let $\lambda(\cdot)$ denote the largest eigenvalue of a matrix. Since $\boldsymbol{X}_{j+1} = \boldsymbol{\Phi}_j(\boldsymbol{X}_j+\boldsymbol{E}_j)$ with each column of $\boldsymbol{E}_j$ being iid sampled from $\mathcal{N}(0,\hat{\sigma}^2/nI)$, it follows from [67][Theorem 4.6.1] that there exists a constant C such that, with probability at least $1-2e^{-n}$, $\lambda(\boldsymbol{X}_{j+1}\boldsymbol{X}_{j+1}^\top) \le \lambda^2(\boldsymbol{\Phi}_j)(\lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)+C\hat{\sigma}^2)$. This together with $\lambda(\boldsymbol{\Phi}_j) = \frac{\lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)}{\lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)+\sigma^2}$ implies that when $\hat{\sigma}^2 \le \sigma^2/C$,

$$\lambda(\boldsymbol{X}_{j+1}\boldsymbol{X}_{j+1}^\top) \le \lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)\lambda(\boldsymbol{\Phi}_j)\frac{\lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)+C\hat{\sigma}^2}{\lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)+\sigma^2} \le \lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)\lambda(\boldsymbol{\Phi}_j) \tag{11}$$

holds with probability at least $1-2e^{-n}$. Denote by $\tau = \lambda(\boldsymbol{X}\boldsymbol{X}^\top)$. In the following, we prove that with probability at least $1-2qe^{-n}$,

$$\left[\lambda(\boldsymbol{X}_q\boldsymbol{X}_q^\top)\right] \le \lambda(\boldsymbol{X}\boldsymbol{X}^\top)(\frac{\tau}{\tau+\sigma^2})^{q-1}. \tag{12}$$

We prove this by induction. It holds when $q=0$. Now assume equation 12 is true at $q=j$. We prove it also holds at $q=j+1$. Since equation 12 holds at j, we have $\lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top) \le \lambda(\boldsymbol{X}\boldsymbol{X}^\top)$, and hence $\lambda(\boldsymbol{\Phi}_j) = \frac{\lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)}{\lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)+\sigma^2} \le \frac{\tau}{\tau+\sigma^2}$. Plugging this into equation 11 gives

$$\lambda(\boldsymbol{X}_{j+1}\boldsymbol{X}_{j+1}^\top) \le \lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)\lambda(\boldsymbol{\Phi}_j) \le \lambda(\boldsymbol{X}\boldsymbol{X}^\top)(\frac{\tau}{\tau+\sigma^2})^j.$$

This proves equation 12. Finally, we can obtain the bound for $\lambda(\boldsymbol{\Phi}_j)$ as

$$\lambda(\boldsymbol{\Phi}_j) = \frac{\lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)}{\lambda(\boldsymbol{X}_j\boldsymbol{X}_j^\top)+\sigma^2} \le \frac{\lambda(\boldsymbol{X}\boldsymbol{X}^\top)(\frac{\tau}{\tau+\sigma^2})^{j-1}}{\lambda(\boldsymbol{X}\boldsymbol{X}^\top)(\frac{\tau}{\tau+\sigma^2})^{j-1}+\sigma^2} \le \frac{\lambda(\boldsymbol{X}\boldsymbol{X}^\top)}{\sigma^2}(\frac{\tau}{\tau+\sigma^2})^{j-1}.$$

□

B.2. Detailed Explanation of the Gap Between Diffusion Models and DAEs

In this appendix, we delve deeper into the connection between diffusion models and sequences of Denoising Autoencoders (DAEs), focusing on the initial step of the diffusion process.

Consider a diffusion model $f_\theta(t,\boldsymbol{x}_t)$ with T time steps (e.g., $T=1000$), which begins the sampling process from pure Gaussian noise $\boldsymbol{x}_0\sim\mathcal{N}(0,\mathbf{I})$. The model predicts the target state using (here we consider the image x-prediction which is equal to noise ϵ-prediction and velocity v-prediction [59]):

$$\boldsymbol{x}_1 = f_\theta(0,\boldsymbol{x}_0), \tag{13}$$

where $f_\theta(0,\boldsymbol{x}_0)$ approximates the denoising function at time $t=0$. This step functions as a DAE with pure Gaussian input.

Subsequent sampling steps involve Euler updates of the form:

$$\begin{aligned} \boldsymbol{x}_{0+\gamma} &= \boldsymbol{x}_0 + \gamma \left(f_\theta(0, \boldsymbol{x}_0) - \boldsymbol{x}_0 \right) \\ &\cdots \\ \boldsymbol{x}_{t+\gamma} &= \boldsymbol{x}_t + \gamma \left(f_\theta(t, \boldsymbol{x}_t) - \boldsymbol{x}_t \right), \end{aligned} \tag{14}$$

where γ is a small time increment. In these steps, each input $\boldsymbol{x}_t$ is a mixture of Gaussian noise and previous model outputs, aligning with the typical input to a DAE trained on such mixtures.

The only significant gap between a sequence of DAEs and the diffusion model arises in the initial step due to the pure Gaussian input. By analyzing the initial step separately, we can better align the recursive DAE framework with the diffusion model. Specifically, if we consider the initial DAE handling pure Gaussian inputs and subsequent DAEs processing mixtures of noise and signal, the entire diffusion process can be viewed as a series of DAEs with varying input distributions.

However, an important question arises: *Will a linear DAE learn any meaningful information from the first step with pure Gaussian input?* In the case of a linear DAE, learning from pure noise is challenging because there is no underlying structure to capture. This limitation highlights why the initial step differs from the rest and underscores the necessity of separating its analysis.

By acknowledging this gap, our analysis becomes more comprehensive, bridging the understanding between DAEs and diffusion models. This perspective not only sheds light on the mechanics of diffusion models but also provides a pathway for leveraging insights from DAE analysis to improve diffusion-based generative models.

B.3. Proof of Proposition 2

Now, we formulate the reflow process of a linear DAE incorporating real data. Recall the settings from 3.1; suppose we have training data $\boldsymbol{X} = [\boldsymbol{x}_1 \quad \cdots \quad \boldsymbol{x}_n]$ with $\boldsymbol{x}_i = \boldsymbol{U}^\star \boldsymbol{U}^{\star\top} \boldsymbol{a}_i$, where $\boldsymbol{a}_i \sim \mathcal{N}(0, \mathbf{I})$. Starting with $\boldsymbol{X}_1 = \boldsymbol{X}$, the scheme for generating synthetic data at the j-th iteration ($j \geq 1$) is outlined as follows.

- **Add real data**: $\hat{\boldsymbol{X}}_j = [\boldsymbol{X}_j \quad \boldsymbol{X}]$.
- **Fit DAE**: $(\boldsymbol{W}_2^j, \boldsymbol{W}_1^j) = \theta^\star(\hat{\boldsymbol{X}}_j)$ by solving equation **??** with training data $\hat{\boldsymbol{X}}_j$.
- **Generate synthetic data for the next iteration**: $\boldsymbol{X}_{j+1} = \boldsymbol{W}_2^j \boldsymbol{W}_1^j (\boldsymbol{X}_j + \boldsymbol{E}_j)$, where each column of the noise matrix $\boldsymbol{E}_j$ is i.i.d. sampled from $\mathcal{N}(0, \hat{\sigma}^2/n^2 \mathbf{I})$.

First, we examine the effect of incorporating real data into the training process. Let $\lambda(\cdot)$ denote the largest eigenvalue of a matrix and $\lambda_{\min}(\cdot)$ denote the smallest eigenvalue of a matrix.

Lemma B.1. *Let $X_j, X_0 \in \mathbb{R}^{n \times d}$ be given matrices, and define the block matrix.*

$$\hat{\boldsymbol{X}}_j = [X_j \quad X_0].$$

Then the maximum eigenvalue of $\hat{\boldsymbol{X}}_j \hat{\boldsymbol{X}}_j^\top$ satisfies the following inequalities:

$$\lambda_{\min}(X_j X_j^\top) + \lambda(X_0 X_0^\top) \leq \lambda(\hat{\boldsymbol{X}}_j \hat{\boldsymbol{X}}_j^\top) \leq \lambda(X_j X_j^\top) + \lambda(X_0 X_0^\top).$$

Proof. First, observe that

$$\hat{X}_j \hat{X}_j^\top = X_j X_j^\top + X_0 X_0^\top. \tag{15}$$

We aim to bound $\lambda(\hat{X}_j \hat{X}_j^\top)$ using the eigenvalues of $X_j X_j^\top$ and $X_0 X_0^\top$. Recall that both $X_j X_j^\top$ and $X_0 X_0^\top$ are symmetric positive semi-definite matrices.

Upper Bound:

Using Weyl's inequality for eigenvalues of Hermitian matrices, we have

$$\lambda(A+B) \leq \lambda(A) + \lambda(B), \tag{16}$$

where A and B are symmetric matrices.

Applying this to $A = X_j X_j^\top$ and $B = X_0 X_0^\top$, we obtain

$$\lambda(\hat{X}_j \hat{X}_j^\top) \leq \lambda(X_j X_j^\top) + \lambda(X_0 X_0^\top).$$

Lower Bound:

Similarly, Weyl's inequality provides a lower bound:

$$\lambda(A+B) \geq \lambda_{\min}(A) + \lambda(B). \tag{17}$$

Applying this to $A = X_j X_j^\top$ and $B = X_0 X_0^\top$, we have

$$\lambda(\hat{X}_j \hat{X}_j^\top) \geq \lambda_{\min}(X_j X_j^\top) + \lambda(X_0 X_0^\top).$$

Combining the upper and lower bounds from Equations equation 16 and equation 17, we establish the inequalities in Equation equation B.1, thus proving the lemma. □

Proposition B.2. *In the above synthetic data generation process 3.1 with adding real data, suppose that the variance of the added noise is not too large, i.e., $\hat{\sigma} \leq C\sigma$ for some universal constant C. Then, with probability at least $1 - 2je^{-n}$, the learned DAE does not suffer from model collapse as*

$$\|\boldsymbol{W}_2^j \boldsymbol{W}_1^j\|^2 \geq \frac{\|\boldsymbol{X}\|^2}{2\|\boldsymbol{X}\|^2 + \sigma^2}. \tag{18}$$

Proof. Following an analysis similar to the proof of Theorem 3.2, we have

$$\boldsymbol{\Phi}_j = (\hat{\mathbf{X}}_j \hat{\mathbf{X}}_j^\top) \left(\hat{\mathbf{X}}_j \hat{\mathbf{X}}_j^\top + \sigma^2 \mathbf{I} \right)^{-1}, \tag{19}$$

where $\hat{\mathbf{X}}_j = [\mathbf{X}_j \quad \mathbf{X}] \in \mathbb{R}^{n \times 2d}$. Since both $\boldsymbol{\Phi}_j$ and $\hat{\mathbf{X}}_j \hat{\mathbf{X}}_j^\top$ are symmetric positive semi-definite matrices, their eigenvalues are real and non-negative. Therefore, the eigenvalues of $\boldsymbol{\Phi}_j$ satisfy

$$\lambda(\boldsymbol{\Phi}_j) = \frac{\lambda(\hat{\mathbf{X}}_j \hat{\mathbf{X}}_j^\top)}{\lambda(\hat{\mathbf{X}}_j \hat{\mathbf{X}}_j^\top) + \sigma^2}. \tag{20}$$

Applying the eigenvalue bounds from Lemma B.1, we obtain

$$\lambda_{\min}(\hat{\mathbf{X}}_j \hat{\mathbf{X}}_j^\top) \geq \lambda_{\min}(\mathbf{X}_j \mathbf{X}_j^\top) + \lambda_{\min}(\mathbf{X}\mathbf{X}^\top), \tag{21}$$

$$\lambda(\hat{\mathbf{X}}_j \hat{\mathbf{X}}_j^\top) \leq \lambda(\mathbf{X}_j \mathbf{X}_j^\top) + \lambda(\mathbf{X}\mathbf{X}^\top). \tag{22}$$

Substituting these bounds into the expression for $\lambda_{\min}(\boldsymbol{\Phi}_j)$, we have

$$\lambda(\boldsymbol{\Phi}_j) \geq \frac{\lambda_{\min}(\mathbf{X}_j \mathbf{X}_j^\top) + \lambda(\mathbf{X}\mathbf{X}^\top)}{\lambda(\mathbf{X}_j \mathbf{X}_j^\top) + \lambda(\mathbf{X}\mathbf{X}^\top) + \sigma^2}. \tag{23}$$

Since $\lambda_{\min}(\mathbf{X}_j \mathbf{X}_j^\top) \geq 0$, it follows that

$$\lambda(\boldsymbol{\Phi}_j) \geq \frac{\lambda(\mathbf{X}\mathbf{X}^\top)}{\lambda(\mathbf{X}_j \mathbf{X}_j^\top) + \lambda(\mathbf{X}\mathbf{X}^\top) + \sigma^2}. \tag{24}$$

Let us denote $\tau = \lambda(\mathbf{X}\mathbf{X}^\top)$ and assume that $\lambda(\mathbf{X}_j \mathbf{X}_j^\top) \leq \tau$ (we will justify this assumption later). Then, we have

$$\lambda(\boldsymbol{\Phi}_j) \geq \frac{\lambda(\mathbf{X}\mathbf{X}^\top)}{2\tau + \sigma^2}.$$

Using a similar analysis as in the proof of Theorem 3.2, and the fact that $\mathbf{X}_{j+1} = \mathbf{\Phi}_j(\mathbf{X}_j + \mathbf{E}_j)$, where each column of $\mathbf{E}_j$ is independently sampled from $\mathcal{N}\left(0, \frac{\hat{\sigma}^2}{n^2}\mathbf{I}\right)$, we have

$$\lambda(\mathbf{X}_{j+1}\mathbf{X}_{j+1}^\top) \leq \lambda^2(\mathbf{\Phi}_j)\left(\lambda(\mathbf{X}_j\mathbf{X}_j^\top) + C\hat{\sigma}^2\right), \tag{25}$$

with probability at least $1 - 2e^{-n}$.

We will now prove that, with probability at least $1 - 2qe^{-n}$, the following holds:

$$\lambda(\mathbf{X}_q\mathbf{X}_q^\top) \leq \tau. \tag{26}$$

We proceed by induction. For $q = 0$, the inequality holds by the definition of τ. Assume that inequality equation 26 holds for $q = j$; we will show it also holds for $q = j + 1$.

Since equation 26 holds at iteration j, we have $\lambda(\mathbf{X}_j\mathbf{X}_j^\top) \leq \tau$. Therefore,

$$\lambda(\mathbf{\Phi}_j) \leq \frac{\lambda(\mathbf{X}_j\mathbf{X}_j^\top) + \lambda(\mathbf{X}\mathbf{X}^\top)}{\lambda(\mathbf{X}_j\mathbf{X}_j^\top) + \lambda(\mathbf{X}\mathbf{X}^\top) + \sigma^2} \leq \frac{2\tau}{2\tau + \sigma^2}.$$

Plugging this bound, along with the assumption $\hat{\sigma}^2 \leq \frac{\sigma^2}{2C}$, into inequality equation 25, we obtain

$$\lambda(\mathbf{X}_{j+1}\mathbf{X}_{j+1}^\top) \leq \left(\frac{2\tau}{2\tau + \sigma^2}\right)^2 \left(\tau + C\hat{\sigma}^2\right) \leq \tau.$$

This completes the induction step and proves inequality equation 26.

Recall the inequality equation 24:

$$\lambda(\mathbf{\Phi}_j) \geq \frac{\lambda(\mathbf{X}\mathbf{X}^\top)}{\lambda(\mathbf{X}_j\mathbf{X}_j^\top) + \lambda(\mathbf{X}\mathbf{X}^\top) + \sigma^2}.$$

Since $\lambda(\mathbf{X}_j\mathbf{X}_j^\top)$ is bounded above by τ and $\lambda(\mathbf{X}\mathbf{X}^\top) > 0$, the right-hand side of inequality equation 24 is bounded below by a positive constant. Therefore, $\lambda(\mathbf{\Phi}_j)$ is bounded below by a positive constant, which implies that the learned DAE does not suffer from model collapse. □

Remark B.3. To prevent model collapse in generative models, a common strategy is to incorporate real data into the training process. Previous studies [9, 20, 22] have shown that mixing real data with synthetic data during training helps maintain model performance and prevents degeneration caused by relying solely on self-generated data. In diffusion models, integrating real samples can enhance model performance and reduce the risk of collapse [68]. By conditioning the model on both real and synthetic data, the training process leverages the structure of real data distributions. Building on these approaches, our work introduces methods to integrate real data into the training of Rectified Flow, even when direct noise-image pairs are not available. By generating noise-image pairs from real data using reverse processes and balancing them with synthetic pairs, we prevent model collapse while retaining efficient sampling.

B.4. Model collapse in Rectified flow

In the appendix, we provide the formal statement of our observation:

Proposition B.4. *Let $v_{\theta_j}(t, \boldsymbol{x})_{j=1}^{\infty}$ be a sequence of vector fields trained via Reflow in Rectified Flow. As $j \to \infty$, due to the sampling process of Rectified Flow, the generated result $\boldsymbol{x}_{j,1}$ at time $t = 1$ (i.e., the output of the j-th Reflow iteration) converges to a constant vector, indicating model collapse.*

Consider the explicit Euler discretization of the Rectified Flow ODE. Starting from $\boldsymbol{x}_{j,0} = \boldsymbol{z}$, where $\boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$, we update:

$$\boldsymbol{x}_{j,t+\gamma} = \boldsymbol{x}_{j,t} + \gamma, v_{\theta_j}(t, \boldsymbol{x}_{j,t}), \quad t \in [0, 1], \tag{27}$$

Algorithm 1 Real-data Augmented Reflow

Require: Reflow iterations $\mathcal{J}$; real dataset $\{\boldsymbol{x}^{(i)}\}$; pre-trained vector field v_{θ_0}; mix ratio λ; ODE solver $\mathrm{ODE}_{v_{\theta_0}}(t_0, t_1, \boldsymbol{x})$; regeneration parameter α.

Ensure: Trained vector fields $\{v_{\theta_j}\}_{j=1}^{\mathcal{J}}$

for $j = 1$ to $\mathcal{J}$ **do**
  Sample $\{\boldsymbol{z}^{(i)}\}$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
  Compute $\hat{\boldsymbol{x}}^{(i)} = \mathrm{ODE}_{v_{\theta_j}}(0, 1, \boldsymbol{z}^{(i)})$ ▷ *Generate synthetic noise-image pairs*
  Compute $\hat{\boldsymbol{z}}^{(i)} = \mathrm{ODE}_{v_{\theta_j}}(1, 0, \boldsymbol{x}^{(i)})$ ▷ *Generate reverse image-noise pairs from real data*
  Randomly select λn synthetic pairs and $(1-\lambda)n$ real reverse pairs
  $\mathcal{D}_j = \{(\boldsymbol{z}_j^{(i)}, \boldsymbol{x}_j^{(i)})\}_{i=1}^{n} = \{(\boldsymbol{z}^{(i)}, \hat{\boldsymbol{x}}^{(i)})\}_{i=1}^{\lambda n} \cup \{(\hat{\boldsymbol{z}}^{(i)}, \boldsymbol{x}^{(i)})\}_{i=1}^{(1-\lambda)n}$ ▷ *Mix Pairs with Ratio λ*
  repeat ▷ *Reflow training*
    for each $(\boldsymbol{z}_j^{(i)}, \boldsymbol{x}_j^{(i)}) \in \mathcal{D}_j$ **do**
      Sample $t \sim \mathcal{U}(0, 1)$
      Compute $\boldsymbol{x}_t^{(i)} = t\, \boldsymbol{x}_j^{(i)} + (1-t)\, \boldsymbol{z}_j^{(i)}$
      Compute loss:

$$\mathcal{L}_{\mathrm{RF}} = \frac{1}{B} \sum_{i=1}^{B} \left\| v_{\theta_j}(t, \boldsymbol{x}_t^{(i)}) - (\boldsymbol{x}_j^{(i)} - \boldsymbol{z}_j^{(i)}) \right\|^2$$

      Update θ_j using gradient descent
    Repeat Steps 4 and 6 every α epoches
  until converged
Output: $\{v_{\theta_j}\}_{j=1}^{\mathcal{J}}$

with step size γ. If each small step of v_{θ_j} acts similarly to a DAE, then based on Theorem **??**, as $j \to \infty$, we have:

$$\lim_{j \to \infty} \mathrm{rank}(v_{\theta_j}) = 0. \tag{28}$$

This implies $v_{\theta_j}(t, \boldsymbol{x}_{j,t}) \to \mathbf{0}$, leading to $\boldsymbol{x}_{j,t+\gamma} \approx \boldsymbol{x}_{j,t}$. Thus, the generated result remains near the initial point, confirming model collapse as stated in Proposition B.4.

Remark B.5. Although there is a theoretical gap between DAEs and Rectified Flow, our experimental results (Figure 7) support this proposition, suggesting that model collapse does occur in Rectified Flow under iterative self-training.

C. Methods Details

C.1. Real-data Augmented Reflow

Here we provide the detailed pseudocode: Section C.1.

C.2. Online Real-data Augmented Reflow

Here we provide the detailed pseudocode: Algorithm 2.

C.3. Does Adding Randomness Help? Reverse SDE Sampling

In the previous methods, we utilized the reverse ODE process to generate noise-image pairs for training. However, when using only the deterministic ODE, the randomness in the training process originates solely from the initial latent variables $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This limited source of randomness may impact the diversity of the generated samples and the robustness of the model [69].

Algorithm 2 Online Real-data Augmented Reflow Training

Require: Reflow iterations $\mathcal{J}$; real dataset $\{\boldsymbol{x}^{(i)}\}$; pre-trained vector field v_{θ_0}; mix ratio λ; SDE/ODE solver $\mathrm{SDE/ODE}(t_0, t_1, \cdot)$; regeneration parameter α
Ensure: Trained vector fields $\{v_{\theta_j}\}_{j=1}^{\mathcal{J}}$

for $j = 1$ to $\mathcal{J}$ **do**
  repeat ▷ *Reflow training*
    for each mini-batch **do**
      Sample $\{\boldsymbol{z}^{(i)}\}$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
      Compute $\hat{\boldsymbol{x}}^{(i)} = \mathrm{SDE/ODE}(0, 1, \boldsymbol{z}^{(i)})$ ▷ *Generate synthetic data*
      Sample $\{\boldsymbol{x}^{(i)}\}$ from real dataset
      Compute $\hat{\boldsymbol{z}}^{(i)} = \mathrm{SDE/ODE}(1, 0, \boldsymbol{x}^{(i)})$ ▷ *Generate reverse data*
      Randomly select λB synthetic pairs and $(1-\lambda)B$ real reverse pairs
      $\mathcal{D}_j = \{(\boldsymbol{z}_j^{(i)}, \boldsymbol{x}_j^{(i)})\} = \{(\boldsymbol{z}^{(i)}, \hat{\boldsymbol{x}}^{(i)})\} \cup \{(\hat{\boldsymbol{z}}^{(i)}, \boldsymbol{x}^{(i)})\}$ ▷ *Mix pairs according to λ*
      Sample $t \sim \mathcal{U}(0, 1)$
      for each $(\boldsymbol{z}_j^{(i)}, \boldsymbol{x}_j^{(i)})$ in $\mathcal{D}_j$ **do**
        Compute $\boldsymbol{x}_t^{(i)} = t\, \boldsymbol{x}_j^{(i)} + (1-t)\, \boldsymbol{z}_j^{(i)}$
        Compute loss:
        $$\mathcal{L}_{\mathrm{RF}} = \frac{1}{B} \sum_{i=1}^{B} \left\| v_{\theta_j}(t, \boldsymbol{x}_t^{(i)}) - (\boldsymbol{x}_j^{(i)} - \boldsymbol{z}_j^{(i)}) \right\|^2$$
      Update θ_j using gradient descent
  until converged
Output: $\{v_{\theta_j}\}_{j=1}^{\mathcal{J}}$

To enhance diversity and potentially improve generation quality, we consider introducing controlled randomness into the reverse process by employing a reverse Stochastic Differential Equation (SDE). The reverse SDE allows us to inject noise at each time step during the sampling process, defined as:

$$d\boldsymbol{x} = \left[f(t, \boldsymbol{x}) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) \right] dt + g(t) d\widetilde{\mathbf{w}}, \tag{29}$$

where $f(t, \boldsymbol{x})$ and $g(t)$ are the drift and diffusion coefficients, respectively, and $d\widetilde{\mathbf{w}}$ denotes the standard Wiener process in reverse time. By introducing the diffusion term $g(t)d\widetilde{\mathbf{w}}$, we inject controlled stochasticity into the reverse sampling.

In practice, we set the noise scale $g(t)$ to be small (e.g., $\sigma = 0.001$) and perform sampling using methods like the Euler-Maruyama scheme with an appropriate number of steps (e.g., 100 steps). This controlled injection of noise increases the variability in the training data without significantly disrupting the straightening effect of the flow. Specifically, the added randomness helps explore the neighborhood of data samples, enriching the learning process. We denote this method as **ORS Refflow**.

D. Experiments Details and Extra Results

Setup for DAE. In the Reflow verification experiment for the DAE, we use a 4-dimensional Gaussian distribution as both the initial and target distributions. The initial distribution is $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and the target distribution is $\mathcal{N}(\mathbf{0}, 5\mathbf{I})$, where $\mathbf{0}$ is a 4-dimensional zero vector, and $\mathbf{I}$ is the identity matrix. We employ a neural network θ composed of two linear layers $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ without activation functions and biases. We train the Reflow process for 20 iterations. The "Ratio" refers to the proportion of synthetic data to real data; a higher value indicates a greater proportion of synthetic data.

This bias-free two-layer MLP is strictly linear. Let $W_1 \in \mathbb{R}^{w \times d}$ be the first-layer weights (or $w \times (d+1)$ if a time scalar is concatenated) and $W_2 \in \mathbb{R}^{d' \times w}$ the second-layer weights. The network realises the

rank-$\leq w$ linear map

$$f_\theta(x) = W_2\, W_1\, x \quad \big(\text{or } f_\theta(x,t) = W_2\, W_1\, [x;\, t]\big),$$

Figure 4 presents the results from the Reflow experiment using a Denoising Autoencoder (DAE) on a 4-dimensional Gaussian distribution.

(**a**) shows the Wasserstein-2 (W2) distance between the true target data distribution and the generated data distribution over Reflow iterations. This metric assesses the fidelity of the generated data in approximating the target distribution.

(**b**) displays the evolution of the first principal component (Dimension 0) of the data as Reflow iterations increase. We compare the original DAE, which does not utilize synthetic data, with our DAE-RA model, which employs various ratios of synthetic data (ranging from 0.1 to 0.9), as well as a fully synthetic DAE. The comparison highlights the effectiveness of our DAE-RA model in maintaining the integrity of principal components, thereby preserving data structure and diversity.

Setup for Rectified Flow. In the Reflow verification experiment for linear neural network Rectified Flow, we augment $\boldsymbol{W}_1$ by adding one dimension corresponding to time, resulting in a neural network $\boldsymbol{W}_1\boldsymbol{W}_2 : \mathbb{R}^{d+1} \to \mathbb{R}^d$. Our experimental results can be found below and confirm our prop B.4 We also test a nonlinear neural network consisting of three linear layers with SELU activation functions and an extra dimension added to the first linear layer. The results are shown in

Model collapse in linear Rctified Flow We experiment on a 10-dimensional Gaussian which starts from the initial distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and the target distribution is $\mathcal{N}(\mathbf{0}, 5\mathbf{I})$. But to demonstrate our inference, we set dimension 1 of the covariance matrix to 1e-3, which reduces the rank of the data as a whole. Figure 7a shows the model collapse process of linear RF, the Figure 7b and Figure 7c demonstrates the correctness of Propositio B.4

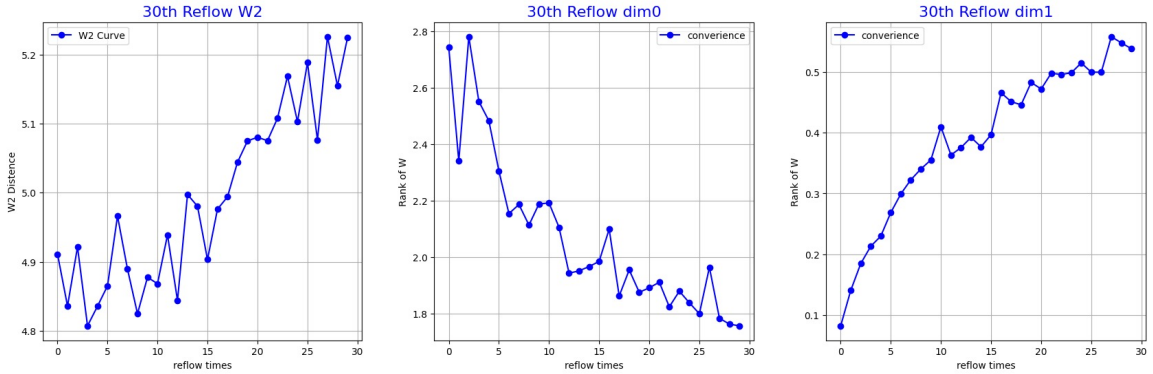
D.1. Straight Flow and Fewer-Step Image Generation

In our RA Reflow experiments [1], due to the high computational cost of Reflow training, we use a half-size U-Net compared to the one used in Flow Matching [6]. We used the standard implementation from the `https://github.com/atong01/conditional-flow-matching` repository provided by Tong et al. [37]. All methods were trained using the same configuration, differing only in the choice of the probability path or Reflow methods. Since the code for Lipman et al. [6] has not been released, some parameters may still differ from the original implementation. We summarize our setup here; the exact parameter choices can be found in our source code. We used the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and no weight decay. To replicate the architecture in Lipman et al. [6], we employed the U-Net model from Dhariwal and Nichol [70] with the following settings: channels set to 256, depth of 2, channel multipliers of [1, 2, 2, 2], number of heads as 4, head channels as 64, attention resolution of 16, and dropout of 0.0. We also used the "ICFM" methods from Tong et al. [37]'s repository to train Rectified Flow instead of using the original repository open-sourced by Liu et al. [7], because they use the same interpolation methods and probability paths.

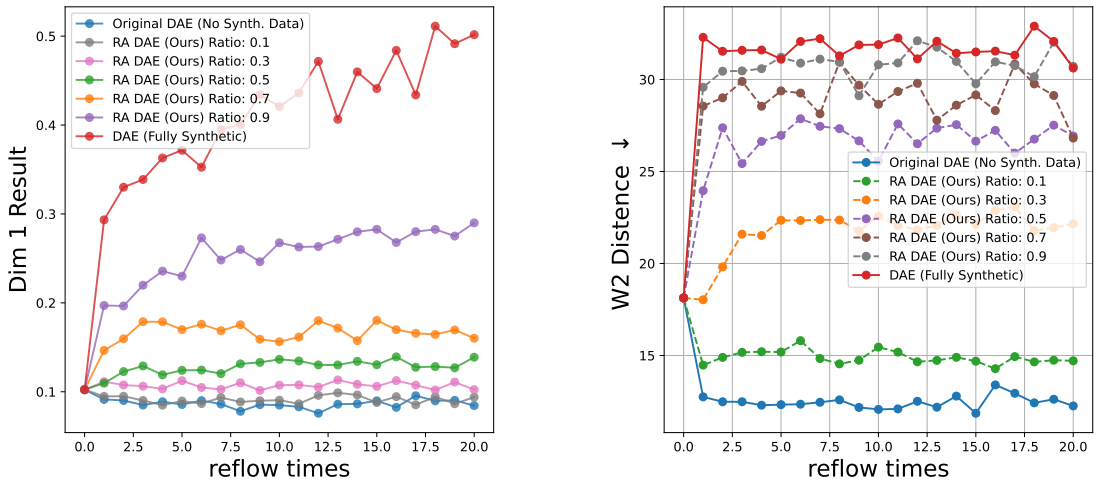
Training was conducted with a batch size of 256 per GPU, using six NVIDIA RTX 4090 GPUs, over a total of 2000 epochs. For Reflow, we generated 500,000 noise-image pairs for every Reflow iteration, according to Liu et al. [7]'s blog[2]. Although Liu et al. [7] mention that they use 40,00,000 noise-image to get the best performance, we keep the regular 500,000 noise-image to save time and training source. The learning rate schedule involved increasing the learning rate linearly from 0 to 5×10^{-4} over the first 45,000 iterations, then decreasing it linearly back to 0 over the remaining epochs. We set the noise scale $\sigma = 10^{-6}$. For sampling, we used Euler integration with the `torchdyn` package and the DOPRI5 solver from the `torchdiffeq` package.

[1] Our results may differ slightly from those reported in the original papers due to variations in neural network settings or random seeds. Nonetheless, our comparisons are fair. Because model collapse experiments require extensive retraining, we used more resource-efficient settings, which can further contribute to these minor discrepancies.

[2] `https://zhuanlan.zhihu.com/p/603740431`



(a) Wasserstein-2 Distance (b) Dim 0 (c) Dim 1



(d) Dim 1 (e) Wasserstein-2 Distance

Figure 7: Results from the reflow experiment with linear Rectified Flow on 10D Gaussian. The first row shows the Wasserstein distance and low-dimensional curves; the second row further illustrates other sampled dimensions.

In the CelebA-HQ experiments, we maintain the image resolution at 256×256. We utilize a pretrained Variational Autoencoder (VAE) from Stable Diffusion [1], where the VAE encoder reduces an RGB image $\mathbf{x} \in \mathbb{R}^{h \times w \times 3}$ to a latent representation $\mathbf{z} = \mathcal{E}(\mathbf{x})$ with dimensions $\frac{h}{8} \times \frac{w}{8} \times 4$. We used the standard implementation from the LFM repository (`https://github.com/VinAIResearch/LFM`) provided by Dao et al. [38]. We also used the DiT-L/2 [71] checkpoint released in Dao et al. [38]'s repository as the starting point for our Reflow training. Training was conducted with 4 NVIDIA A800 GPUs.

For RA Reflow, we tested $\lambda \in 0.1, 0.3, 0.5, 0.7, 0.9, 1.0$ with $\alpha = 4$. Note that when $\lambda = 0.0$, we are using 100% real reverse image-noise pairs, which is not equivalent to the original Reflow of Rectified Flow. Therefore, we train the original Reflow as the baseline. For the regeneration parameter α, we fixed $\lambda = 0.5$ and compared $\alpha \in 2, 4, 10, \infty$, where ∞ means we never regenerate new data within a single Reflow training. We evaluated the models using both the adaptive sampler "dopri5" (consistent with Lipman et al. [6]) and fixed, low numbers of function evaluations (NFEs) $10, 20, 50$ to demonstrate the elimination of model collapse and the maintenance of flow straightness by our method. This allows us to assess both generation quality and sampling efficiency simultaneously.

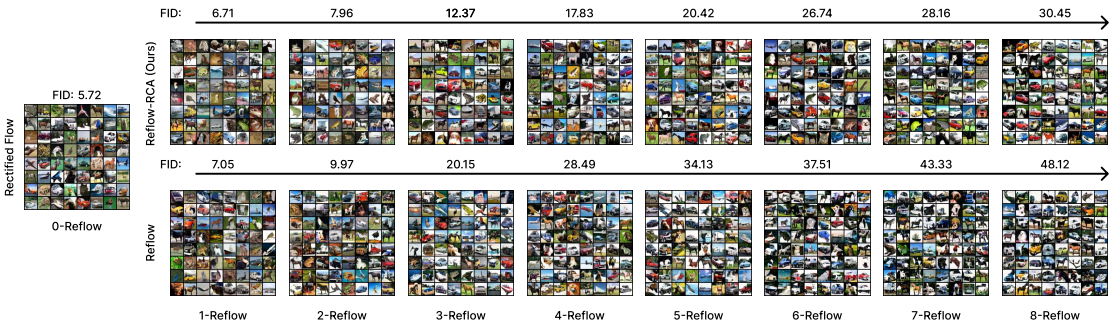


Figure 8: Results from the reflow experiment in CIFAR-10 using half-scale U-net.

Table 2: Summary of Configuration Parameters Across Experiments

	CIFAR10-figure 1	**CIFAR10-figure 5**	**CIFAR10-Table 4**
Channels	256	128	256
Channels multiple	1,2,2,2	1,2,2,2	1,2,2,2
Heads	4	4	4
Heads Channels	64	64	64
Attention resolution	16	16	16
Dropout	0.0	0.0	0.0
Effective Batch size	256	256	256
GPUs	6	6	6
Noise-image pairs	100k	500k	500k
Reflow Sampler	dopri5	Euler (100 NFE)	dopri5
α	2	4	2
λ	0.1	/	0.5
Learning Rate	2e-4	5e-4	5e-4

RF++ training set-up. Before training the 2-round Rectified Flow we create synthetic image–noise pairs with the EDM sampler (Heun's second–order solver). For CIFAR-10 we generate 0.8 M pairs using 35 NFEs; for ImageNet we generate 4 M pairs using 79 NFEs. To avoid division-by-zero in EDM initialisation we sample $t \sim \mathrm{U}[10^{-5}, 0.99999]$. For Imagenet, mixed-precision Adam is employed with dynamic loss scaling and an EMA decay of 0.9999. We train CIFAR-10 for roughly 3 days on 8 NVIDIA A800 GPUs, respectively, and ImageNet for more than 10 days on 16 A800s (batch size 2048 throughout). Dataset licences: CIFAR-10 (unknown), ImageNet (research, non-commercial).

D.2. Extra Results for Latent RF

Parameter Ablation Here we set the setting in Table 2.

D.3. Additional results for strong RF++ baselines

This section reports the full numerical results corresponding to Figure 6 in the main text. Tables 6 and 7 list Fréchet Inception Distance (FID), Recall, and the number of image–noise pairs used in each Reflow round for CIFAR-10 and ImageNet 64×64, respectively. All methods use the same RF++ backbone [30]; RA, ORA, and ORAS differ only in how real data are mixed during Reflow.

Table 3: **Comparison on FID score (↓) for unconditional generation on CelebA-HQ.** $\lambda = 0.5$, $\alpha = 2$, DiT-L/2. Best NFE shows the result with adaptive ODE sampler in (FID/NFE)

	CelebA-HQ (256×256)			
	10 NFE	20 NFE	50 NFE	Best NFE
1-RF(LFM) [38]	16.51	8.40	5.87	5.45/89
2-RF	12.04	7.34	5.76	5.73/71
2-RF-RA (**Ours**)	11.39	7.27	5.61	5.57/69
2-RF-ORA (**Ours**)	10.89	7.12	5.60	5.52/69
2-RF-ORAS (**Ours**)	**10.86**	**6.99**	**5.53**	**5.49/70**

Table 4: Performance of RF-RCA Models under Different λ Values

λ	0.1	0.3	0.5	0.7	0.9	1.0
1-RF-RCA	5.87	6.21	6.37	6.81	6.93	7.05
2-RF-RCA	6.37	7.10	7.96	8.53	8.98	9.97
3-RF-RCA	8.02	10.29	12.37	14.74	18.01	20.15

Table 5: Performance of RF-RCA Models under Different α Values

α	2	4	8	∞
1-RF-RCA	6.09	6.37	6.70	7.05
2-RF-RCA	6.92	7.10	8.14	9.97
3-RF-RCA	9.71	10.29	13.37	20.15

Table 6: **CIFAR-10 experiments with RF++ and real-data mixing.** Comparison of two-round RF++ and its real-data augmented variants. "Pairs" denotes the number of image–noise pairs used in each Reflow round; "–" indicates values not reported in the original source, and "/" indicates that the method does not maintain a persistent pair pool (pairs are generated online).

	CIFAR-10 (32×32)			
Method	NFE (↓)	FID (↓)	Rec. (↑)	Pairs (↓)
1-RF [7]	127	2.58	0.57	–
2-RF++ [30]	1	3.38	–	1M
2-RA-RF++ (**Ours**)	1	3.35	0.57	0.8M
2-ORA-RF++ (**Ours**)	1	3.29	0.59	/
2-ORAS-RF++ (**Ours**)	1	**3.27**	**0.60**	/
2-RF++(RA) (**Ours**)	100	2.42	0.61	0.8M
2-RF++(ORA) (**Ours**)	100	2.34	0.63	/
2-RF++(ORAS) (**Ours**)	100	**2.34**	**0.64**	/

Table 7: **ImageNet** 64×64 **experiments with RF++ and real-data mixing.** All models use the RF++ configuration with EDM-based initialization [30, 72]. Real-data mixing (RA) improves or matches the baseline FID with fewer image–noise pairs, while the online variants (ORA and ORAS) achieve the best FID without storing a pair pool.

	ImageNet (64×64), class-conditional			
Method	NFE (↓)	FID (↓)	Rec. (↑)	Pairs (↓)
2-RF++ [30]	1	4.31	–	5M
2-RA-RF++ (**Ours**)	1	4.26	0.49	4M
2-ORA-RF++ (**Ours**)	1	4.17	0.55	/
2-ORAS-RF++ (**Ours**)	1	**4.15**	**0.56**	/
2-RF++(RA) (**Ours**)	100	3.71	0.57	4M
2-RF++(ORA) (**Ours**)	100	3.70	**0.63**	/
2-RF++(ORAS) (**Ours**)	100	**3.60**	0.62	/