

# Balanced Conic Rectified Flow

## Supplementary Material

### A Detailed settings

**CPU and GPU settings** All experiments were conducted on a machine with an Intel Core i9-10980XE CPU (18 cores, 3.00GHz) and four NVIDIA GeForce RTX 3090 GPUs (24GB VRAM each). The system was equipped with 128GB of DDR4 RAM.

**Using exponential distribution of timesteps for training** The trajectory crossovers during the reflow process are more frequent near the noise or image endpoints (i.e., when closer to  $X_0$  or  $X_1$ ) [22]. To focus the training more effectively on these regions with high crossover frequency, we employed an exponential distribution. This phenomenon can be clearly observed in A1a, where we computed the top-k indices of the predicted velocity during the sampling process. The curvature is significantly higher near the start and end indices, indicating more crossovers at those points. Based on this observation, we adopted an exponential distribution<sup>5</sup> rather than a uniform distribution, as illustrated in A1b.

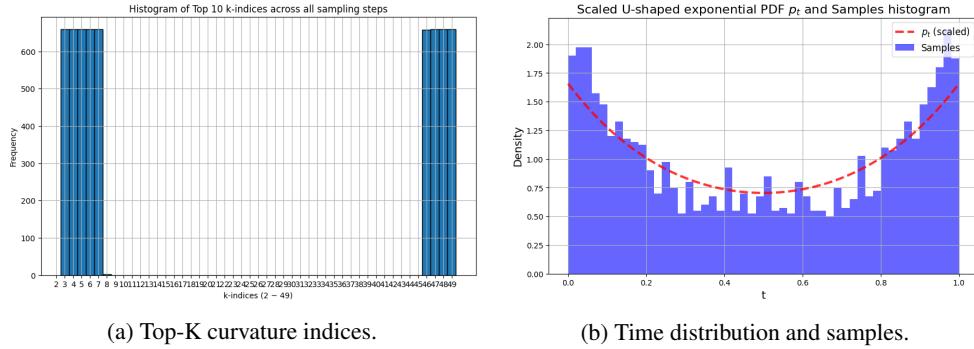


Figure A1: Top-K ( $K=10$ ) curvature indices and U-shape exponential time distribution with sampled data. For visualization convenience, the U-shaped probability density function  $p_t$  was scaled so that the area under the curve equals 1.

**$U_{\text{fake}}$  and  $U_{\text{real}}$**  The training schedule for conic reflow and original reflow is split into two phases. In the first half, conic and original reflows alternate to prevent bias toward either real or fake pairs. In the second half, only the original reflow is used to address the data imbalance, as fake pairs are far more abundant than real pairs. For instance, if total training step is  $\mathbb{N} = 100$ , then  $U_{\text{real}} = \{1, 3, 5, 7, \dots, 49\}$  and  $U_{\text{fake}} = \{2, 4, 6, 8, \dots, 50, 51, 52, \dots, 100\}$ .

### B Recall and Precision

In this section, we evaluate the performance of our method compared to the original rectified flow using recall and precision metrics. These metrics allow us to analyze how well the generated data covers the real data distribution (recall) and how accurate the generated samples are compared to the real data (precision) [20].

We conduct the evaluation on the CIFAR-10 dataset, which consists of 60,000 real images combined with 50,000 synthetic images. For sampling, we utilize Euler sampling to ensure consistency across experiments.

As shown in A1, the results highlight key differences between multi-step and 1-step sampling:

<sup>5</sup>More explicitly, we use  $p_t(u) \propto \exp(au) + \exp(-au)$  on  $u \in [0, 1]$  with  $a = 3$

- **Multi-step sampling:** Both our method and the original exhibit similar precision and recall values, indicating comparable performance in this setting.
- **1-step sampling:** Precision shows only a slight difference of approximately 0.85% on average, suggesting that both methods perform nearly identically in terms of precision. However, recall reveals a more significant advantage for our method:
  - On average, our method achieves 4.5% higher recall across all settings.
  - Specifically, for 2-Rectified Flow, our method outperforms the original by 5.5% in recall.

These findings indicate that while the precision of our method is nearly identical to the original, its higher recall demonstrates superior coverage of the real data distribution. Therefore, we can interpret these results as evidence that our method produces a more comprehensive and balanced representation of the underlying data distribution compared to the original.

Method	NFE	Precision ( $\uparrow$ )	Recall ( $\uparrow$ )
<b>2-Rectified Flow</b>			
<i>Full Step Generation</i>			
Ours	104	0.691	<b>0.605</b> (+0.005)
Original	104	<b>0.696</b> (+0.005)	0.600
<i>One step Generation</i>			
Ours	1	0.687	<b>0.583</b> (+0.055)
Original	1	<b>0.695</b> (+0.008)	0.528
<b>3-Rectified Flow</b>			
<i>Full Step Generation</i>			
Ours	104	0.691	<b>0.599</b> (+0.007)
Original	104	<b>0.698</b> (+0.007)	0.592
<i>One step Generation</i>			
Ours	1	0.682	<b>0.592</b> (+0.03)
Original	1	<b>0.691</b> (+0.009)	0.562

Table A1: Comparison of 2- and 3-Rectified Flows under different NFE settings, highlighting the better results in bold.

## C Extreme number of reflow process ( $k = 4$ )

In this section, we compare the generative quality, curvature, IVD of our method and the original under a setting with an extreme number of reflow processes ( $k = 4$ ), which is higher than the typical settings of  $k = 2$  or  $k = 3$ .

Method	NFE	IS ( $\uparrow$ )	FID ( $\downarrow$ )	Precision ( $\uparrow$ )	Recall ( $\uparrow$ )
<b>4-Rectified Flow</b>					
Ours	100	<b>9.076</b>	<b>4.195</b>	0.696	<b>0.585</b>
Original	100	8.951	4.490	<b>0.705</b>	0.584
Ours	1	<b>8.808</b>	<b>5.662</b>	<b>0.690</b>	<b>0.581</b>
Original	1	8.597	6.580	0.688	0.576

Table A2: Comparison of IS, FID, precision, and recall for 4-Rectified Flow between Ours and Original. The better values are highlighted in bold.

Method	Curvature ( $\downarrow$ )	IVD ( $\downarrow$ )
<b>4-Rectified Flow</b>		
Ours	<b>0.00176</b>	<b>0.20787</b>
Original	0.00186	0.21812

Table A3: Comparison of curvature and IVD for 4-Rectified Flow between Ours and Original rectified flow. The better values are highlighted in bold.

As shown in Table A2 and A3:

- **1-Step generation quality:** Our method outperforms the original in both FID and IS, demonstrating superior generative performance.
- **Multi-step generation quality:** Similarly, our method shows better results compared to the original.

364 • **Additional metrics:** Our method achieves better curvature and IVD compared to the original.  
 365 This indicates that our method forms a velocity field that enables the solution trajectory  
 366 to be straighter during the reflow process and better preserves the direction of the initial  
 367 velocity, ensuring it aligns more closely with the overall trajectory.

368 These findings show that our method preserves the distribution of real images while preventing bias  
 369 toward fake images. Even with  $k > 3$ , our method improves the reflow process and makes it robust  
 370 for extreme reflow step settings.

## 371 D Fine-tuning with real pairs improves a pre-trained rectified flow model

372 We demonstrate the effect of fine-tuning a pretrained rectified flow model using only 60,000 real  
 373 pairs. For this experiment, we used the official rectified flow CIFAR-10 checkpoints available on  
 374 GitHub.<sup>6</sup> With minimal additional training, we observe a noticeable improvement in 1-step quality  
 375 as shown in Figure A2(a). Additionally, both curvature and IVD values decrease rapidly as illustrated  
 376 in Figure A2(b). These results show that applying our method to a 2- or 3-rectified flow model,  
 377 previously trained with standard techniques, effectively improves 1-step generation quality even with  
 378 a small number of real pairs. Figure A2(b) also shows fine-tuned version has lower curvature and  
 379 IVD than the original, indicating that our method is straighter than the original. Furthermore, the  
 380 fine-tuned version has lower recon and p-recon differences between real and fake images, indicating  
 381 that our method reduces the bias toward fake samples. For visualization convenience, IVD was scaled  
 382 by  $10^{-2}$ .

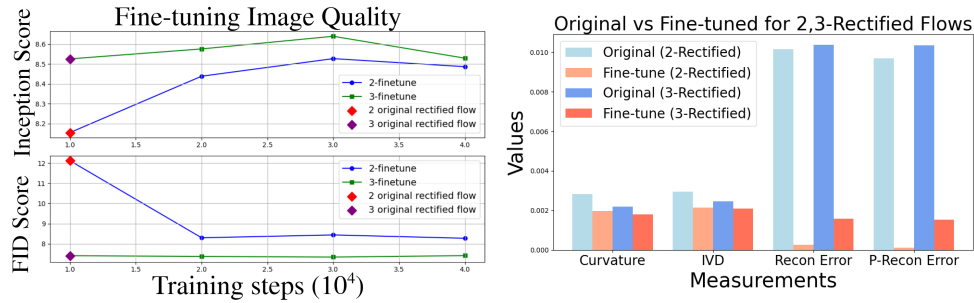


Figure A2: (a) Comparison of image quality between the original rectified model and our fine-tuned model across training steps (left). (b) Comparison of measurements for original and fine-tuned models for 2- and 3-rectified flows (right).

## 383 E Using even fewer fake samples and reflow just real pair

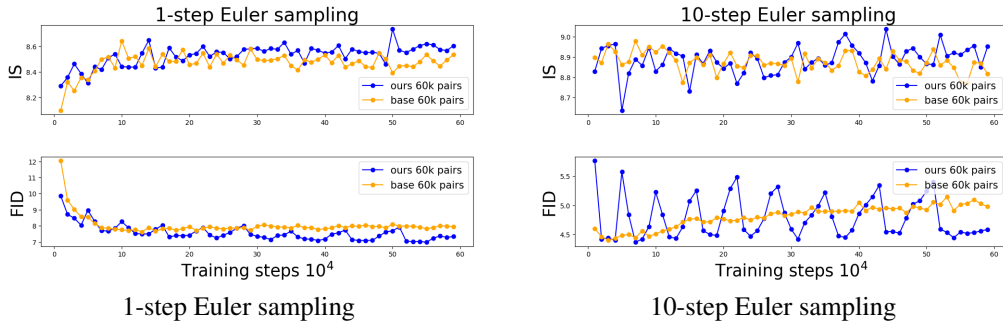


Figure A3: FID and IS score for original and ours 2-rectified flow (using 60k fake pairs)

384 **Extreme case (60k fake pair(original) and + 60k fake pair Real pair(ours))** Figure A3 compares  
 385 the generation quality of the 2-rectified flow trained with an extremely small number of fake pairs.

<sup>6</sup>Checkpoints are available at: <https://github.com/gnabitab/RectifiedFlow>.

As shown in the figure, when the number of fake pairs is very limited, the 1-step image generation quality shows slight improvement. However, reflow training under such limited data coverage tends to overfit to the narrow fake distribution, which leads to a decline in generation quality over multiple steps, despite the gains observed in 1-step performance.

In contrast, our approach preserves the path for real images during the conic reflow step, even with an extremely small number of fake pairs (60k). This results in not only improved 1-step quality but also mitigates the degradation in multi-step generation quality compared to the original.

**Reflow just using real pair** We also conduct experiments under the extreme setting where reflow is trained using only real image pairs, excluding any synthetic samples. While this setup leads to slight improvements in 1-step generation quality, we observe progressive degradation in full-step performance as training continues. This limitation arises from the fact that real pairs offer accurate supervision only in the local neighborhood of the data manifold. Without the diversity and coverage provided by synthetic samples, the model struggles to generalize its velocity field across the entire generation trajectory.

To validate this observation, we compare the following three settings: (1) Real only (no Slerp), (2) Real only with Slerp-based supervision, and (3) Our full method combining real and fake pairs with Slerp. As shown in Table A4, setting (2) improves over (1), but both underperform compared to our full method:

Setting	1-step (IS / FID)	RK-step (IS / FID)
(1) Real only (no Slerp)	8.21 / 6.93	8.65 / 5.13
(2) Real only (+ Slerp)	8.53 / 6.71	9.02 / 4.47
(3) Ours (real + fake + Slerp)	<b>8.79 / 4.16</b>	<b>9.30 / 3.24</b>

Table A4: Performance comparison using only real pairs, real pairs with Slerp supervision, and our full method.

These results show that while Slerp-based perturbed supervision enhances the effectiveness of real image, the combination of real and synthetic samples achieves better 1-step and multi-step generation quality.

## F Generalization to complex dataset: Imagenet 64×64

In this section, we evaluate how well our method generalizes on the ImageNet 64×64 dataset. We compare our method with the original by using 1) reconstruction and perturbed reconstruction errors, and 2) their difference. The evaluation shows that our approach is not only limited to smaller datasets like CIFAR-10 but also extends effectively to more challenging datasets. All precision and recall values are computed using 100k real images and 50k synthetic images. Reconstruction and perturbed reconstruction errors are obtained using a 2-step Euler solver, where reverse noise is perturbed with Slerp by  $\epsilon = 0.1$ . Both reconstruction and perturbed reconstruction errors are computed over a total of 12k images.

As shown in Table A5, our method significantly reduces reconstruction errors for both real and fake images compared to the original. This includes perturbed cases, where the gap between real and fake reconstruction errors (Difference and Difference Perturbed) is notably smaller in our approach. These results demonstrate that our method effectively mitigates the increasing discrepancy between real and fake images during the reflow process. For qualitative results on reconstruction and perturbed reconstruction images for both fake and real images, please refer to Figure A4 and A5.

	Fake Recon	Real Recon	Perturbed Fake Recon	Perturbed Real Recon	Difference	Difference Perturbed
Ours (Euler 2-step)	0.01727	0.02138	0.02735	0.03076	0.00410	0.00340
Original (Euler 2-step)	0.02566	0.03273	0.03665	0.04261	0.00707	0.00596

Table A5: Comparison of Reconstruction errors and differences between Original and Ours on ImageNet 64×64 using the Euler 2-step solver.

## G RF++<sup>†</sup> and RF++<sup>†</sup>(+Ours) extended training results (500K steps)

In this section, we report extended training results beyond 300K iterations to demonstrate that our method continues to outperform the original RF++<sup>†</sup>.

As shown in Table A6, RF++<sup>†</sup>(+Ours) achieves better FID and IS scores compared to RF++<sup>†</sup>. This suggests that our Slerp-based perturbation method leads to more efficient reflow training, effectively preserving straight paths to real image distributions. All experiments use the same training configurations as described in Appendix J.

Model	Steps	IS (↑)	FID (↓)	Solver
RF++ <sup>†</sup> (+Ours)	1-step	<b>9.15</b>	<b>3.84</b>	Heun 2nd
RF++ <sup>†</sup>	1-step	9.04	4.14	Heun 2nd
RF++ <sup>†</sup> (+Ours)	2-step	<b>9.36</b>	<b>3.03</b>	Heun 2nd
RF++ <sup>†</sup>	2-step	9.24	3.16	Heun 2nd

Table A6: Extended training results (500K steps) for RF++<sup>†</sup> and RF++<sup>†</sup>(+Ours).

## H Qualitative results of reconstruction and perturbed reconstruction

Figure A12, A13, A14 provide a qualitative comparison of reconstruction between the original and ours. Figure A15 provide a qualitative comparison of the perturbed reconstruction between the original and ours (Cifar 10). As shown in the figures, the rectified flow model trained with our reflow procedure preserves the structure of real objects more effectively, whereas the original generates images that deviate significantly from the original object after undergoing reconstruction or perturbed reconstruction.

## I Qualitative comparison on Lsun dataset

We provide additional qualitative results in Figure A6, A7, A8, A9, A10, A11 demonstrating that our method achieves better generation quality than the original under few-step sampling settings.

## J Misc. configurations

Our experiments on CIFAR10 are configured as follows. For rectified flow, we utilize the same network architecture as the rectified flow model based on DDPM++ in [33, 24] with batch size 256. The training process is smoothed using an exponential moving average (EMA) with a decay rate of 0.999999, following the approach in Song et al. [33]. The Adam optimizer [8] is used with a learning rate of 2e-4 and a dropout rate of 0.15 is applied.

We train RF++<sup>†</sup> following the configuration (F) from Lee et al. [22], which includes EDM initialization, an exponential time distribution, and LPIPS-Huber-1/t loss, with a batch size of 128. For training, we use 800K fake pairs for RF++ and 600K fake pairs with an additional 50K real pairs for RF++<sup>†</sup>(+ours). Both models are optimized using Adam with an EMA decay of 0.9999. We generate 50,000 synthetic images to compute FID using the official RF++ implementation, which employs Heun’s second-order solver [22]. For evaluation, we report results using the best checkpoint selected during 300K training iterations. The best FID and IS score obtained with 500K iterations (see Appendix G) is also based on the same configuration, and evaluated using the official RF++ and guided-diffusion [A3] implementation without modification.

For the high-resolution (LSUN), both our method and the original 2-rectified flow were trained for 600,000 steps with a batch size of 16. Training started from the same 1-rectified flow checkpoints from the official rectified flow repository. During inference, all models used fixed seeds and identical noise conditions corresponding to the sampling steps.

## 458 K Pseudocode

---

**Algorithm 1:** Full Algorithm (Red text indicates our addition)

---

**Input:** Coupling  $(X_0, X_1)$  of  $\pi_0$  and  $\pi_1$ ; velocity model  $v_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^d$

**Output:**  $v_{\hat{\theta}_{\text{conic}}}$

```

1 Training:
2  $\hat{\theta} = \arg \min_{\theta} \mathbb{E} [\|X_1 - X_0 - v_{\theta}(tX_1 + (1-t)X_0, t)\|^2]$ , where  $t \sim \text{Uniform}([0, 1])$ 
3 Sampling:
4 Draw forward flow:  $dZ_{t,F} = v_{\hat{\theta}}(Z_{t,F}, t) dt$ , starting from  $Z_{0,F} \sim \pi_0$ 
5 Draw backward flow:  $(Z_{0,R}, X_1)$  following  $dZ_{t,R} = v_{\hat{\theta}}^{-1}(Z_{t,R}, t) dt$ , with  $X_1 \sim \pi_1$ 
6 Balanced Conic Reflow:
7 Initialize : Total training step =  $\mathbb{N}$ , Repairing step =  $\mathcal{T}$ ,  $\text{cnt} = 0$ ,  $\chi_{\text{fake}}, \chi_{\text{real}}$ 
8 for  $t \in (0, \mathbb{N})$  do
459 9   if  $t == t_{\zeta^{\max}}$  then
10   |   Compute  $\zeta^{\max}$  using Eqn. 13
11   if  $\text{cnt} == \mathcal{T}$  then
12   |   Generate new real pair:  $(Z_{0,R}, X_1) \leftarrow (v_{\hat{\theta}_{\text{conic}}}^{-1}(X_1), X_1)$ 
13   |    $\zeta_k^{\max} \leftarrow \zeta^{\max} / k$ 
14   |    $\text{cnt} \leftarrow 0$ 
15   else
16   |   Train: Update  $\hat{\theta}_{\text{conic}}$ 
17   |    $\arg \min_{\theta} \mathbb{E} [\|\chi_{\text{fake}} \cdot (\dot{Z}_{t,F} - v_{\theta}(Z_{t,F})) + \chi_{\text{real}} \cdot (X_1 - \text{slerp}(Z_{0,R}, \epsilon, \zeta) - v_{\theta}(\text{Conic}(X_1, \epsilon, \zeta, t)))\|^2]$ 
18   |    $\text{cnt} += 1$ 
19 (Optional) Distillation: Learn neural network  $\hat{T}$  to distill the  $k$ -rectified flow, so that  $Z_{1,F}^k \approx \hat{T}(Z_{0,F}^k)$ 

```

---

## 460 L Limitation

461 While our method demonstrates strong improvements in reflow process, one limitation remains in the  
462 choice of the maximum Slerp noise magnitude  $\zeta^{\max}$ , which is dependent on the training dataset. We  
463 select  $\zeta^{\max}$  by measuring reconstruction discrepancy between real and fake samples after a warm-up  
464 phase, and choosing the noise scale that maximizes this gap. While this aligns well with our problem  
465 formulation and intuition, providing a theoretically explicit solution remains challenging due to the  
466 unknown intrinsic dimension of the training dataset. It may be beneficial to incorporate additional  
467 factors into the decision process. For example, from the perspective of information theory, the  
468 ratio between real and fake pairs may serve as a guideline by comparing their information content  
469 (e.g., entropy), and the intrinsic dimension of the data manifold may also provide useful signals for  
470 determining an appropriate noise scale [A1, A2].

## 471 M Implementation details

472 **Slerp.** Below is the Python implementation of our spherical linear interpolation (Slerp), used to  
 473 interpolate between the reverse noise  $Z_{0,R}$  and a randomly sampled noise  $\epsilon \sim \mathcal{N}(0, I)$  with ratio  $\zeta$ :

```

474 def slerp_ours(t, v0, v1, DOT_THRESHOLD=0.9995):
475     c = True
476     if not isinstance(v0, np.ndarray):
477         c = True
478         v0 = v0.detach().cpu().numpy()
479     if not isinstance(v1, np.ndarray):
480         c = True
481         v1 = v1.detach().cpu().numpy()
482
483     v0_copy = np.copy(v0)
484     v1_copy = np.copy(v1)
485
486     v0 = v0 / np.linalg.norm(v0)
487     v1 = v1 / np.linalg.norm(v1)
488     dot = np.sum(v0 * v1)
489
490     if np.abs(dot) > DOT_THRESHOLD:
491         print("do lerp")
492         return torch.lerp(t, v0_copy, v1_copy)
493
494     theta_0 = np.arccos(dot)
495     sin_theta_0 = np.sin(theta_0)
496     theta_t = theta_0 * t
497     sin_theta_t = np.sin(theta_t)
498
499     s0 = np.sin(theta_0 - theta_t) / sin_theta_0
500     s1 = sin_theta_t / sin_theta_0
501     v2 = s0 * v0_copy + s1 * v1_copy
502
503     return torch.from_numpy(v2).to("cuda") if c else v2
504
505 
```

506 **Noise scheduler.** We define the nonlinear Slerp noise schedule as:

```

507 def noise_scheduler(max_steps, max_noise):
508     schedule = []
509     for step in range(max_steps):
510         t = step / max_steps
511         noise = max_noise * 2 * (1 - 1 / (1 + t**2))
512         schedule.append(noise)
513     return schedule
514
515 
```

516 **Scheduled reflow during training.** During training, conic reflow is periodically triggered and the  
 517 noise schedule is dynamically adjusted:

```

518 if i_effective == warmup_steps:
519     max_noise = compute_zeta_max(0.01, 0.5)
520     # Equation (8): starting noise magnitude = 0.01, end = 0.5
521     (...)
522 if i_effective % 100000 == 0:
523     add_noise = True
524     c_n = 0
525     partial_max_noise = max_noise / noise_decay
526     noise_decay += decay_direction
527     # Start decay_direction is -1
528     if noise_decay == 1:
529         decay_direction = 1
530     elif noise_decay == k:
531
532 
```

```

532         decay_direction = -1
533         alpha = noise_scheduler(max_steps + 5, patial_max_noise)
534

```

535 **Applying conic reflow.** At scheduled noise injection steps, we replace  $z$  using Slerp between the  
536 original  $z$  and Gaussian noise:

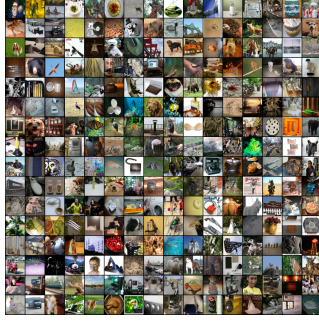
```

537
538 if conic_rf:
539     if add_noise:
540         x, z, c, _ = next(train_iter_conic)
541         noise_batch = torch.randn_like(z).to(device) * (1 - 1e-5)
542         z = slerp_ours(alpha[-1 - c_n], z, noise_batch)
543         if cnt % gradient_accumulation_steps == 0:
544             print('conic reflow(slerp):', alpha[-1 - c_n], 'noise_step:', c_n)
545             add_noise = False
546     (...)
547     # Sample t, zt
548     t = sample_t(exponential_distribution, x.shape[0], arg.a).to(device)
549     zt = (1 - t).view(-1, 1, 1, 1) * x + t.view(-1, 1, 1, 1) * z
550     target = z - x
551
552     # Forward pass
553     pred = model(zt, t, c)
554     # Predicted x
555     pred_x = zt - pred * t.view(-1, 1, 1, 1)
556

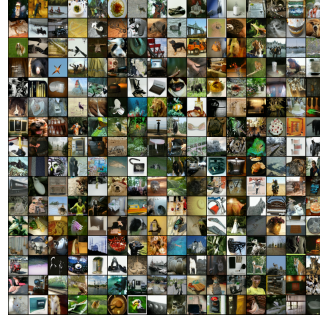
```

Method	NFE ( $\downarrow$ )	IS ( $\uparrow$ )	FID ( $\downarrow$ )
<i>Full Simulation (Euler Solver, <math>N=2000</math>)</i>			
VP SDE [33]	2000	9.58	2.55
sub-VP SDE [33]	2000	9.56	2.61
NCSN++ (VE SDE) [33]	2000	9.83	2.31
DDPM [13]	1000	9.46	3.21
<i>Adaptive Step Simulation (Runge–Kutta (RK45), Adaptive <math>N</math>)</i>			
VP ODE [33]	140	9.37	3.93
sub-VP ODE [33]	146	9.46	3.16
NCSN++ (VE ODE) [33]	176	9.35	5.38
LSGM [37]	147	-	2.10
PFGM [44]	110	9.68	2.35
EDM [15]	35	9.84	2.04
1-Rectified Flow	127	9.60	2.58
2-Rectified Flow	110	9.24	3.36
<b>2-Rectified Flow Ours</b>	104	9.30	3.24
3-Rectified Flow	104	9.01	3.96
<b>3-Rectified Flow Ours</b>	98	9.14	3.70
<i>One-Step Simulation (Euler Solver, <math>N=1</math>)</i>			
VP ODE (+Distill) [33]	1	1.20 (8.73)	451 (16.23)
sub-VP ODE (+Distill) [33]	1	1.21 (8.80)	451 (14.32)
NCSN++ (VE ODE) (+Distill) [33]	1	1.18 (2.57)	461 (254)
1-Rectified Flow (+Distill)	1	1.13 (9.08)	378 (6.18)
2-Rectified Flow (+Distill)	1	8.08 (9.01)	12.21 (4.85)
2-Rectified Flow++ <sup>†</sup> [22]	1	9.03	4.14
2-Rectified Flow++ <sup>†</sup> ( <b>Ours</b> )	1	9.15	3.84
<b>2-Rectified Flow Ours (+Distill)</b>	1	8.79 (9.11)	5.98 (4.16)
3-Rectified Flow (+Distill)	1	8.47 (8.79)	8.15 (5.21)
<b>3-Rectified Flow Ours (+Distill)</b>	1	8.84 (8.96)	5.48 (4.68)
<i>Diffusion + Distillation</i>			
DDIM Distillation [26]	1	8.36	9.36
DMD [46]	1	-	3.77
Diff-Instruct [27]	1	9.89	4.53
PD [30]	1	8.69	8.34
DFNO [48]	1	(-)	4.15
SID, ( $\alpha = 1.2$ ) [49]	1	9.98	1.92
<i>Consistency Model</i>			
CD [34]	1	9.48	3.55
CT [34]	1	8.49	8.70
ICT [32]	1	9.54	2.83
CTM [17]	1	-	5.19
CTM + GAN [17]	1	-	1.98

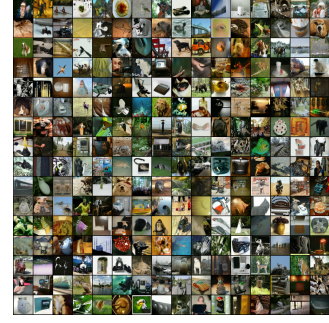
Table A7: Unconditional generation quality with various diffusion-based models on CIFAR-10. Blue rows highlight the top-5 baselines for 1-NFE, and red rows for Adaptive NFE (RK-45). The lowest FID and highest IS scores in each setting are **bolded**.



(a) Real Image



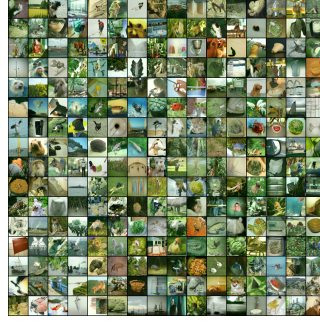
(b) Reconstruction Image (Ours)



(c) Perturbed Recon Image (Ours)



(d) Real Image

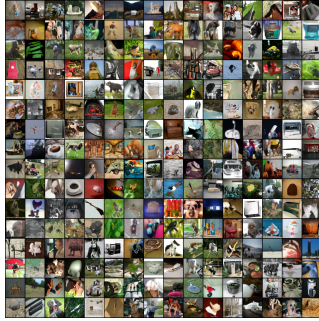


(e) Reconstruction Image (Base)

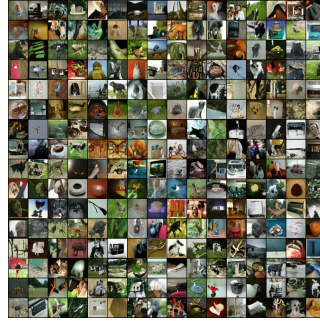


(f) Perturbed Recon Image (Base)

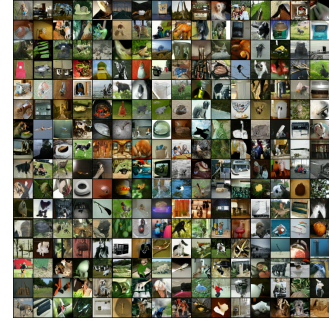
Figure A4: Qualitative comparison of **real** image reconstruction and perturbed reconstruction results between the baseline and our method using the 2-rectified flow trained on the ImageNet dataset. The **bottom** row presents the original method, while the **top** row presents our method.



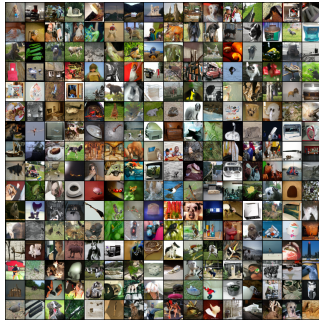
(a) Fake Image



(b) Reconstruction Image (Ours)



(c) Perturbed Recon Image (Ours)



(d) Fake Image



(e) Reconstruction Image (Base)



(f) Perturbed Recon Image (Base)

Figure A5: Qualitative comparison of **fake** image reconstruction and perturbed reconstruction results between the original and our method using the 2-rectified flow trained on the ImageNet dataset. The **bottom** row presents the original method, while the **top** row presents our method.

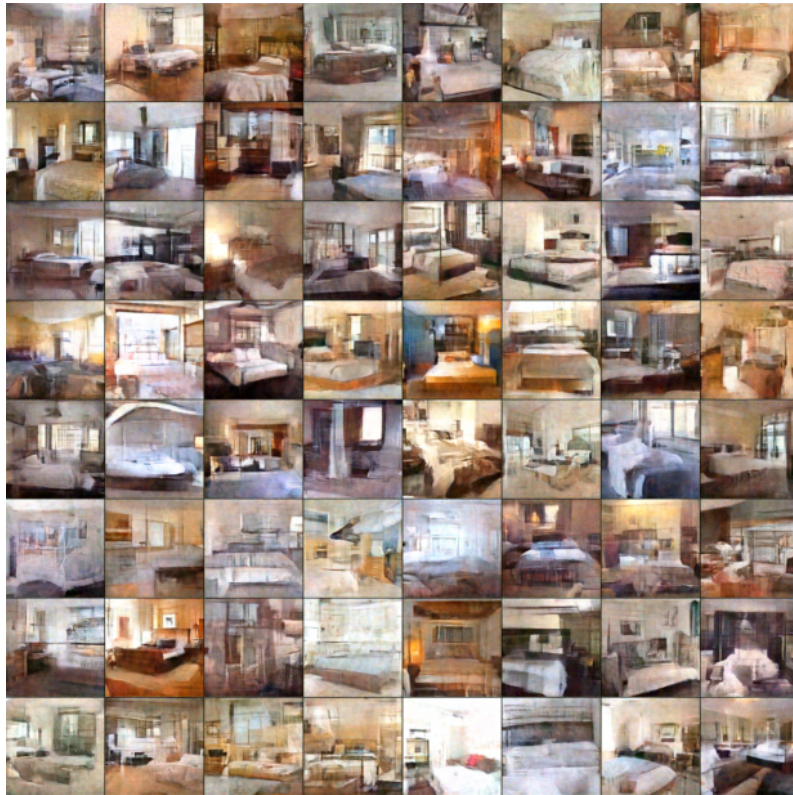
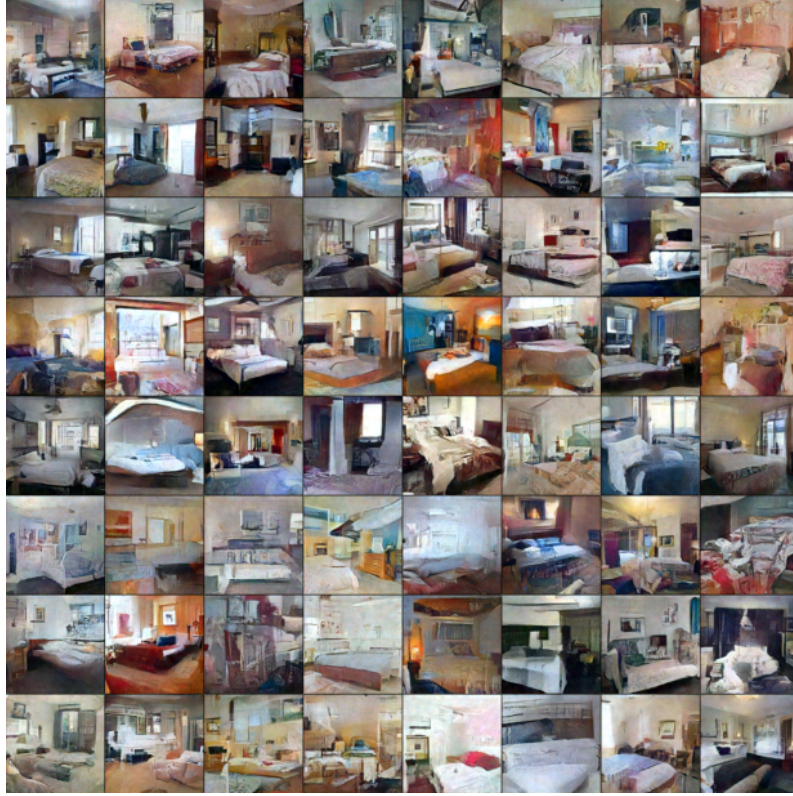


Figure A6: Ours (**up**) and Original (**down**) 2-rectified flow (1-step, seed 1)

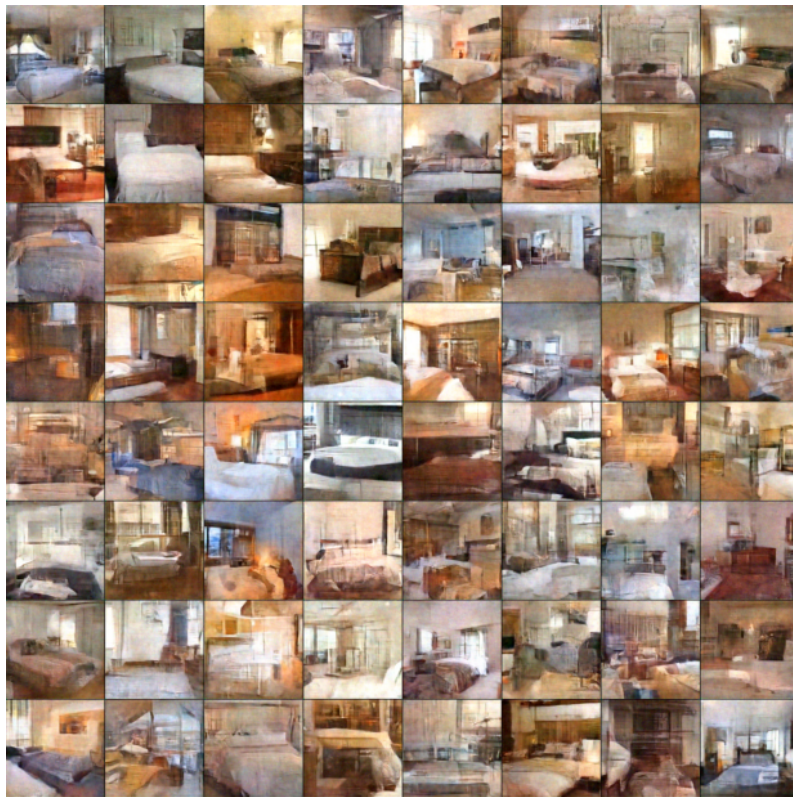


Figure A7: Ours (**up**) and Original (**down**) 2-rectified flow (1-step, seed 2)

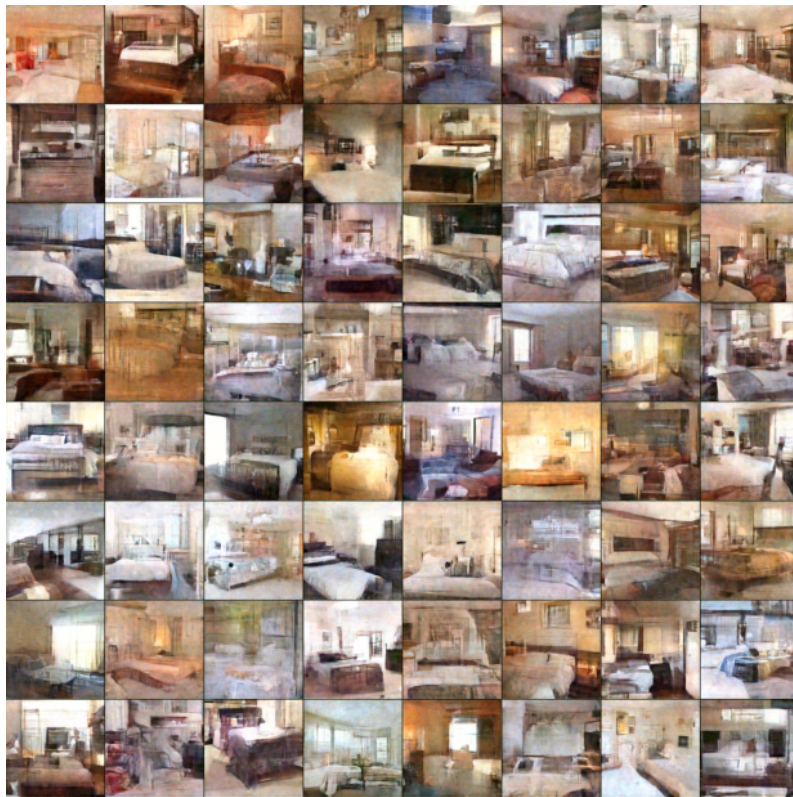


Figure A8: Ours (**up**) and Original (**down**) 2-rectified flow (1-step, seed 3)

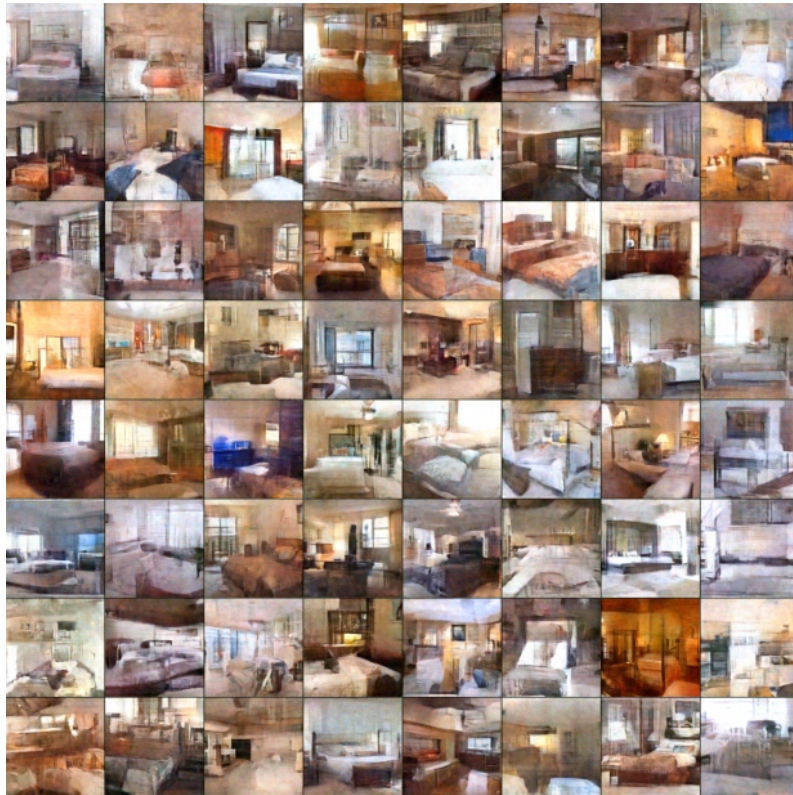


Figure A9: Ours (**up**) and Original (**down**) 2-rectified flow (1-step, seed 333)

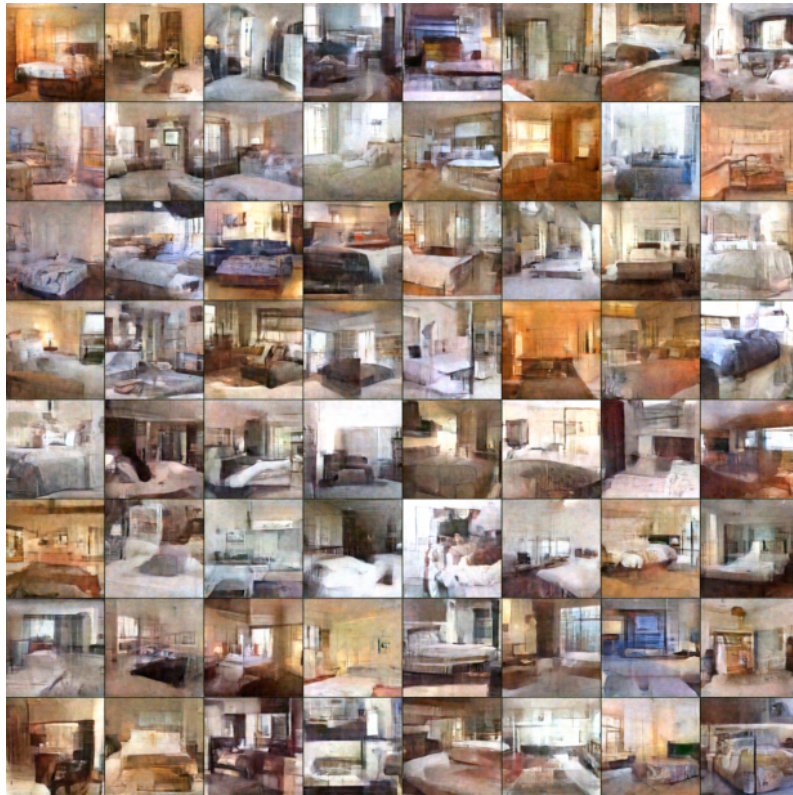


Figure A10: Ours (**up**) and Original (**down**) 2-rectified flow (1-step, seed 555)

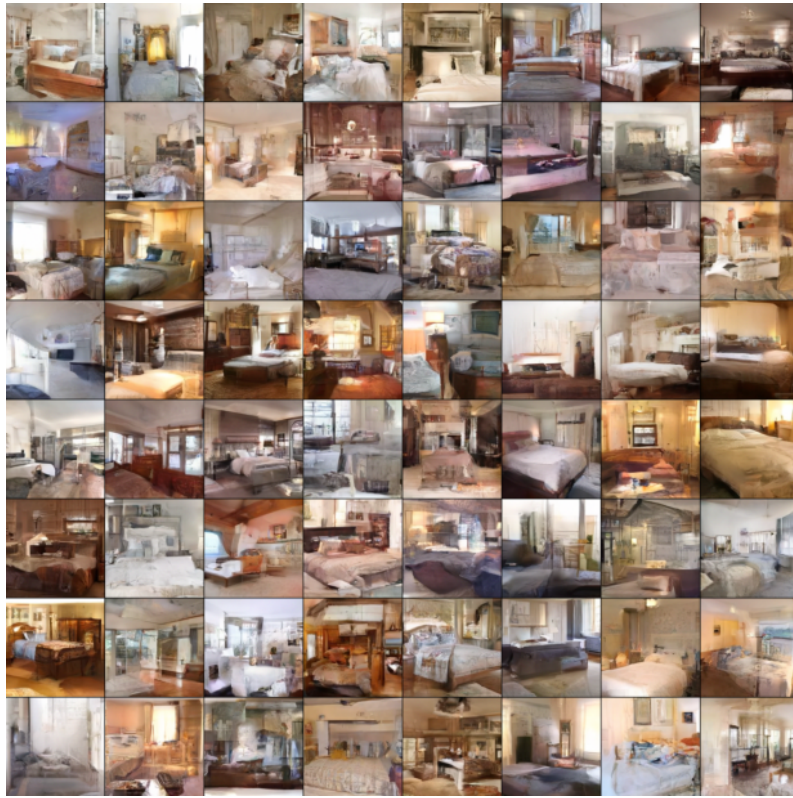
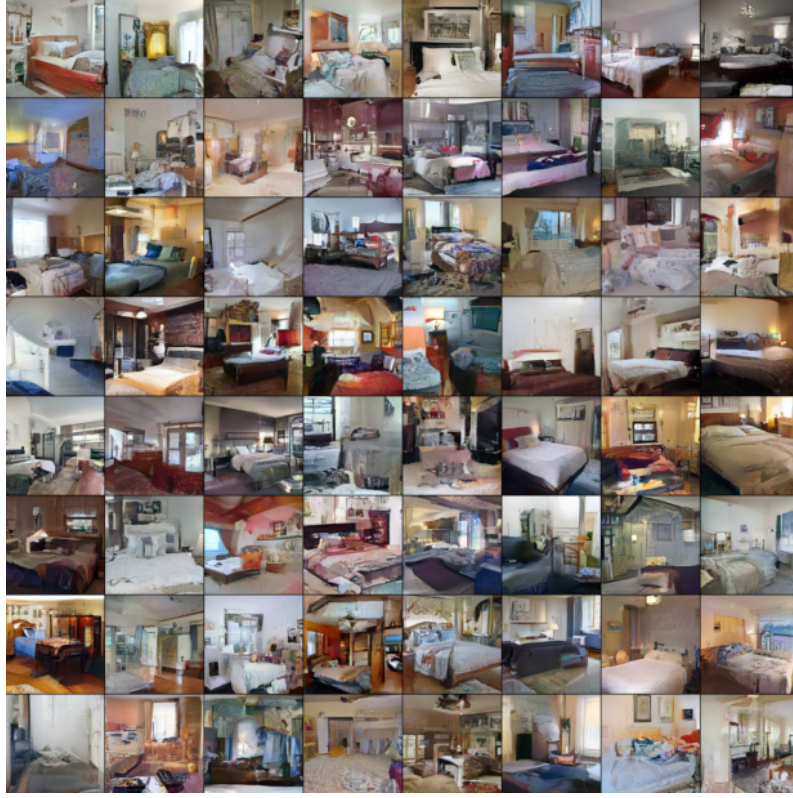
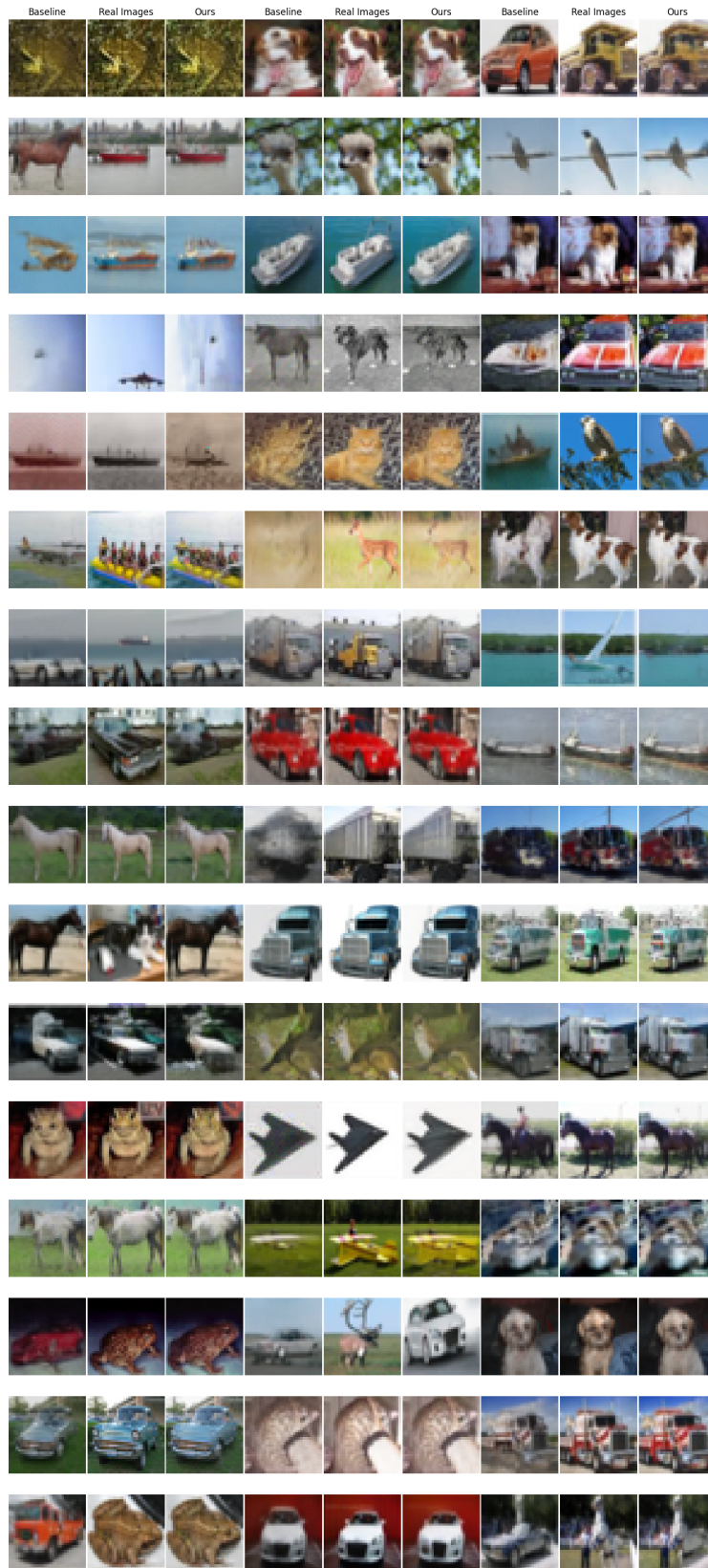
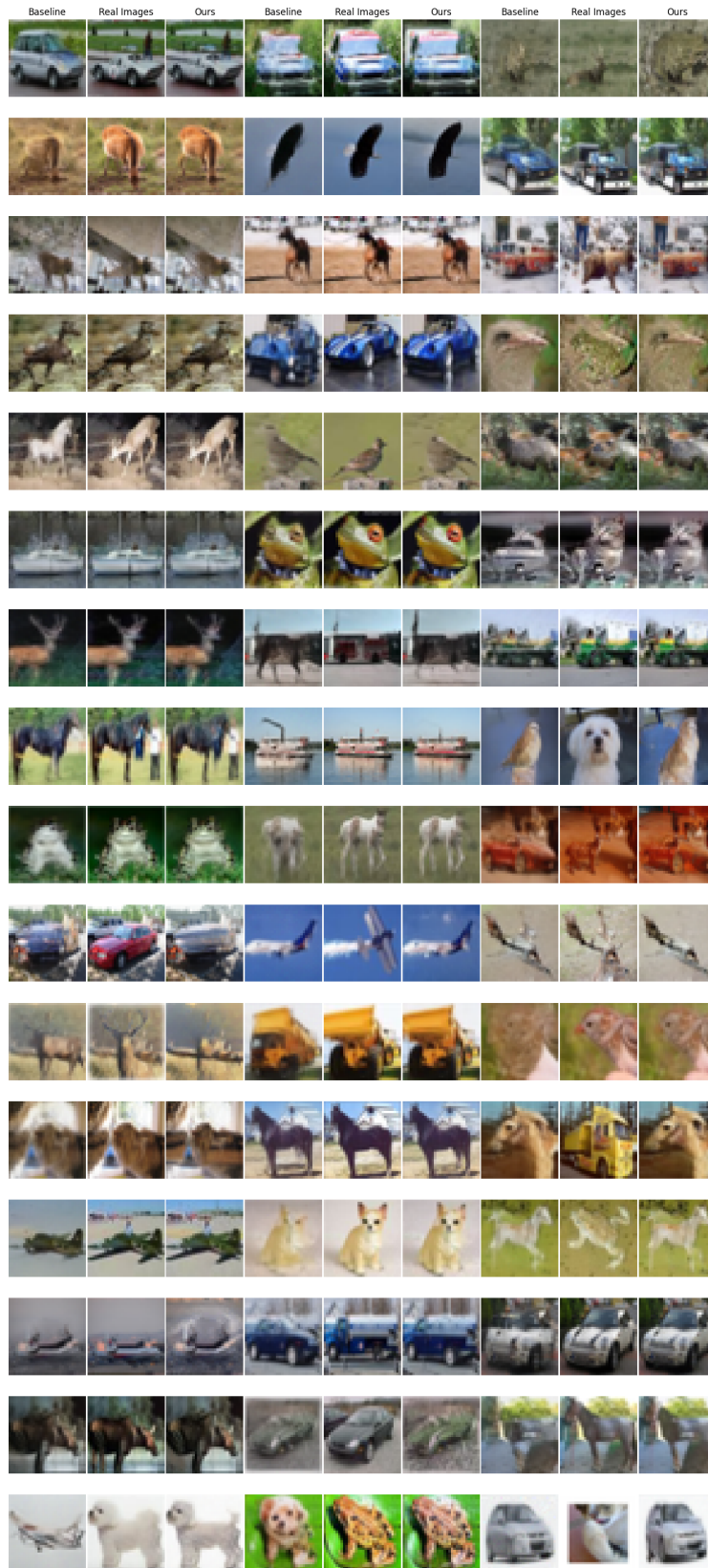
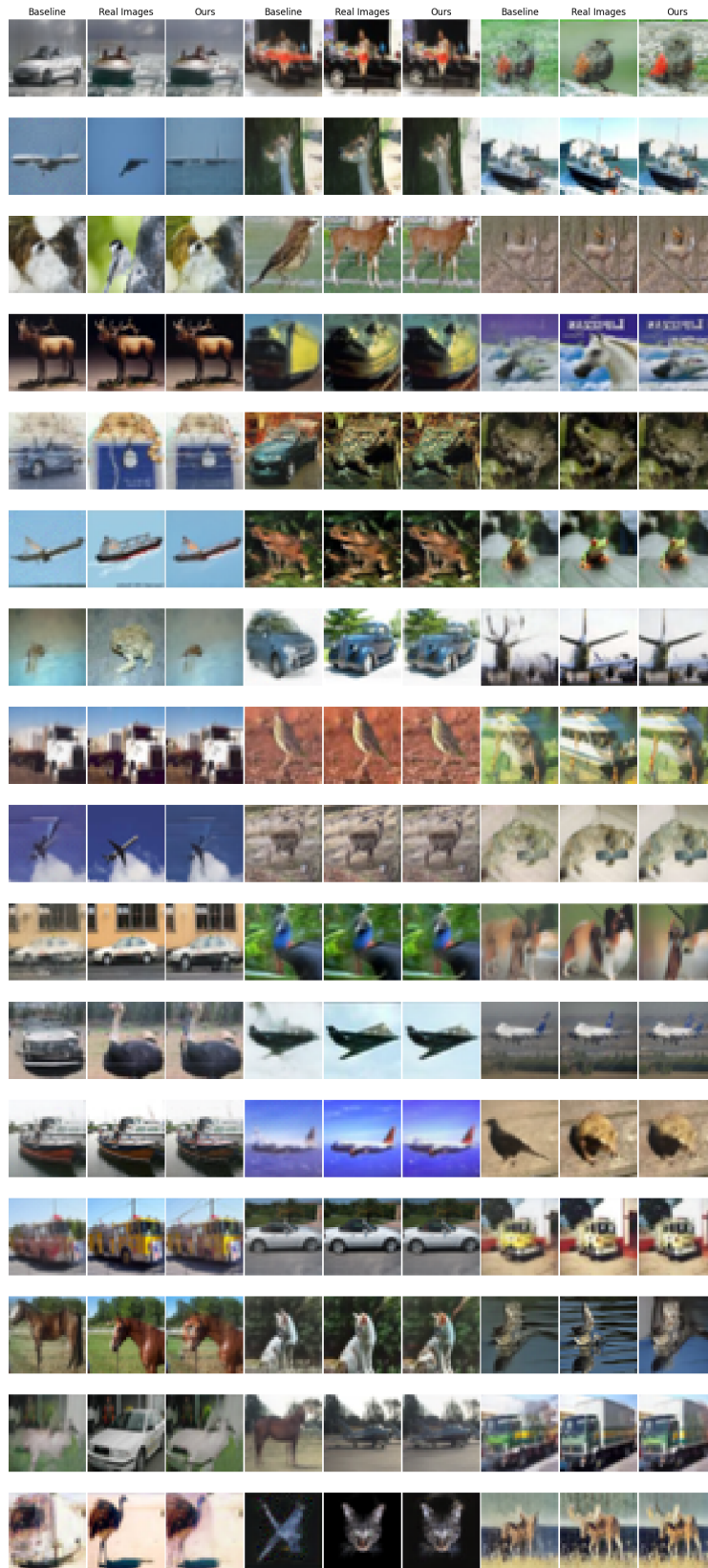


Figure A11: Ours (**up**) and Original (**down**) 2-rectified flow (2-step, seed 785)







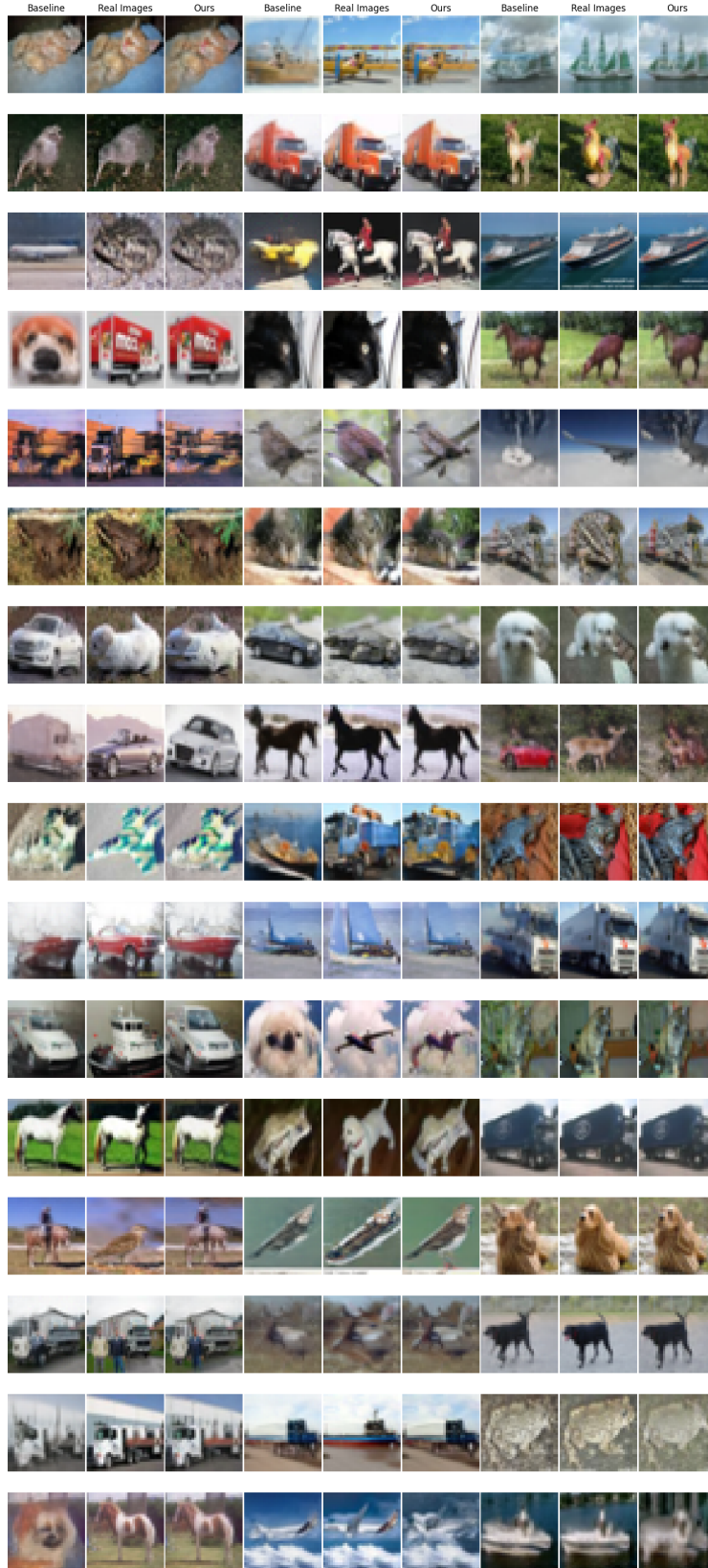


Figure A15: Compare with perturbed ( $\varepsilon = 0.04$ ) reconstruction image to original (1-step Euler)

## References

- [1] Jonas Adler, Emil Kobler, Sebastian Lunz, Carola-Bibiane Schönlieb, and Simon Arridge. Task adapted reconstruction for inverse problems. In *International Conference on Machine Learning*, pages 74–84. PMLR, 2021.
- [2] Luigi Ambrosio, Gianluca Crippa, Camillo De Lellis, Felix Otto, Michael Westdickenberg, Luigi Ambrosio, and Gianluca Crippa. Existence, uniqueness, stability and differentiability properties of the flow associated to weakly differentiable vector fields. *Transport equations and multi-D hyperbolic conservation laws*, pages 3–57, 2008.
- [3] Velin Antun, Francesco Renna, Clarice Poon, Ben Adcock, and Anders C Hansen. On instabilities of deep learning in image reconstruction and the potential costs of ai. *Proceedings of the National Academy of Sciences*, 117(48):30088–30095, 2020.
- [4] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.
- [5] Hongrui Chen, Holden Lee, and Jianfeng Lu. Improved analysis of score-based generative modeling: User-friendly bounds under minimal smoothness assumptions. In *International Conference on Machine Learning*, pages 4735–4763. PMLR, 2023.
- [6] 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.
- [7] 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 (CVPR)*, pages 248–255. IEEE, 2009.
- [8] P Kingma Diederik. Adam: A method for stochastic optimization. (*No Title*), 2014.
- [9] 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.
- [10] Alessio Figalli and Federico Glaudo. An invitation to optimal transport, wasserstein distances, and gradient flows. *Springer*, 2021.
- [11] R Flamary, N Courty, D Tuia, and A Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Trans. Pattern Anal. Mach. Intell*, 2016.
- [12] 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.
- [13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.
- [14] Young Kyun Jang, Dat Huynh, Ashish Shah, Wen-Kai Chen, and Ser-Nam Lim. Spherical linear interpolation and text-anchoring for zero-shot composed image retrieval. *arXiv preprint arXiv:2405.00571*, 2024.
- [15] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.
- [16] 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.
- [17] 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.
- [18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, ON, Canada, 2009. URL <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>. Technical report.
- [19] Thomas G Kurtz. Equivalence of stochastic equations and martingale problems. *Stochastic analysis 2010*, pages 113–130, 2011.

- [20] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.
- [21] Sangyun Lee, Beomsu Kim, and Jong Chul Ye. Minimizing trajectory curvature of ode-based generative models. In *International Conference on Machine Learning*, pages 18957–18973. PMLR, 2023.
- [22] Sangyun Lee, Zinan Lin, and Giulia Fanti. Improving the training of rectified flows. *arXiv preprint arXiv:2405.20320*, 2024.
- [23] Hangyu Li, Xiangxiang Chu, Dingyuan Shi, and Lin Wang. Flowdreamer: exploring high fidelity text-to-3d generation via rectified flow, 2024. URL <https://arxiv.org/abs/2408.05008>.
- [24] 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.
- [25] 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.
- [26] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.
- [27] Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.
- [28] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [29] Saiprasad Ravishankar, Jong Chul Ye, and Jeffrey A Fessler. Image reconstruction: From sparsity to data-adaptive methods and machine learning. *Proceedings of the IEEE*, 108(1):86–109, 2019.
- [30] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.
- [31] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.
- [32] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.
- [33] 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.
- [34] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.
- [35] Hans J Stetter et al. *Analysis of discretization methods for ordinary differential equations*, volume 23. Springer, 1973.
- [36] Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Conference on Learning Theory*, pages 3084–3114. PMLR, 2019.
- [37] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.
- [38] Francisco Vargas, Pierre Thodoroff, Austen Lamacraft, and Neil Lawrence. Solving schrödinger bridges via maximum likelihood. *Entropy*, 23(9):1134, 2021.
- [39] Cedric Villani. Optimal transport: old and new. *Springer*, 2009.
- [40] Cedric Villani. Topics in optimal transportation. *American Mathematical Soc*, 2021.
- [41] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

- 658 [42] Clinton Wang and Polina Golland. Interpolating between images with diffusion models, 2023. URL  
659 <https://openreview.net/pdf?id=L2D9Gybx0P>. OpenReview preprint.
- 660 [43] Fu-Yun Wang, Ling Yang, Zhaoyang Huang, Mengdi Wang, and Hongsheng Li. Rectified diffusion:  
661 Straightness is not your need in rectified flow. *arXiv preprint arXiv:2410.07303*, 2024.
- 662 [44] Yilun Xu, Ziming Liu, Max Tegmark, and Tommi Jaakkola. Poisson flow generative models. *Advances in*  
663 *Neural Information Processing Systems*, 35:16782–16795, 2022.
- 664 [45] Hanshu Yan, Xingchao Liu, Jiachun Pan, Jun Hao Liew, Qiang Liu, and Jiashi Feng. Perflow: Piecewise rec-  
665 tified flow as universal plug-and-play accelerator, 2024. URL <https://arxiv.org/abs/2405.07510>.
- 666 [46] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and  
667 Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF*  
668 *Conference on Computer Vision and Pattern Recognition*, pages 6613–6623, 2024.
- 669 [47] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Con-  
670 struction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint*  
671 *arXiv:1506.03365*, 2015.
- 672 [48] Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast  
673 sampling of diffusion models via operator learning. In *International conference on machine learning*,  
674 pages 42390–42402. PMLR, 2023.
- 675 [49] Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity  
676 distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In  
677 *Forty-first International Conference on Machine Learning*, 2024.
- 678 [50] Huminhao Zhu, Fangyikang Wang, Tianyu Ding, Qing Qu, and Zhihui Zhu. Analyzing and improving  
679 model collapse in rectified flow models. *arXiv preprint arXiv:2412.08175*, 2024.

## 680 Additional References

- 681 [A1] Stanczuk, Jan Pawel, Georgios Batzolis, Teo Deveney, and Carola-Bibiane Schönlieb. *Diffusion models*  
682 *encode the intrinsic dimension of data manifolds*. International Conference on Machine Learning (ICML),  
683 2024.
- 684 [A2] Mingi Kwon, Shin seong Kim, Jaeseok Jeong, Yi Ting Hsiao, and Youngjung Uh. Tcfg: *Tangential*  
685 *damping classifier-free guidance*. 2025. URL <https://arxiv.org/abs/2503.18137>.
- 686 [A3] Dhariwal, Prafulla and Alexander Nichol. *Diffusion models beat GANs on image synthesis*. Advances in  
687 Neural Information Processing Systems (NeurIPS), 34:8780–8794, 2021.