A Related Works

Diffusion Models and Flow-based Models Diffusion models [76; 28; 80; 81; 79; 78; 33; 13; 31; 21; 65; 102; 51; 92; 12] have achieved unprecedented results in various generative modeling tasks, including image/video generation [27; 101; 86; 94; 71], audio generation [36], point cloud generation [53; 54; 46; 87], biological generation [91; 55; 86; 30], etc.. Most of the works are based on stochastic differential equations (SDEs), and researchers have explored techniques to transform them into marginal-preserving probability flow ordinary differential equations (ODEs) [80; 77]. Recently, [45; 43; 40; 1; 23] propose to directly learn probability flow ODEs by constructing linear or non-linear interpolations between two distributions. These ODEs obtain comparable performance as diffusion models, but require much fewer inference steps. Among these approaches, Rectified Flow [45; 43] introduces a special *reflow* procedure which enhances the coupling between distributions and squeezes the generative ODE to one-step generation. However, the effectiveness of reflow has only been examined on small datasets like CIFAR10, thus raising questions about its suitability on large-scale models and big data. In this paper, we demonstrate that the Rectified Flow pipeline can indeed enable high-quality one-step generation in large-scale text-to-image diffusion models, hence brings ultra-fast T2I foundation models with pure supervised learning.

Large-Scale Text-to-Image Generation Early research on text-to-image generation focused on small-scale datasets, such as flowers and birds [68; 69; 97]. Later, the field shifted its attention to more complex scenarios, particularly in the MS COCO dataset [39], leading to advancements in training and generation [83; 98; 37]. DALL-E [66] was the pioneering transformer-based model that showcased the amazing zero-shot text-to-image generation capabilities by scaling up the network size and the dataset scale. Subsequently, a series of new methods emerged, including autoregressive models [14; 15; 18; 96], GAN inversion [11; 44], GAN-based approaches [103], and diffusion models [59; 71; 67; 64; 31]. Among them, Stable Diffusion is an open-source text-to-image generator based on latent diffusion models [70]. It is trained on the LAION 5B dataset [74] and achieves the state-of-the-art generalization ability. Additionally, GAN-based models like StyleGAN-T [73] and GigaGAN [32] are trained with adversarial loss to generate high-quality images rapidly. Our work provides a novel approach to yield ultra-fast, one-step, large-scale generative models without the delicate adversarial training.

Acceleration of Diffusion Models Despite the impressive generation quality, diffusion models are known to be slow during inference due to the requirement of multiple iterations to reach the final result. To accelerate inference, there are two categories of algorithms. The first kind focuses on fast post-hoc samplers [33; 42; 49; 50; 77; 4; 10]. These fast samplers can reduce the number of inference steps for pre-trained diffusion models to 20-50 steps. However, relying solely on inference to boost performance has its limitations, necessitating improvements to the model itself [85; 90; 38]. Distillation [25] has been applied to pre-trained diffusion models [52], squeezing the number of inference steps to below 10. Progressive distillation [72] is a specially tailored distillation procedure for diffusion models, and has successfully produced 2/4-step Stable Diffusion [58]. Consistency models [82] are a new family of generative models that naturally operate in a one-step manner. There are several works concurrent with InstaFlow for accelerating large-scale text-to-image models: BOOT [20] designs a data-free distillation pipeline for pre-trained diffusion models with a bootstrapping method; Latent Consistency Model [56] adopts consistency distillation with a skipping scheme; UFOGen [93] combines diffusion models with GAN for high-quality distillation; Yin et al. [95] proposes Distribution Matching Distillation to align the distribution generated from the one-step model with the teacher Stable Diffusion. Instead of employing sophisticated distillation or GAN loss, InstaFlow uses Rectified Flow [45; 43] and its unique *reflow* procedure to straighten the ODE trajectories. With simple supervised learning on a least-squares problem, it refines the coupling between the noise distribution and the image distribution, thereby improving the performance of direct distillation.

B Neural Network Structure

The whole pipeline of our text-to-image generative model consists of three parts: the text encoder, the generative model in the latent space, and the decoder. We use the same text encoder and decoder as Stable Diffusion: the text encoder is adopted from CLIP ViT-L/14 and the latent decoder is

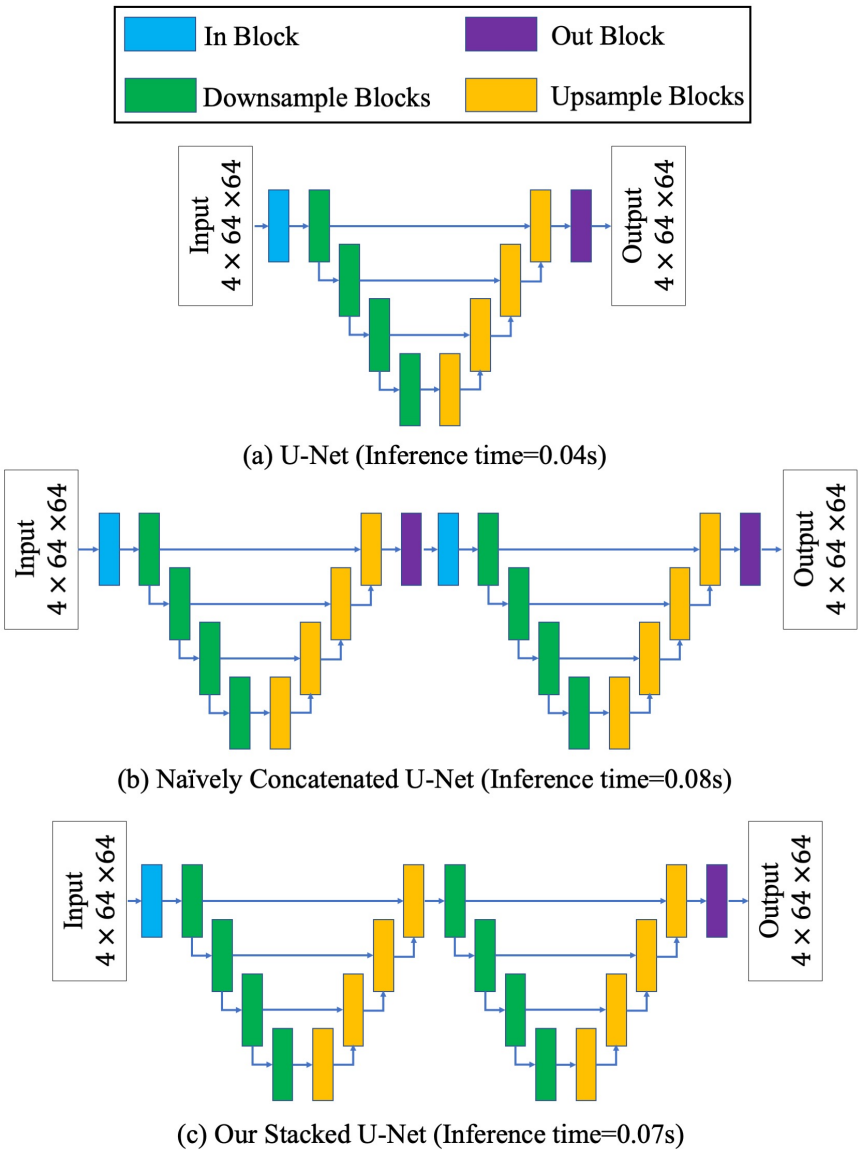

Figure 12: Different neural network structures for distillation and their inference time. The blocks with the same colors can share weights.

adopted from a pre-trained auto-encoder with a downsampling factor of 8. During training, the parameters in the text encoder and the latent decoder are frozen. On average, to generate 1 image on NVIDIA A100 GPU with a batch size of 1, text encoding takes 0.01s and latent decoding takes 0.04s.

By default, the generative model in the latent space is a U-Net structure. For reflow, we do not change any of the structure, but just fine-tune the model. For distillation, we tested three network structures, as shown in Figure 12. The first structure is the original U-Net structure in SD. The second structure is obtained by directly concatenating two U-Nets with shared parameters. We found that the second structure significantly decrease the distillation loss and improve the quality of the generated images after distillation, but it doubles the computational time.

To reduce the computational time, we tested a family of networks structures by deleting different blocks in the second structure. By this, we can examine the importance of different blocks in this concatenated network in distillation, remove the unnecessary ones and thus further decrease inference time. We conducted a series of ablation studies, including:

1. Remove 'Downsample Blocks 1 (the green blocks on the left)'
2. Remove 'Upsample Blocks 1 (the yellow blocks on the left)'
3. Remove 'In+Out Block' in the middle (the blue and purple blocks in the middle).
4. Remove 'Downsample Blocks 2 (the green blocks on the right)'
5. Remove 'Upsample blocks 2 (the yellow blocks on the right)'

The only one that would not hurt performance is Structure 3, and it gives us a 7.7% reduction in inference time ($\frac{0.13-0.12}{0.13} = 0.0769$). This third structure, Stacked U-Net, is illustrated in Figure 12 (c).

C Additional Details and Results on the Preliminary Experiments

General Experiment Settings In this section, we use the pre-trained Stable Diffusion 1.4 provided in the official open-sourced repository[3] to initialize the weights, since otherwise the convergence is unbearably slow.

In our experiment, we set $D_{\mathcal{T}}$ to be a subset of text prompts from laion2B-en [74], pre-processed by the same filtering as SD. $\texttt{ODE}[v_{\texttt{SD}}]$ is implemented as the pre-trained Stable Diffusion with 25-step DPMSolver [49] and a fixed guidance scale of 6.0. We set the similarity loss $\mathbb{D}(\cdot, \cdot)$ for distillation to be the LPIPS loss [100]. The neural network structure for both reflow and distillation are kept to the SD U-Net. We use a batch size of 32 and 8 A100 GPUs for training with AdamW optimizer [48]. Our training script is based on the official fine-tuning script provided by HuggingFace[4] and the choice of optimizer follows the default protocol. We use exponential moving average with a factor of 0.9999, following the default configuration. We clip the gradient to reach a maximal gradient norm of 1. We warm-up the training process for 1,000 steps in both reflow and distillation. BF16 format is adopted during training to save GPU memory. To compute the LPIPS loss, we used its official 0.1.4 version[5] and its model based on AlexNet. The learning rate for reflow is 10^{-6}.

We measure the inference time of our models on a server with NVIDIA A100 GPU, and a batch size of 1. We use `PyTorch 2.0.1` and `Hugging Face Diffusers 0.19.3`. For fair comparison, we use the inference time of standard SD on our computational platform for Progressive Distillation-SD as their model is not available publicly. The inference time contains the text encoder and the latent decoder, but does NOT contain NSFW detector.

[3] `https://github.com/CompVis/stable-diffusion`

[4] `https://huggingface.co/docs/diffusers/training/text2image`

[5] `https://github.com/richzhang/PerceptualSimilarity`

FID: weight decay / learning rate	10^{-1}	10^{-2}	10^{-3}
10^{-5}	44.04	45.90	44.03
10^{-6}	137.05	134.83	139.31
10^{-7}	$\sim$297	$\sim$297	$\sim$297

Table 3: FID of different distilled SD models measured with 5000 images on MS COCO2017.

C.1 Additional Details and Results of Direct Distillation

Grid Search To achieve the best empirical performance, we conduct grid search on learning rate and weight decay to the limit of our computational resources. Particularly, the learning rates are selected from $\{10^{-5}, 10^{-6}, 10^{-7}\}$ and the weight decay coefficients are selected from $\{10^{-1}, 10^{-2}, 10^{-3}\}$. For all the 9 models, we train them for $100,000$ steps. We generate $32 \times 100,000 = 3,200,000$ pairs of $(X_0, \texttt{ODE}[v_{\text{SD}}](X_0))$ as the training set for distillation.

Additional Results We provide additional results on direct distillation of Stable Diffusion 1.4, shown in Figure 18, 19, 20, 21 and Table 3. Although increasing the learning rate boosts the performance, we found that a learning rate of $\geq 10^{-4}$ leads to unstable training and NaN errors. A small learning rate, like 10^{-6} and 10^{-7}, results in slow convergence and blurry generation even after training $100,000$ steps.

C.2 Additional Quantitative Comparison

We provide additional quantitative results with parameter-sharing Stacked U-Net and multiple reflow in Table 4 and 5. According to equation 5, the reflow procedure can be repeated for multiple times. We repeat reflow for one more time to get 3-Rectified Flow (v_3), which is initialized from 2-Rectified Flow (v_2). 3-Rectified Flow is trained to minimize equation 5 for $50,000$ steps. Then we get its distilled version by generating $1,600,000$ new pairs of $(X_0, \texttt{ODE}[v_3](X_0))$ and distill for another $50,000$ steps. We found that to stabilize the training process of 3-Rectified Flow and its distillation, we have to decrease the learning rate from 10^{-6} to 10^{-7}.

C.3 Estimation of Training Cost

Measured on our platform, when training with batch size of 4 and U-Net, one A100 GPU day can process $100,000$ iterations using L2 loss, $86,000$ iterations using LPIPS loss; when generating pairs with batch size of 16, one A100 GPU day can generate $200,000$ data pairs. We compute the computational cost according to this.

Estimated Training Cost of (Pre) 2-Rectified Flow+Distill (U-Net): $3,200,000/200,000$ (Data Generation) + $32/4 \times 50,000/100,000$ (Reflow) + $32/4 \times 50,000/86,000$ (Distillation) $\approx$ 24.65 A100 GPU days.

Estimated Training Cost of Progressive Distillation: We refer to Appendix C.2.1 (LAION-5B 512×512) of [58] and estimate the training cost. PD starts from 512 steps, and progressively applies distillation to 1 step with a batch size of 512. Quoting the statement 'For stage-two, we train the model with 2000-5000 gradient updates except when the sampling step equals to 1,2, or 4, where we train for 10000-50000 gradient updates', a lower-bound estimation of gradient updates would be 2000 (512 to 256) + 2000 (256 to 128) + 2000 (128 to 64) + 2000 (64 to 32) + 2000 (32 to 16) + 5000 (16 to 8) + 10000 (8 to 4) + 10000 (4 to 2) + 50000 (2 to 1) = 85,000 iterations. Therefore, one-step PD at least requires $512/4 \times 85000/100000 = 108.8$ A100 GPU days. Note that we ignored the computational cost of stage 1 of PD and '2 steps of DDIM with teacher' during PD, meaning that the real training cost is higher than 108.8 A100 GPU days.

Method	Inference Time	FID-5k	CLIP
SD 1.4-DPM Solver (25 step)[70; 49]	0.88s	22.8	**0.315**
(Pre) 2-Rectified Flow (25 step)	0.88s	**22.1**	0.313
(Pre) 3-Rectified Flow (25 step)	0.88s	23.6	0.309
Progressive Distillation-SD (1 step)[58]	0.09s	37.2	0.275
SD 1.4+Distill (U-Net)	0.09s	40.9	0.255
(Pre) 2-Rectified Flow (1 step)	0.09s	68.3	0.252
(Pre) 2-Rectified Flow+Distill (U-Net)	0.09s	31.0	**0.285**
(Pre) 3-Rectified Flow (1 step)	0.09s	37.0	0.270
(Pre) 3-Rectified Flow+Distill (U-Net)	0.09s	**29.3**	0.283
Progressive Distillation-SD (2 step)[58]	0.13s	26.0	0.297
Progressive Distillation-SD (4 step)[58]	0.21s	26.4	0.300
SD 1.4+Distill (Stacked U-Net)	0.12s	52.0	0.269
(Pre) 2-Rectified Flow+Distill (Stacked U-Net)	0.12s	**24.6**	0.306
(Pre) 3-Rectified Flow+Distill (Stacked U-Net)	0.12s	26.3	**0.307**

Table 4: Comparison of FID on MS COCO 2017 following the evaluation setup in [58]. As in [32; 73], the inference time is measured on NVIDIA A100 GPU, with a batch size of 1, `PyTorch 2.0.1` and `Huggingface Diffusers 0.19.3`. 2-Rectified Flow+Distill outperforms Progressive Distillation within the same inference time using much less training cost. The numbers for Progressive Distillation are measured from Figure 10 in [58]. 'Pre' is added to distinguish the models from Table 2.

Method	Inference Time	# Param.	FID-30k
SD* [70]	2.9s	0.9B	**9.62**
(Pre) 2-Rectified Flow (25 step)	0.88s	0.9B	13.4
SD 1.4+Distill (U-Net)	0.09s	0.9B	34.6
(Pre) 2-Rectified Flow+Distill (U-Net)	0.09s	0.9B	**20.0**
SD 1.4+Distill (Stacked U-Net)	0.12s	0.9B	41.5
(Pre) 2-Rectified Flow+Distill (Stacked U-Net)	0.12s	0.9B	**13.7**

Table 5: Comparison of FID on MS COCO 2014 with $30,000$ images. Note that the models distilled after reflow has noticeable advantage compared with direct distillation even when (Pre) 2-Rectified Flow has worse performance than the original SD due to insufficient training. $*$ denotes that the numbers are measured by [32]. 'Pre' is added to distinguish the models from Table 2. As in StyleGAN-T [73] and GigaGAN [32], our generated images are downsampled to 256×256 before computing FID.

D Additional Training Details on InstaFlow

Implementation Details and Training Pipeline for InstaFlow-0.9B We switch to Stable Diffusion 1.5, and keep the same $D_{\mathcal{T}}$ as in Section C. The ODE solver sticks to 25-step DPMSolver [49] for $\texttt{ODE}[v_{\texttt{SD}}]$. Guidance scale is slightly decreased to 5.0 because larger guidance scale makes the images generated from 2-Rectified Flow over-saturated. Since distilling from 2-Rectified Flow yields satisfying results, 3-Rectiifed Flow is not trained. We still generate $1,600,000$ pairs of data for reflow and distillation, respectively. To expand the batch size to be larger than $4 \times 8 = 32$, gradient accumulation is applied. The overall training pipeline for 2-Rectified Flow+Distill (U-Net) is summarized as follows:

1. **Reflow (Stage 1)**: We train the model using the reflow objective equation 5 with a batch size of 64 for 70,000 iterations. The model is initialized from the pre-trained SD 1.5 weights. (11.2 A100 GPU days)
2. **Reflow (Stage 2)**: We continue to train the model using the reflow objective equation 5 with an increased batch size of 1024 for 25,000 iterations. The final model is **2-Rectified Flow**. (64 A100 GPU days)
3. **Distill (Stage 1)**: Starting from the 2-Rectified Flow checkpoint, we fix the time $t = 0$ for the neural network, and fine-tune it using the distillation objective equation 6 with a batch size of 1024 for 21,500 iterations. The guidance scale α of the teacher model, 2-Rectified Flow, is set to 1.5 and the similarity loss $\mathbb{D}$ is L2 loss. (54.4 A100 GPU days)

4. **Distill (Stage 2)**: We switch the similarity loss $\mathbb{D}$ to LPIPS loss, then we continue to train the model using the distillation objective equation 6 and a batch size of 1024 for another 18,000 iterations. The final model is 2-Rectified Flow+Distill (U-Net). We name it **InstaFlow-0.9B**. (53.6 A100 GPU days)

The total training cost for InstaFlow-0.9B is $3,200,000/200,000$ (Data Generation) + 11.2 + 64 + 54.4 + 53.6 = 199.2 A100 GPU days.

Implementation Details and Training Pipeline for InstaFlow-1.7B We adopt the Stacked U-Net structure in Section C, but abandon the parameter-sharing strategy. This gives us a Stacked U-Net with 1.7B parameters, almost twice as large as the original U-Net. Starting from 2-Rectified Flow, 2-Rectified Flow+Distill (Stacked U-Net) is trained by the following distillation steps:

1. **Distill (Stage 1)**: The Stacked U-Net is initialized from the weights in the 2-Rectified Flow checkpoint. Then we fix the time $t = 0$ for the neural network, and fine-tune it using the distillation objective equation 6 with a batch size of 64 for 110,000 iterations. The similarity loss $\mathbb{D}$ is L2 loss. (35.2 A100 GPU days)
2. **Distill (Stage 2)**: We switch the similarity loss $\mathbb{D}$ to LPIPS loss, then we continue to train the model using the distillation objective equation 6 and a batch size of 320 for another 2,500 iterations. The final model is 2-Rectified Flow+Distill (Stacked U-Net). We name it **InstaFlow-1.7B**. (4.4 A100 GPU days)

Discussion 1 (Experiment Observations) During training, we made the following observations: (1) the 2-Rectified Flow model did not fully converge and its performance could potentially benefit from even longer training duration; (2) distillation showed faster convergence compared to reflow; (3) the LPIPS loss had an immediate impact on enhancing the visual quality of the distilled one-step model. Based on these observations, we believe that with more computational resources, further improvements can be achieved for the one-step models.

Discussion 2 (One-Step Stacked U-Net and Two-Step Progressive Distillation) Although one-step Stacked U-Net and 2-step progressive distillation (PD) need similar inference time, they have two key differences: (1) 2-step PD additionally minimizes the distillation loss at $t = 0.5$, which may be unnecessary for one-step generation from $t = 0$; (2) by considering the consecutive U-Nets as one model, we are able to examine and remove redundant components from this large neural network, further reducing the inference time by approximately 8% (from $0.13s$ to $0.12s$).

E Additional Qualitative Results

We provide additional qualitative results in Figure 13, 14, 15, 16, including inspections on the latent spaces of InstaFlow-0.9B and InstaFlow-1.7B, and visual comparison of few-step generation and guidance scale α. We also show uncurated images generated from 20 random LAION text prompts with the same random noises for visual comparison. The images from different models are shown in Figure 22, 23, 24, 25.

Alignment between 2-Rectified Flow and the One-Step Models The learned latent spaces of generative models have intriguing properties. By properly exploiting their latent structure, prior works succeeded in image editing [89; 34; 24; 84; 57; 47; 101], semantic control [61; 11; 44], disentangled control direction discovery [22; 63; 88; 75], etc.. In general, the latent spaces of one-step generators, like GANs, are usually easier to analyze and use than the multi-step diffusion models. One advantage of our pipeline is that it gives a multi-step continuous flow and the corresponding one-step models simultaneously. Figure 17 shows that the latent spaces of our distilled one-step models align with 2-Rectified Flow. Therefore, the one-step models can be good surrogates to understand and leverage the latent spaces of continuous flow, since the latter one has higher generation quality.

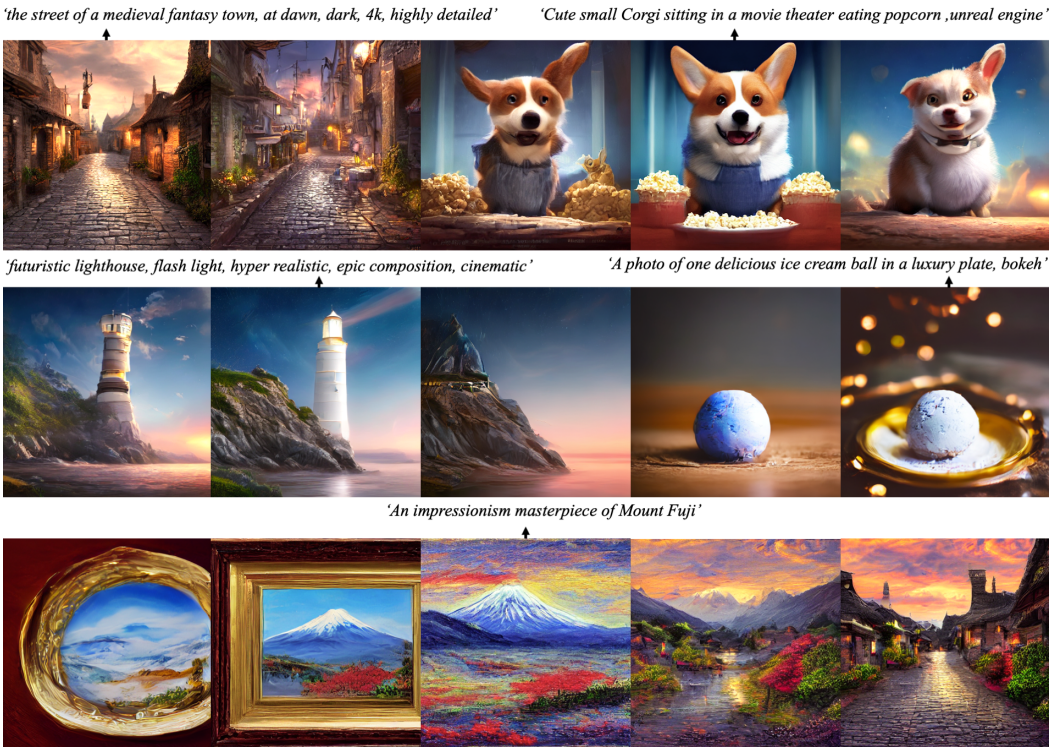

Figure 13: Latent space interpolation of our one-step InstaFlow-0.9B. The images are generated in $0.09s$, saving $\sim 90\%$ of the computational time from the 25-step SD-1.5 teacher model in the inference stage.

Figure 14: Images generated from our one-step InstaFlow-1.7B in $0.12s$. With the same random noise, the pose and lighting are preserved across different text prompts.

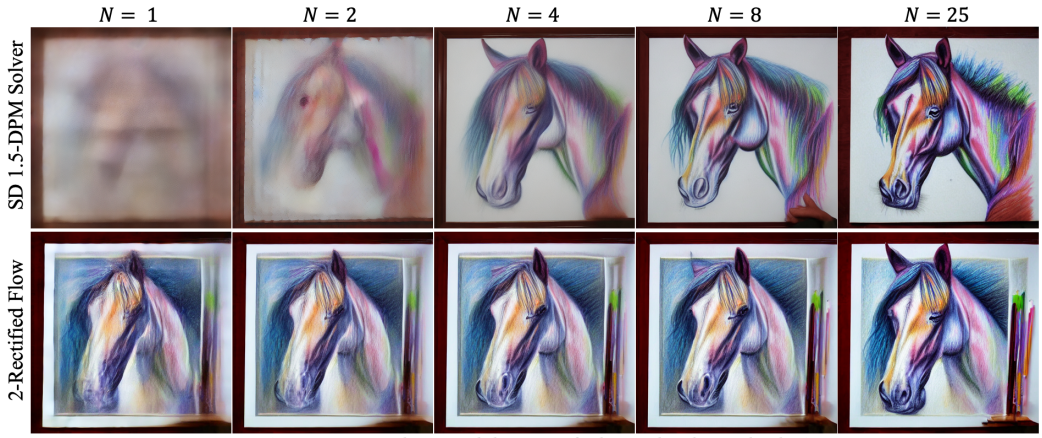

Figure 15: Visual comparison with different number of inference steps N. With the same random seed, 2-Rectified Flow can generate clear images when $N \leq 4$, while SD 1.5-DPM Solver cannot.

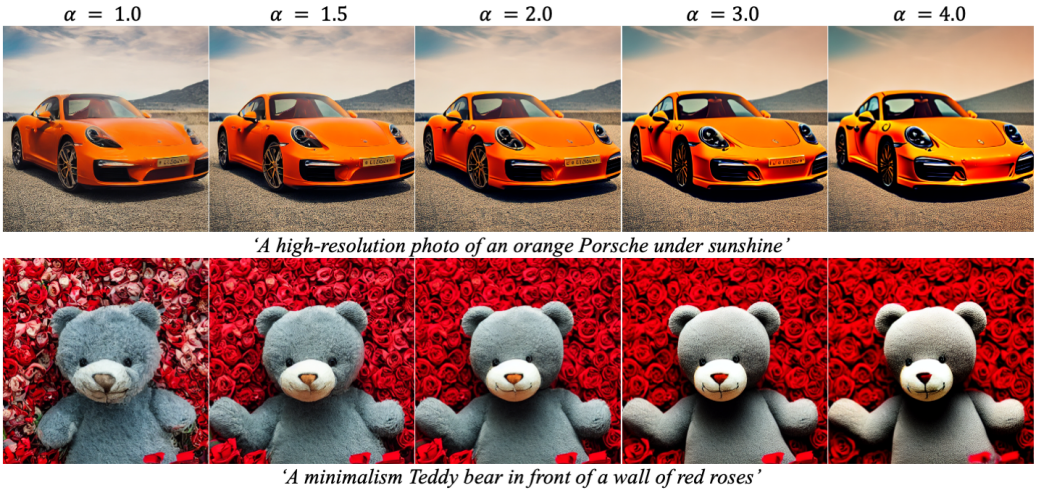

Figure 16: Visual comparison with different guidance scale α on 2-Rectified Flow. When $\alpha = 1.0$, the generated images have blurry edges and twisted details; when $\alpha \geq 2.0$, the generated images gradually gets over-saturated.

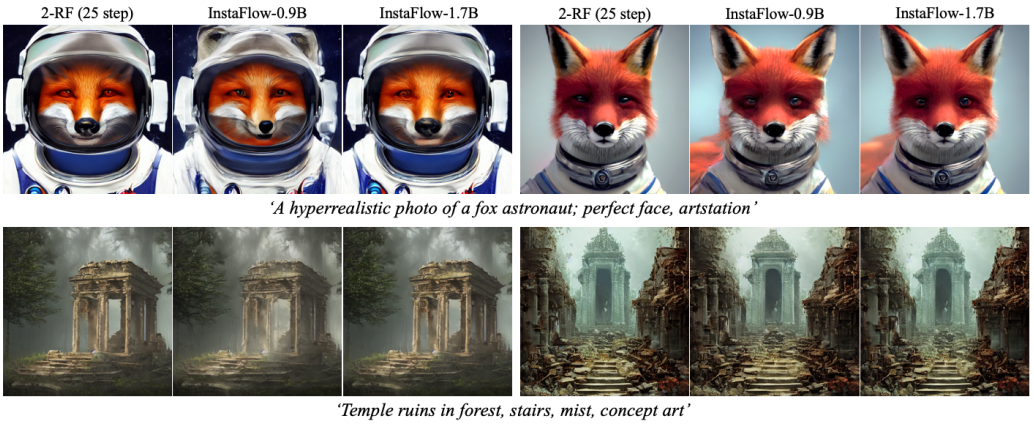

Figure 17: With the same random noise and text prompts, the one-step models generate similar images with the continuous 2-Rectified Flow, indicating their latent space aligns. Therefore, the one-step models can be good surrogates to analyze the properties of the latent space of the continuous flow.

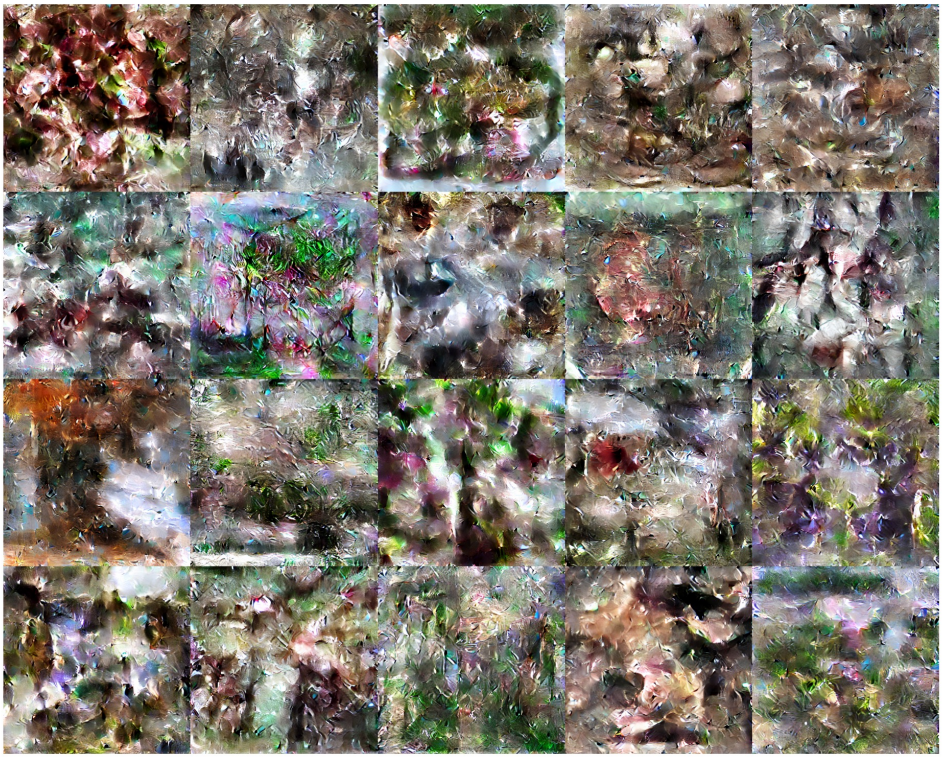

Figure 18: Uncurated samples from SD+Distill (U-Net) trained with a learning rate of 10^{-7} and a weight decay coefficient of 10^{-3}.

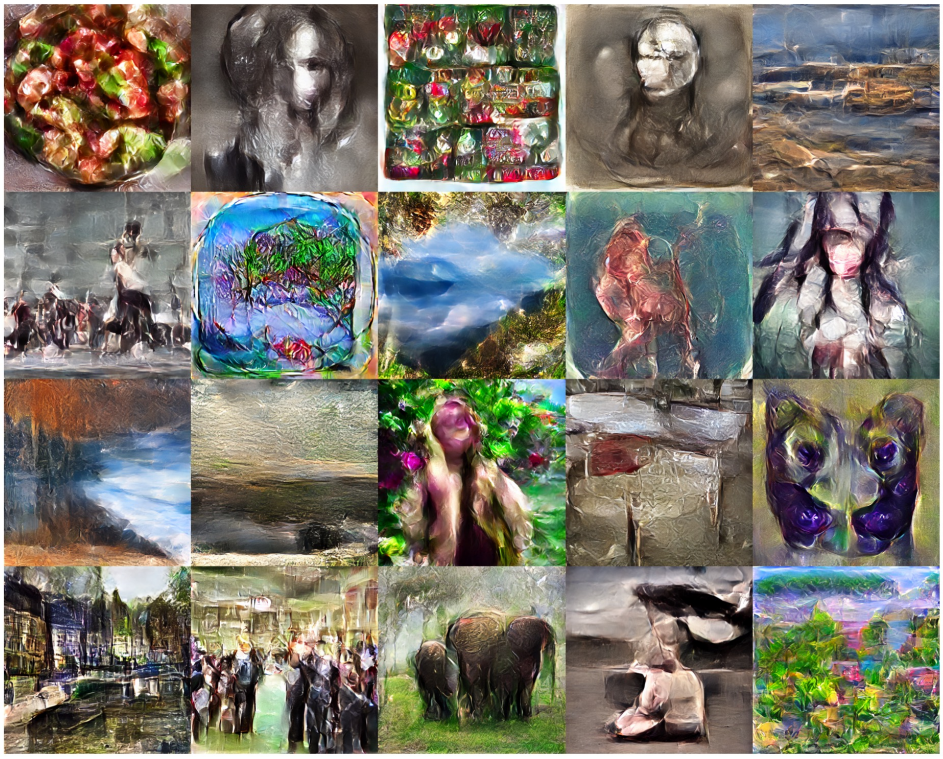

Figure 19: Uncurated samples from SD+Distill (U-Net) trained with a learning rate of 10^{-6} and a weight decay coefficient of 10^{-3}.

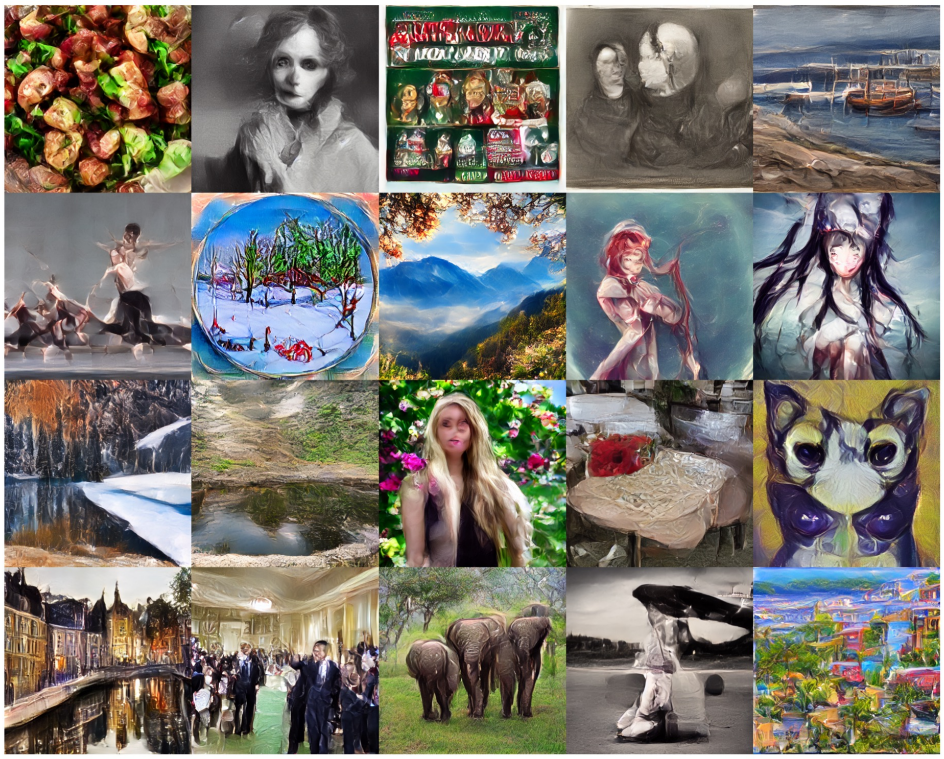

Figure 20: Uncurated samples from SD+Distill (U-Net) trained with a learning rate of 10^{-5} and a weight decay coefficient of 10^{-3}.

Figure 21: Uncurated samples from SD+Distill (Stacked U-Net) trained with a learning rate of 10^{-5} and a weight decay coefficient of 10^{-3}.

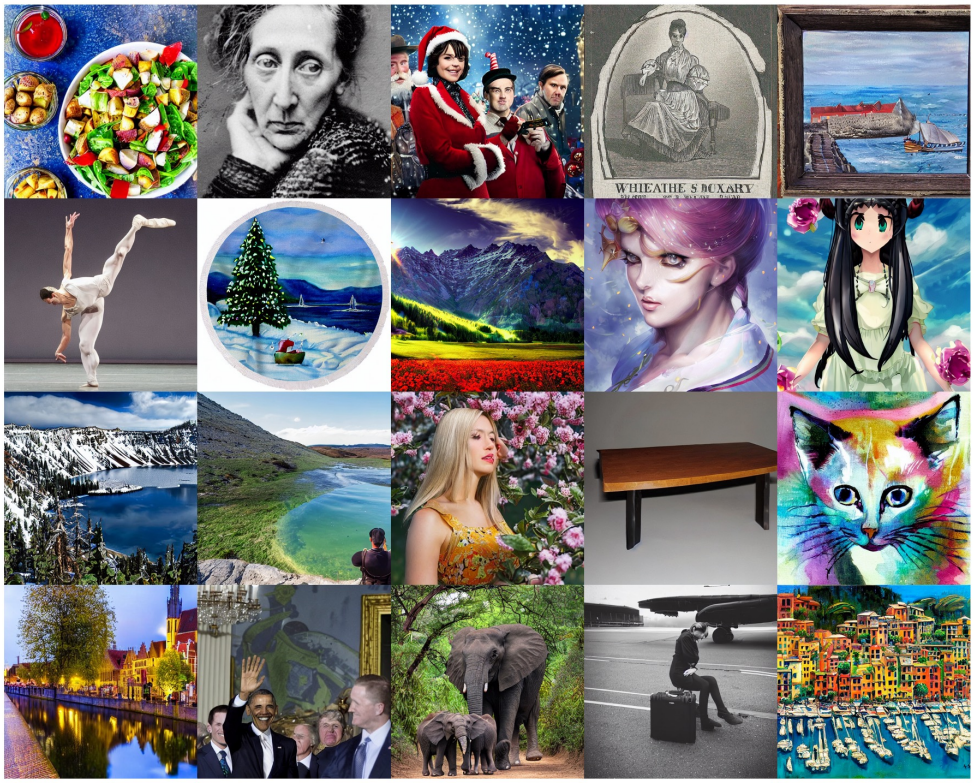

Figure 22: Uncurated samples from Stable Diffusion 1.5 with 25-step DPMSolver [49] and guidance scale 5.0.

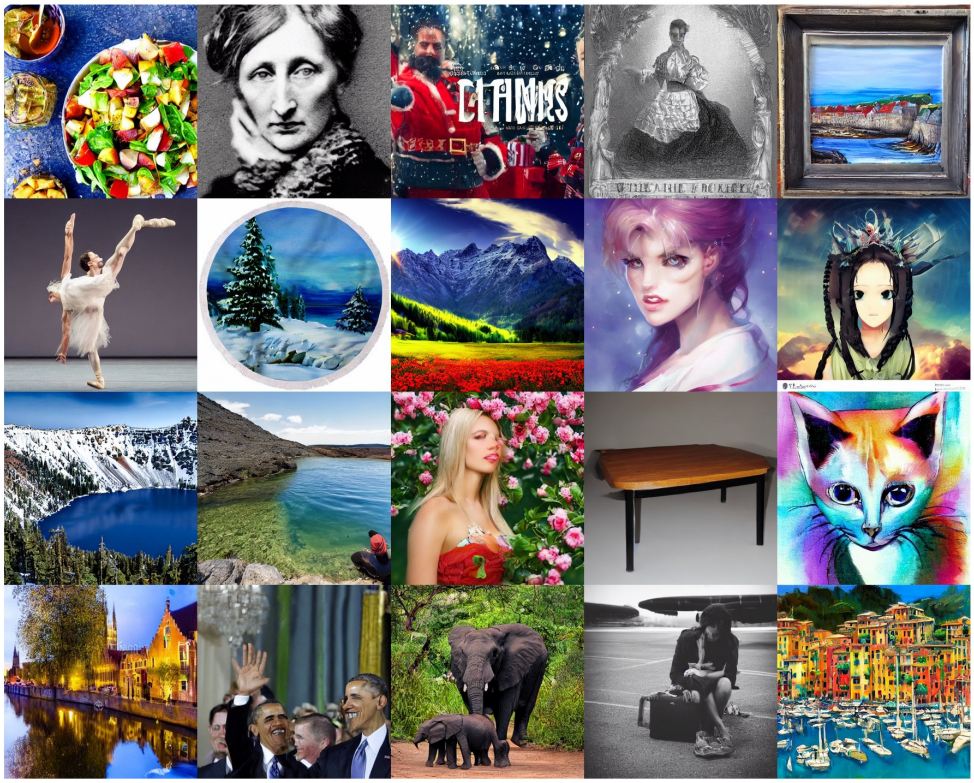

Figure 23: Uncurated samples from 2-Rectified Flow with guidance scale 1.5 and 25-step Euler solver.

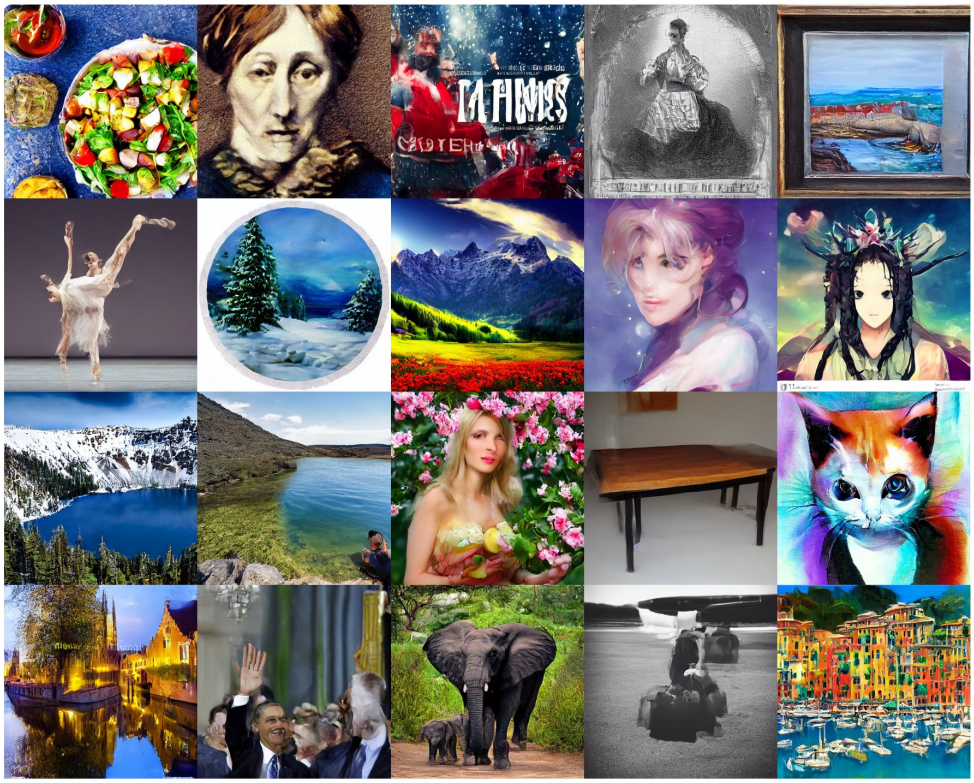

Figure 24: Uncurated samples from one-step InstaFlow-0.9B

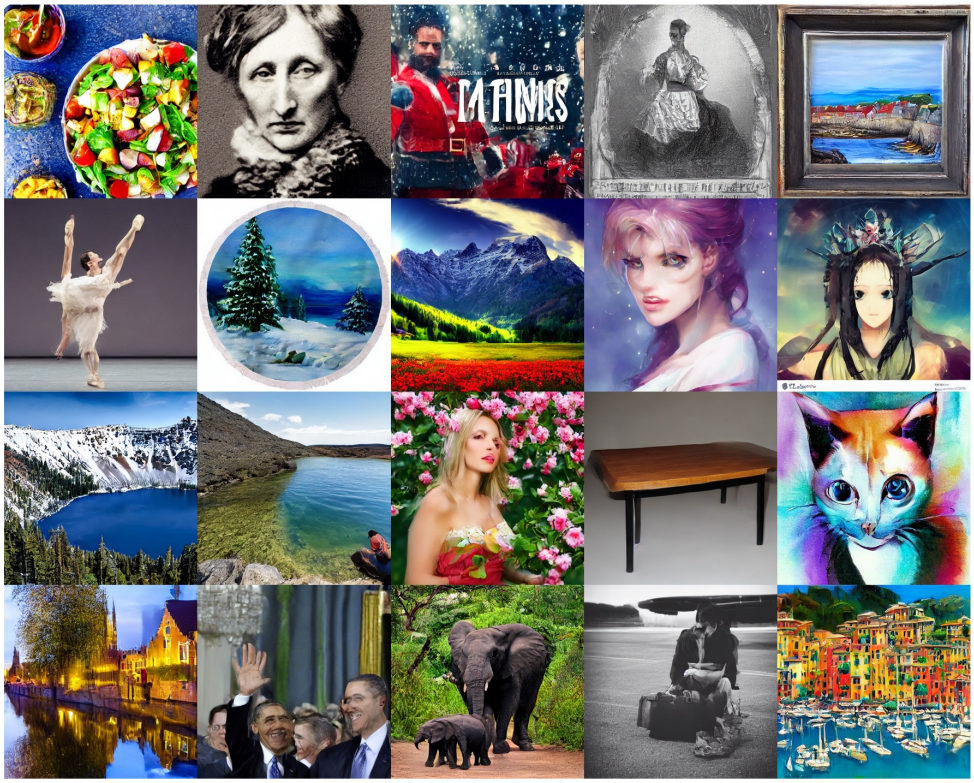

Figure 25: Uncurated samples from one-step InstaFlow-1.7B

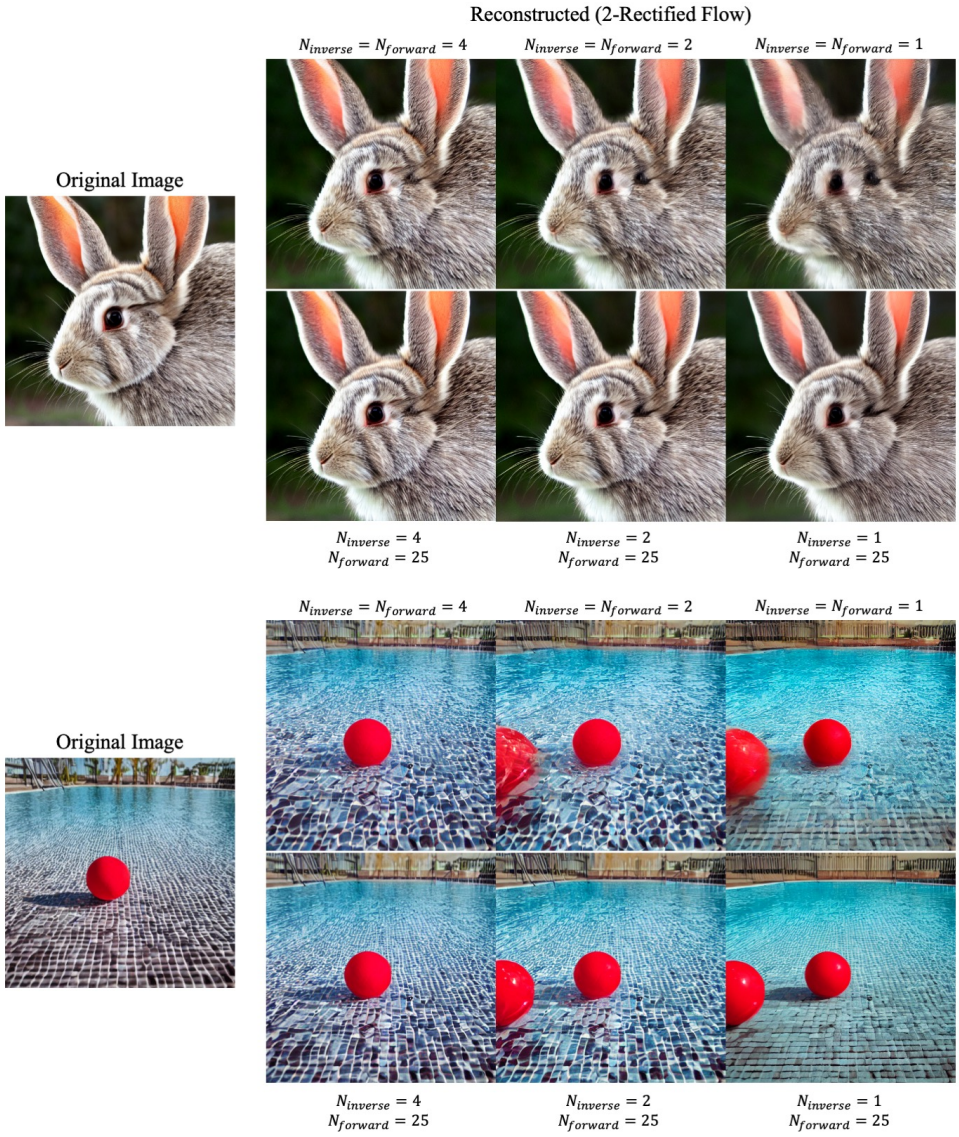

Figure 26: Examples of image encoding and reconstruction with 2-Rectified Flow. Here, we encode an image X_1 to the latent noise space by simulating the inverse probability flow ODE, $X_0 = X_1 + \int_1^0 v(X_t, t)\mathrm{d}t$. Then we reconstruct the image from the latent encoding X_0 by simulating the forward probability flow ODE, $\hat{X}_1 = X_0 + \int_0^1 v(X_t, t)\mathrm{d}t$. We show the reconstructed image $\hat{X}_1$. For both stages, we adopt Euler solver and use $N_{inverse}$ and $N_{forward}$ steps respectively. With as few as 4 steps, 2-Rectified Flow can encode and reconstruct the original image successfully. Since 2-Rectified Flow is straighter, the encoding stage works well with even 1 step.

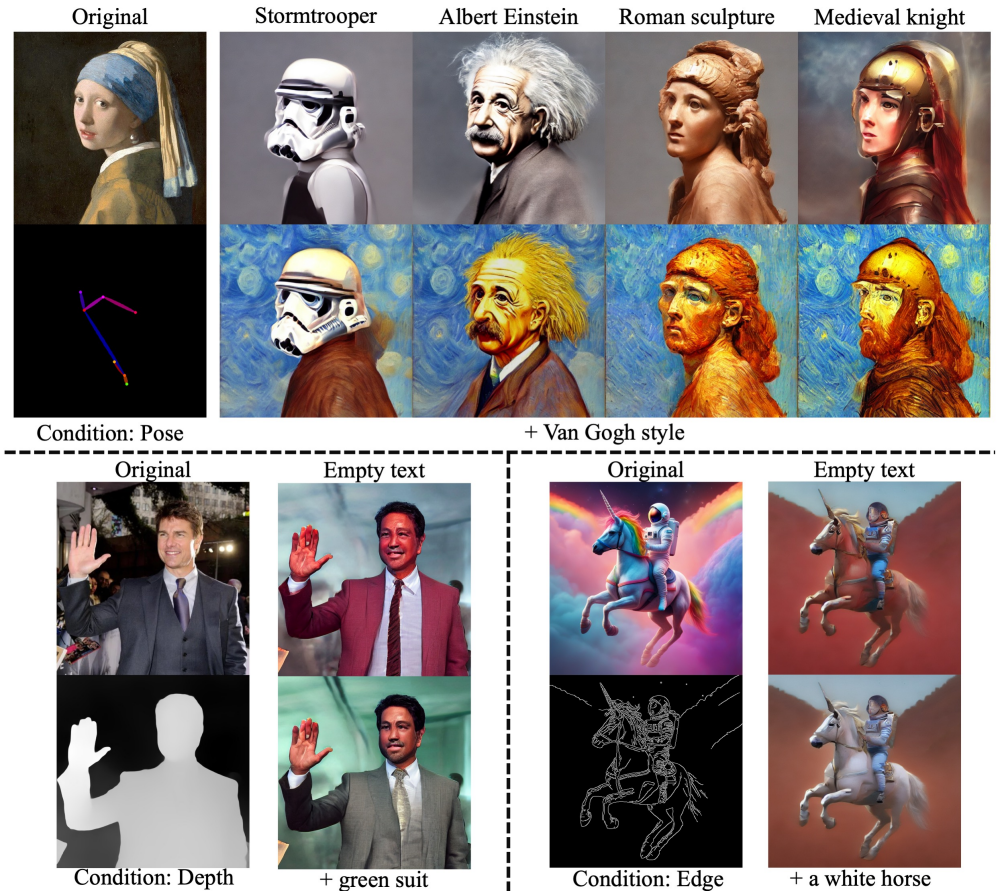

Figure 27: We surprisingly find that pre-trained InstaFlow is fully compatible with ControlNets [99] pre-trained with Stable Diffusion. The images shown here are generated with one-step InstaFlow + pre-trained ControlNets.