PeRFlow: Piecewise Rectified Flow as Universal Plug-and-Play Accelerator

Hanshu Yan[*], **Xingchao Liu**[+], **Jiachun Pan**[#], **Jun Hao Liew**[*], **Qiang Liu**[+], **Jiashi Feng**[*]
[*]ByteDance, [+]Univeristy of Texas at Austin, [#]National University of Singapore
hanshu.yan@outlook.com

Abstract

We present Piecewise Rectified Flow (PeRFlow), a flow-based method for accelerating diffusion models. PeRFlow divides the sampling process of generative flows into several time windows and straightens the trajectories in each interval via the reflow operation, thereby approaching piecewise linear flows. PeRFlow achieves superior performance in a few-step generation. Moreover, through dedicated parameterizations, the PeRFlow models inherit knowledge from the pretrained diffusion models. Thus, the training converges fast and the obtained models show advantageous transfer ability, serving as universal plug-and-play accelerators that are compatible with various workflows based on the pre-trained diffusion models. Codes for training and inference have been publicly released. [1].

1 Introduction

Diffusion models have exhibited impressive generation performances across different modalities, such as image [34, 9, 39, 2], video [10, 55, 42, 15, 48], and audio [11]. Diffusion models generate samples by reversing pre-defined complicated diffusion processes, thus requiring many inference steps to synthesize high-quality results. Such expensive computational cost hinders their deployment [14, 40, 31] in real-world applications.

Diffusion models can be efficiently sampled by solving the corresponding probability flow ordinary differential equations (PF-ODEs) [39, 38]. Researchers have designed many advanced samplers, such as DDIM [38], DPM-solver [24], and DEIS [53], to accelerate generation, inspired by the semi-linear structure and adaptive solvers in ODEs. However, these samplers still require tens of inference steps to generate satisfying results. Researchers have also explored distilling pretrained diffusion models into few-step generative models [35, 28, 5, 50, 30, 3], which have succeeded in synthesizing images within 8 inference steps. Progressive Distillation [35] separates the whole sampling process into multiple segments and learns the mapping from starting points to endpoints for each segment. Distribution Matching Distillation [50] and SwiftBrush [30] use the score distillation loss to align the distributions of teacher and one-step student generators. UFOGen [47], SDXL-Turbo [36] and SDXL-Lightning [16] resort to adversarial training for learning few-step/one-step image generators. They initialize the students from pretrained diffusion models and use adversarial and/or MSE losses to align the student model's generation with the pretrained ones. These methods suffer from the difficult tuning of the adversarial training procedure and the mode collapse issue. Latent Consistency Model (LCM) [26, 27] adopts consistency distillation [40] to train a generator that directly maps noises to the terminal images. LCM only utilizes supervised distillation where the training procedure will be more stable and easier in comparison to adversarial training. However, the generated images have fewer details compared with SDXL-Lighting.

[1] https://github.com/magic-research/piecewise-rectified-flow

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

https://doi.org/10.52202/079017-2497

Unlike the existing methods above, which mainly learn the mappings from noises to images, we aim to simplify the flow trajectories and preserve the continuous flow trajectories of the original pretrained diffusion models. Specifically, we attempt to straighten the trajectories of the original PF-ODEs via a piecewise reflow operation. Previously, InstaFlow [21] leverages the rectified flow framework[20, 19] to learn the transformation from initial random noise to images. It bridges the two distributions with linear interpolation and trains the model by matching the interpolation. With the reflow operation, it may be able to learn straight-line flows for one-step generation via pure supervised learning. InstaFlow provides a simple pipeline for accelerating pretrained diffusion models, however, it suffers from poor sampling quality which can be attributed to synthetic data generation. The reflow operation requires generating data from the pretrained diffusion models with ODE solvers (e.g., DDIM or DPM-Solver [24, 25]) to construct a training dataset. Synthesizing training data brings two problems: (1) constructing and storing the dataset requires excessive time and space, which limits its training efficiency; (2) synthetic data has a noticeable gap with real training data in quality due to the numerical error of solving ODEs. Thus, the performance of the learned straighter flow is bounded.

To address the problems, *we propose piecewise rectified flow (PeRFlow), which divides the flow trajectories into several time windows and conducts reflow in each window*. By solving the ODEs in the shortened time interval, PeRFlow avoids simulating the entire ODE trajectory for preparing the training data. This significantly reduces the target synthesis time, enabling the simulation to be performed in real time along with the training procedure. Besides, PeRFlow samples the starting noises by adding random noises to clean images according to the marginal distributions, and solves the endpoints of a shorter time interval, which has a lower numerical error than integrating the entire trajectories. Through such a divide-and-conquer strategy, PeRFlow can straighten the sampling trajectories with large-scale real training data. Besides the training framework, *we also design a dedicated parameterization method for PeRFlow to inherit sufficient knowledge from the pretrained diffusion models*. Diffusion models are usually trained with ϵ-prediction, but flow-based generative models generate data by following the velocity field. We derive the correspondence between ϵ-prediction and the velocity field of flow, thus narrowing the gap between the pretrained diffusion models and the student PeRFlow model. Consequently, PeRFlow acceleration converges fast and the resultant model can synthesize highly-detailed images within very few steps. PeRFlow does not require unstable adversarial training or a complete modification of the training paradigm. It is a lightweight acceleration framework and can be easily applied to training unconditional/conditional generative models of different data modalities.

We conducted extensive experiments to verify the effectiveness of PeRFlow on accelerating pretrained diffusion models, including Stable Diffusion (SD) 1.5, SD 2.1, SDXL [32], and AnimateDiff [6]. PeRFlow-accelerated models can generate high-quality results within four steps. Moreover, we find that the variation of the weights, $\Delta W = \theta - \phi$, between the trained student model θ and the pretrained diffusion model ϕ, can serve as universal accelerators of almost all workflows that are only trained on the pretrained diffusion models. These workflows include customized SD models, ControlNets, and multiview 3D generation. We compared PeRFlow with state-of-the-art acceleration methods. PeRFlow shows advantages in terms of FID values, visual quality, and generation diversity.

In summary, PeRFlow has the following favorable features: 1) it is simple and flexible for accelerating various diffusion pipelines with fast convergence; 2) The accelerated generators support fast generation; 3) The obtained ΔW shows superior plug-and-play compatibility with the workflows of the pretrained models.

2 Methodology

2.1 Rectified Flow and Reflow

Flow-based generative models aim to learn a velocity field $\boldsymbol{v}_\theta(\boldsymbol{z}_t, t)$ that transports random noise $\boldsymbol{z}_1 \sim \pi_1$ sampled from a noise distribution into certain data distribution $\boldsymbol{z}_0 \sim \pi_0$. Then, one can generate samples by solving (1) from $t = 1$ to 0:

$$\mathrm{d}\boldsymbol{z}_t = \boldsymbol{v}_\theta(\boldsymbol{z}_t, t)\mathrm{d}t, \quad \boldsymbol{z}_1 \sim \pi_1. \tag{1}$$

Recently, simulation-free learning of flow-based models has become prevalent [20, 19, 18, 1]. A representative method is Rectified flow [20, 19, 18], which adopts linear interpolation between the noise distribution $\mathbf{z}_1$ and the data distribution $\mathbf{z}_0$. It trains a neural network $\boldsymbol{v}_\theta$ to approximate the

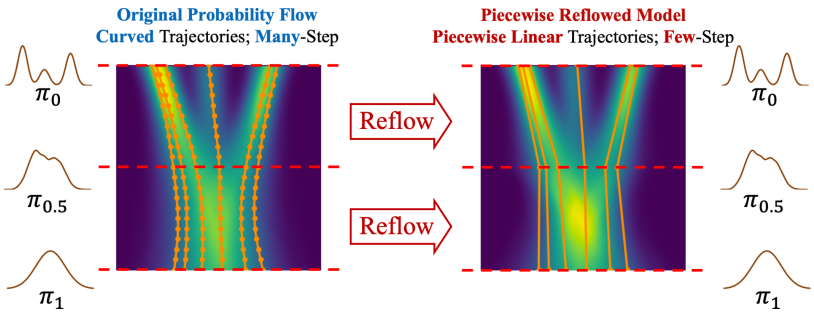

Figure 1: Our few-step generator PeRFlow is trained by a divide-and-conquer strategy. We divide the ODE trajectories into several intervals and perform reflow in each time window to straighten the sampling trajectories.

velocity field via the conditional flow matching loss. The corresponding optimization procedure is termed reflow [20, 19],

$$\min_{\theta} \mathbb{E}_{\boldsymbol{z}_1\sim\pi_1,\boldsymbol{z}_0\sim\pi_0}\left[\int_0^1 \|(\boldsymbol{z}_1-\boldsymbol{z}_0)-v_\theta(\boldsymbol{z}_t,t)\|^2\mathrm{d}t\right], \quad \text{with} \quad \boldsymbol{z}_t=(1-t)\boldsymbol{z}_0+t\boldsymbol{z}_1. \tag{2}$$

InstaFlow [21] proposed to accelerate pretrained diffusion-based text-to-image models via reflow. Given a pretrained diffusion model f_ϕ, one can generate new data by solving the corresponding probability flow ODE. We denote $\Phi(\boldsymbol{z}_t,t,s)$ as the ODE solver, such as the DPM-Solver [23]. For simplicity, our notation drops the parameters in the ODE solvers. By simulating with $\boldsymbol{z}_0=\Phi(\boldsymbol{z}_1,1,0)$, where $\boldsymbol{z}_1$ is sampled from the random Gaussian distribution π_1, it synthesizes a dataset of `(text, noise, image)` triplets for reflow. Since it usually takes tens of inference steps to generate high-quality data with $\Phi(\boldsymbol{z}_1,1,0)$, InstaFlow is expensive to scale up. Moreover, since InstaFlow is trained with generated images, it lacks the supervision of real data and thus compromises the resulting generation quality. In the following subsections, we target solving these problems.

2.2 Piecewise Rectified Flow

We present Piecewise Rectified Flow (PeRFlow), aiming at training a piecewise linear flow to approximate the sampling process of a pretrained diffusion model. PeRFlow sticks to the idea of trajectory straightening. It further allows using high-quality real training data and one-the-fly optimization. PeRFlow is easier to scale up and succeeds in accelerating large-scale diffusion models, including the Stable Diffusion family.

A pretrained diffusion model f_ϕ corresponds to a probability flow ODE defined by a noise schedule $\sigma(t)$. In the Stable Diffusion family, the forward diffusion process follows $\boldsymbol{z}_t=\sqrt{1-\sigma^2(t)}\boldsymbol{z}_0+\sigma(t)\epsilon$, where $\boldsymbol{z}_0$ and ϵ are sampled from the data distribution and random Gaussian respectively. The sampling trajectories are usually complicated curves. Even for an advanced ODE solver $\Phi(\boldsymbol{z}_t,t,s)$, it still requires many steps to generate an artifact-free image. We accelerate the pretrained model by applying a divide-and-conquer strategy, that is, we divide the ODE trajectories into multiple time windows and straighten the trajectories in each time window via the reflow operation.

We create K time windows $\{[t_k,t_{k-1})\}_{k=K}^1$ where $1=t_K>\cdots>t_k>t_{k-1}>\cdots>t_0=0$. For each time window $[t_k,t_{k-1})$, the starting distribution π_k will be the marginal distribution of the diffusion process at time t_k. It can be derived from $\boldsymbol{z}_{t_k}=\sqrt{1-\sigma^2(t_k)}\boldsymbol{z}_0+\sigma(t_k)\epsilon$. The target end distribution π_{k-1} is constructed by $\Phi(\boldsymbol{z}_{t_k},t_k,t_{k-1})$. We train the PeRFlow model, denoted by θ, to fit the linear interpolation between $\mathbf{z}_{t_k}$ and $\mathbf{z}_{t_{k-1}}$ for all $k\in[1,\dots,K]$.

$$\begin{aligned}&\min_{\theta}\sum_{k=1}^{K}\mathbb{E}_{\boldsymbol{z}_{t_k}\sim\pi_k}\left[\int_{t_{k-1}}^{t_k}\left\|\frac{\boldsymbol{z}_{t_{k-1}}-\boldsymbol{z}_{t_k}}{t_{k-1}-t_k}-v_\theta(\boldsymbol{z}_t,t)\right\|^2\mathrm{d}t\right],\\ &\text{with}\quad \boldsymbol{z}_{t_{k-1}}=\Phi(\boldsymbol{z}_{t_k},t_k,t_{k-1})\quad\text{and}\quad \boldsymbol{z}_t=\frac{t-t_{k-1}}{t_k-t_{k-1}}\boldsymbol{z}_{t_k}+\frac{t_k-t}{t_k-t_{k-1}}\boldsymbol{z}_{t_{k-1}}.\end{aligned} \tag{3}$$

Parameterization The pretrained diffusion models are usually trained by two parameterization tricks, namely ϵ-*prediction* and *velocity-prediction*. To inherit knowledge from the pretrained network,

https://doi.org/10.52202/079017-2497

we parameterize the PeRFlow model as the same type of diffusion and initialize network θ from the pretrained diffusion model ϕ. For the *velocity-prediction*, we can train the PeRFlow model by velocity-matching in (3). To accommodate ϵ-*prediction*, we can represent the denoised state $\boldsymbol{z}_{t_{k-1}}$ with the starting state $\boldsymbol{z}_{t_k}$ and ϵ:

$$\boldsymbol{z}_{t_{k-1}} = \lambda_k \boldsymbol{z}_{t_k} + \eta_k \epsilon, \tag{4}$$

where $\lambda_k > 1$ and η_k are defined by the user. We propose to train a neural network $\epsilon_\theta(\boldsymbol{z}_t, t)$ to estimate the noise ϵ in (4) based on $\boldsymbol{z}_t$ for all $t \in [t_k, t_{k-1})$:

$$\begin{aligned} &\min_\theta \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{z}_{t_k}\sim\pi_k}\left[\int_{t_{k-1}}^{t_k} \left\| \frac{\boldsymbol{z}_{t_{k-1}} - \lambda_k \boldsymbol{z}_{t_k}}{\eta_k} - \epsilon_\theta(\boldsymbol{z}_t, t)\right\|^2 \mathrm{d}t\right], \\ &\text{with} \quad \boldsymbol{z}_{t_{k-1}} = \Phi(\boldsymbol{z}_{t_k}, t_k, t_{k-1}) \quad \text{and} \quad \boldsymbol{z}_t = \frac{t - t_{k-1}}{t_k - t_{k-1}}\boldsymbol{z}_{t_k} + \frac{t_k - t}{t_k - t_{k-1}}\boldsymbol{z}_{t_{k-1}}. \end{aligned} \tag{5}$$

The optimum of (3) and (5) are,

$$v^*(\boldsymbol{z}_t, t) = \mathbb{E}\left[\frac{\boldsymbol{z}_{t_{k-1}} - \boldsymbol{z}_{t_k}}{t_{k-1} - t_k}\middle| \boldsymbol{z}_t\right], \quad \text{and} \quad \epsilon^*(\boldsymbol{z}_t, t) = \mathbb{E}\left[\frac{\boldsymbol{z}_{t_{k-1}} - \lambda_k \boldsymbol{z}_{t_k}}{\eta_k}\middle| \boldsymbol{z}_t\right].$$

Using calculus and the fact $\boldsymbol{z}_t = \frac{t-t_{k-1}}{t_k - t_{k-1}}\boldsymbol{z}_{t_k} + \frac{t_k - t}{t_k - t_{k-1}}\boldsymbol{z}_{t_{k-1}}$, we get,

$$v^*(\boldsymbol{z}_t, t) = \frac{(1-\lambda_k)\boldsymbol{z}_t - \eta_k \epsilon^*(\boldsymbol{z}_t, t)}{t - t_{k-1} + \lambda_k t_k - \lambda_k t} \tag{6}$$

The sampling process involves first computing $\epsilon_\theta(\boldsymbol{z}_t, t)$ from $\boldsymbol{z}_t$, then estimating the velocity $\boldsymbol{v}(\boldsymbol{z}_t)$ via (6) for solving the ODE (1). In this paper, we consider two choices for λ and η:

- *Parameterization [A]*: According to the definition of the diffusion process, we have $\boldsymbol{z}_{t_k} = \gamma \boldsymbol{z}_{t_{k-1}} + \sqrt{1-\gamma^2}\epsilon$ with $\gamma = \sqrt{(1-\sigma_k^2)/(1-\sigma_{k-1}^2)}$. We can represent $\boldsymbol{z}_{t_k}$ with $\boldsymbol{z}_{t_{k-1}}$ and yield,

$$\lambda_k = \frac{\sqrt{1-\sigma_{k-1}^2}}{\sqrt{1-\sigma_k^2}}, \quad \eta_k = -\frac{\sqrt{\sigma_k^2 - \sigma_{k-1}^2}}{\sqrt{1-\sigma_k^2}}. \tag{7}$$

- *Parameterization [B]*: We can also follow the DDIM solver [38], i.e.,

$$\boldsymbol{z}_{t_{k-1}} = \sqrt{\frac{\alpha_{t_{k-1}}}{\alpha_{t_k}}}\boldsymbol{z}_{t_k} + \sqrt{\alpha_{t_{k-1}}}\left(\sqrt{\frac{1-\alpha_{t_{k-1}}}{\alpha_{t_{k-1}}}} - \sqrt{\frac{1-\alpha_{t_k}}{\alpha_{t_k}}}\right)\epsilon_\theta(\boldsymbol{z}_{t_k}, t_k),$$

  where $\alpha_k = 1 - \sigma_k^2$. We can correspondingly set,

$$\lambda_k = \frac{\sqrt{\alpha_{k-1}}}{\sqrt{\alpha_k}}, \quad \eta_k = \sqrt{1-\alpha_{t_{k-1}}} - \frac{\sqrt{\alpha_{k-1}}}{\sqrt{\alpha_k}}\sqrt{1-\alpha_k}. \tag{8}$$

  This parameterization initializes the student flow from the update rule of DDIM, which is equivalent to the Euler discretization of the probability flow ODE. We empirically observe that it gives faster training convergence.

Scaling Up with Real Training Data PeRFlow divides the time range $[1, 0]$ into multiple windows. For each window, the starting point $\boldsymbol{z}_{t_k}$ is obtained by adding random noise to *real* training data z_0, and it only requires several inference steps to solve the ending point $\boldsymbol{z}_{t_{k-1}}$. The computational cost is significantly reduced for each training iteration compared to InstaFlow, allowing us to train PeRFlow on large-scale training datasets with fast online simulation of the ODE trajectory. Besides, solving endpoints of a shorter time window $[\boldsymbol{z}_{t_k}, \boldsymbol{z}_{t_{k-1}})$ has lower numerical errors in comparison to the entire time range. High-quality supervision yields significant improvement in the generation results.

Classifier-Free Guidance in Training Classifier-free guidance (CFG) [7] is a common technique to improve the generation quality of text-to-image models. During training, we solve the endpoints $\boldsymbol{z}_{t_{k-1}}$ for each time window $[t_k, t_{k-1})$ in an online manner via an ODE solver $\Phi(\boldsymbol{z}_{t_k}, t_k, t_{k-1}, \boldsymbol{c}, w)$, where $w \geq 1$ denotes the CFG scale, $\boldsymbol{c}$ denotes the text prompt. CFG is turned off when $w = 1$. PeRFlow supports two modes: *CFG-sync* and *CFG-fixed*:

Algorithm 1: Piecewise Rectified Flow

Input: Training dataset $\mathcal{D}$, ϵ- or $\boldsymbol{v}$-prediction teacher model f_ϕ, Noise schedule $\sigma(t)$, ODE solver $\Phi(z_t, t, s, f_\phi)$, Number of windows K, student model ϵ_θ or $\boldsymbol{v}_\theta$,

Create K time windows $\{(t_{k-1}, t_k]\}_{k=1}^K$ with $t_K = 1$ and $t_0 = 0$;
Initialize $\theta = \phi$;
repeat
  Sample $z_0 \sim \mathcal{D}$;
  Sample k from $\{1, \cdots, K\}$ uniformly, then randomly sample time $t \in (t_{k-1}, t_k]$;
  Sample random noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$;
  Get $\boldsymbol{z}_{t_k} = \sqrt{1 - \sigma^2(t_k)}\boldsymbol{z}_0 + \sigma(t_k)\epsilon$;
  Solve the endpoint of the time window $z_{t_{k-1}} = \Phi(\boldsymbol{z}_{t_k}, t_k, t_{k-1})$;
  Get $\boldsymbol{z}_t = \boldsymbol{z}_{t_k} + \frac{\boldsymbol{z}_{t_k} - \boldsymbol{z}_{t_{k-1}}}{t_k - t_{k-1}}(t - t_k)$;
  if *ϵ-prediction* **then**
    Compute loss $\ell = \left\| \epsilon_\theta(\boldsymbol{z}_t, t) - \frac{\boldsymbol{z}_{t_{k-1}} - \lambda_k \boldsymbol{z}_{t_k}}{\eta_k} \right\|^2$;
  else
    Compute loss $\ell = \left\| \boldsymbol{v}_\theta(\boldsymbol{z}_t, t) - \frac{\boldsymbol{z}_{t_k} - \boldsymbol{z}_{t_{k-1}}}{t_k - t_{k-1}} \right\|^2$;
  end
  Update θ with gradient-based optimizer using $\nabla_\theta \ell$.
until *convergence*;
$\Delta W = \theta - \phi$.

Return: Fast PeRFlow f_θ and ΔW.

- *CFG-sync*: We disable CFG by setting $w = 1$ for $\Phi(\boldsymbol{z}_{t_k}, t_k, t_{k-1}, \boldsymbol{c}, w)$. The obtained PeRFlow model can use similar CFG scales as the pretrained diffusion models to guide the sampling.
- *CFG-fixed*: We use a pre-defined $w = w^* > 1$ for $\Phi(\boldsymbol{z}_{t_k}, t_k, t_{k-1}, \boldsymbol{c}, w)$ during training. The obtained PeRFLow model learns to straighten the specific ODE trajectories corresponding to $\Phi(\boldsymbol{z}_{t_k}, t_k, t_{k-1}, \boldsymbol{c}, w^*)$. One should use a smaller CFG scale (e.g., 1.0-2.5) to adjust guidance when sampling from PeRFLow trained with *CFG-fixed*.

Through empirical comparison, we observe that PeRFlow+*CFG-sync* preserves the sampling diversity of the original diffusion models with occasional failure in generating complex structures, while PeRFlow+*CFG-fixed* trades off sampling diversity in exchange for fewer failure cases.

Our recommendations are as follows: When using powerful pre-trained diffusion models (e.g., SDXL) and prioritizing generation quality, PeRFLow+*CFG-fixed* is the better choice. On the other hand, when the goal is to maintain the sampling diversity and adaptability of customized fine-tuned plug-ins, such as Dreamshaper, PeRFLow+*CFG-sync* is the more suitable option.

PeRFlow as Universal Plug-and-Play Accelerator PeRFlow initializes the weights of the student model θ with the pretrained diffusion model ϕ. After training with piecewise reflow, we find that the change of weights $\Delta W = \theta - \phi$ can be used to seamlessly accelerate many other workflows pretrained with the diffusion model. For exmaple, ΔW of PeRFlow+SD-v1.5 can accelerate the ControlNets [52], IP-Adaptor [49] and multiview generation [22] pipelines trained with the original SD v1.5. The accelerated pipelines achieve nearly lossless few-step generation as the original many-step generation. Please refer to Section 3.2 for detailed results.

Number of Time Windows The number of training segments depends on our expected minimum steps for the inference stage. Suppose the number of minimum steps for the inference stage is N, the number of training segments K should be less or equal to N. The reason is that we cannot approximate the velocity of a time window by the velocity of its previous time window. So, for each window, we should allocate at least a one-step computation budget. This paper evaluates 4-step, 6-step, and 8-step generation capabilities, so we set the number of training segments as four. In some special cases (e.g., Wonder3D in gigure 8 Appendix), the trajectory across the whole time window is almost linear after 4-piece PeRFlow acceleration. We can generate multi-view results with one step.

https://doi.org/10.52202/079017-2497

But in most cases, we should use an inference step larger or equal to the training segments. On the computational cost, PeRFlow only requires the $1/K$ amount of steps for synthesizing the training target in each iteration, compared to that of InstaFlow.

3 Experiments

We use PeRFlow to accelerate several large-scale text-to-image and text-to-video models, including SD-v1.5, SD-v2.1, SDXL, and AnimateDiff. In this section, we will illustrate the experiment configurations and empirical results.

Experiment Configuration All the PeRFlow models are initialized from their diffusion teachers. PeRFlow-SD-v1.5 is trained with images in resolution of 512×512 using ϵ-*prediction* defined in (7). PeRFlow-SD-v2.1 is trained with images in resolution of 768×768 using $\boldsymbol{v}$-*prediction*. PeRFlow-SDXL is trained with images in resolution of 1024×1024 using ϵ-*prediction* defined in (8). Images are all sampled from the LAION-Aesthetics-5+ dataset [37] and center-cropped. We also train PeRFlow-AnimateDiff with video clips in size of $16 \times 384 \times 384$ using ϵ-*prediction* defined in (8). We randomly drop out the text captions with a low probability (10%) to enable classifier-free guidance during sampling. We divide the time range $[0, 1]$ into four windows uniformly. For each window, we use the DDIM solver to solve the endpoints with 8 steps. We refer to the Hugging Face scripts for training Stable Diffusion [2] to set other hyper-parameters, including learning rate and weight decay. All experiments are conducted with 16 NVIDIA A100 GPUs.

3.1 Few-step generation

PeRFlow succeeds in accelerating pretrained Stable Diffusion models to few-step generators. As shown in figure 2 and 3, PeRFlow can generate astonishing pictures with only 4 steps. If increasing the number of inference steps (e.g., 5 or 6), we can obtain images with much richer details. We compare the generation results with recent acceleration methods, including InstaFlow, LCM-LORA, and SDXL-lightning. PeRFlow enjoys richer visual texture and better alignment between text prompts and images. Refer to figure 10, 12, and 13 in **Appendix** for more results.

We compute the FID values of PeRFlow-accelerated SDs in table 1 using images on three different reference distributions: (1) LAION-5B-Aesthetics [37], which is the training set of PeRFlow and other methods; (2) MS COCO 2014 [17] validation dataset; (3) images generated from SD- v1.5/XL with JourneyDB [41] prompts. We generate 30,000 images for the SD-v1.5 models and 10,000 for the SDXL series. We set the inference steps to 4 and 8 steps, respectively. In comparison to LCM-LoRA, we observe that PeRFlow models have obviously lower FID values. When increasing the number of inference steps, FID values of PeRFlow decrease because the numerical errors of solving ODE are better controlled. However, FID values of LCM-LoRA unexpectedly increase.

Domain shift caused by acceleration When accelerating diffusion models, we expect to preserve the performance and properties of the pretrained models. In table 1, we compute the FID values between the generation of the original SD models and the accelerated models. We observe the FID values of PeRFlow are smaller than LCM-LORA, InstaFlow, and SDXL-Lightning. This implies the distribution shift to the original SD models caused by PeRFlow is much smaller than other counterparts. The numerical comparison corresponds to the results in figure 5. The color style and layout of PeRFlow's results match the results of the pretrained models, while an obvious domain shift appears in the results of LCM-LoRA. Besides, the sampling diversity of PeRFlow is similar to the original SD-v1.5 and appears to be better than LCM-LoRA in figure 6.

3.2 PeRFlow as Universal Plug-and-Play Accelerator on SD Work Flows

PeRFlow-ΔW serves as a universal accelerator that can be simply plugged into various pipelines trained on the pretrained Stable Diffusion models, including (but not limited to) ControlNet [52], IP-Adaptor [49], and multiview generation. For example, plugging PeRFlow-ΔW into the SD-v1.5 ControlNet-Tile gives a 4-step image enhancement module (figure 7). Combining this module with the 4-step PeRFlow-SD-v1.5, we can generate high-quality 1024×1024 images with lightweight SD-v1.5 backbones. For multiview generation, plugging PeRFlow-ΔW into the Wonder3D [22]

[2] https://github.com/huggingface/diffusers/tree/main/examples/text_to_image

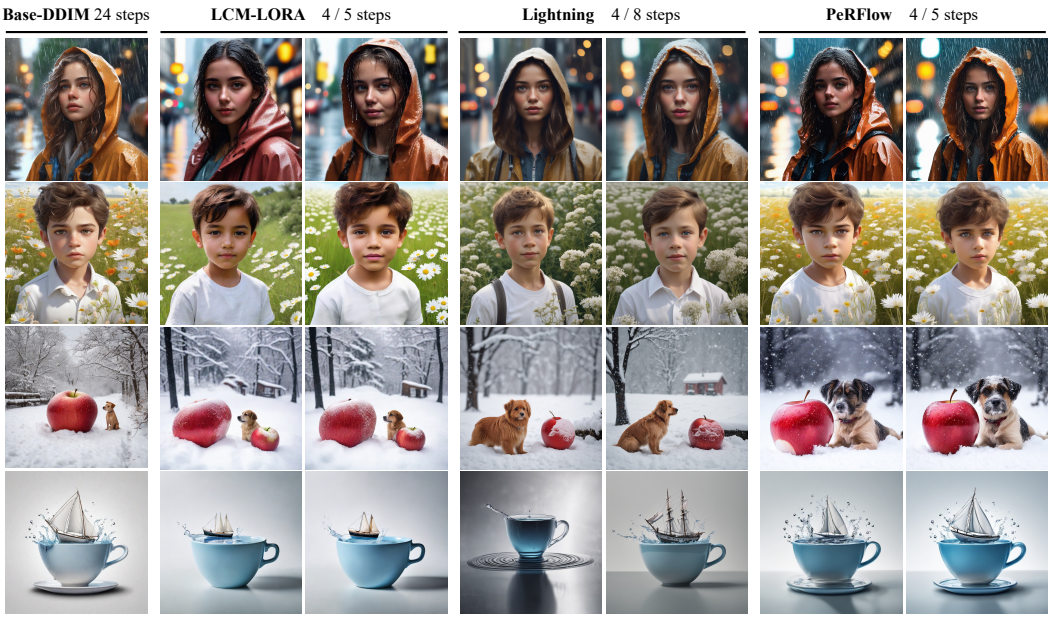

Figure 2: The 1024×1024 images generated by PeRFlow enjoy richer details and better text-image consistency in comparison to other acceleration methods on SDXL. Prompt #1: *"a closeup face photo of girl, wearing a raincoat, in the street, heavy rain, bokeh"*; Prompt #2: *"a closeup face photo of a boy in white shirt standing on the grassland, flowers"*; Prompt #3: *"a huge red apple in front of a small dog, heavy snow"*. Prompt #4: *"front view of a boat sailing in a cup of water"*.

Table 1: FID values of different acceleration methods (lower values indicate better quality).

	LAION-5B		COCO2014		SD-v1.5	
Method	**4-step**	**8-step**	**4-step**	**8-step**	**4-step**	**8-step**
InstaFlow	14.32	10.98	13.86	11.40	16.67	10.45
LCM-LoRA	15.28	19.21	23.49	29.63	15.63	21.19
PeRFlow	**8.60**	**8.52**	**11.31**	14.16	**8.28**	**5.03**

(a) SD-v1.5

	LAION-5B		COCO2014		SDXL	
Method	**4-step**	**8-step**	**4-step**	**8-step**	**4-step**	**8-step**
Lightning	15.47	14.37	22.86	20.44	11.41	10.49
LCM-LoRA	13.66	13.31	19.74	21.70	9.42	9.90
PeRFlow	**13.30**	**13.06**	**18.48**	**19.21**	**9.28**	**9.12**

(b) SDXL

pipeline leads to **one-step** generation of multi-view images (figure 8). More results are shown in figure 9.

3.3 Additional Discussion

Inference Budget Allocation PeRFlow divides the entire sampling trajectory into K time windows $\{[t_k, t_{k-1})\}_{k=K}^{1}$, with $1 = t_K > \cdots > t_k > t_{k-1} > \cdots > t_0 = 0$ indicating noisy to clean states. After training, K-step inference (one for each window) will yield high-quality images in most cases. However, for pictures with complex structures, such as motorcycles with well-crafted wheels and engines, PeRFlow may require more steps. Ho *et al.* [8] found that diffusion models generate images by synthesizing the layout and structure first and then refining the local details. We denote the

https://doi.org/10.52202/079017-2497

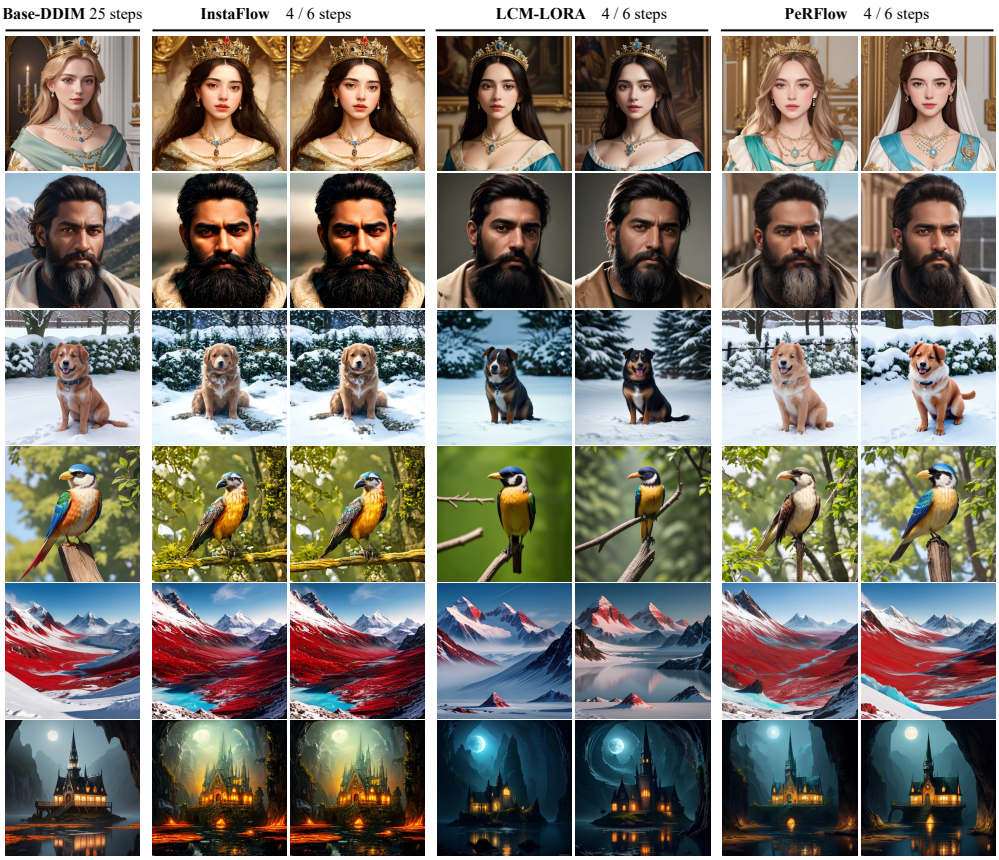

Figure 3: The 512×512 images generated by PeRFlow enjoy richer details and color styles in comparison to other acceleration methods on SD-v1.5 (w/ DreamShaper). Images in each row are generated with the same random seed.

Figure 4: 6-step generation ($16 \times 512 \times 512$) via PeRFlow-AnimateDiff (motion module-v3 with DreamShaper). The text prompts used are *"A young woman smiling, in the park, sunshine"* and *"A dog sitting in the garden, snow, trees"*.

number of inference steps by N. Inspired by this observation, we first allocate each window with $N//K$ steps. If $N \mod K > 0$, then the extra budget is given to time windows in noisy regions. Specifically, we give one extra step for windows, whose index i satisfies $K - i < N \mod K$. In practice, PeRFlow creates 4 time windows for acceleration training, and 5-step inference consistently generates high-quality images.

Dynamic Classifier-Free Guidance CFG is a useful technique to improve the layout, structure, and text alignment of the generated images. However, a large CFG scale sometimes leads to over-saturated color blocks [12, 43]. To mitigate this issue, we use a dynamic CFG strategy for few-step sampling, i.e., the corresponding CFG scales decrease for window K to 1. For example, when sampling with

(a) SD-v1.5 (b) PeRFlow (c) LCM-LoRA

Figure 5: PeRFlow has better compatibility with customized SD models compared to LCM-LoRA. The top is ArchitectureExterior and the bottom is DisneyPixarCartoon.

(a) SD-v1.5 (b) PeRFlow (c) LCM-LORA

Figure 6: Three random samples from two models with the same prompts. PeRFlow has better sampling diversity compared to LCM-LoRA.

5 steps, the CFG schedule is 7.5-4.0-4.0-4.0 for the *CFG-sync* mode and 2.5-1.5-1.5-1.5 for the *CFG-fixed* mode.

4 Related Works

Few-Step Diffusion Models Diffusion models have demonstrated impressive generative capabilities, but their iterative sampling process often suffers from slow inference speed [8, 39, 38]. To accelerate these models, various methods have been proposed. Progressive Distillation [35, 28] iteratively reduces the number of inference steps to 4-8, but the error can accumulate during the process. Alternative approaches [45, 44, 54, 47, 16, 36] leverage adversarial losses to align the distributions and reduce the number of inference steps, but these methods often struggle with training instability and mode collapse. To avoid adversarial training, recent works [50, 30, 56] employ additional models to estimate the score of the generated data for distilling one-step generators, but this adds extra cost to the training pipeline. Consistency Distillation [40, 26] is a novel pipeline for distilling few-step diffusion models by optimizing a consistency loss. However, the substantial difference between consistency models and the original diffusion models can hurt their adaptability to pre-trained modules. In our work, PeRFlow provides a simple, clean, and efficient framework for training few-step generative flows. By using different parameterizations as described in Section 2.2, PeRFlow achieves minimal gap with diffusion models, making it suitable for various pre-trained workflows.

Straight Probability Flows Learning straight probability flow is a promising principle for obtaining fast generative flows [20, 19, 21, 4]. Reflow is an effective way to learn such straight flows, but it requires constructing a large synthetic dataset [20, 21], which can introduce computational overhead and distribution shift. To avoid dataset construction,[13, 46] use an extra neural network to estimate the initial noise corresponding to an image, but training this network can be challenging. [33] employs mini-batch optimal transport to directly learn a straighter trajectory, but it is unclear how to apply this method to conditional generation scenarios, such as text-to-image generation. [29] finds the best step-size schedule for the pretrained generative model before reflow to improve efficiency, but it cannot avoid dataset generation and the resulting distribution shift. PeRFlow provides a new method to avoid using synthetic datasets. It uses real training data to mitigate distribution shift and a divide-and-conquer strategy to efficiently perform reflow, leading to advanced few-step text-to-image generators. Sequential reflow[51] is a concurrent work to ours. Compared to their work, we additionally provide different parameterization strategies to enhance the empirical performance in accelerating pre-trained text-to-image models.

https://doi.org/10.52202/079017-2497

5 Conclusions

In this work, we present Piecewise Rectified Flow (PeRFlow), a novel technique to learn few-step flow-based generative models. PeRFlow adopts a divide-and-conquer strategy, separating the generation trajectory into intervals and applying the reflow operation within each interval. This yields two key advantages: (1) using real training data to mitigate distribution shift from synthetic data, and (2) avoiding the need to generate and store a synthetic dataset prior to training. PeRFlow also designs proper parameterizations to inherit knowledge from pre-trained diffusion models for fast convergence. Consequently, PeRFlow accelerates powerful diffusion models like SD v1.5, SD v2.1, and SDXL, producing high-quality few-step image generators. Moreover, PeRFlow can be seamlessly combined with various SD workflows to create their accelerated versions.

Limitations Currently, PeRFlow divides the time range into 4 windows, balancing inference and training costs. It needs 4 steps or more for generation. To enable 1-2 step inference, we plan to explore multi-stage training and will focus on avoiding target synthesizing in the future.

Broader Impacts

This work proposes an acceleration technique for generative models. It can reduce the computational cost to less than 20% of the original and thus reduce the power cost. The proposed acceleration technique makes generative models more environmentally friendly.

Acknowledgment

The authors appreciate Yuanzhi Zhu for his valuable input and suggestions.

References

[1] M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

[2] Y. Balaji, S. Nah, X. Huang, A. Vahdat, J. Song, Q. Zhang, K. Kreis, M. Aittala, T. Aila, S. Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

[3] D. Berthelot, A. Autef, J. Lin, D. A. Yap, S. Zhai, S. Hu, D. Zheng, W. Talbott, and E. Gu. Tract: Denoising diffusion models with transitive closure time-distillation. *arXiv preprint arXiv:2303.04248*, 2023.

[4] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. Oberman. How to train your neural ode: the world of jacobian and kinetic regularization. In *International conference on machine learning*, pages 3154–3164. PMLR, 2020.

[5] J. Gu, S. Zhai, Y. Zhang, L. Liu, and J. M. Susskind. Boot: Data-free distillation of denoising diffusion models with bootstrapping. In *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*, 2023.

[6] Y. Guo, C. Yang, A. Rao, Y. Wang, Y. Qiao, D. Lin, and B. Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023.

[7] J. Ho and T. Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[8] J. Ho, A. Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models, Dec. 2020. arXiv:2006.11239 [cs, stat].

[9] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022.

[10] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[11] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[12] T. Kynkäänniemi, M. Aittala, T. Karras, S. Laine, T. Aila, and J. Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. *arXiv preprint arXiv:2404.07724*, 2024.

[13] S. Lee, B. Kim, and J. C. Ye. Minimizing trajectory curvature of ode-based generative models. In *International Conference on Machine Learning*, pages 18957–18973. PMLR, 2023.

[14] Y. Li, H. Wang, Q. Jin, J. Hu, P. Chemerys, Y. Fu, Y. Wang, S. Tulyakov, and J. Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] J. H. Liew, H. Yan, J. Zhang, Z. Xu, and J. Feng. Magicedit: High-fidelity and temporally coherent video editing. *arXiv preprint arXiv:2308.14749*, 2023.

[16] S. Lin, A. Wang, and X. Yang. Sdxl-lightning: Progressive adversarial diffusion distillation, 2024.

[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[18] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2022.

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

[20] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[21] X. Liu, X. Zhang, J. Ma, J. 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.

[22] X. Long, Y.-C. Guo, C. Lin, Y. Liu, Z. Dou, L. Liu, Y. Ma, S.-H. Zhang, M. Habermann, C. Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. *arXiv preprint arXiv:2310.15008*, 2023.

[23] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. 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.

[24] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, Aug. 2022. arXiv:2206.00927 [cs, stat].

[25] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models, 2023.

[26] S. Luo, Y. Tan, L. Huang, J. Li, and H. Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023.

[27] S. Luo, Y. Tan, S. Patil, D. Gu, P. von Platen, A. Passos, L. Huang, J. Li, and H. Zhao. Lcm-lora: A universal stable-diffusion acceleration module, 2023.

[28] C. Meng, R. Rombach, R. Gao, D. Kingma, S. Ermon, J. Ho, and T. Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023.

 https://doi.org/10.52202/079017-2497

[29] B. Nguyen, B. Nguyen, and V. A. Nguyen. Bellman optimal step-size straightening of flow-matching models. In *The Twelfth International Conference on Learning Representations*, 2023.

[30] T. H. Nguyen and A. Tran. Swiftbrush: One-step text-to-image diffusion model with variational score distillation, 2024.

[31] J. Pan, H. Yan, J. H. Liew, V. Y. Tan, and J. Feng. Adjointdpm: Adjoint sensitivity method for gradient backpropagation of diffusion probabilistic models. *arXiv preprint arXiv:2307.10711*, 2023.

[32] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[33] A.-A. Pooladian, H. Ben-Hamu, C. Domingo-Enrich, B. Amos, Y. Lipman, and R. T. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *International Conference on Machine Learning*, pages 28100–28127. PMLR, 2023.

[34] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, Apr. 2022. arXiv:2112.10752 [cs].

[35] T. Salimans and J. Ho. PROGRESSIVE DISTILLATION FOR FAST SAMPLING OF DIFFUSION MODELS. page 21, 2022.

[36] A. Sauer, D. Lorenz, A. Blattmann, and R. Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

[37] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35: 25278–25294, 2022.

[38] J. Song, C. Meng, and S. Ermon. Denoising Diffusion Implicit Models, June 2022. arXiv:2010.02502 [cs].

[39] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS. page 36, 2021.

[40] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[41] K. Sun, J. Pan, Y. Ge, H. Li, H. Duan, X. Wu, R. Zhang, A. Zhou, Z. Qin, Y. Wang, et al. Journeydb: A benchmark for generative image understanding. *Advances in Neural Information Processing Systems*, 36, 2024.

[42] W. Wang, J. Liu, Z. Lin, J. Yan, S. Chen, C. Low, T. Hoang, J. Wu, J. H. Liew, H. Yan, et al. Magicvideo-v2: Multi-stage high-aesthetic video generation. *arXiv preprint arXiv:2401.04468*, 2024.

[43] X. Wang, N. Dufour, N. Andreou, M.-P. Cani, V. F. Abrevaya, D. Picard, and V. Kalogeiton. Analysis of classifier-free guidance weight schedulers. *arXiv preprint arXiv:2404.13040*, 2024.

[44] Z. Wang, H. Zheng, P. He, W. Chen, and M. Zhou. Diffusion-gan: Training gans with diffusion. In *The Eleventh International Conference on Learning Representations*, 2022.

[45] Z. Xiao, K. Kreis, and A. Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. In *International Conference on Learning Representations*, 2021.

[46] S. Xing, J. Cao, H. Huang, X.-Y. Zhang, and R. He. Exploring straighter trajectories of flow matching with diffusion guidance. *arXiv preprint arXiv:2311.16507*, 2023.

[47] Y. Xu, Y. Zhao, Z. Xiao, and T. Hou. Ufogen: You forward once large scale text-to-image generation via diffusion gans. *arXiv preprint arXiv:2311.09257*, 2023.

[48] Z. Xu, J. Zhang, J. H. Liew, H. Yan, J.-W. Liu, C. Zhang, J. Feng, and M. Z. Shou. Magicanimate: Temporally consistent human image animation using diffusion model. *arXiv preprint arXiv:2311.16498*, 2023.

[49] H. Ye, J. Zhang, S. Liu, X. Han, and W. Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

[50] T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park. One-step diffusion with distribution matching distillation, 2023.

[51] J. Yoon and J. Lee. Sequential flow matching for generative modeling. *arXiv preprint arXiv:2402.06461*, 2024.

[52] L. Zhang and M. Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models, Feb. 2023. arXiv:2302.05543 [cs].

[53] Q. Zhang and Y. Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022.

[54] H. Zheng, P. He, W. Chen, and M. Zhou. Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders. In *The Eleventh International Conference on Learning Representations*, 2022.

[55] D. Zhou, W. Wang, H. Yan, W. Lv, Y. Zhu, and J. Feng. MagicVideo: Efficient Video Generation With Latent Diffusion Models, May 2023. arXiv:2211.11018 [cs].

[56] M. Zhou, H. Zheng, Z. Wang, M. Yin, and H. Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. *arXiv preprint arXiv:2404.04057*, 2024.

 https://doi.org/10.52202/079017-2497

Appendix

A More generation results

Table 2: Clip scores of different acceleration methods (high values indicate better quality). We select 5000 text prompts from COCO2014 and generate one image for each prompt for computing clip cosine similarity scores. PeRFlow's results align better with text prompts in comparison to other methods.

Method	SDXL 25 steps	Lightning 4 steps	LCM-LORA 4 steps	PeRFlow 4 steps
Score	**0.337**	0.330	0.327	**0.337**

Figure 7: 4-step image enhancement (128 $\rightarrow$ 1024) with PeRFlow-SD v1.5+ControlNet-tile [52]

Figure 8: One-step multiview generation of PeRFlow-SD v1.5+Wonder3D [22]

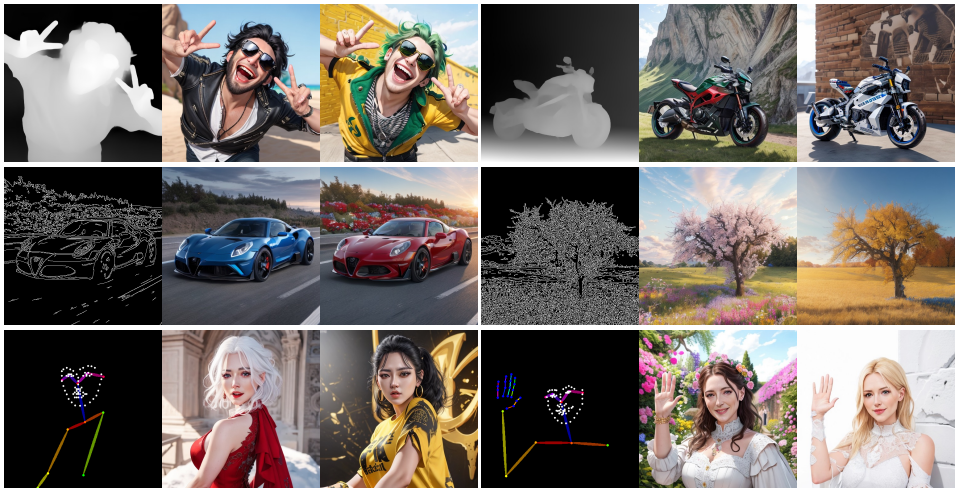

Figure 9: Fast generation via PeRFlow accelerated depth-/edge-/pose-ControlNet [52]

https://doi.org/10.52202/079017-2497

Figure 10: 4-step generation (512×512) via PeRFlow-SD-v1.5.

Figure 11: 8-step generation (512×512) via PeRFlow-SD-v1.5.

Figure 12: 4-step generation (768×768) via PeRFlow-SD-v2.1.

https://doi.org/10.52202/079017-2497

Figure 13: 4-step generation (1024×1024) via PeRFlow-SDXL.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: all claims are supported via theoretical or empirical evidence.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Discussed in the conclusion section.

https://doi.org/10.52202/079017-2497

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

   Justification: Math derivations are all provided in the main article.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: Important details about experiments are provided.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: We use public data for training. Codes are also submitted together with the Supplementary Material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

https://doi.org/10.52202/079017-2497

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: We follow the standard settings of training diffusion models, and provide the related public links.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

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

   Answer: [NA]

   Justification: We follow the standard settings of computing FID values.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We provide the GPU types.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: No violations to the Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: Yes, discussed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: All models used here are publicly released.

    Guidelines:

    - The answer NA means that the paper poses no such risks.

https://doi.org/10.52202/079017-2497

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [NA]

    Justification: No new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [NA]

    Justification: No crowdsourcing experiments.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [NA]

    Justification: Not related.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-2497