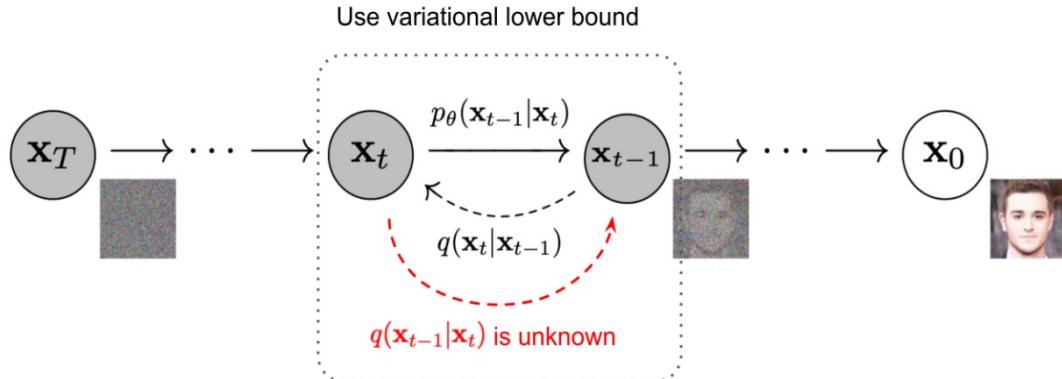


The Principle and Application of Diffusion Acceleration

Fu-Yun Wang

fywang0126@gmail.com

Diffusion Models: Markovian Perspective



■ Assumption:

- $p(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

■ Forward Process:

- $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1 - \alpha_t)\mathbf{I})$
- $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

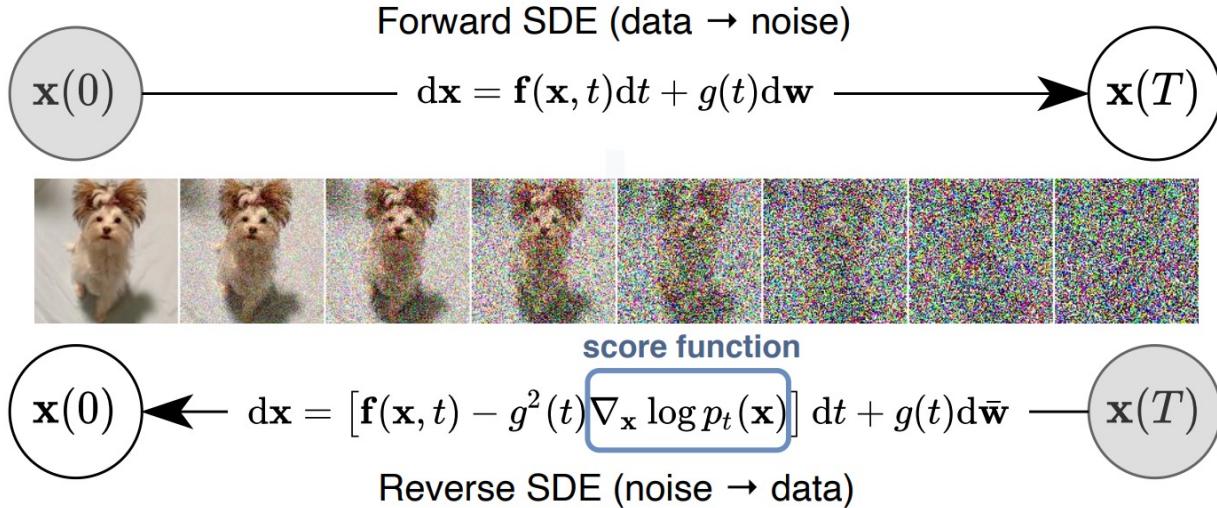
■ Reverse Process:

- $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}$

■ Maximum Likelihood Estimation (MLE) is Equivalent to

$$\arg \min_{\theta} \mathbb{E}_{t \sim U\{2, T\}} [\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} [D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))]]$$

Diffusion Models: Stochastic Differential Equation Perspective



The only unknown term is the **score function**.

Train a neural network through score matching!

Probability Flow ODE:

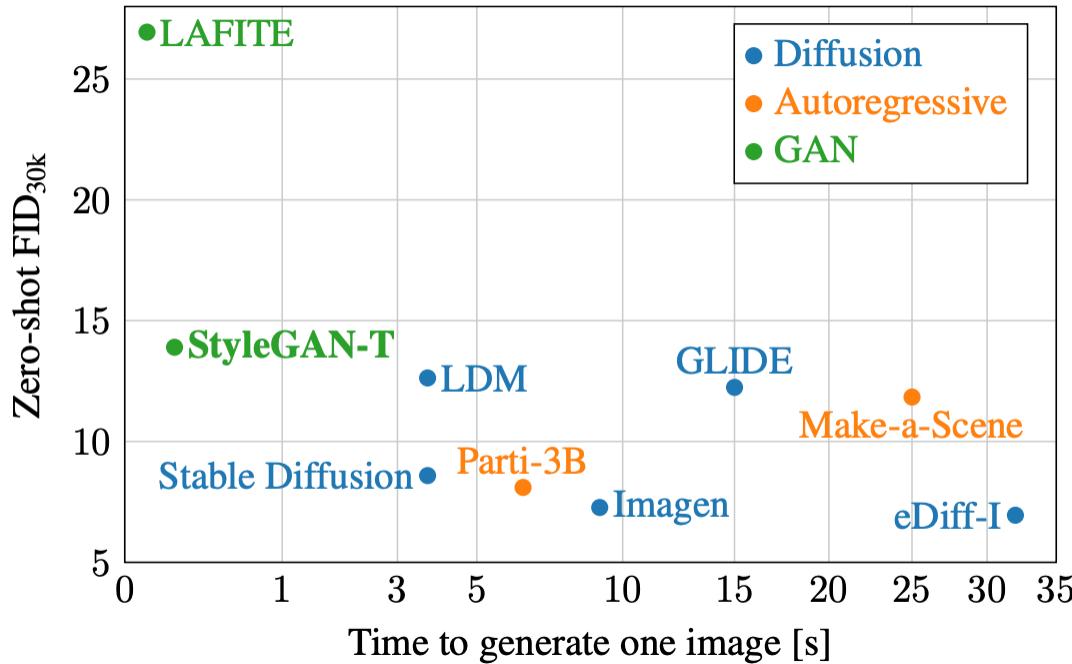
A deterministic reverse process

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g(t)^2\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] dt$$

Exact Solution form of PF-ODE

$$\mathbf{x}_t = \frac{\alpha_t}{\alpha_s} \mathbf{x}_s - \alpha_t \int_{\lambda_s}^{\lambda_t} e^{-\lambda} \hat{\epsilon}_{\theta}(\hat{\mathbf{x}}_{\lambda}, \lambda) d\lambda.$$

Diffusion Models: Slow Inference Speed



How to speed up the diffusion generation?

- Reducing the number of function evaluation (NFE).
 - Better Solvers.
- Adversarial post-training.
- Parallel Sampling.
- Distillation.
 - Naïve distillation.
 - Guided distillation.
 - Score distillation.
 - Consistency distillation.
 - Rectification.

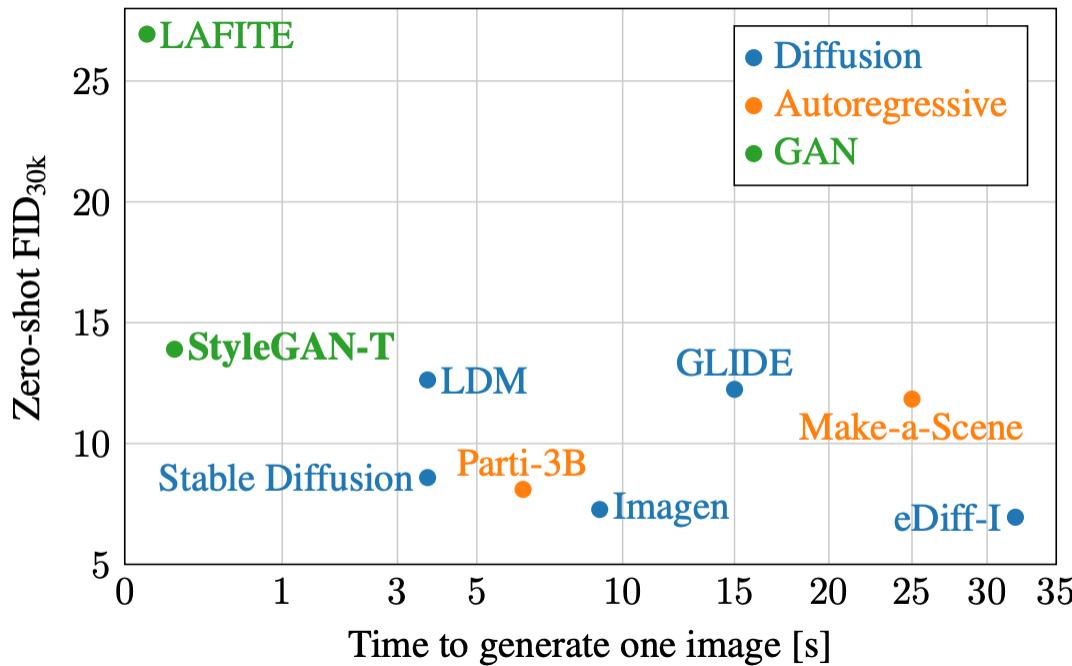
DPM-Solver

$$\mathbf{x}_{t_{i-1} \rightarrow t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} \tilde{\mathbf{x}}_{t_{i-1}} - \alpha_{t_i} \int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda} \hat{\epsilon}_\theta(\hat{\mathbf{x}}_\lambda, \lambda) d\lambda.$$

$$\hat{\epsilon}_\theta(\hat{\mathbf{x}}_\lambda, \lambda) = \sum_{n=0}^{k-1} \frac{(\lambda - \lambda_{t_{i-1}})^n}{n!} \hat{\epsilon}_\theta^{(n)}(\hat{\mathbf{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}}) + \mathcal{O}((\lambda - \lambda_{t_{i-1}})^k),$$

$$\mathbf{x}_{t_{i-1} \rightarrow t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} \tilde{\mathbf{x}}_{t_{i-1}} - \alpha_{t_i} \sum_{n=0}^{k-1} \hat{\epsilon}_\theta^{(n)}(\hat{\mathbf{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}}) \int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda} \frac{(\lambda - \lambda_{t_{i-1}})^n}{n!} d\lambda + \boxed{\mathcal{O}(h_i^{k+1})}$$

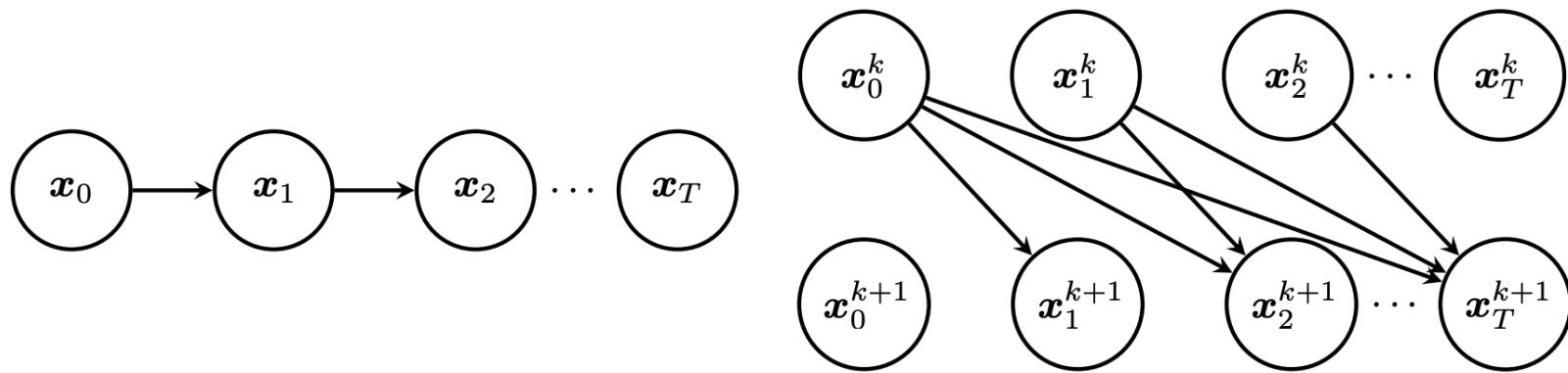
Diffusion Models: Slow Inference Speed



How to speed up the diffusion generation?

- Reducing the number of function evaluation (NFE).
 - Better Solvers.
 - Adversarial post-training.
 - **Parallel Sampling.**
- Distillation.
 - Naïve distillation.
 - Guided distillation.
 - Score distillation.
 - Consistency distillation.
 - Rectification.

Picard Iteration



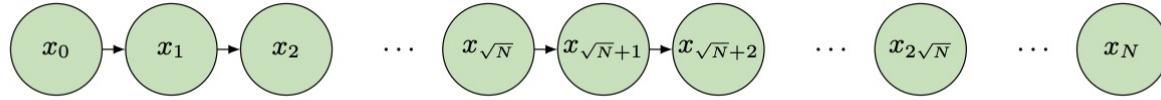
$$\mathbf{x}_t = \mathbf{x}_0 + \int_0^t s(\mathbf{x}_u, u) du.$$

$$\mathbf{x}_t^{k+1} = \mathbf{x}_0^k + \int_0^t s(\mathbf{x}_u^k, u) du.$$

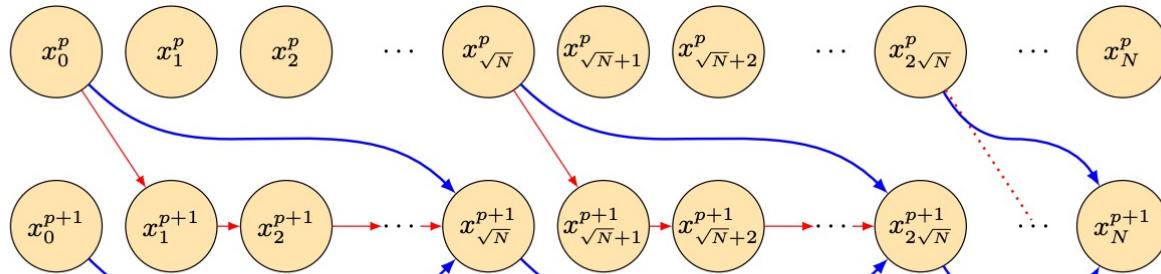
Lower Bound of Picard Iteration = Sequential Denoising

$$\begin{aligned}\mathbf{x}_{t+1}^{k+1} &= \mathbf{x}_0^k + \frac{1}{T} \sum_{i=0}^t s(\mathbf{x}_i^k, \frac{i}{T}) \\&= \left(\mathbf{x}_0^k + \frac{1}{T} \sum_{i=0}^{t-1} s(\mathbf{x}_i^k, \frac{i}{T}) \right) + \frac{1}{T} s(\mathbf{x}_t^k, \frac{t}{T}) \\&= \mathbf{x}_t^{k+1} + \frac{1}{T} s(\mathbf{x}_t^k, \frac{t}{T}) \\&= \mathbf{x}_t^{k+1} + \frac{1}{T} s(h_{t-1}(\dots h_2(h_1(\mathbf{x}_0))), \frac{t}{T}) \\&= \mathbf{x}_t^\star + \frac{1}{T} s(\mathbf{x}_t^\star, \frac{t}{T}) = \mathbf{x}_{t+1}^\star.\end{aligned}$$

Parareal Algorithm



(a) Sequential Sampling



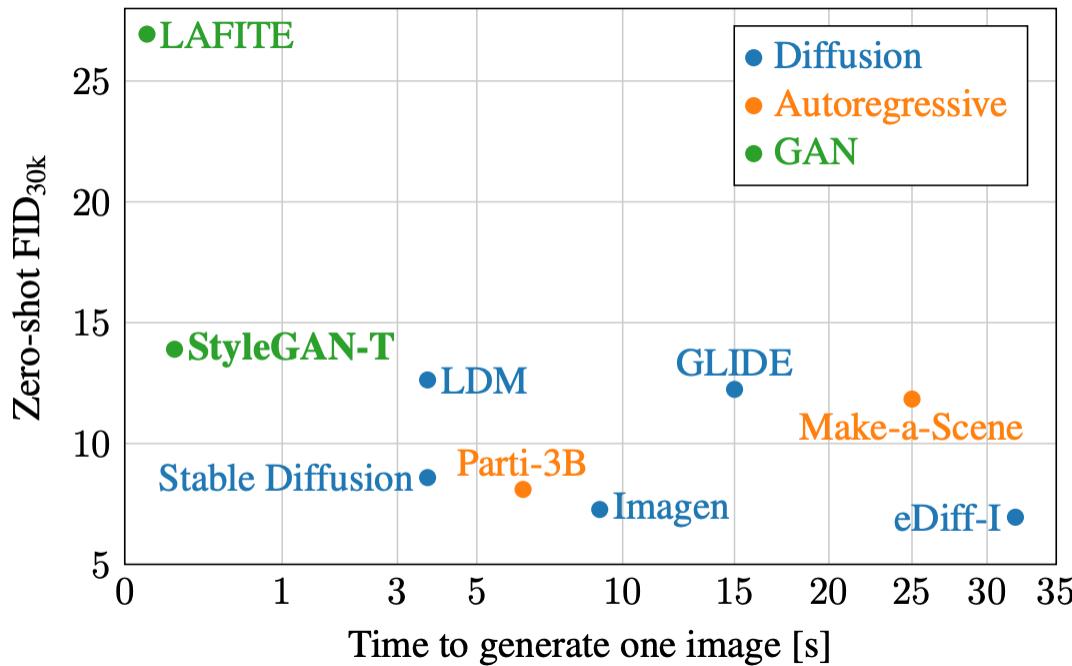
(b) SRDS

$$x_{i+1}^{p+1} = \boxed{\mathcal{F}(x_i^p, t_i, t_{i+1})} + \left(\boxed{\mathcal{G}(x_i^{p+1}, t_i, t_{i+1}) - \mathcal{G}(x_i^p, t_i, t_{i+1})} \right)$$

Fine Solver (Parallel)

Coarse Solver (Sequential)

Diffusion Models: Slow Inference Speed



How to speed up the diffusion generation?

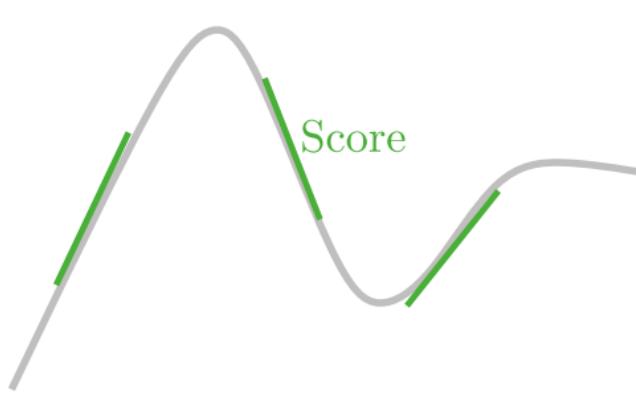
- Reducing the number of function evaluation (NFE).
 - Better Solvers.
 - Adversarial post-training.
 - Parallel Sampling.

- Distillation.
 - Naïve distillation.
 - Guided distillation.
 - Score distillation.
 - Consistency distillation.
 - Rectification.

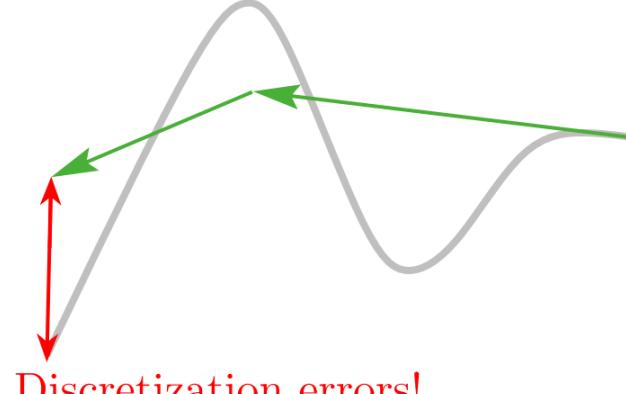
Understanding Diffusion Models from the PF-ODE path

We know the derivative w.r.t. time t .

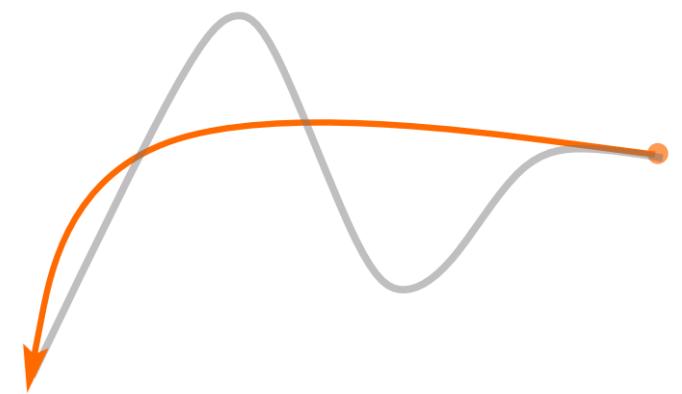
$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] dt$$



PF-ODE

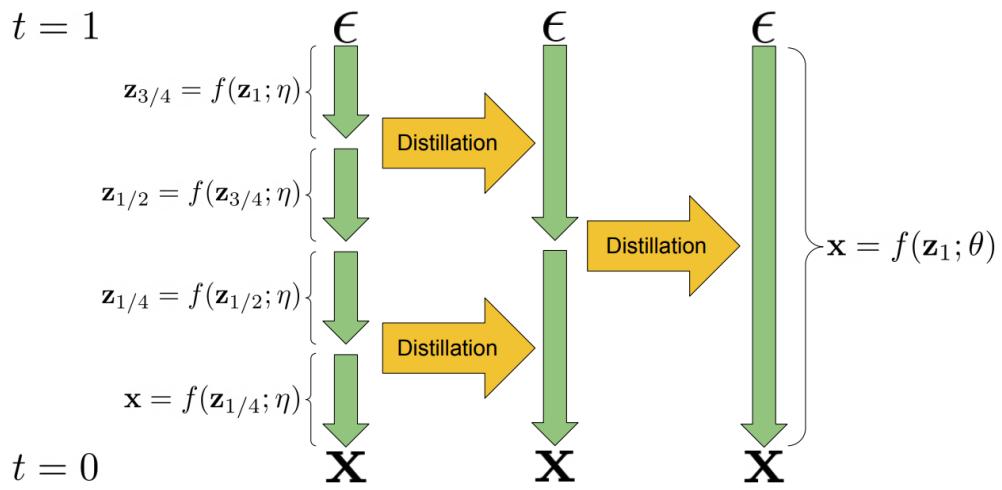


Discretized numerical solving.



Naïve distillation.

Distillation Techniques: Progressive Distillation



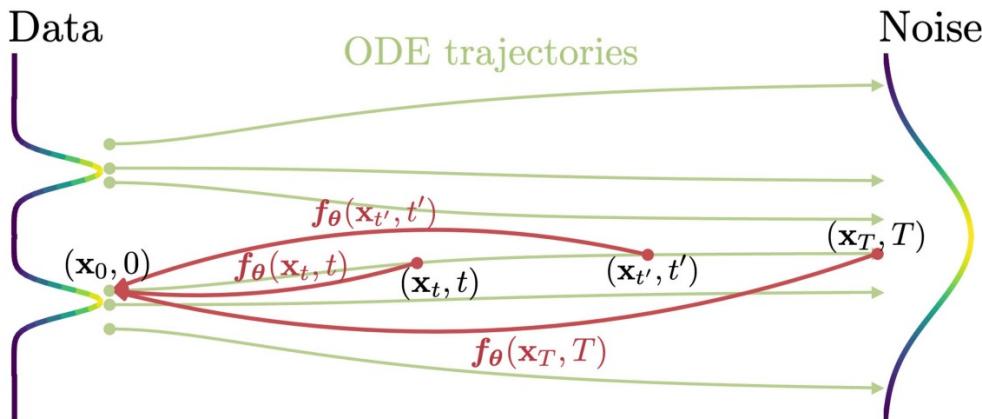
Algorithm 2 Progressive distillation

```

Require: Trained teacher model  $\hat{\mathbf{x}}_\eta(\mathbf{z}_t)$ 
Require: Data set  $\mathcal{D}$ 
Require: Loss weight function  $w()$ 
Require: Student sampling steps  $N$ 
for  $K$  iterations do
     $\theta \leftarrow \eta$                                  $\triangleright$  Init student from teacher
    while not converged do
         $\mathbf{x} \sim \mathcal{D}$ 
         $t = i/N, i \sim \text{Cat}[1, 2, \dots, N]$ 
         $\epsilon \sim N(0, I)$ 
         $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$ 
        # 2 steps of DDIM with teacher
         $t' = t - 0.5/N, t'' = t - 1/N$ 
         $\mathbf{z}_{t'} = \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_t) + \frac{\sigma_{t'}}{\sigma_t} (\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\eta(\mathbf{z}_t))$ 
         $\mathbf{z}_{t''} = \alpha_{t''} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}} (\mathbf{z}_{t'} - \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}))$ 
         $\tilde{\mathbf{x}} = \frac{\mathbf{z}_{t''} - (\sigma_{t''}/\sigma_t) \mathbf{z}_t}{\alpha_{t''} - (\sigma_{t''}/\sigma_t) \alpha_t}$   $\triangleright$  Teacher  $\hat{\mathbf{x}}$  target
         $\lambda_t = \log[\alpha_t^2/\sigma_t^2]$ 
         $L_\theta = w(\lambda_t) \|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)\|_2^2$ 
         $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$ 
    end while
     $\eta \leftarrow \theta$                                  $\triangleright$  Student becomes next teacher
     $N \leftarrow N/2$   $\triangleright$  Halve number of sampling steps
end for

```

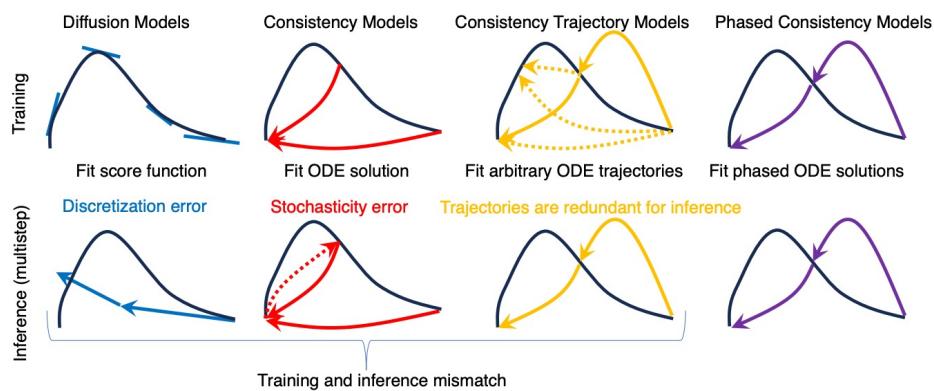
Distillation Techniques: Consistency Distillation



Algorithm 2 Consistency Distillation (CD)

Input: dataset \mathcal{D} , initial model parameter θ , learning rate η , ODE solver $\Phi(\cdot, \cdot; \phi)$, $d(\cdot, \cdot)$, $\lambda(\cdot)$, and μ
 $\theta^- \leftarrow \theta$
repeat
 Sample $\mathbf{x} \sim \mathcal{D}$ and $n \sim \mathcal{U}[1, N - 1]$
 Sample $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t_{n+1}^2 \mathbf{I})$
 $\hat{\mathbf{x}}_{t_n}^\phi \leftarrow \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \phi)$
 $\mathcal{L}(\theta, \theta^-; \phi) \leftarrow$
 $\lambda(t_n)d(f_\theta(\mathbf{x}_{t_{n+1}}, t_{n+1}), f_{\theta^-}(\hat{\mathbf{x}}_{t_n}^\phi, t_n))$
 $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta, \theta^-; \phi)$
 $\theta^- \leftarrow \text{stopgrad}(\mu \theta^- + (1 - \mu) \theta)$
until convergence

Distillation Techniques: Phased Consistency Distillation



Algorithm 1 Phased Consistency Distillation with CFG-augmented ODE solver (PCD)

Input: dataset \mathcal{D} , initial model parameter θ , learning rate η , ODE solver $\Psi(\cdot, \cdot, \cdot, \cdot)$, distance metric $d(\cdot, \cdot)$, EMA rate μ , noise schedule α_t, σ_t , guidance scale $[w_{\min}, w_{\max}]$, number of ODE step k , discretized timesteps $t_0 = \epsilon < t_1 < t_2 < \dots < t_N = T$, edge timesteps $s_0 = t_0 < s_1 < s_2 < \dots < s_M = t_N \in \{t_i\}_{i=0}^N$ to split the ODE trajectory into M sub-trajectories.

Training data : $\mathcal{D}_x = \{(\mathbf{x}, \mathbf{c})\}$

$\theta^- \leftarrow \theta$

repeat

 Sample $(\mathbf{z}, \mathbf{c}) \sim \mathcal{D}_z$, $n \sim \mathcal{U}[0, N - k]$ and $\omega \sim [\omega_{\min}, \omega_{\max}]$

 Sample $\mathbf{x}_{t_{n+k}} \sim \mathcal{N}(\alpha_{t_{n+k}} \mathbf{z}; \sigma_{t_{n+k}}^2 \mathbf{I})$

 Determine $[s_m, s_{m+1}]$ given n

$\mathbf{x}_{t_n}^\phi \leftarrow (1 + \omega)\Psi(\mathbf{x}_{t_{n+k}}, t_{n+k}, t_n, \mathbf{c}) - \omega\Psi(\mathbf{x}_{t_{n+k}}, t_{n+k}, t_n, \emptyset)$

$\tilde{\mathbf{x}}_{s_m} = \mathbf{f}_\theta^m(\mathbf{x}_{t_{n+k}}, t_{n+k}, \mathbf{c})$ and $\hat{\mathbf{x}}_{s_m} = \mathbf{f}_{\theta^-}(\mathbf{x}_{t_n}^\phi, t_n, \mathbf{c})$

 Obtain $\tilde{\mathbf{x}}_s$ and $\hat{\mathbf{x}}_s$ through adding noise to $\tilde{\mathbf{x}}_{s_m}$ and $\hat{\mathbf{x}}_{s_m}$

$\mathcal{L}(\theta, \theta^-) = d(\tilde{\mathbf{x}}_{s_m}, \hat{\mathbf{x}}_{s_m}) + \lambda(\text{ReLU}(1 + \tilde{\mathbf{x}}_s) + \text{ReLU}(1 - \hat{\mathbf{x}}_s))$

$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta, \theta^-)$

$\theta^- \leftarrow \text{stopgrad}(\mu \theta^- + (1 - \mu) \theta)$

until convergence

Application: AnimateLCM

- [AnimateLCM](#) support
 - NOTE: You will need to use `autoselect` or `lcm` or `lcm[100_ots]` `beta_schedule`. To use fully with LCM, be sure to use appropriate LCM lora, use the `lcm` `sampler_name` in KSampler nodes, and lower `cfg` to somewhere around 1.0 to 2.0. Don't forget to decrease steps (minimum = ~4 steps), since LCM converges faster (less steps). Increase step count to increase detail as desired.
- [AnimateLCM-I2V](#) support, big thanks to [Fu-Yun Wang](#) for providing me the original diffusers code he created during his work on the paper
 - NOTE: Requires same settings as described for AnimateLCM above. Requires `Apply AnimateLCM-I2V Model` Gen2 node usage so that `ref_latent` can be provided; use `Scale Ref Image` and `VAE Encode` node to preprocess input images. While this was intended as an img2video model, I found it works best for vid2vid purposes with `ref_drift=0.0`, and to use it for only at least 1 step before switching over to other models via chaining with toher `Apply AnimateDiff Model (Adv.)` nodes. The `apply_ref_when_disabled` can be set to True to allow the `img_encoder` to do its thing even when the `end_percent` is reached. AnimateLCM-I2V is also extremely useful for maintaining coherence at higher resolutions (with ControlNet and SD LoRAs active, I could easily upscale from 512x512 source to 1024x1024 in a single pass). TODO: add examples

Downloads last month

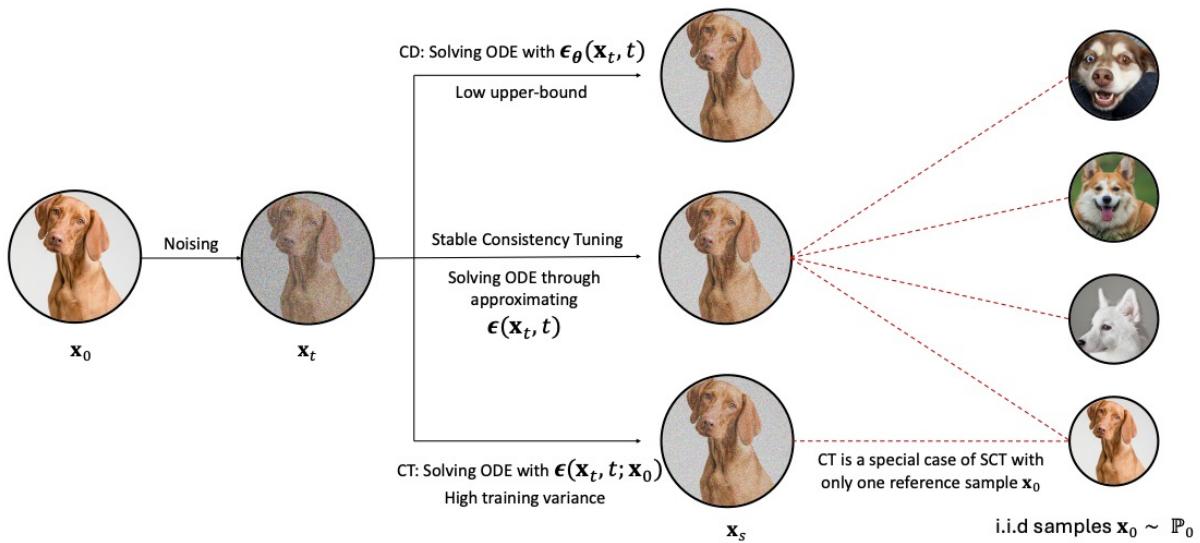
73,760



Application: AnimateLCM



Consistency Training



$$\mathbf{h}_\theta(\mathbf{x}_t, t) \xleftarrow{\text{fit}} \mathbf{r} + \mathbf{h}_{\theta^-}(\mathbf{x}_r, r)$$

Bootstrapping

$$\mathbf{r} \approx \epsilon_\phi(\mathbf{x}_t, t) \int_{\lambda_t}^{\lambda_r} e^{-\lambda} d\lambda + \mathcal{O}((\lambda_r - \lambda_t)^2)$$

Consistency Distillation

$$\mathbf{r} \approx \epsilon(\mathbf{x}_t, t; \mathbf{x}_0) \int_{\lambda_t}^{\lambda_r} e^{-\lambda} d\lambda + \mathcal{O}((\lambda_r - \lambda_t)^2)$$

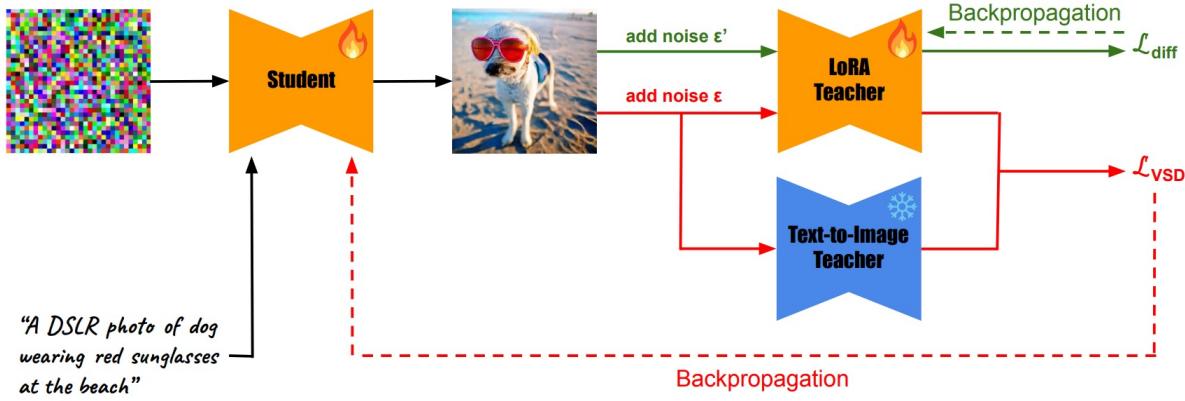
Consistency Training

Ground Truth of Score Estimation: Stable Consistency Tuning

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \mathbf{c}) &= \mathbb{E}_{\mathbb{P}(\mathbf{x}_0 \mid \mathbf{x}_t, \mathbf{c})} [\nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \mathbf{x}_0, \mathbf{c})] \\
&= \mathbb{E}_{\mathbb{P}(\mathbf{x}_0 \mid \mathbf{c})} \left[\frac{\mathbb{P}(\mathbf{x}_0 \mid \mathbf{x}_t, \mathbf{c})}{\mathbb{P}(\mathbf{x}_0 \mid \mathbf{c})} \nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \mathbf{x}_0, \mathbf{c}) \right] \\
&= \mathbb{E}_{\mathbb{P}(\mathbf{x}_0 \mid \mathbf{c})} \left[\frac{\mathbb{P}(\mathbf{x}_t \mid \mathbf{x}_0, \mathbf{c})}{\mathbb{P}(\mathbf{x}_t \mid \mathbf{c})} \nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \mathbf{x}_0, \mathbf{c}) \right] \\
&= \mathbb{E}_{\mathbb{P}(\mathbf{x}_0 \mid \mathbf{c})} \left[\frac{\mathbb{P}(\mathbf{x}_t \mid \mathbf{x}_0)}{\mathbb{P}(\mathbf{x}_t \mid \mathbf{c})} \nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \mathbf{x}_0) \right] \\
&\approx \frac{1}{n} \sum_{\{\mathbf{x}_0^{(i)}\} \sim \mathbb{P}(\mathbf{x}_0 \mid \mathbf{c})}^{i=0, \dots, n-1} \frac{\mathbb{P}(\mathbf{x}_t \mid \mathbf{x}_0^{(i)})}{\mathbb{P}(\mathbf{x}_t \mid \mathbf{c})} \nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \mathbf{x}_0^{(i)}) \\
&\approx \frac{1}{n} \sum_{\{\mathbf{x}_0^{(i)}\} \sim \mathbb{P}(\mathbf{x}_0 \mid \mathbf{c})}^{i=0, \dots, n-1} \frac{\mathbb{P}(\mathbf{x}_t \mid \mathbf{x}_0^{(i)})}{\sum_{\mathbf{x}_0^{(j)} \in \{\mathbf{x}_0^{(i)}\}} \mathbb{P}(\mathbf{x}_t \mid \mathbf{x}_0^{(j)}, \mathbf{c})} \nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \mathbf{x}_0^{(i)}) \\
&= \frac{1}{n} \sum_{\{\mathbf{x}_0^{(i)}\} \sim \mathbb{P}(\mathbf{x}_0 \mid \mathbf{c})}^{i=0, \dots, n-1} \frac{\mathbb{P}(\mathbf{x}_t \mid \mathbf{x}_0^{(i)})}{\sum_{\mathbf{x}_0^{(j)} \in \{\mathbf{x}_0^{(i)}\}} \mathbb{P}(\mathbf{x}_t \mid \mathbf{x}_0^{(j)})} \nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \mathbf{x}_0^{(i)})
\end{aligned}$$

Distillation Techniques: Score Distillation

$$\nabla_{\theta} \mathcal{L}_{\text{VSD}}(\theta) \triangleq \mathbb{E}_{t, \epsilon, c} \left[\omega(t) (\epsilon_{\text{pretrain}}(\mathbf{x}_t, t, y^c) - \epsilon_{\phi}(\mathbf{x}_t, t, c, y)) \frac{\partial \mathbf{g}(\theta, c)}{\partial \theta} \right]$$



Algorithm 1 SwiftBrush Distillation

- 1: **Require:** a pretrained text-to-image teacher ϵ_{ψ} , a LoRA teacher ϵ_{ϕ} , a student model f_{θ} , two learning rates η_1 and η_2 , a weighting function ω , a prompts dataset Y , the maximum number of time steps T and the noise schedule $\{(\alpha_t, \sigma_t)\}_{t=1}^T$ of the teacher model
 - 2: **Initialize:** $\phi \leftarrow \psi, \theta \leftarrow \psi$
 - 3: **while** not converged **do**
 - 4: Sample input noise $z \sim \mathcal{N}(0, I)$
 - 5: Sample text caption input $y \sim Y$
 - 6: Compute student output $\hat{x}_0 = f_{\theta}(z, y)$
 - 7: Sample timestep $t \sim \mathcal{U}(0.02T, 0.98T)$
 - 8: Sample added noise $\epsilon \sim \mathcal{N}(0, I)$
 - 9: Compute noisy sample $\hat{x}_t = \alpha_t \hat{x}_0 + \sigma_t \epsilon$
 - 10: $\theta \leftarrow \theta - \eta_1 [\omega(t) (\epsilon_{\psi}(\hat{x}_t, t, y) - \epsilon_{\phi}(\hat{x}_t, t, y)) \frac{\partial \hat{x}_0}{\partial \theta}]$
 - 11: Sample timestep $t' \sim \mathcal{U}(0, T)$
 - 12: Sample added noise $\epsilon' \sim \mathcal{N}(0, I)$
 - 13: Compute noisy sample $\hat{x}_{t'} = \alpha_{t'} \hat{x}_0 + \sigma_{t'} \epsilon'$
 - 14: $\phi \leftarrow \phi - \eta_2 \nabla_{\phi} \|\epsilon_{\phi}(\hat{x}_{t'}, t', y) - \epsilon'\|^2$
 - 15: **end while**
 - 16: **return** trained student model f_{θ}
-

Distillation Techniques: Score Distillation

$$\mathcal{L}(\theta) = \mathcal{D}^{[0,T]}(p_\theta, q) = \int_{t=0}^T w(t) \mathbb{E}_{\mathbf{x}_t \sim \pi_t} [\mathbf{d}(\mathbf{s}_{p_{\theta,t}}(\mathbf{x}_t) - \mathbf{s}_{q_t}(\mathbf{x}_t))] dt,$$

Generator $g_\theta : p_z \rightarrow p_\theta$, $p_{\theta,t} = p_\theta * \mathcal{N}(0, I)$, $\mathbf{s}_{\theta,t}(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log p_{\theta,t}(\mathbf{x}_t)$

impossible to compute $\frac{d}{\theta} \mathbf{s}_{\theta,t}(\mathbf{x}_t)$

Score Divergence Gradient Theorem

$$\mathcal{L}(\theta) = \mathcal{D}^{[0,T]}(p_\theta, q) = \int_{t=0}^T w(t) \mathbb{E}_{\mathbf{x}_t \sim \pi_t} [\mathbf{d}(\mathbf{s}_{p_{\theta,t}}(\mathbf{x}_t) - \mathbf{s}_{q_t}(\mathbf{x}_t))] dt,$$

太虚伪 太善变 不完美

$$\mathbb{E}_{\mathbf{x}_t \sim p_{\text{sg}[\theta],t}} \left[\mathbf{d}'(\mathbf{s}_{p_{\theta,t}}(\mathbf{x}_t) - \mathbf{s}_{q_t}(\mathbf{x}_t)) \frac{\partial}{\partial \theta} \mathbf{s}_{p_{\theta,t}}(\mathbf{x}_t) \right] \quad (3.6)$$

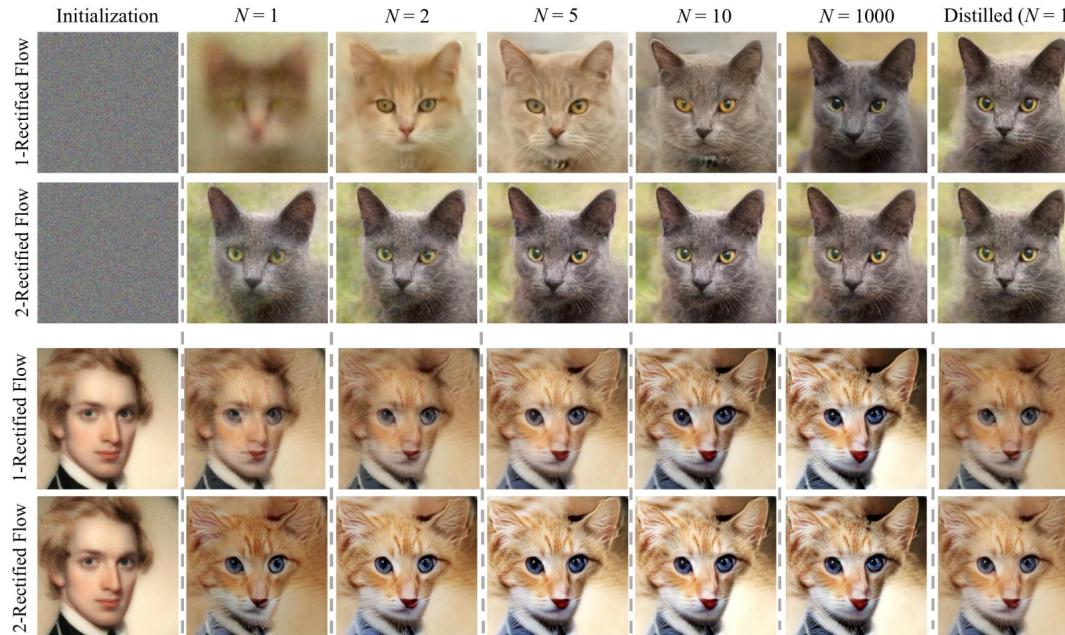
$$= -\frac{\partial}{\partial \theta} \mathbb{E}_{\substack{\mathbf{x}_0 \sim p_{\theta,0}, \\ \mathbf{x}_t | \mathbf{x}_0 \sim q_t(\mathbf{x}_t | \mathbf{x}_0)}} \left[\left\{ \mathbf{d}'(\mathbf{s}_{p_{\text{sg}[\theta],t}}(\mathbf{x}_t) - \mathbf{s}_{q_t}(\mathbf{x}_t)) \right\}^T \left\{ \mathbf{s}_{p_{\text{sg}[\theta],t}}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t | \mathbf{x}_0) \right\} \right].$$

Simplify

$$\mathbf{d}_\psi(\mathbf{x}_t, t) - \mathbf{x}_0$$

Ignore

Distillation Techniques: Rectified Flow



Advantages:

- High-quality few-step generation.
- Flexibility on inference steps.
- Simple forms.

Distillation Techniques: Rectified Flow

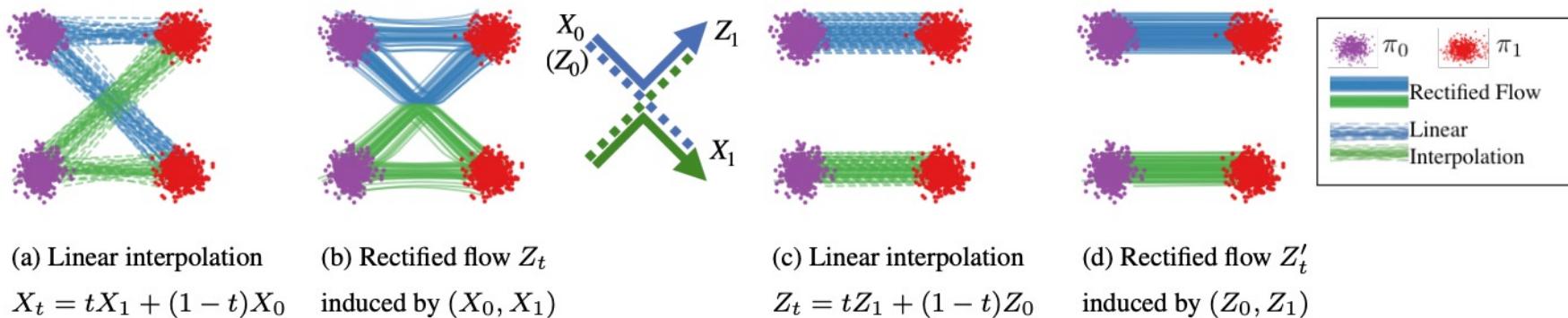
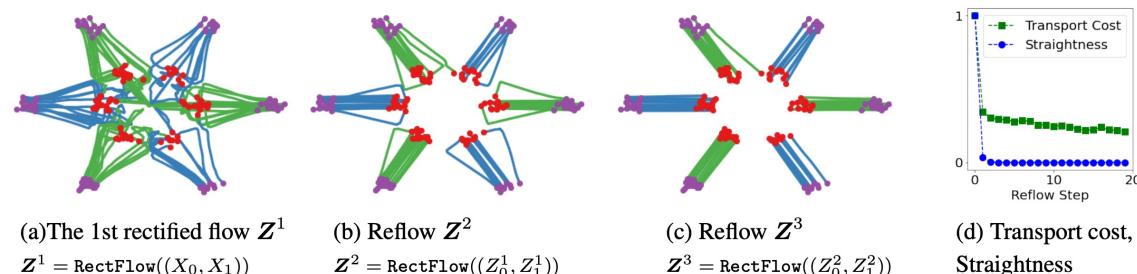
- Linear interpolation.

$$X_t = tX_1 + (1 - t)X_0$$

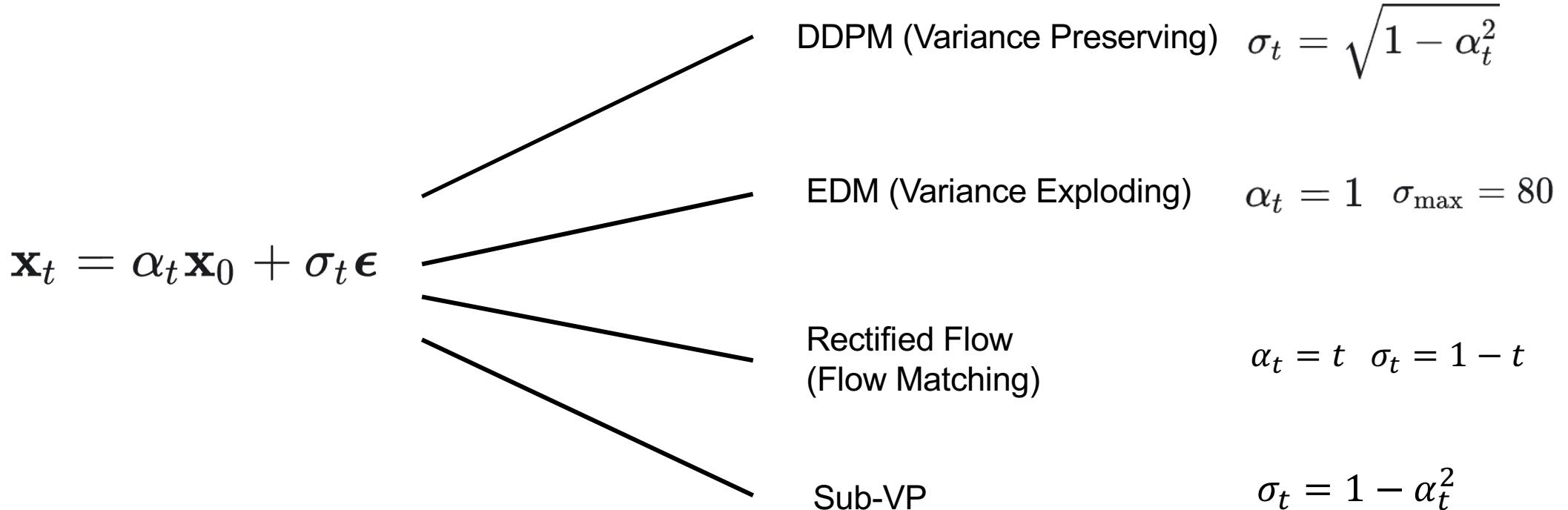
- v -prediction.

$$dX_t = (X_1 - X_0)dt$$

- Rectification (Reflow).



Diffusion Models: A (relative) Unified Perspective



The Magic of Rectified Flow: Retraining with Matched Noise-Sample Pairs

Algorithm 1 Flow Matching v -Prediction

Input:

Sample \mathbf{x}_0 from the data distribution
Sample time t from a predefined schedule or uniformly from $[0, 1]$
Sample noise ϵ from normal distribution
Compute \mathbf{x}_t : $\mathbf{x}_t = (1 - t) \cdot \mathbf{x}_0 + t \cdot \epsilon$
Predict velocity $\hat{\mathbf{v}}$ using the model: $\hat{\mathbf{v}} = \text{Model}(\mathbf{x}_t, t)$
Compute loss: $\mathcal{L} = \|\hat{\mathbf{v}} - (\mathbf{x}_0 - \epsilon)\|_2^2$
Backpropagate and update parameters

Algorithm 3 Rectified Flow v -Prediction

Input: noise-data pair $(\epsilon, \hat{\mathbf{x}}_0)$

~~Sample \mathbf{x}_0 from the data distribution~~
~~Sample time t from a predefined schedule or uniformly from $[0, 1]$~~
~~Sample noise ϵ from normal distribution~~
~~Compute \mathbf{x}_t : $\mathbf{x}_t = (1 - t) \cdot \hat{\mathbf{x}}_0 + t \cdot \epsilon$~~
~~Predict velocity $\hat{\mathbf{v}}$ using the model: $\hat{\mathbf{v}} = \text{Model}(\mathbf{x}_t, t)$~~
~~Compute loss: $\mathcal{L} = \|\hat{\mathbf{v}} - (\hat{\mathbf{x}}_0 - \epsilon)\|_2^2$~~
Backpropagate and update parameters

Flow Matching Training Is a Subset of Diffusion Training

Algorithm 1 Flow Matching v -Prediction

Input:

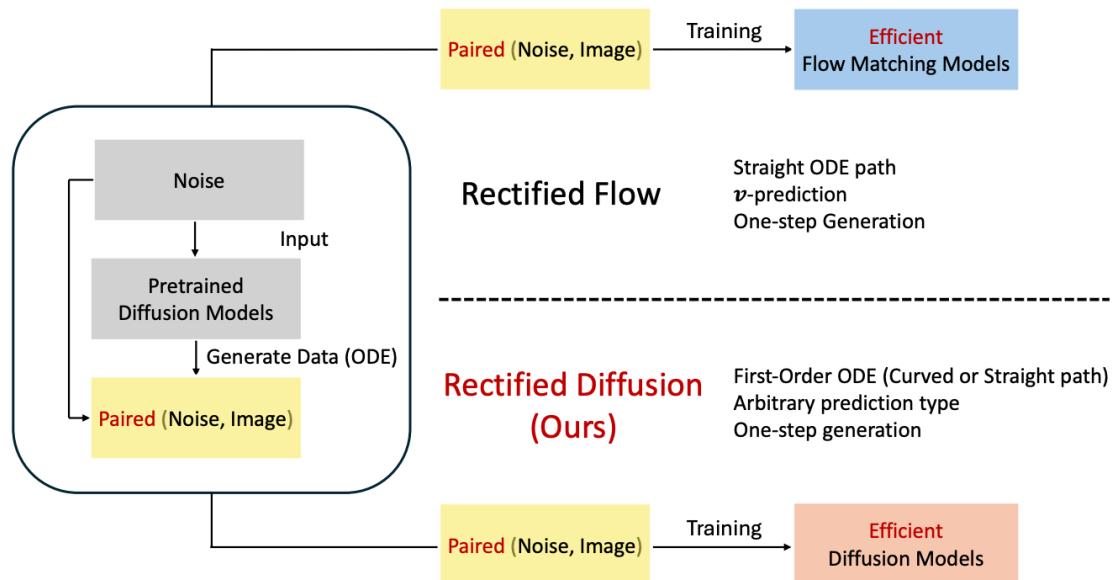
Sample \mathbf{x}_0 from the data distribution
Sample time t from a predefined schedule or uniformly from $[0, 1]$
Sample noise ϵ from normal distribution
Compute $\mathbf{x}_t : \mathbf{x}_t = (1 - t) \cdot \mathbf{x}_0 + t \cdot \epsilon$
Predict velocity $\hat{\mathbf{v}}$ using the model: $\hat{\mathbf{v}} = \text{Model}(\mathbf{x}_t, t)$
Compute loss: $\mathcal{L} = \|\hat{\mathbf{v}} - (\mathbf{x}_0 - \epsilon)\|_2^2$
Backpropagate and update parameters

Algorithm 2 Diffusion Training ϵ -Prediction

Input: α_t, σ_t

Sample \mathbf{x}_0 from the data distribution
Sample time t from a predefined schedule or uniformly from $[0, 1]$
Sample noise ϵ from normal distribution
Compute $\mathbf{x}_t : \mathbf{x}_t = \alpha_t \cdot \mathbf{x}_0 + \sigma_t \cdot \epsilon$
Predict noise $\hat{\epsilon}$ using the model: $\hat{\epsilon} = \text{Model}(\mathbf{x}_t, t)$
Compute loss: $\mathcal{L} = \|\hat{\epsilon} - \epsilon\|_2^2$
Backpropagate and update parameters

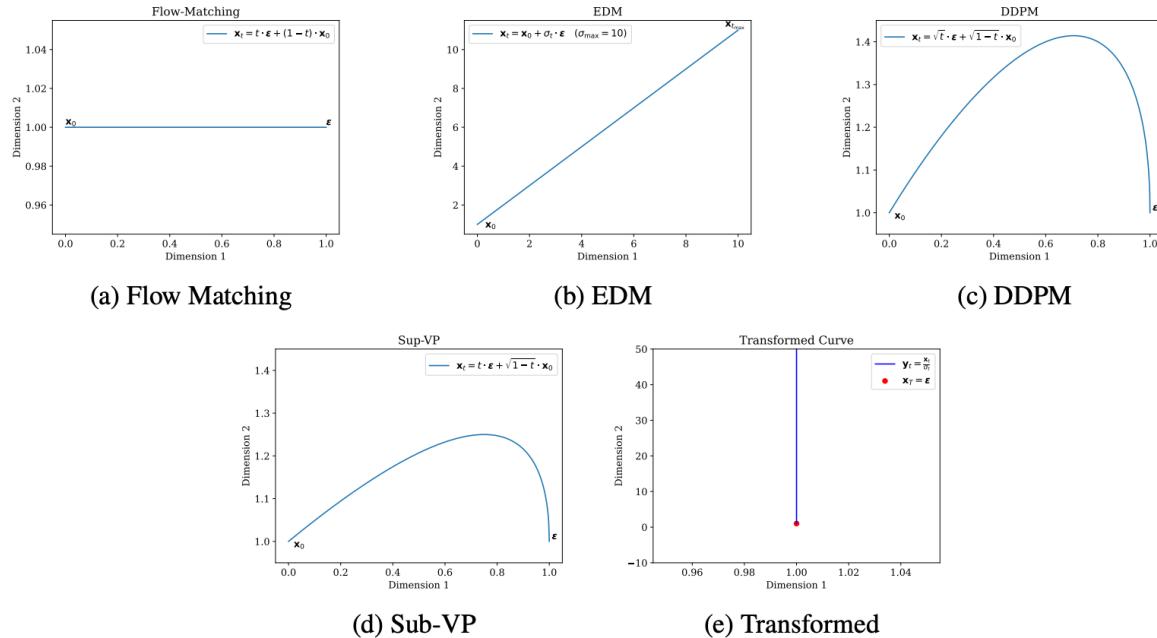
Rectified Diffusion: Extending Rectified Flow to General Diffusion Models



Algorithm 4 Rectified Diffusion ϵ -Prediction

Input: noise-data pair (ϵ, \hat{x}_0) , α_t, σ_t
Sample x_0 from the data distribution
Sample time t from a predefined schedule or uniformly from $[0, 1]$
Sample noise ϵ from normal distribution
Compute $x_t : x_t = \alpha_t \cdot \hat{x}_0 + \sigma_t \cdot \epsilon$
Predict noise $\hat{\epsilon}$ using the model: $\hat{\epsilon} = \text{Model}(x_t, t)$
Compute loss: $\mathcal{L} = \|\hat{\epsilon} - \epsilon\|_2^2$
Backpropagate and update parameters

Rectified Diffusion: the Essential Training Target Is First-Order ODE



Important Points of First-Order ODE:

- It has the same form of predefined diffusion forms.

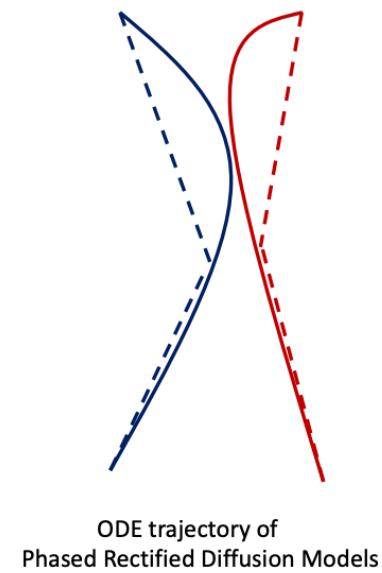
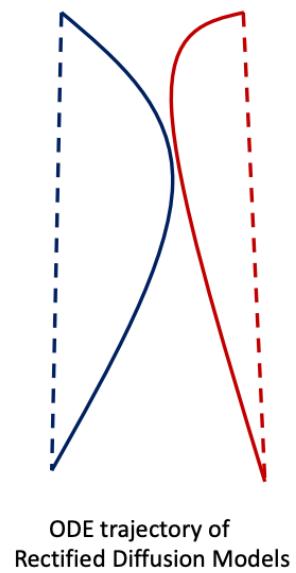
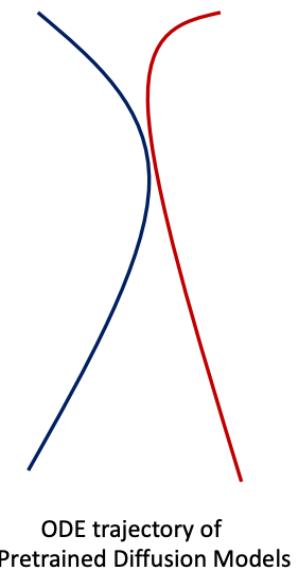
$$\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \epsilon$$

- It can be inherently curved.

- It can be transformed into straight lines with timestep dependent scaling.

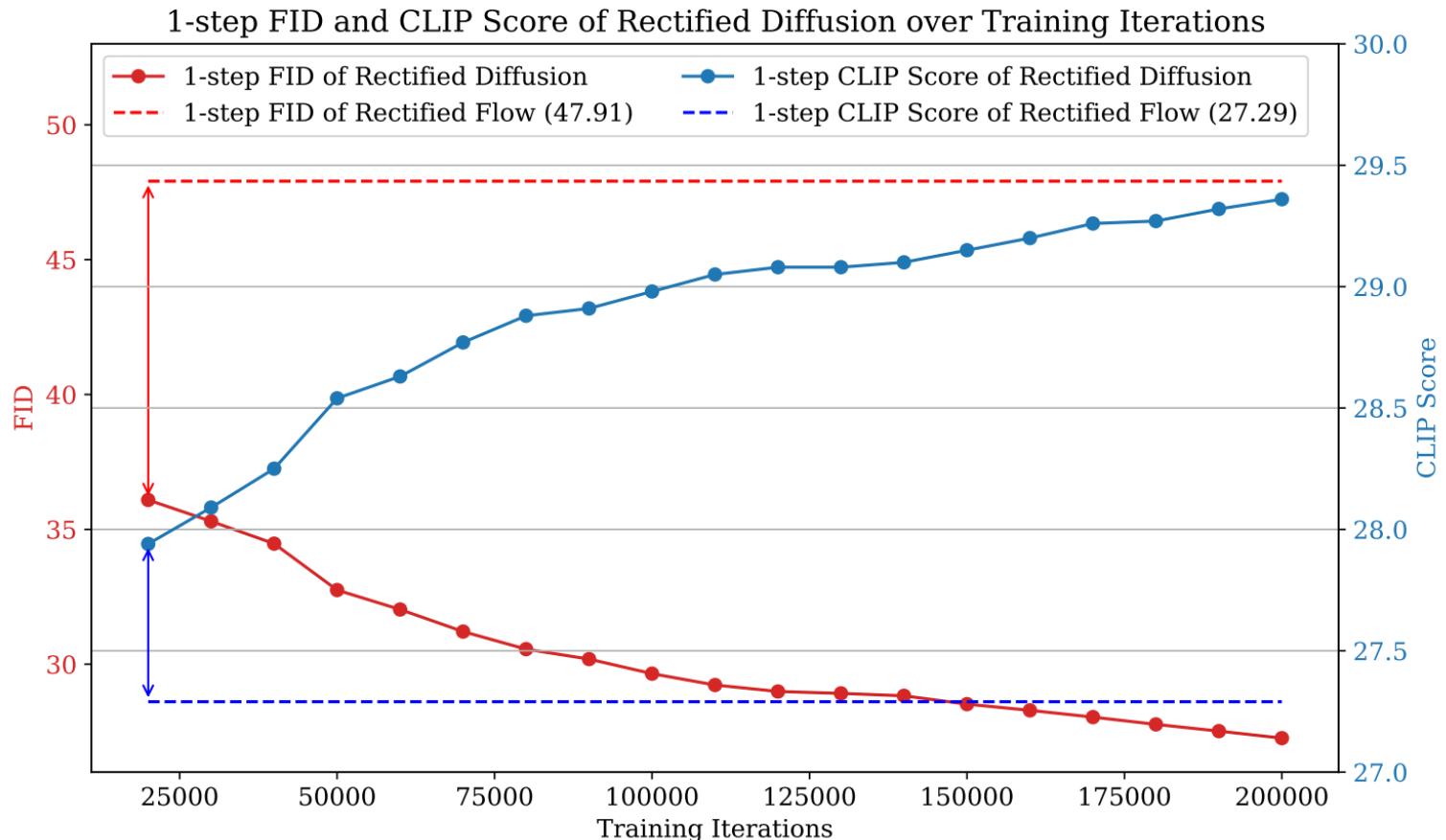
$$\mathbf{y}_t = \frac{\alpha_t}{\sigma_t} \mathbf{x}_0 + \epsilon$$

Rectified Diffusion (Phased)



$$\epsilon = \frac{\frac{\mathbf{x}_{s_{m-1}}}{\alpha_{s_{m-1}}} - \frac{\mathbf{x}_{s_m}}{\alpha_{s_m}}}{\frac{\sigma_{s_{m-1}}}{\alpha_{s_{m-1}}} - \frac{\sigma_{s_m}}{\alpha_{s_m}}} = \frac{\Delta \mathbf{z}}{\Delta \text{NSR}}$$

Rectified Diffusion Vs Rectified Flow



Rectified Diffusion Vs Rectified Flow

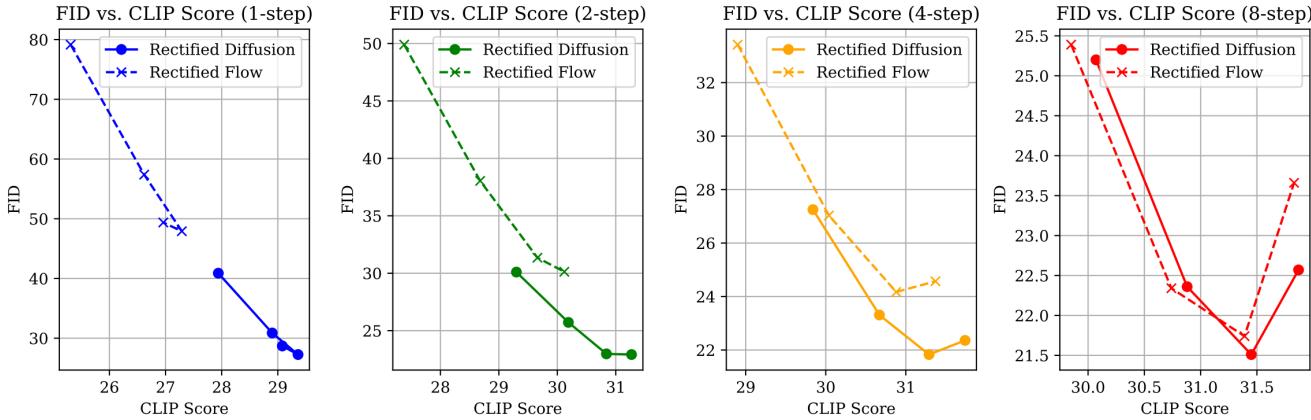


Rectified-Flow



Rectified-Diffusion

Rectified Diffusion Vs Rectified Flow



Method	Res.	Time (↓)	# Steps	# Param.	FID (↓)	CLIP (↑)
SDv1-5+DPMsolver (Upper-Bound) (Lu et al., 2022)	512	0.88s	25	0.9B	20.1	0.318
Rectified Flow (Liu et al., 2023)	512	0.88s	25	0.9B	21.65	0.315
Rectified Flow (Liu et al., 2023)	512	0.09s	1	0.9B	47.91	0.272
Rectified Flow (Liu et al., 2023)	512	0.13s	2	0.9B	31.35	0.296
Rectified Diffusion (Ours)	512	0.09s	25	0.9B	21.28	0.316
Rectified Diffusion (Ours)	512	0.09s	1	0.9B	27.26	0.295
Rectified Diffusion (Ours)	512	0.13s	2	0.9B	22.98	0.309
Rectified Flow (Distill) (Liu et al., 2023)	512	0.09s	1	0.9B	23.72	0.302
Rectified Flow (Distill) (Liu et al., 2023)	512	0.13s	2	0.9B	73.49	0.261
Rectified Flow (Distill) (Liu et al., 2023)	512	0.21s	4	0.9B	103.48	0.245
Rectified Diffusion (CD) (Ours)	512	0.09s	1	0.9B	22.83	0.305
Rectified Diffusion (CD) (Ours)	512	0.13s	2	0.9B	21.66	0.312
Rectified Diffusion (CD) (Ours)	512	0.21s	4	0.9B	21.43	0.314
PeRFFlow (Yan et al., 2024)	512	0.21s	4	0.9B	22.97	0.294
Rectified Diffusion (Phased) (Ours)	512	0.21s	4	0.9B	20.64	0.311
PeRFFlow-SDXL (Yan et al., 2024)	1024	0.71s	4	3B	27.06	0.335
Rectified Diffusion-SDXL (Phased) (Ours)	1024	0.71s	4	3B	25.81	0.341

References

- [1] Denoising Diffusion Probabilistic Models.
- [2] Denoising Diffusion Implicit Models.
- [3] Score-Based Generative Modeling through Stochastic Differential Equations.
- [4] Flow Matching for Generative Modeling.
- [5] Elucidating the Design Space of Diffusion-Based Generative Models.
- [6] DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps.
- [7] Discrete Flow Matching.
- [8] Consistency Models.
- [9] Consistency Models Made Easy.
- [10] Latent Consistency Models: Synthesizing High-Resolution Images with Few-step Inference.
- [11] Phased Consistency Model.
- [12] Multistep Consistency Models.
- [13] PeRFlow: Piecewise Rectified Flow as Universal Plug-and-Play Accelerator.
- [14] Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow.
- [15] InstaFlow: One Step is Enough for High-Quality Diffusion-Based Text-to-Image Generation.
- [16] StyleGAN-T: Unlocking the Power of GANs for Fast Large-Scale Text-to-Image Synthesis.
- [17] Stable Consistency Tuning: Understanding and Improving Consistency Models.
- [18] SwiftBrush : One-Step Text-to-Image Diffusion Model with Variational Score Distillation.
- [19] One-step Diffusion with Distribution Matching Distillation.
- [20] Progressive Distillation for Fast Sampling of Diffusion Models

Our Works

- [1] Stable Consistency Tuning: Understanding and Improving Consistency Models.
- [2] Rectified Diffusion: Straightness Is Not Your Need in Rectified Flow.
- [3] Phased Consistency Model.
- [4] AnimateLCM: Computation-Efficient Personalized Style Video Generation without Personalized Video Data.

Thank you!

Fu-Yun Wang

fywang@link.cuhk.edu.hk