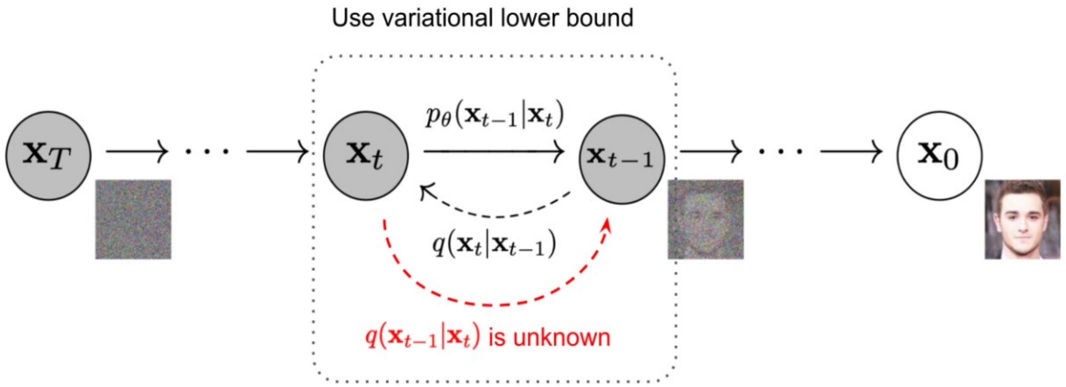
The Principle and Application of Diffusion Acceleration

Fu-Yun Wang

fywang0126@gmail.com

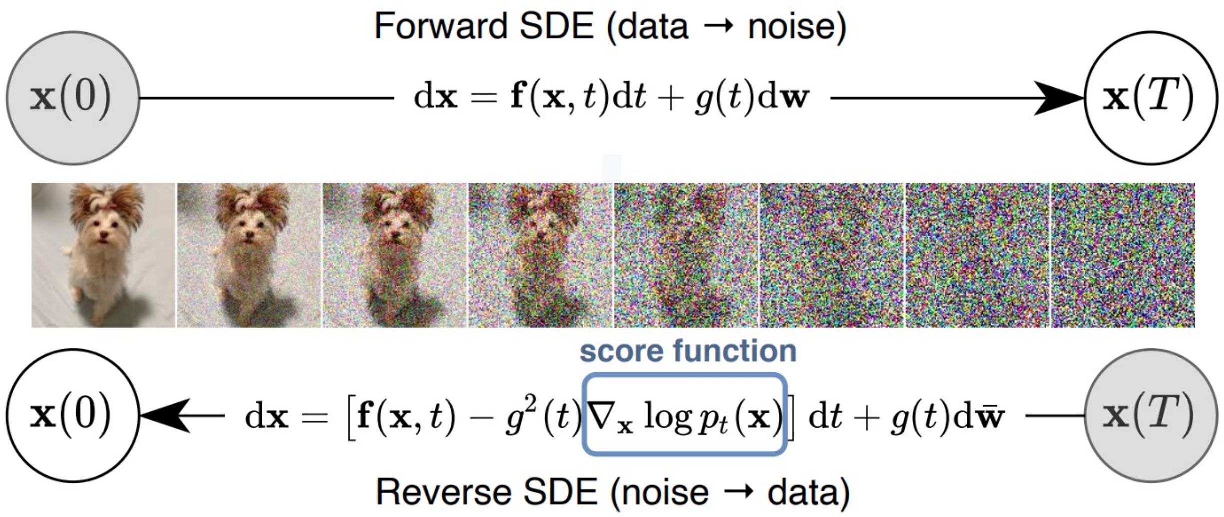
Diffusion Models: Markovian Perspective



- Assumption:
  - $$p(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$$
- Forward Process:
  - $$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$
  - $$p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \mathbf{I})$$
- Reverse Process:
  - $$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}$$
  - Maximum Likelihood Estimation (MLE) is Equivalent to

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

Diffusion Models: Stochastic Differential Equation Perspective



The only unknown term is the score function.

Train a neural network through score matching!

Probability Flow ODE:

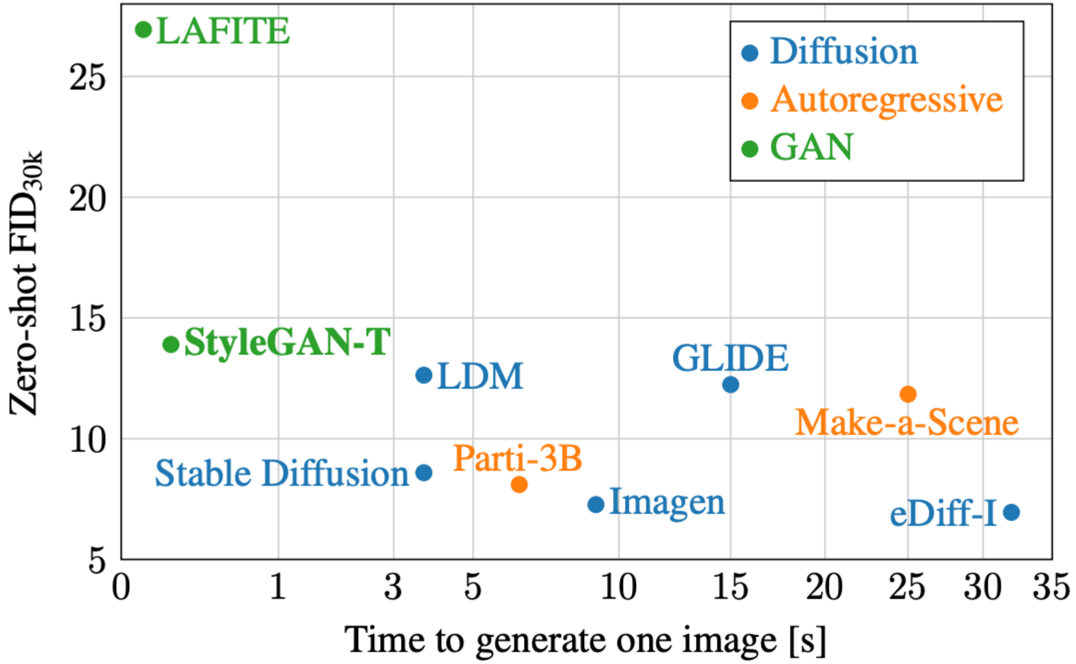
A deterministic reverse process

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

Exact Solution form of PF-ODE

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

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.

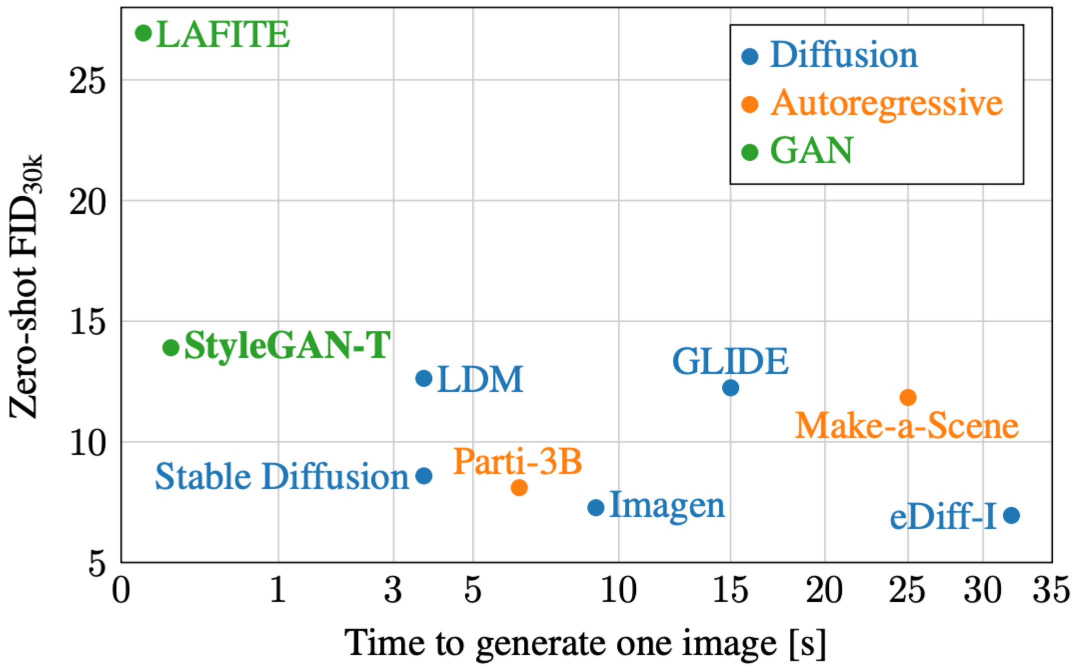
DPM-Solver

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

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

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

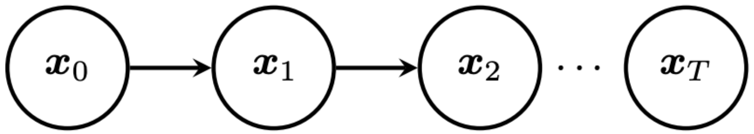
Diffusion Models: Slow Inference Speed

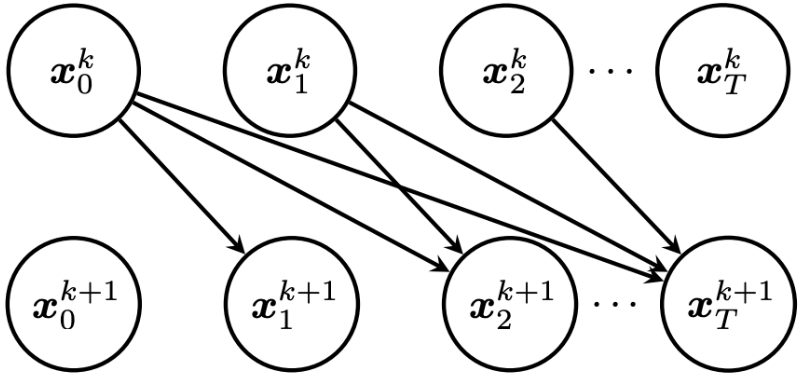


How to speed up the diffusion generation?

- 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





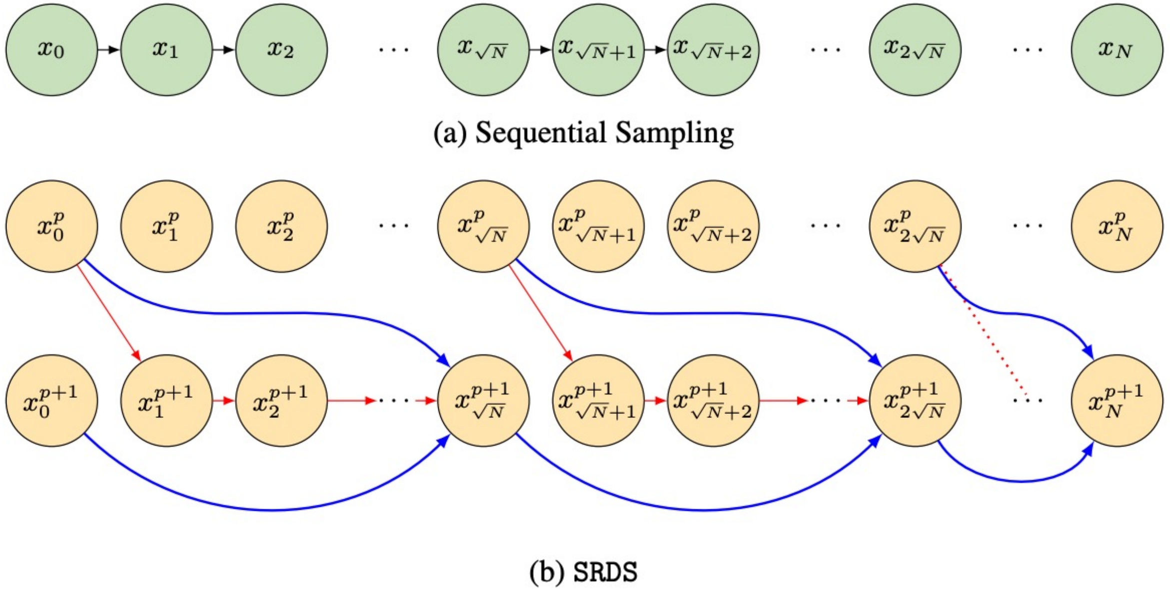
$$\boldsymbol{x}_t = \boldsymbol{x}_0 + \int_0^t s(\boldsymbol{x}_u, u)du.$$

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

Lower Bound of Picard Iteration = Sequential Denoising

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

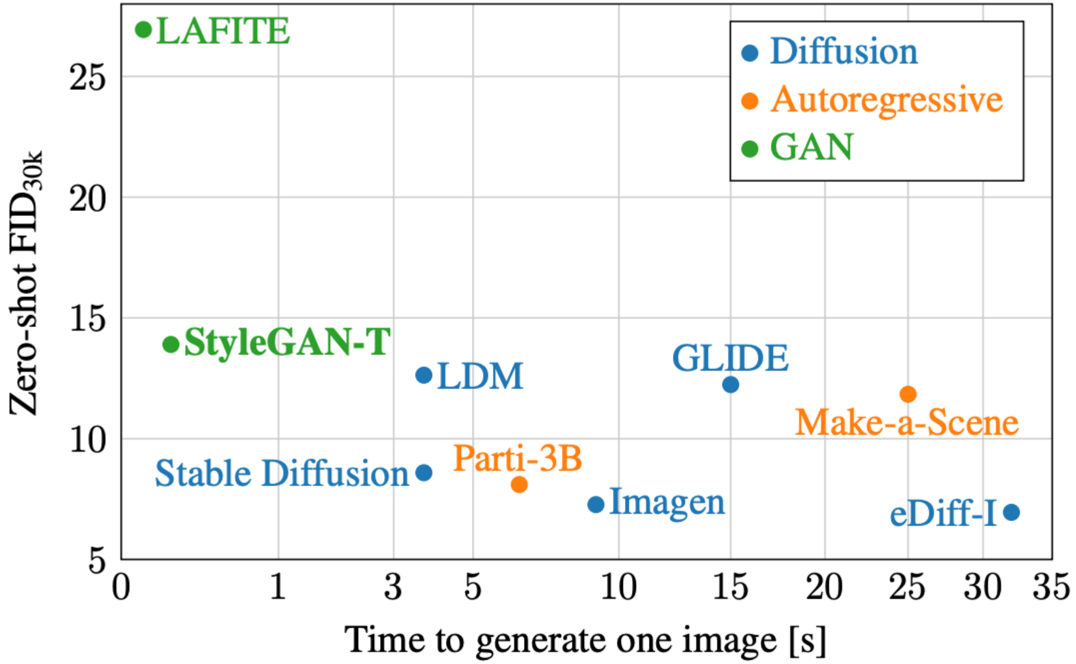
Parareal Algorithm



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

Fine Solver (Parallel) Coarse Solver (Sequential)

Diffusion Models: Slow Inference Speed



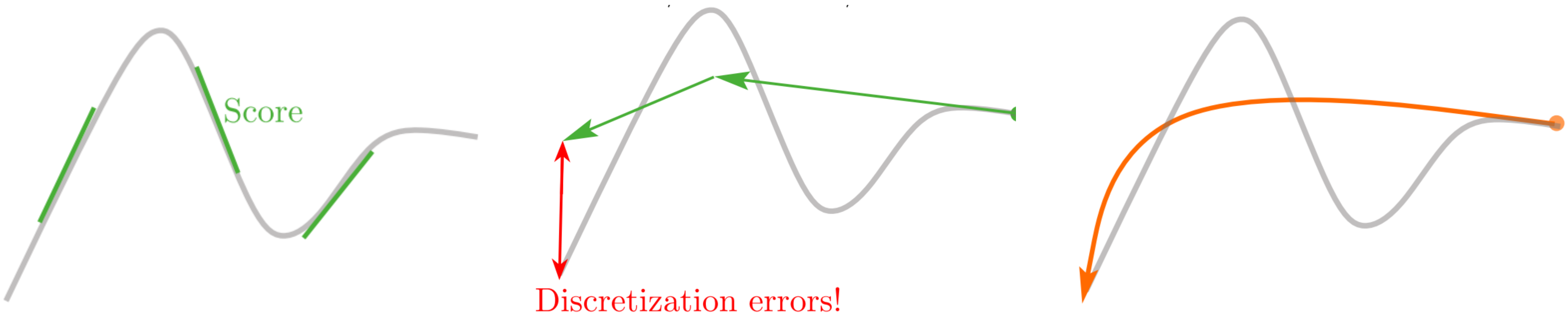
How to speed up the diffusion generation?

- 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.

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

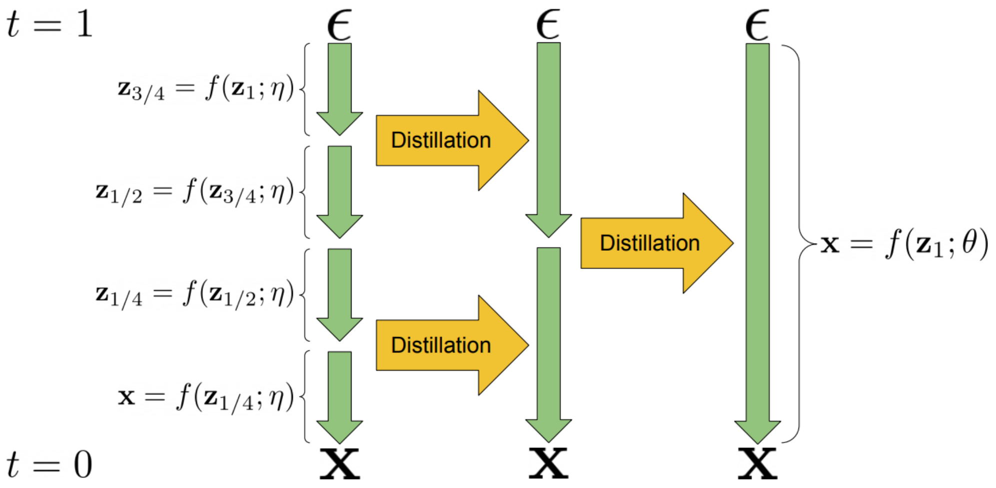


PF-ODE

Discretized numerical solving.

Naïve distillation.

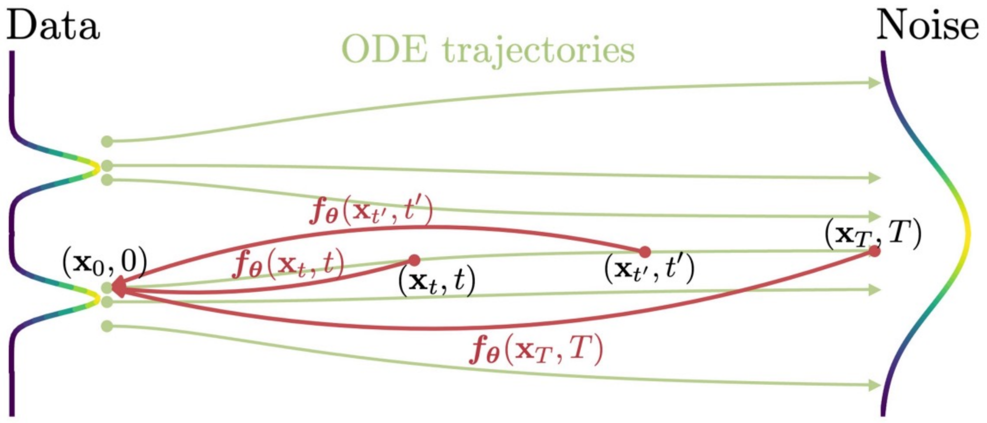
Distillation Techniques: Progressive Distillation



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**
    $\theta \leftarrow \eta$ ▷ Init student from teacher
    while not converged **do**
      $\mathbf{x} \sim \mathcal{D}$
      $t = i/N, \;\; i \sim Cat[1, 2, \ldots, N]$
      $\epsilon \sim N(0, I)$
      $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$
      `# 2 steps of DDIM with teacher`
      $t' = t - 0.5/N, \quad t'' = t - 1/N$
      $\mathbf{z}_{t'} = \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_t) + \frac{\sigma_{t'}}{\sigma_t}(\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\eta(\mathbf{z}_t))$
      $\mathbf{z}_{t''} = \alpha_{t''} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}}(\mathbf{z}_{t'} - \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}))$
      $\tilde{\mathbf{x}} = \frac{\mathbf{z}_{t''} - (\sigma_{t''}/\sigma_t)\mathbf{z}_t}{\alpha_{t''} - (\sigma_{t''}/\sigma_t)\alpha_t}$ ▷ 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
    $\eta \leftarrow \theta$ ▷ Student becomes next teacher
    $N \leftarrow N/2$ ▷ Halve number of sampling steps
  end for

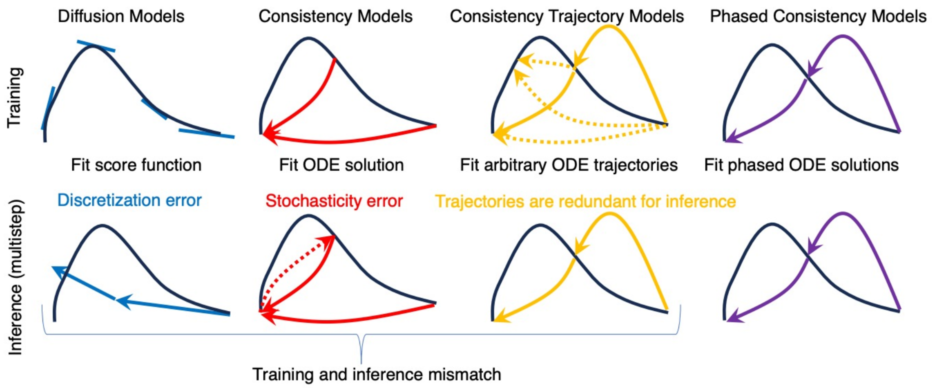
Distillation Techniques: Consistency Distillation



Algorithm 2 Consistency Distillation (CD)

Input: dataset $\mathcal{D}$, initial model parameter $\boldsymbol{\theta}$, learning rate η, ODE solver $\Phi(\cdot, \cdot; \boldsymbol{\phi})$, $d(\cdot, \cdot)$, $\lambda(\cdot)$, and μ
$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
repeat
  Sample $\mathbf{x} \sim \mathcal{D}$ and $n \sim \mathcal{U}[\![1, N-1]\!]$
  Sample $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t_{n+1}^2 \boldsymbol{I})$
  $\hat{\mathbf{x}}_{t_n}^{\phi} \leftarrow \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi})$
  $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) \leftarrow$
    $\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\phi}, t_n))$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$
  $\boldsymbol{\theta}^- \leftarrow \text{stopgrad}(\mu \boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta})$
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 $\boldsymbol{\theta}$, 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 $t_0 = \epsilon < t_1 < t_2 < \cdots < t_N = T$, edge timesteps $s_0 = t_0 < s_1 < s_2 < \cdots < s_M = t_N \in \{t_i\}_{i=0}^{N}$ to split the ODE trajectory into M sub-trajectories.
Training data : $\mathcal{D}_{\mathbf{x}} = \{(\mathbf{x}, \boldsymbol{c})\}$
$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
repeat
  Sample $(\boldsymbol{z}, \boldsymbol{c}) \sim \mathcal{D}_z$, $n \sim \mathcal{U}[0, N-k]$ and $\omega \sim [\omega_{\min}, \omega_{\max}]$
  Sample $\mathbf{x}_{t_{n+k}} \sim \mathcal{N}(\alpha_{t_{n+k}} \boldsymbol{z}; \sigma^2_{t_{n+k}} \mathbf{I})$
  Determine $[s_m, s_{m+1}]$ given n
  $\mathbf{x}^{\phi}_{t_n} \leftarrow (1+\omega)\Psi(\mathbf{x}_{t_{n+k}}, t_{n+k}, t_n, \boldsymbol{c}) - \omega\Psi(\mathbf{x}_{t_{n+k}}, t_{n+k}, t_n, \varnothing)$
  $\tilde{\mathbf{x}}_{s_m} = \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+k}}, t_{n+k}, \boldsymbol{c})$ and $\hat{\mathbf{x}}_{s_m} = \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}^{\phi}_{t_n}, t_n, \boldsymbol{c})$
  Obtain $\tilde{\mathbf{x}}_s$ and $\hat{\mathbf{x}}_s$ through adding noise to $\tilde{\mathbf{x}}_{s_m}$ and $\hat{\mathbf{x}}_{s_m}$
  $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = d(\tilde{\mathbf{x}}_{s_m}, \hat{\mathbf{x}}_{s_m}) + \lambda(\mathrm{ReLU}(1 + \tilde{\mathbf{x}}_s) + \mathrm{ReLU}(1 - \hat{\mathbf{x}}_s))$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$
  $\boldsymbol{\theta}^- \leftarrow \mathrm{stopgrad}(\mu \boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta})$
until convergence

Application: AnimateLCM

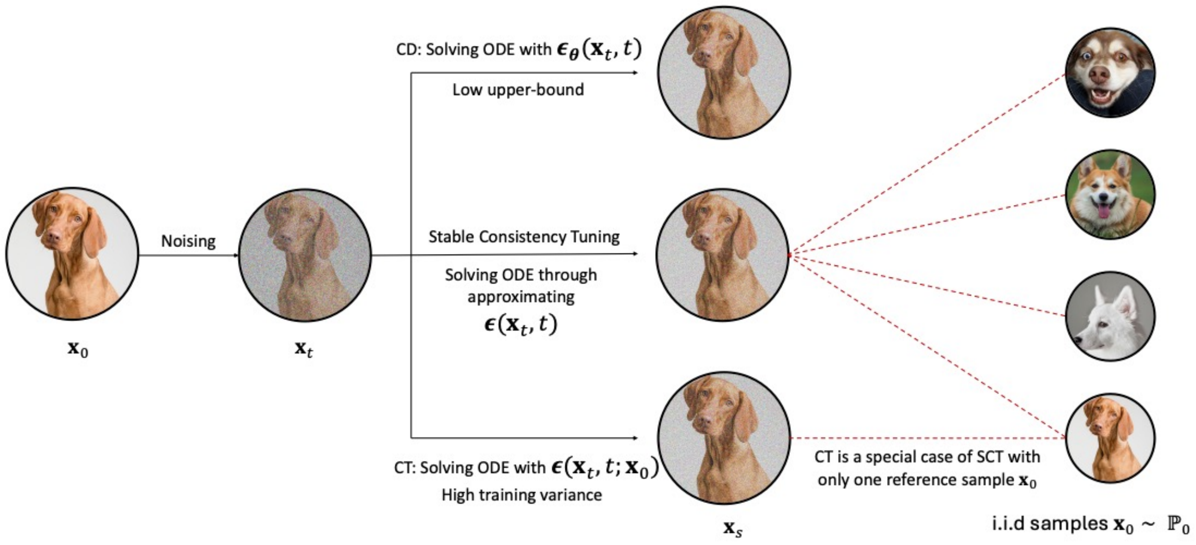
- AnimateLCM support
  - NOTE: You will need to use `autoselect` or `lcm` or `lcm[100_ots]` beta_schedule. To use fully with LCM, be sure to use appropriate LCM lora, use the `lcm` sampler_name in KSampler nodes, and lower cfg to somewhere around 1.0 to 2.0. Don't forget to decrease steps (minimum = ~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



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

Bootstrapping

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

Consistency Distillation

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

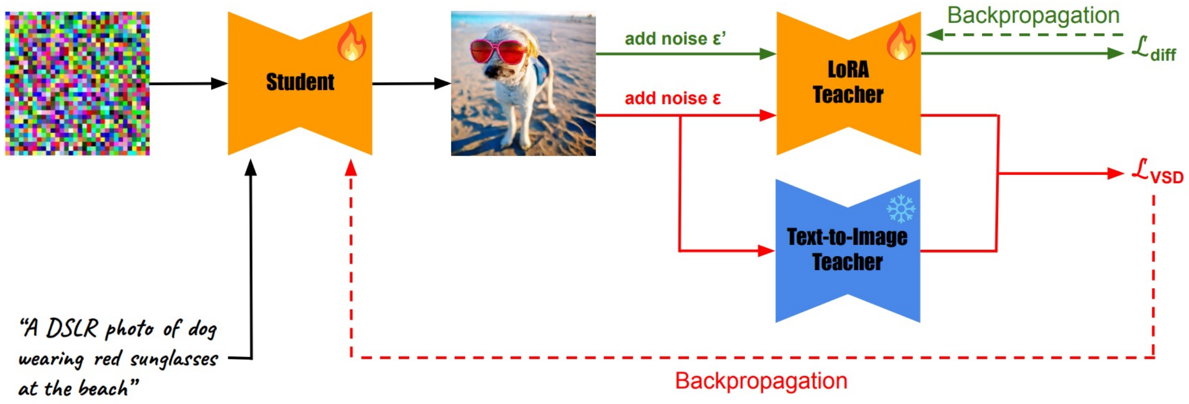
Consistency Training

Ground Truth of Score Estimation: Stable Consistency Tuning

$$
\begin{aligned}
\nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \boldsymbol{c}) &= \mathbb{E}_{\mathbb{P}(\mathbf{x}_0 \mid \mathbf{x}_t, \boldsymbol{c})} \left[\nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \mathbf{x}_0, \boldsymbol{c})\right] \\
&= \mathbb{E}_{\mathbb{P}(\mathbf{x}_0 \mid \boldsymbol{c})} \left[\frac{\mathbb{P}(\mathbf{x}_0 \mid \mathbf{x}_t, \boldsymbol{c})}{\mathbb{P}(\mathbf{x}_0 \mid \boldsymbol{c})} \nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \mathbf{x}_0, \boldsymbol{c})\right] \\
&= \mathbb{E}_{\mathbb{P}(\mathbf{x}_0 \mid \boldsymbol{c})} \left[\frac{\mathbb{P}(\mathbf{x}_t \mid \mathbf{x}_0, \boldsymbol{c})}{\mathbb{P}(\mathbf{x}_t \mid \boldsymbol{c})} \nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t \mid \mathbf{x}_0, \boldsymbol{c})\right] \\
&= \mathbb{E}_{\mathbb{P}(\mathbf{x}_0 \mid \boldsymbol{c})} \left[\frac{\mathbb{P}(\mathbf{x}_t \mid \mathbf{x}_0)}{\mathbb{P}(\mathbf{x}_t \mid \boldsymbol{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 c)}^{i=0,\ldots n-1} \frac{\mathbb{P}(\mathbf{x}_t \mid \mathbf{x}_0^{(i)})}{\mathbb{P}(\mathbf{x}_t \mid \boldsymbol{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 c)}^{i=0,\ldots 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)}, \boldsymbol{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 c)}^{i=0,\ldots 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

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



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: $\theta \leftarrow \theta - \eta_1 \left[\omega(t)\left(\epsilon_\psi(\hat{x}_t, t, y) - \epsilon_\phi(\hat{x}_t, t, y)\right)\frac{\partial \hat{x}_0}{\partial \theta}\right]$
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

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

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

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

Score Divergence Gradient Theorem

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

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

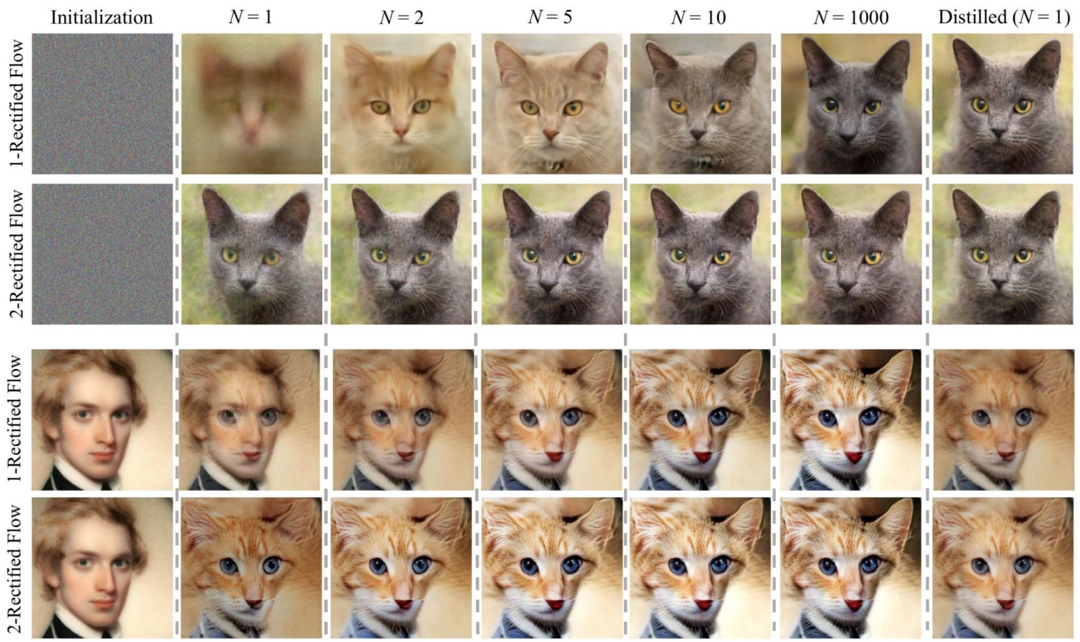
$$= -\frac{\partial}{\partial\theta}\mathbb{E}_{\substack{\boldsymbol{x}_0 \sim p_{\theta,0},\\ \boldsymbol{x}_t|\boldsymbol{x}_0 \sim q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}}\left[\left\{\mathbf{d}'(\boldsymbol{s}_{p_{\mathrm{sg}[\theta],t}}(\boldsymbol{x}_t) - \boldsymbol{s}_{q_t}(\boldsymbol{x}_t))\right\}^T\left\{\boldsymbol{s}_{p_{\mathrm{sg}[\theta],t}}(\b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

Simplify

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

Ignore

Distillation Techniques: Rectified Flow



Advantages:

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

Distillation Techniques: Rectified Flow

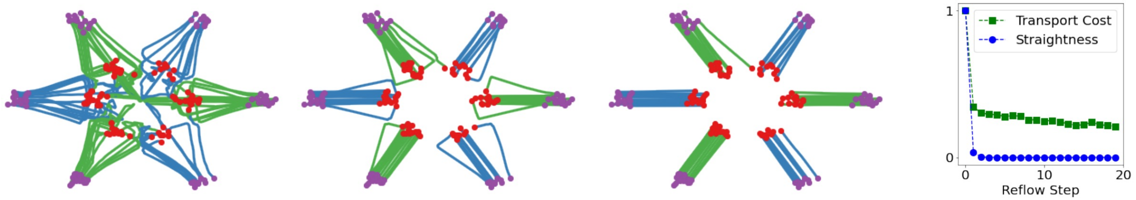
- Linear interpolation.

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

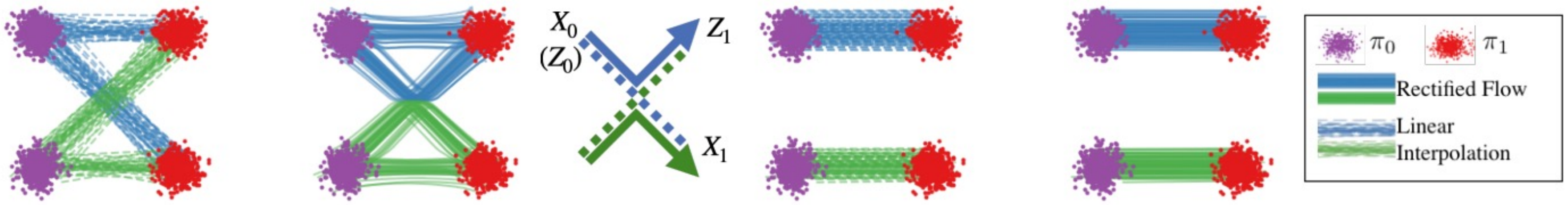
- v-prediction.

$$\mathrm{d}X_t = (X_1 - X_0)\mathrm{d}t$$

- Rectification (Reflow).



(a)The 1st rectified flow $\boldsymbol{Z}^1$ $\boldsymbol{Z}^1 = \mathtt{RectFlow}((X_0, X_1))$ (b) Reflow $\boldsymbol{Z}^2$ $\boldsymbol{Z}^2 = \mathtt{RectFlow}((Z_0^1, Z_1^1))$ (c) Reflow $\boldsymbol{Z}^3$ $\boldsymbol{Z}^3 = \mathtt{RectFlow}((Z_0^2, Z_1^2))$ (d) Transport cost, Straightness



(a) Linear interpolation $X_t = tX_1 + (1-t)X_0$ (b) Rectified flow Z_t induced by (X_0, X_1) (c) Linear interpolation $Z_t = tZ_1 + (1-t)Z_0$ (d) Rectified flow Z'_t induced by (Z_0, Z_1)

Diffusion Models: A (relative) Unified Perspective

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

- DDPM (Variance Preserving) $\sigma_t = \sqrt{1 - \alpha_t^2}$
- EDM (Variance Exploding) $\alpha_t = 1 \quad \sigma_{\max} = 80$
- Rectified Flow (Flow Matching) $\alpha_t = t \quad \sigma_t = 1 - t$
- Sub-VP $\sigma_t = 1 - \alpha_t^2$

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

Algorithm 1 Flow Matching $\boldsymbol{v}$-Prediction

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

Algorithm 3 Rectified Flow $\boldsymbol{v}$-Prediction

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

Flow Matching Training Is a Subset of Diffusion Training

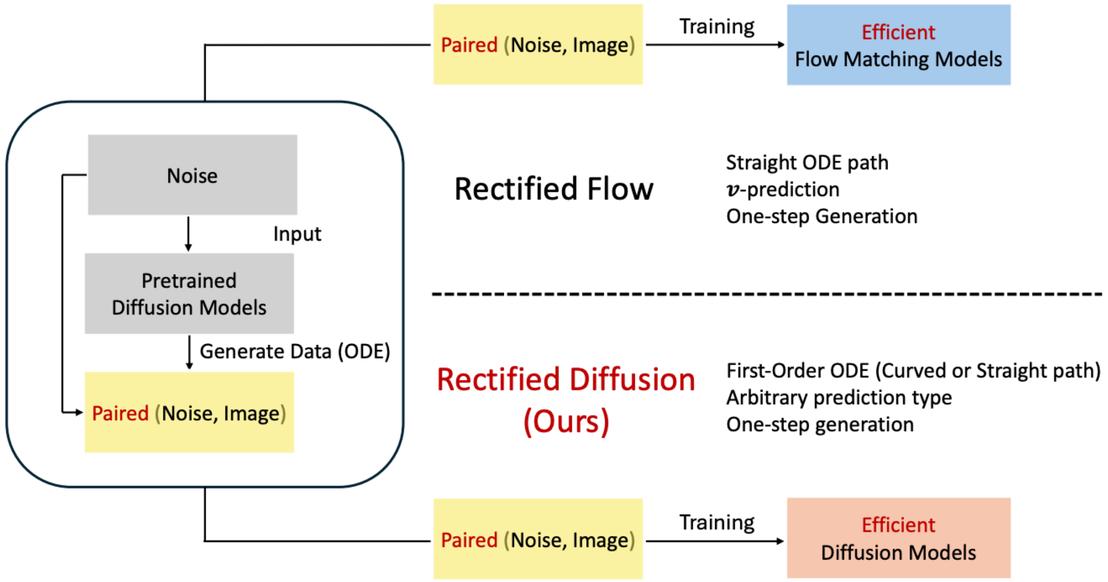
Algorithm 1 Flow Matching $\boldsymbol{v}$-Prediction

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

Algorithm 2 Diffusion Training $\boldsymbol{\epsilon}$-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 $\boldsymbol{\epsilon}$ from normal distribution
Compute $\mathbf{x}_t$: $\mathbf{x}_t = \alpha_t \cdot \mathbf{x}_0 + \sigma_t \cdot \boldsymbol{\epsilon}$
Predict noise $\hat{\boldsymbol{\epsilon}}$ using the model: $\hat{\boldsymbol{\epsilon}} = \text{Model}(\mathbf{x}_t, t)$
Compute loss: $\mathcal{L} = \|\hat{\boldsymbol{\epsilon}} - \boldsymbol{\epsilon}\|_2^2$
Backpropagate and update parameters

Rectified Diffusion: Extending Rectified Flow to General Diffusion Models



Algorithm 4 Rectified Diffusion ϵ-Prediction

Input: noise-data pair $(\boldsymbol{\epsilon}, \hat{\mathbf{x}}_0)$, 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 $\boldsymbol{\epsilon}$ from normal distribution~~

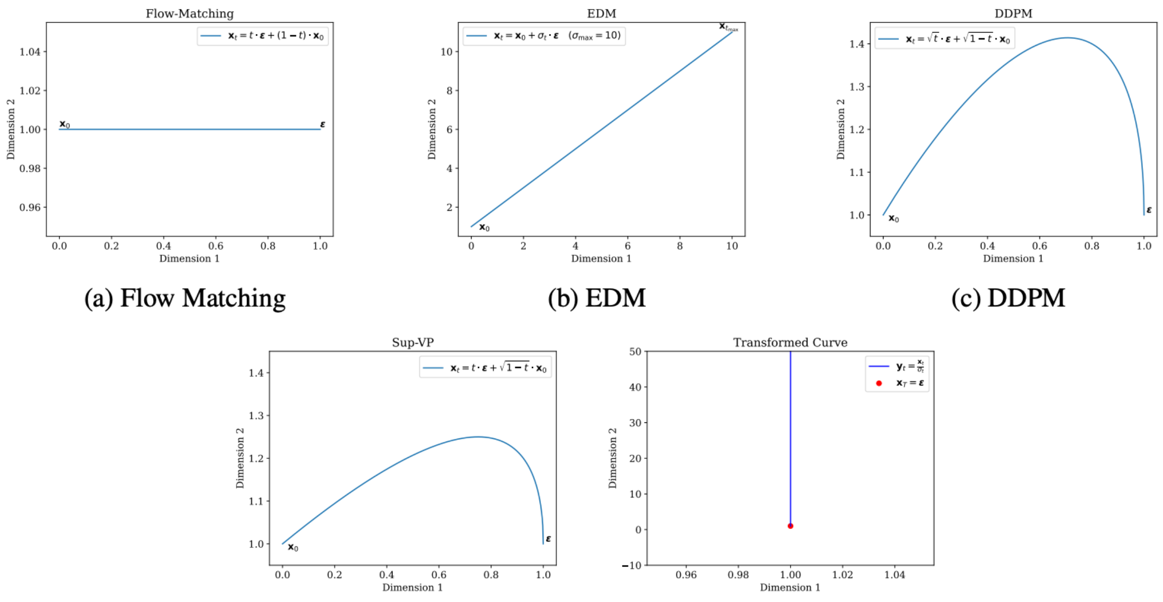
Compute $\mathbf{x}_t$: $\mathbf{x}_t = \alpha_t \cdot \hat{\mathbf{x}}_0 + \sigma_t \cdot \boldsymbol{\epsilon}$

Predict noise $\hat{\boldsymbol{\epsilon}}$ using the model: $\hat{\boldsymbol{\epsilon}} = \text{Model}(\mathbf{x}_t, t)$

Compute loss: $\mathcal{L} = \|\hat{\boldsymbol{\epsilon}} - \boldsymbol{\epsilon}\|_2^2$

Backpropagate and update parameters

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



(a) Flow Matching (b) EDM (c) DDPM

(d) Sub-VP (e) Transformed

Important Points of First-Order ODE:

- It has the same form of predefined diffusion forms.

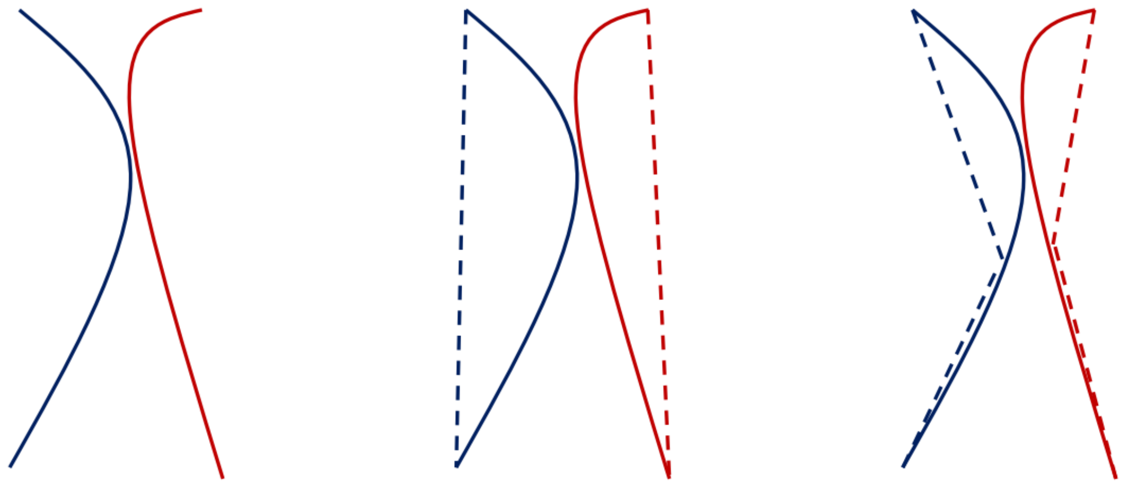
$$\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$$

- It can be inherently curved.

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

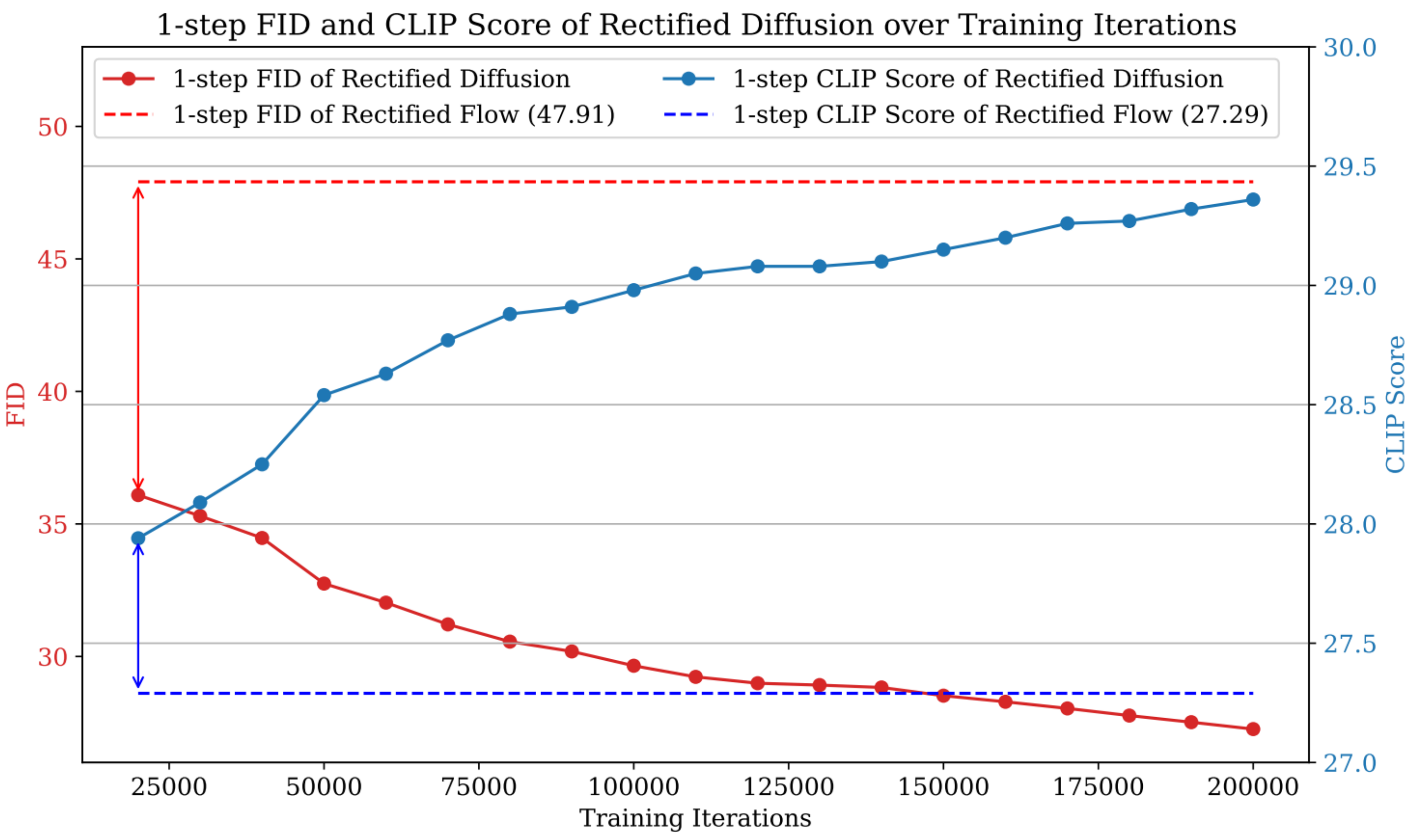
$$\mathbf{y}_t = \frac{\alpha_t}{\sigma_t}\mathbf{x}_0 + \boldsymbol{\epsilon}$$

Rectified Diffusion (Phased)

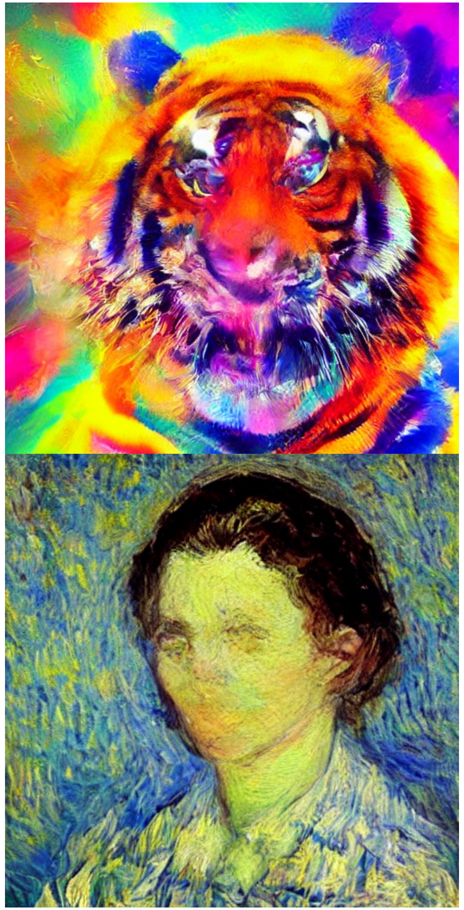


$$\boldsymbol{\epsilon} = \frac{\frac{\mathbf{x}_{s_{m-1}}}{\alpha_{s_{m-1}}} - \frac{\mathbf{x}_{s_m}}{\alpha_{s_m}}}{\frac{\sigma_{s_{m-1}}}{\alpha_{s_{m-1}}} - \frac{\sigma_{s_m}}{\alpha_{s_m}}} = \frac{\Delta \mathbf{z}}{\Delta \mathrm{NSR}}$$

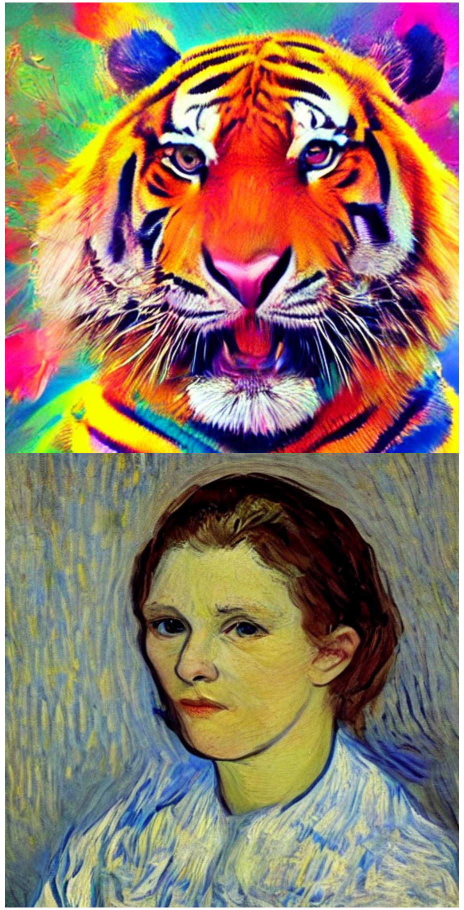
Rectified Diffusion Vs Rectified Flow



Rectified Diffusion Vs Rectified Flow

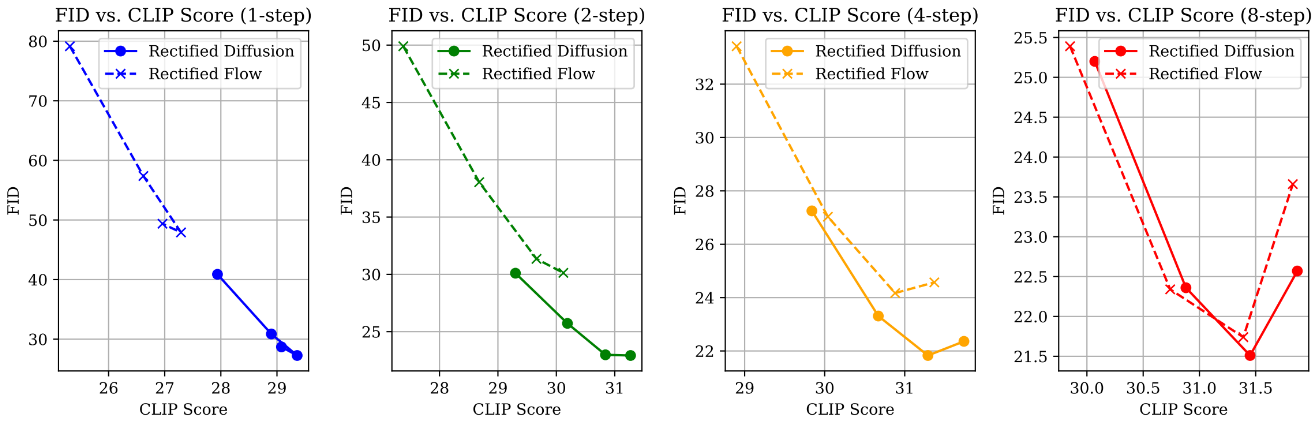


Rectified-Flow



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
PeRFlow (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
PeRFlow-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