DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps (Neurips 2022 Oral, accept rate ~1.7%)

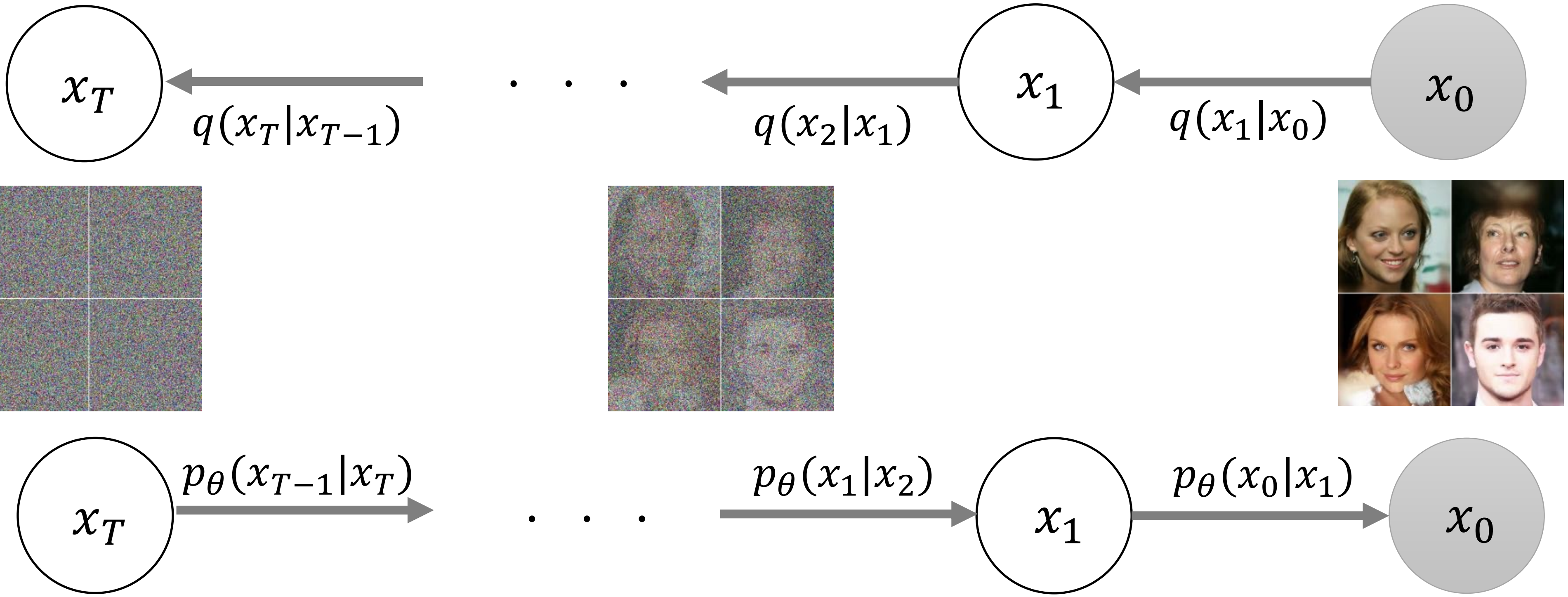
Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, Jun Zhu

Tsinghua University

Denoising Diffusion Probabilistic Models (DDPM)

Learning to gradually denoise (Sohl-Dickstein et al., 2015; Ho et al., 2020;)

$q(x_T|x_0) \approx \mathcal{N}(0, I)$ $q(x_t|x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ $q(x_0) = p_{data}(x)$



$p(x_T) = \mathcal{N}(0, I)$ $p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$ $p_\theta(x_0)$

Why Diffusion Models?

Diffusion models are simple but effective

- No need to learn an "encoder" $q(z|x)$.
  - VAEs: Learn models by "**searching**" both p_θ and q_ϕ.
  - Diffusion models: Learn models by "**imitating**" a fixed forward process q.
- Training objective is simple: (MSE loss)

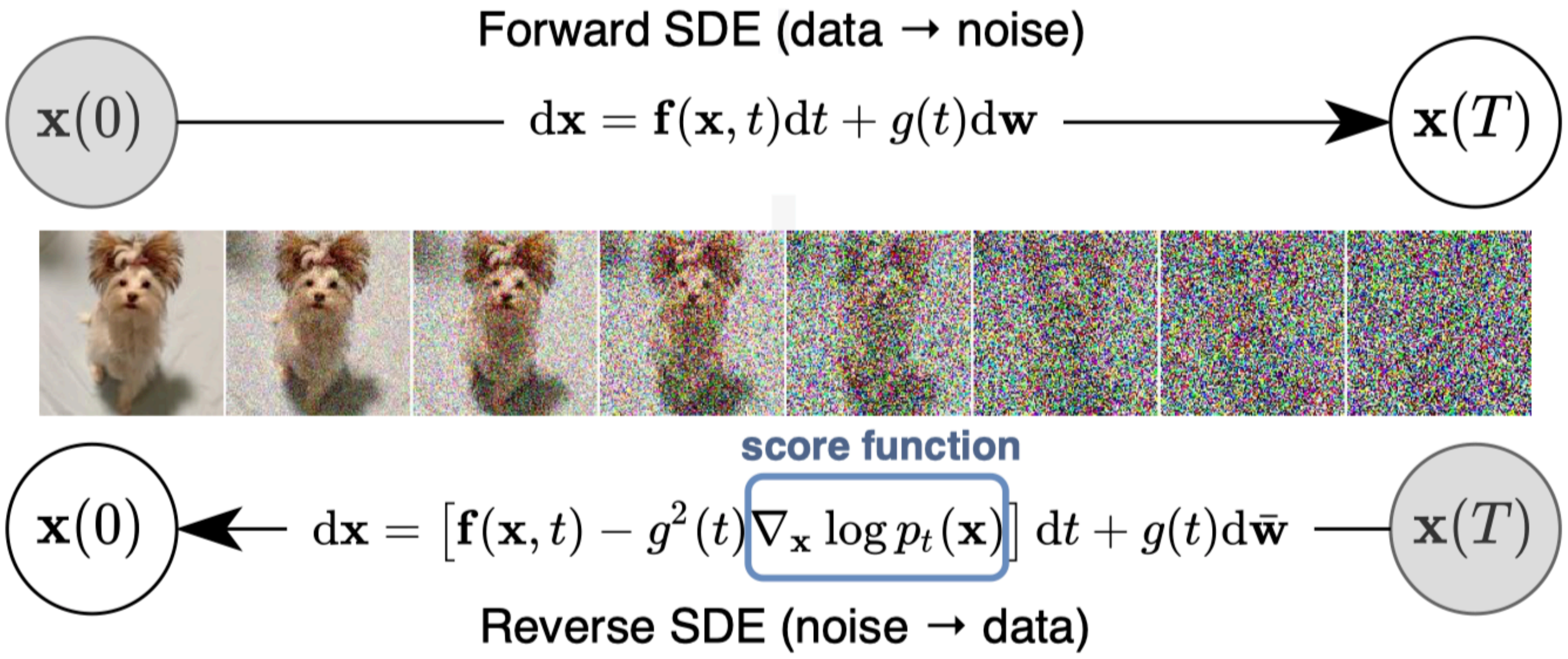
$$\frac{1}{2}\int_0^T \omega(t)\mathbb{E}_{q_0(\boldsymbol{x}_0)}\mathbb{E}_{q(\boldsymbol{\epsilon})}\left[\|\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t)-\boldsymbol{\epsilon}\|_2^2\right]\mathrm{d}t$$

- Convergence guarantee:

  For large enough T, the reverse process is indeed Gaussian!

Going to Infinity: Continuous-time Diffusion Models

Score-based Generative Models (Song et al.,2021)



The forward SDE and the reverse SDE has **the same path measure (“joint distribution”)**.

By **estimating the score function** at each time t, we can get a generative model from noise distribution to data distribution.

Diffusion Probabilistic Models (Score-based Generative Models)

Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al.,2021

Forward SDE (data → noise)

$$\mathbf{x}(0) \longrightarrow \mathrm{d}\boldsymbol{x}_t = f(t)\boldsymbol{x}_t \mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}_t, \longrightarrow \mathbf{x}(T)$$

$$\mathbf{x}(0) \longleftarrow \mathrm{d}\boldsymbol{x}_t = \left[f(t)\boldsymbol{x}_t + \frac{g^2(t)}{\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) \right] \mathrm{d}t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t, \longleftarrow \mathbf{x}(T)$$

Reverse SDE (noise → data)

- Transition is a linear Gaussian:

$$q_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\alpha(t)\boldsymbol{x}_0, \sigma^2(t)\boldsymbol{I}),$$

- Training by denoising:

$$\frac{1}{2}\int_0^T \omega(t)\mathbb{E}_{q_0(\boldsymbol{x}_0)}\mathbb{E}_{q(\boldsymbol{\epsilon})}\left[\|\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}\|_2^2\right]\mathrm{d}t$$

Continuous Perspective: “Equivalent ODE” of an SDE

Same marginal distribution (Song et al.,2021)

Proposition. (Song, et al, 2021)

Starting from the distribution $q_0(x_0)$, define the distribution through the following SDE at time t as $q_t(x_t)$:

$$\mathrm{d}\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}_t$$

Then the following ODE has the same **marginal** distribution $q_t(x_t)$ at each time t :

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{x}_t, t) - \frac{1}{2}g(t)^2 \boxed{\nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x}_t)}$$

Two Types of Diffusion Probabilistic Models

Diffusion SDEs and Diffusion ODEs (Song et al.,2021)

- **Diffusion SDEs:**

$$\mathrm{d}\boldsymbol{x}_t = \left[f(t)\boldsymbol{x}_t + \frac{g^2(t)}{\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) \right] \mathrm{d}t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t, \quad \boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \tilde{\sigma}^2 \boldsymbol{I}).$$

Sampling by **DDPM** is equivalent to a **first-order** discretization of diffusion SDEs (Song et al.,2021).
Sampling by **Analytic-DPM** is also equivalent to discretization of diffusion SDEs.

- **Diffusion ODEs:**

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = f(t)\boldsymbol{x}_t + \frac{g^2(t)}{2\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t), \quad \boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \tilde{\sigma}^2 \boldsymbol{I})$$

Deterministic(No more noises, Generally faster than SDE);
Invertible(The encoded noise can be used for downstream tasks, such as inpainting);

Sampling by **DDIM** is equivalent to a **first-order** discretization of diffusion ODEs (Salimans et al., 2022).

Slow Sampling Speed is (one of) the Most Critical Issues of DPMs

Usually needs at least 100 sequential steps to converge

- Sampling Trajectory of DPMs: gradually denoising from a Gaussian noise.

- Sampling from DPMs need to **discretize diffusion SDEs / ODEs**, which needs SDE / ODE solvers. Generally, ODE solvers converge faster than SDE solvers.
- However, they both need at least **100** sequential steps to converge.

$$\mathrm{d}\boldsymbol{x}_t = \left[f(t)\boldsymbol{x}_t + \frac{g^2(t)}{\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right]\mathrm{d}t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t,$$

Diffusion SDE -> SDE solver (**200~1000 steps**)

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = f(t)\boldsymbol{x}_t + \frac{g^2(t)}{2\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$$

Diffusion ODE -> ODE solver (**~100 steps**)

DPM-Solver: A Training-Free Fast ODE Solver for DPMs

Sampling by DPM-Solver needs only 10~20 steps, without any further training

NFE = 10 | NFE = 15 | NFE = 20 | NFE = 100 | NFE = 10

(a) DDIM [19] (b) DPM-Solver (ours)

Solving Diffusion ODEs by Traditional Runge-Kutta Methods

The "black-box" assumption ignores known information of diffusion ODEs

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = f(t)\boldsymbol{x}_t + \frac{g^2(t)}{2\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$$

Given an initial value x_s at time s, the exact solution x_t at time t satisfies:

$$\boldsymbol{x}_t = \boldsymbol{x}_s + \underbrace{\int_s^t \left(f(\tau)\boldsymbol{x}_\tau + \frac{g^2(\tau)}{2\sigma_\tau}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\tau, \tau) \right) \mathrm{d}\tau}$$

A "**black-box**" $\boldsymbol{h}_\theta(\boldsymbol{x}_t, t)$

losses known information of $f(t)$ and $g(t)$.

Traditional Runge-Kutta methods (RK45) **cannot converge for < 20 steps**. (Song et al.,2021)

Observation 1: Exactly Computing the Linear Part

Diffusion ODE has a semi-linear structure

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \boxed{f(t)\boldsymbol{x}_t} + \frac{g^2(t)}{2\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$$

Linear function

↓ "*variation of constants*" formula

The exact solution x_t at time t :

$$\boldsymbol{x}_t = \boxed{e^{\int_s^t f(\tau)\mathrm{d}\tau}\boldsymbol{x}_s} + \int_s^t \left(e^{\int_\tau^t f(r)\mathrm{d}r}\frac{g^2(\tau)}{2\sigma_\tau}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\tau, \tau)\right)\mathrm{d}\tau$$

Exactly Computed

Observation 2: Simplifying by log-SNR

A simple exponentially weighted integral

$$q_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\alpha(t)\boldsymbol{x}_0, \sigma^2(t)\boldsymbol{I}),$$

signal-to-noise-ratio **(SNR)**: α_t^2/σ_t^2

Define $\lambda_t := \log(\alpha_t/\sigma_t)$ (half of log-SNR)

We can prove that:

$$f(t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}$$

$$g^2(t) = -2\sigma_t^2\frac{\mathrm{d}\lambda_t}{\mathrm{d}t}$$

$$\boldsymbol{x}_t = e^{\int_s^t f(\tau)\mathrm{d}\tau}\boldsymbol{x}_s + \int_s^t \left(e^{\int_\tau^t f(r)\mathrm{d}r}\frac{g^2(\tau)}{2\sigma_\tau}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\tau, \tau)\right)\mathrm{d}\tau$$

"*change-of-variable*" formula (from t to λ)

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

Linear term
Exactly Computed

Nonlinear term
Exponentially weighted integral

Summary: Exact Solutions of Diffusion ODEs

A very simple formulation

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

Linear term — **Exactly Computed**

Nonlinear term — **Exponentially weighted integral**

All we need to do is to approximate the exponentially weighted integral.

Designing High-Order Solvers for Diffusion ODEs

Foundation of DPM-Solver

Based on our analysis, given $\tilde{x}_{t_{i-1}}$ at time t_{i-1}, the exact solution at time t_i is:

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

Taylor expansion:

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

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

Derivatives **Coefficients**

Observation 3: Exactly Computing the Coefficients

By integration-by-parts

Because of the **change-of-variable for λ**, the coefficients can be **analytically computed**:

$$\begin{aligned}
\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda} \frac{(\lambda - \lambda_{t_{i-1}})^n}{n!} \mathrm{d}\lambda &= -\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} \frac{(\lambda - \lambda_{t_{i-1}})^n}{n!} \mathrm{d}(e^{-\lambda}) \\
&= \left(-\frac{(\lambda - \lambda_{t_{i-1}})^n}{n!} e^{-\lambda}\right)\Bigg|_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} + \int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda} \frac{(\lambda - \lambda_{t_{i-1}})^{n-1}}{(n-1)!} \mathrm{d}\lambda \\
&= \cdots
\end{aligned}$$

Repeatedly applying n times of integration-by-parts

Observation 4: Approximating Derivatives without Autograd

A classical way for designing high-order ODE solvers

The high-order derivatives can be approximated by traditional numerical methods, which are similar to designing traditional ODE solvers.

E.g. for the first-order derivative, we can use some intermediate point or previous point x_{s_i}:

$$\hat{\boldsymbol{\epsilon}}_\theta^{(1)}(\hat{\boldsymbol{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}}) \approx \boxed{\frac{\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_{\lambda_{s_i}}, \lambda_{s_i}) - \hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}})}{\lambda_{s_i} - \lambda_{t_{i-1}}}}$$

Only function evaluation for $\hat{\epsilon}_\theta$,
without applying autograd.

DPM-Solver: Customized Solver for Diffusion ODEs

Reduce the discretization error as much as possible

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

Linear term	Derivatives	Coefficients	High-order errors
Exactly Computed	Approximated	Exactly computed	Omitted
(Observation 1)	(Observation 4)	(Observation 2 & 3)	

We only approximate the terms about the neural network, and exactly compute all of the other terms.

DDIM is the first-order DPM-Solver

That's why DDIM works well

For $k = 2$:

$$\boldsymbol{x}_{t_{i-1}\to t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\mathrm{d}\lambda + \mathcal{O}(h_i^2)$$

$$= \underbrace{\frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}(e^{h_i} - 1)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})} + \mathcal{O}(h_i^2).$$

Denoising Diffusion Implicit Model (**DDIM**, Song et al., 2021)

Therefore, DDIM is the first-order diffusion ODE solver which analytically computes the known terms.

DPM-Solver-2 and DPM-Solver-3

DPM-Solver is the high-order generalization of DDIM

Algorithm 1 DPM-Solver-2.

Require: initial value $\boldsymbol{x}_T$, time steps $\{t_i\}_{i=0}^M$, model $\boldsymbol{\epsilon}_\theta$

$\tilde{\boldsymbol{x}}_{t_0} \leftarrow \boldsymbol{x}_T$
for $i \leftarrow 1$ to M **do**
  $s_i \leftarrow t_\lambda\left(\frac{\lambda_{t_{i-1}}+\lambda_{t_i}}{2}\right)$
  $\boldsymbol{u}_i \leftarrow \frac{\alpha_{s_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_i}\left(e^{\frac{h_i}{2}}-1\right)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$
  $\tilde{\boldsymbol{x}}_{t_i} \leftarrow \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}\left(e^{h_i}-1\right)\boldsymbol{\epsilon}_\theta(\boldsymbol{u}_i, s_i)$
end for
return $\tilde{\boldsymbol{x}}_{t_M}$

Algorithm 2 DPM-Solver-3.

Require: initial value $\boldsymbol{x}_T$, time steps $\{t_i\}_{i=0}^M$, model $\boldsymbol{\epsilon}_\theta$

$\tilde{\boldsymbol{x}}_{t_0} \leftarrow \boldsymbol{x}_T$, $r_1 \leftarrow \frac{1}{3}$, $r_2 \leftarrow \frac{2}{3}$
for $i \leftarrow 1$ to M **do**
  $s_{2i-1} \leftarrow t_\lambda\left(\lambda_{t_{i-1}} + r_1 h_i\right)$, $s_{2i} \leftarrow t_\lambda\left(\lambda_{t_{i-1}} + r_2 h_i\right)$
  $\boldsymbol{u}_{2i-1} \leftarrow \frac{\alpha_{s_{2i-1}}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_{2i-1}}\left(e^{r_1 h_i}-1\right)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$
  $\boldsymbol{D}_{2i-1} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{u}_{2i-1}, s_{2i-1}) - \boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$
  $\boldsymbol{u}_{2i} \leftarrow \frac{\alpha_{s_{2i}}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_{2i}}\left(e^{r_2 h_i}-1\right)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}) - \frac{\sigma_{s_{2i}} r_2}{r_1}\left(\frac{e^{r_2 h_i}-1}{r_2 h_i}-1\right)\boldsymbol{D}_{2i-1}$
  $\boldsymbol{D}_{2i} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{u}_{2i}, s_{2i}) - \boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$
  $\tilde{\boldsymbol{x}}_{t_i} \leftarrow \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}\left(e^{h_i}-1\right)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}) - \frac{\sigma_{t_i}}{r_2}\left(\frac{e^{h_i}-1}{h}-1\right)\boldsymbol{D}_{2i}$
end for
return $\tilde{\boldsymbol{x}}_{t_M}$

Comparison with Traditional Runge-Kutta Methods

DPM-Solver is customized for DPMS

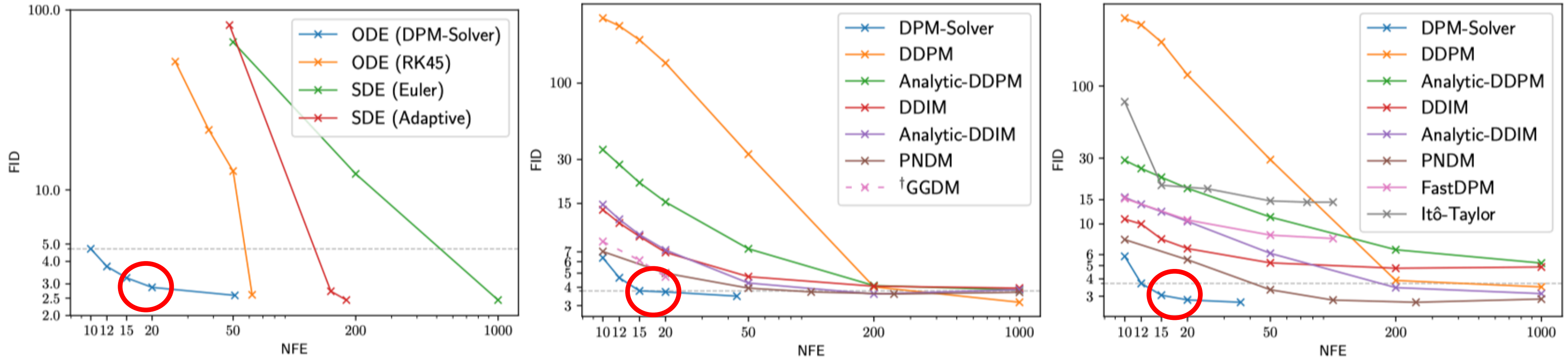
Table 1: FID ↓ on CIFAR-10 for different orders of Runge-Kutta (RK) methods and DPM-Solvers, varying the number of function evaluations (NFE). For RK methods, we evaluate diffusion ODEs w.r.t. both t (Eq. (2.7)) and λ (Eq. (E.1)). We use uniform step size in t for RK (t), and uniform step size in λ for RK (λ) and DPM-Solvers.

Sampling method \ NFE	12	18	24	30	36	42	48
RK2 (t)	16.40	7.25	3.90	3.63	3.58	3.59	3.54
RK2 (λ)	107.81	42.04	17.71	7.65	4.62	3.58	3.17
DPM-Solver-2	**5.28**	**3.43**	**3.02**	**2.85**	**2.78**	**2.72**	**2.69**
RK3 (t)	48.75	21.86	10.90	6.96	5.22	4.56	4.12
RK3 (λ)	34.29	4.90	3.50	3.03	2.85	2.74	2.69
DPM-Solver-3	**6.03**	**2.90**	**2.75**	**2.70**	**2.67**	**2.65**	**2.65**

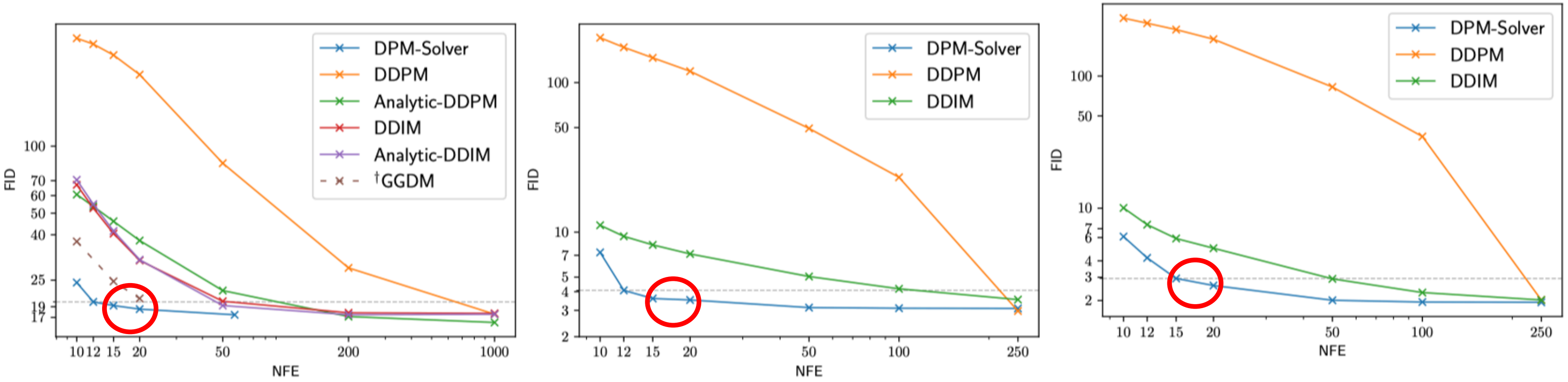
Experiments: SOTA Acceleration for Sampling of DPMs

Almost converges in 15~20 steps

Sample FID ↓ , varying number of function evaluations (NFE).



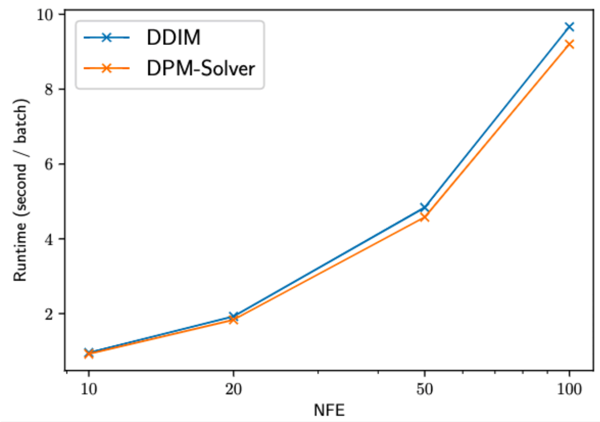
(a) CIFAR-10 (continuous) (b) CIFAR-10 (discrete) (c) CelebA 64x64 (discrete)



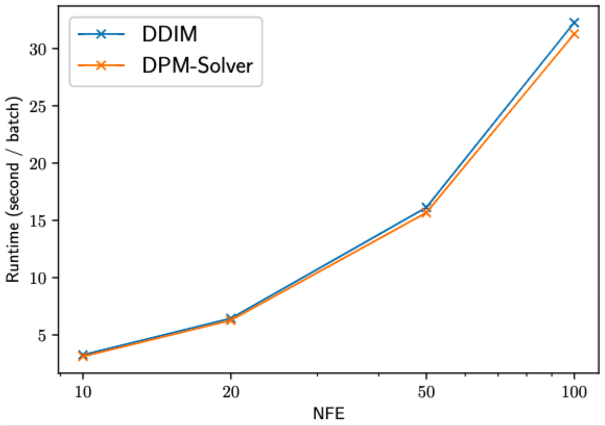
(d) ImageNet 64x64 (discrete) (e) ImageNet 128x128 (discrete) (f) LSUN bedroom 256x256 (discrete)

Additional Computation Costs are Neglectable

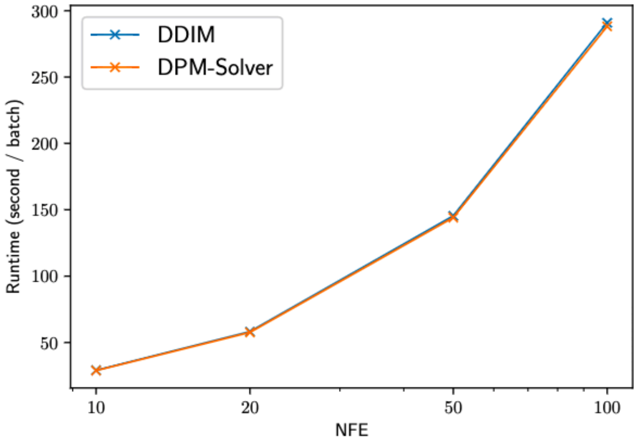
Because we analytically computes all the known terms



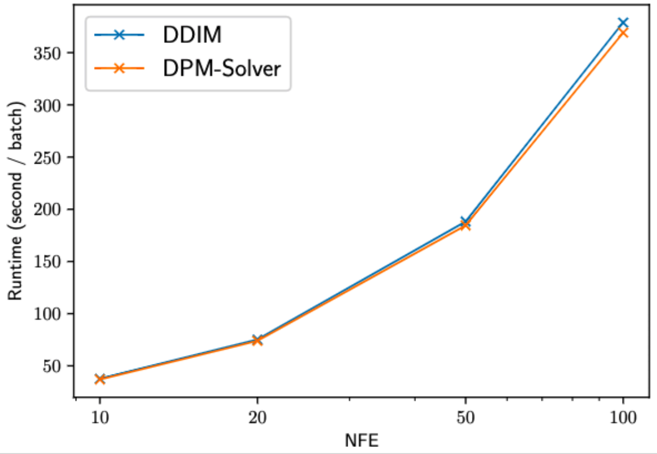
CIFAR10 32x32



CelebA 64x64



ImageNet 128x128



LSUN bedroom 256x256

Runtime (second / batch) on a single GPU, varying different NFEs.

Under the same NFE, The computation costs of DDIM and DPM-Solver are almost the same.
(Our implementation is even slightly faster than the original DDIM.)

DPM-Solver++: Fast Solver for Guided Sampling of Diffusion Probabilistic Models

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, Jun Zhu

Tsinghua University

(Conditional) Guided Sampling by DPMs

Only need to modify the noise prediction model

Classifier guidance:

$$\tilde{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_t, t, c) := \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) - \boxed{s} \cdot \sigma_t \nabla_{\boldsymbol{x}_t} \underbrace{\log p_\phi(c|\boldsymbol{x}_t, t)}_{\text{Classifier}}$$

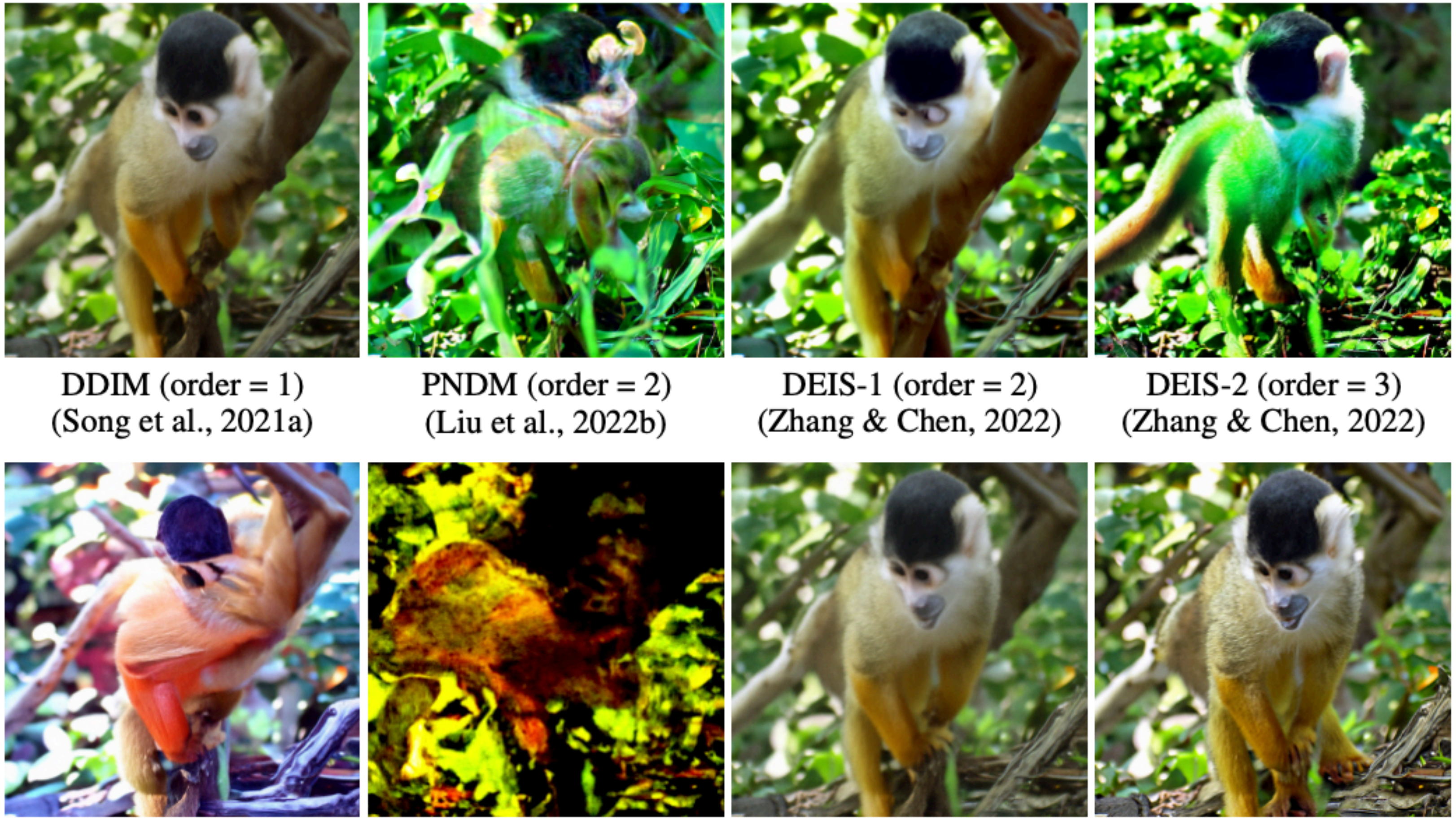
Classifier-free guidance:

$$\tilde{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_t, t, c) := \boxed{s} \cdot \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t, c) + (1 - s) \cdot \underbrace{\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t, \varnothing)}_{\text{Unconditional model}}$$

The **guidance scale** ***s*** is usually **large** for improving the condition-sample alignment.

Challenges for Guided Sampling with Large Guidance Scale

Challenge: unstable high-order solvers



DDIM (order = 1) (Song et al., 2021a)

PNDM (order = 2) (Liu et al., 2022b)

DEIS-1 (order = 2) (Zhang & Chen, 2022)

DEIS-2 (order = 3) (Zhang & Chen, 2022)

DPM-Solver (order = 2) (Lu et al., 2022)

DPM-Solver-3 (order = 3) (Lu et al., 2022)

†DDIM (thresholding) (Saharia et al., 2022b)

DPM-Solver++ (order = 2) (**ours**)

ImageNet 256x256.

Guidance scale is 8.0.

15 function evaluations.

DPM-Solver++: DPM-Solver for Data Prediction Model

Foundation of DPM-Solver++

$$\boldsymbol{x}_\theta(\boldsymbol{x}_t, t) := (\boldsymbol{x}_t - \sigma_t \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t))/\alpha_t$$

Given $\tilde{x}_{t_{i-1}}$ at time t_{i-1}, the exact solution at time t_i is:

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

Taylor expansion:

$$\tilde{\boldsymbol{x}}_{t_i} = \frac{\sigma_{t_i}}{\sigma_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} + \sigma_{t_i}\sum_{n=0}^{k-1}\underbrace{\boldsymbol{x}_\theta^{(n)}(\hat{\boldsymbol{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}})}_{\text{estimated}}\underbrace{\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{\lambda}\frac{(\lambda-\lambda_{t_{i-1}})^n}{n!}\mathrm{d}\lambda}_{\text{analytically computed (Appendix A)}} + \underbrace{\mathcal{O}(h_i^{k+1})}_{\text{omitted}}$$

Single-Step and Multi-Step Solvers

We provide two types solvers

Algorithm 1 DPM-Solver++(2S).

Require: initial value $\boldsymbol{x}_T$, time steps $\{t_i\}_{i=0}^{M}$ and $\{s_i\}_{i=1}^{M}$, data prediction model $\boldsymbol{x}_\theta$.

$\tilde{\boldsymbol{x}}_{t_0} \leftarrow \boldsymbol{x}_T$.
for $i \leftarrow 1$ to M **do**
  $h_i \leftarrow \lambda_{t_i} - \lambda_{t_{i-1}}$
  $r_i \leftarrow \frac{\lambda_{s_i} - \lambda_{t_{i-1}}}{h_i}$
  $\boldsymbol{u}_i \leftarrow \frac{\sigma_{s_i}}{\sigma_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{s_i}\left(e^{-r_i h_i} - 1\right)\boldsymbol{x}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$
  $\boldsymbol{D}_i \leftarrow (1 - \frac{1}{2r_i})\boldsymbol{x}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_i) + \frac{1}{2r_i}\boldsymbol{x}_\theta(\boldsymbol{u}_i, s_i)$
  $\tilde{\boldsymbol{x}}_{t_i} \leftarrow \frac{\sigma_{t_i}}{\sigma_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\left(e^{-h_i} - 1\right)\boldsymbol{D}_i$
end for
return $\tilde{\boldsymbol{x}}_{t_M}$

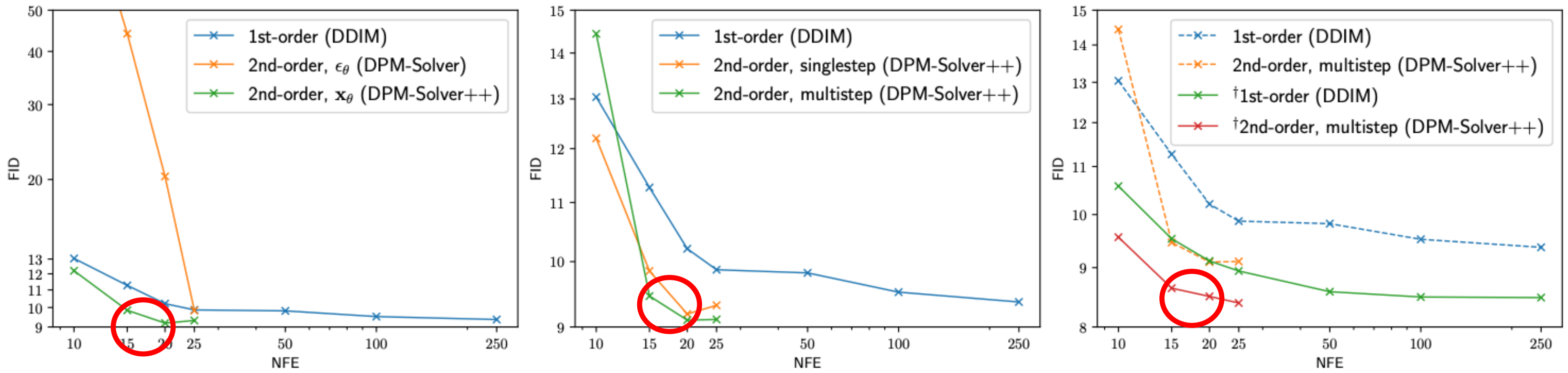
Algorithm 2 DPM-Solver++(2M).

Require: initial value $\boldsymbol{x}_T$, time steps $\{t_i\}_{i=0}^{M}$, data prediction model $\boldsymbol{x}_\theta$.

Denote $h_i := \lambda_{t_i} - \lambda_{t_{i-1}}$ for $i = 1, \ldots, M$.
$\tilde{\boldsymbol{x}}_{t_0} \leftarrow \boldsymbol{x}_T$. Initialize an empty buffer Q.
$Q \xleftarrow{\text{buffer}} \boldsymbol{x}_\theta(\tilde{\boldsymbol{x}}_{t_0}, t_0)$
$\tilde{\boldsymbol{x}}_{t_1} \leftarrow \frac{\sigma_{t_1}}{\sigma_{t_0}}\tilde{\boldsymbol{x}}_0 - \alpha_{t_1}\left(e^{-h_1} - 1\right)\boldsymbol{x}_\theta(\tilde{\boldsymbol{x}}_{t_0}, t_0)$
$Q \xleftarrow{\text{buffer}} \boldsymbol{x}_\theta(\tilde{\boldsymbol{x}}_{t_1}, t_1)$
for $i \leftarrow 2$ to M **do**
  $r_i \leftarrow \frac{h_{i-1}}{h_i}$
  $\boldsymbol{D}_i \leftarrow \left(1 + \frac{1}{2r_i}\right)\boldsymbol{x}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}) - \frac{1}{2r_i}\boldsymbol{x}_\theta(\tilde{\boldsymbol{x}}_{t_{i-2}}, t_{i-2})$
  $\tilde{\boldsymbol{x}}_{t_i} \leftarrow \frac{\sigma_{t_i}}{\sigma_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\left(e^{-h_i} - 1\right)\boldsymbol{D}_i$
  If $i < M$, then $Q \xleftarrow{\text{buffer}} \boldsymbol{x}_\theta(\tilde{\boldsymbol{x}}_{t_i}, t_i)$
end for
return $\tilde{\boldsymbol{x}}_{t_M}$

Ablation Study

DPM-Solver++ can greatly improve the sample quality for large guidance scale



(a) From ϵ_θ to $\boldsymbol{x}_\theta$. (b) From singlestep to multistep. (c) Thresholding.

DPM-Solver-2 (ϵ_θ, singelstep) | DPM-Solver++(2S) ($\boldsymbol{x}_\theta$, singlestep) | DPM-Solver++(2M) ($\boldsymbol{x}_\theta$, multistep, thresholdin)

Example: Stable-Diffusion with DPM-Solver++

Example: Stable-Diffusion with DPM-Solver++

Steps 10 15 20 25 50 999

DDIM

PLMS (PNDM)

DPM-Solver

Example: Image Editing by Stable-Diffusion with DPM-Solver++

In only 20 steps

In only **20** steps

Source image: “A bowl of fruits”

Target text: “A bowl of **pears**”

DPM-Solver is Easy to Use

Official code example: discrete-time DPMs

We support both **continuous-time** and **discrete-time** DPMs. Here we take an example for discrete-time DPMs.
Code is released at: https://github.com/LuChengTHU/dpm-solver (**Github Stars: 600+**)

1. Define noise schedule by the discrete β_i (defined by the training process of the discrete-time DPM) .

```
ns = NoiseScheduleVP('discrete', betas=betas)
```

2. Define DPM-Solver by noise prediction model and noise schedule.

```
dpm_solver = DPM_Solver(model_fn, ns)
```

3. Sample by DPM-Solver.

```
x = dpm_solver.sample(x_T, steps=20, order=2, method="multistep", skip_type="time_uniform")
```

Supported Model Types

https://github.com/LuChengTHU/dpm-solver

We support the following four types of diffusion models. You can set the model type by the argument `model_type` in the function `model_wrapper`.

Model Type	Training Objective	Example Paper
"noise": noise prediction model ϵ_θ	$E_{x0,\epsilon,t}\left[\omega_1(t)\|\|\epsilon_\theta(x_t,t)-\epsilon\|\|_2^2\right]$	DDPM, Stable-Diffusion
"x_start": data prediction model x_θ	$E_{x0,\epsilon,t}\left[\omega_2(t)\|\|x_\theta(x_t,t)-x_0\|\|_2^2\right]$	DALL·E 2
"v": velocity prediction model v_θ	$E_{x0,\epsilon,t}\left[\omega_3(t)\|\|v_\theta(x_t,t)-(\alpha_t\epsilon-\sigma_t x_0)\|\|_2^2\right]$	Imagen Video
"score": marginal score function s_θ	$E_{x0,\epsilon,t}\left[\omega_4(t)\|\|\sigma_t s_\theta(x_t,t)+\epsilon\|\|_2^2\right]$	ScoreSDE

Supported Sampling Types

https://github.com/LuChengTHU/dpm-solver

We support the following three types of sampling by diffusion models. You can set the argument `guidance_type` in the function `model_wrapper`.

Sampling Type	Equation for Noise Prediction Model	Example Paper
"uncond": unconditional sampling	$\tilde{\epsilon}_\theta(x_t, t) = \epsilon_\theta(x_t, t)$	DDPM
"classifier": classifier guidance	$\tilde{\epsilon}_\theta(x_t, t, c) = \epsilon_\theta(x_t, t) - s \cdot \sigma_t \nabla_{x_t} \log q_\phi(x_t, t, c)$	ADM, GLIDE
"classifier-free": classifier-free guidance	$\tilde{\epsilon}_\theta(x_t, t, c) = s \cdot \epsilon_\theta(x_t, t, c) + (1 - s) \cdot \epsilon_\theta(x_t, t)$	DALL·E 2, Imagen, Stable-Diffusion

Supported Algorithms

https://github.com/LuChengTHU/dpm-solver

Method	Supported Orders	Supporting Thresholding	Remark
DPM-Solver, singlestep	1, 2, 3	No	Recommended for **unconditional sampling** (with order = 3). See this paper.
DPM-Solver, multistep	1, 2, 3	No	
DPM-Solver++, singlestep	1, 2, 3	Yes	
DPM-Solver++, multistep	1, 2, 3	Yes	Recommended for **guided sampling** (with order = 2). See this paper.

DPM-Solver in Diffusers Library

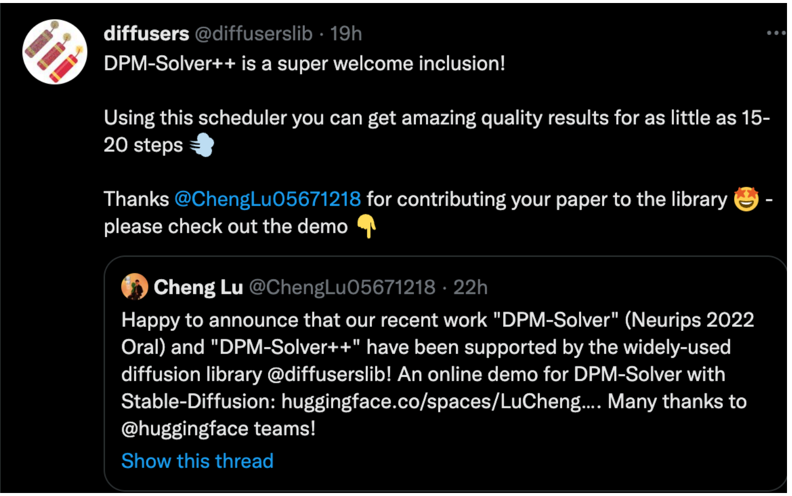
Very easy to use in stable-diffusion

```python
from diffusers import StableDiffusionPipeline
from diffusers import DPMSolverMultistepScheduler



steps = 10

scheduler = DPMSolverMultistepScheduler.from_config("./stable-diffusion-v1-5", subfolder="scheduler")


pipe = StableDiffusionPipeline.from_pretrained(
    "./stable-diffusion-v1-5",
    scheduler=scheduler,
)
pipe = pipe.to("cuda")

prompt = "a photo of an astronaut riding a horse on mars"

images = pipe(prompt, num_inference_steps=steps, num_images_per_prompt=5).images

for i, image in enumerate(images):
    image.save(f"dpm_{steps}_{i}.png")

```

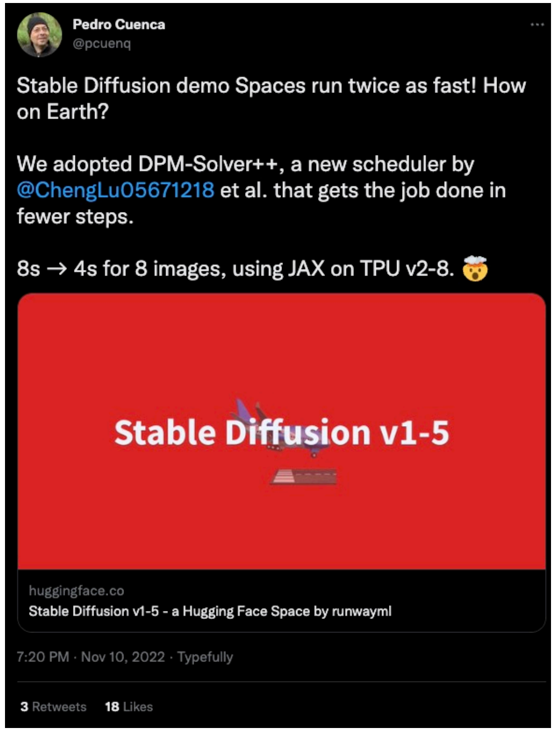
High Impact of DPM-Solver

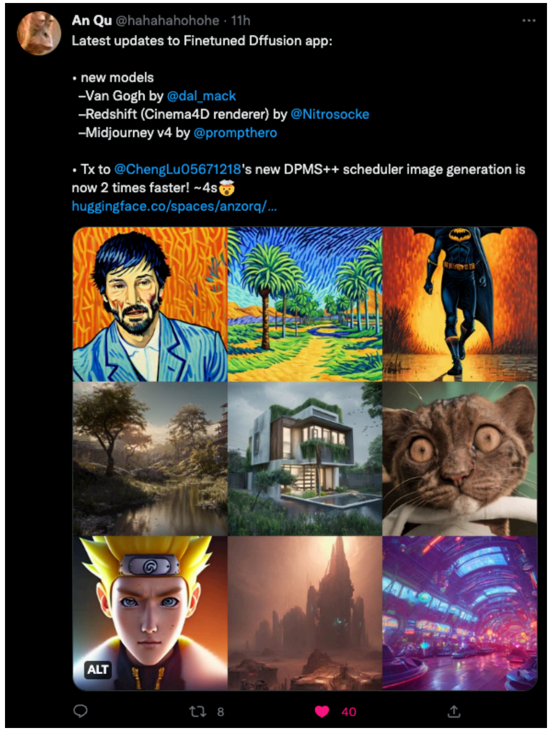
DPM-Solver has addressed more and more attentions

Diffusers official account:



Official Stable Diffusion Demos (both v1 and v2).
(SDM and Finetuned-SDM)





Stable-diffusion-WebUI for:

(the name with "DPM" or "DPM++")

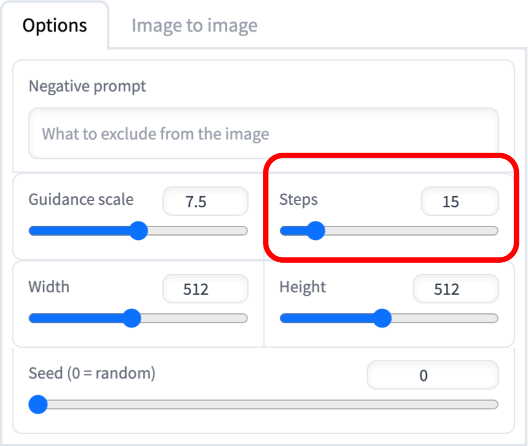
DreamStudio
beta

STABLEBOOST

Online Demo for Stable-Diffusion with DPM-Solver

https://huggingface.co/spaces/LuChengTHU/dpmsolver_sdm

- DPM-Solver can generate high-quality samples within only **20-25** steps, and for some samples even within **10-15** steps.



Summary

- We propose a highly simplified formulation of the exact solutions of diffusion ODEs.
- We propose a customized solver for diffusion ODEs, which can generate high-quality samples in around **10** steps and almost converge in **20** steps.
- Code is released at: https://github.com/LuChengTHU/dpm-solver
- A Chinese tutorial for diffusion models on Zhihu: