DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps

NeurIPS 2022 Oral

Cheng Lu†, Yuhao Zhou†, Fan Bao†, Jianfei Chen†, Chongxuan Li‡, Jun Zhu†
†Tsinghua University
‡Renmin University

Ma Yiyang
2023/01/29

Content

- Authors
- Background
- Algorithm
- Results

Content

- Authors
- Background
- Algorithm
- Results

Content

- Authors
- Background
- Algorithm
- Results

Background

First, we review the algorithm of DDPM.

It consists of two processes: *forward* and *reverse*.

- *forward* process:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad (1)$$

- *reverse* process:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I}) \quad (2)$$

where

$$\tilde{\beta}_t = 1/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = (\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}\mathbf{x}_0)/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 \quad (3)$$

Background

The two processes are shown in the discrete way.

For simplicity, we rewrite the forward process as:

$$q_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\alpha(t)\boldsymbol{x}_0, \sigma^2(t)\boldsymbol{I}) \tag{4}$$

*Yang Song** proved that the forward process can be described in the continuous way based on SDE (stochastic differential equation):

$$\mathrm{d}\boldsymbol{x}_t = f(t)\boldsymbol{x}_t\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}_t, \quad \boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0) \tag{5}$$

where

$$f(t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}, \quad g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2 \tag{6}$$

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations”, in ICLR 2021.

Background

The reverse process via SDE is given as:

$$\mathrm{d}\boldsymbol{x}_t = \left[f(t)\boldsymbol{x}_t + \frac{g^2(t)}{\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) \right] \mathrm{d}t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t, \quad \boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \tilde{\sigma}^2 \boldsymbol{I}) \tag{7}$$

Yang Song also proved that we can depict the reverse process of DDPM via ODE by analyzing the marginal distribution at each time *t* of SDE.

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \boldsymbol{h}_\theta(\boldsymbol{x}_t, t) := f(t)\boldsymbol{x}_t + \frac{g^2(t)}{2\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t), \quad \boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \tilde{\sigma}^2 \boldsymbol{I}) \tag{8}$$

When discretizing SDEs, the step size is limited by the randomness of the Wiener process.

For sampling with fewer steps, we consider the associated ODE.

This paper proposed the k-th-order solution of Eqn. 8 and a method of sampling DDPM with ODE.

Background

It should be noticed that the work of analyzing and solving SDE is *Analytic DPM*, which is the outstanding paper of ICLR 22'. This work is also done by the group of *Jun Zhu*.

Content

- Authors
- Background
- Algorithm
- Results

Algorithm

The diffusion ODE is shown below:

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \boldsymbol{h}_\theta(\boldsymbol{x}_t, t) := f(t)\boldsymbol{x}_t + \frac{g^2(t)}{2\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t), \quad \boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \tilde{\sigma}^2\boldsymbol{I}) \tag{8}$$

which a semi-linear ODE. This kind of ODEs can be solved by *variation of constants*. The solution is shown below:

$$\boldsymbol{x}_t = e^{\int_s^t f(\tau)\mathrm{d}\tau}\boldsymbol{x}_s + \int_s^t \left(e^{\int_\tau^t f(r)\mathrm{d}r} \frac{g^2(\tau)}{2\sigma_\tau}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\tau, \tau) \right) \mathrm{d}\tau \tag{9}$$

where *s>t*.

Algorithm

We propose:

$$g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2 = 2\sigma_t^2\left(\frac{\mathrm{d}\log\sigma_t}{\mathrm{d}t} - \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\right) = -2\sigma_t^2\frac{\mathrm{d}\lambda_t}{\mathrm{d}t} \tag{10}$$

where $\lambda_t := \log(\alpha_t/\sigma_t)$

The Eqn. 9 can be simplified as:

$$\boldsymbol{x}_t = \frac{\alpha_t}{\alpha_s}\boldsymbol{x}_s - \alpha_t\int_s^t\left(\frac{\mathrm{d}\lambda_\tau}{\mathrm{d}\tau}\right)\frac{\sigma_\tau}{\alpha_\tau}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\tau,\tau)\mathrm{d}\tau \tag{11}$$

By replacing the integrand variable from τ to λ, we can get:

$$\boldsymbol{x}_t = \frac{\alpha_t}{\alpha_s}\boldsymbol{x}_s - \alpha_t\int_{\lambda_s}^{\lambda_t} e^{-\lambda}\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda,\lambda)\mathrm{d}\lambda \tag{12}$$

Algorithm

By replacing t and s, we can get:

$$\boldsymbol{x}_{t_{i-1}\to t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda, \lambda)\mathrm{d}\lambda \quad (13)$$

The $(k-1)$-th-order of Taylor Expansion of $\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda, \lambda)$ w.r.t λ at $\lambda_{t_{i-1}}$ is:

$$\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda, \lambda) = \sum_{n=0}^{k-1} \frac{(\lambda - \lambda_{t_{i-1}})^n}{n!}\hat{\boldsymbol{\epsilon}}_\theta^{(n)}(\hat{\boldsymbol{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}}) + \mathcal{O}((\lambda - \lambda_{t_{i-1}})^k) \quad (14)$$

Combining Eqn. 10 and 11, we can get:

$$\boldsymbol{x}_{t_{i-1}\to t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\sum_{n=0}^{k-1}\hat{\boldsymbol{\epsilon}}_\theta^{(n)}(\hat{\boldsymbol{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}})\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\frac{(\lambda - \lambda_{t_{i-1}})^n}{n!}\mathrm{d}\lambda + \mathcal{O}(h_i^{k+1}) \quad (15)$$

Algorithm

Considering $k = 1$:

$$\tilde{\boldsymbol{x}}_{t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}(e^{h_i} - 1)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}), \quad \text{where } h_i = \lambda_{t_i} - \lambda_{t_{i-1}} \tag{16}$$

We name it DPM-Solver-1. We can find out that this is the formulation of DDIM. In other words, DDIM is the $k - 1 = 0$-th-order Taylor Expansion of diffusion ODE. That is not a good estimation.

J. Song, C. Meng, and S. Ermon, “Denoising Diffusion Implicit Models,” in ICLR 2021.

Algorithm

Considering $k = 2$:

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**

$\quad s_i \leftarrow t_\lambda\left(\frac{\lambda_{t_{i-1}} + \lambda_{t_i}}{2}\right)$

$\quad \boldsymbol{u}_i \leftarrow \frac{\alpha_{s_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_i}\left(e^{\frac{h_i}{2}} - 1\right)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$

$\quad \tilde{\boldsymbol{x}}_{t_i} \leftarrow \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}\left(e^{h_i} - 1\right)\boldsymbol{\epsilon}_\theta(\boldsymbol{u}_i, s_i)$

end for

return $\tilde{\boldsymbol{x}}_{t_M}$

Algorithm

Considering $k = 3$:

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**

$s_{2i-1} \leftarrow t_\lambda\left(\lambda_{t_{i-1}} + r_1 h_i\right), \quad s_{2i} \leftarrow t_\lambda\left(\lambda_{t_{i-1}} + r_2 h_i\right)$

$\boldsymbol{u}_{2i-1} \leftarrow \frac{\alpha_{s_{2i-1}}}{\alpha_{t_{i-1}}} \tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_{2i-1}}\left(e^{r_1 h_i} - 1\right) \boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$

$\boldsymbol{D}_{2i-1} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{u}_{2i-1}, s_{2i-1}) - \boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$

$\boldsymbol{u}_{2i} \leftarrow \frac{\alpha_{s_{2i}}}{\alpha_{t_{i-1}}} \tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_{2i}}\left(e^{r_2 h_i} - 1\right) \boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}) - \frac{\sigma_{s_{2i}} r_2}{r_1}\left(\frac{e^{r_2 h_i} - 1}{r_2 h_i} - 1\right) \boldsymbol{D}_{2i-1}$

$\boldsymbol{D}_{2i} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{u}_{2i}, s_{2i}) - \boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$

$\tilde{\boldsymbol{x}}_{t_i} \leftarrow \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} \tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}\left(e^{h_i} - 1\right) \boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}) - \frac{\sigma_{t_i}}{r_2}\left(\frac{e^{h_i} - 1}{h} - 1\right) \boldsymbol{D}_{2i}$

end for

return $\tilde{\boldsymbol{x}}_{t_M}$

Algorithm

DPM-Solver-k takes k diffusion steps per time. How to deploy it to a sampling process?

- Indicate the number of steps NFE of the entire diffusion process.
- Apply DPM-Solver-3 as much as possible. If the NFE is not divisible by 3, add a single step of DPM-Solver-2 or DPM-Solver-1 (dependent on the remainder).

The sampling step schedule is uniformly split the interval of $[\lambda_T,\ \lambda_0]$, i.e.:

$$\lambda_{t_i} = \lambda_T + \frac{i}{M}(\lambda_0 - \lambda_T),\ i = 0, \ldots, M$$

which is different from the uniformly β in previous works.

Content

- Authors
- Background
- Algorithm
- Results

Results

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

(a) DDIM [19]

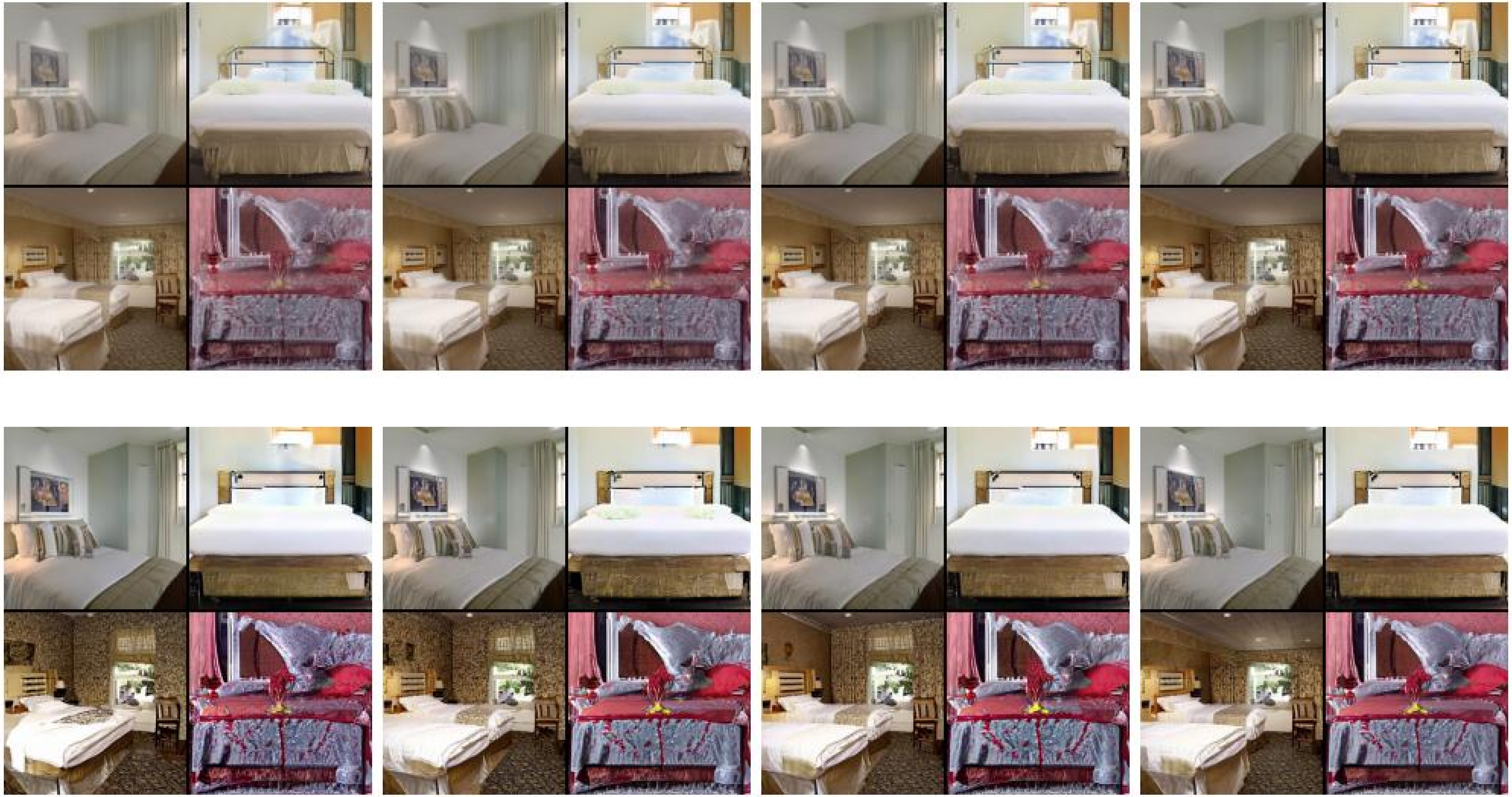
(b) DPM-Solver (ours)

Results

NFE = 10 | NFE = 12 | NFE = 15 | NFE = 20

DDIM [19]

DPM-Solver (ours)



Any drawbacks?

- DPM Solver performs quite bad when using classifier-free guidance with large scale factors.

Further work to solve the problem:

C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, “DPM-Solver++: Fast Solver for Guided Sampling of Diffusion Probabilistic Models”

Thanks for watching.

Ma Yiyang
2023/01/29