Auto-Regressive and Diffusion Probabilistic Models

Jun Zhu

The Institute for Artificial Intelligence
Department of Computer Science and Technology
Tsinghua University

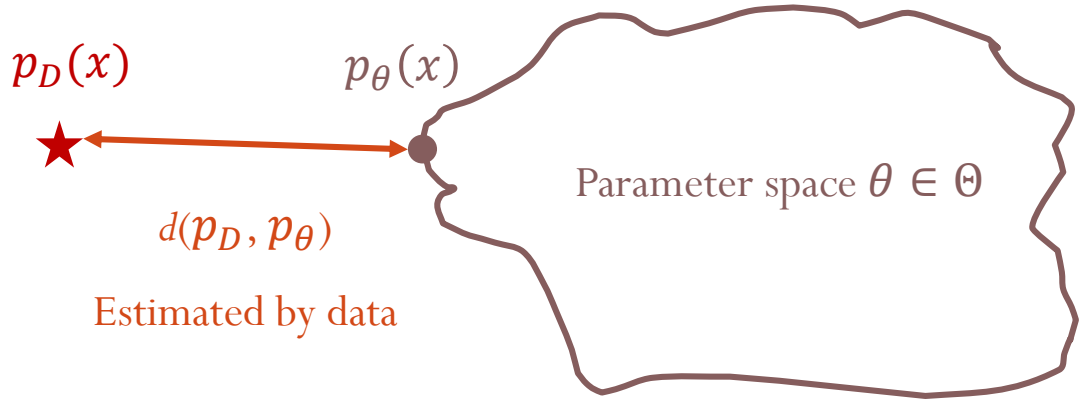
Basics of Generative Modeling

- Given a set of iid data from an unknown data distribution

$$x_i \sim_{iid} p_D(x), \qquad i = 1,2,3 \ldots$$

where $p_D(x)$ is the underlying distribution of the data.

- Learn a model distribution $p_\theta(x)$ that approximates the data distribution: $p_\theta(x) \approx p_D(x)$



Deep Generative Models with (Differentiable) Neural Networks

- **More flexible by using differential function mapping between random variables**

- If z is uniformly distributed over (0, 1), then $y = f(z)$ has the distribution

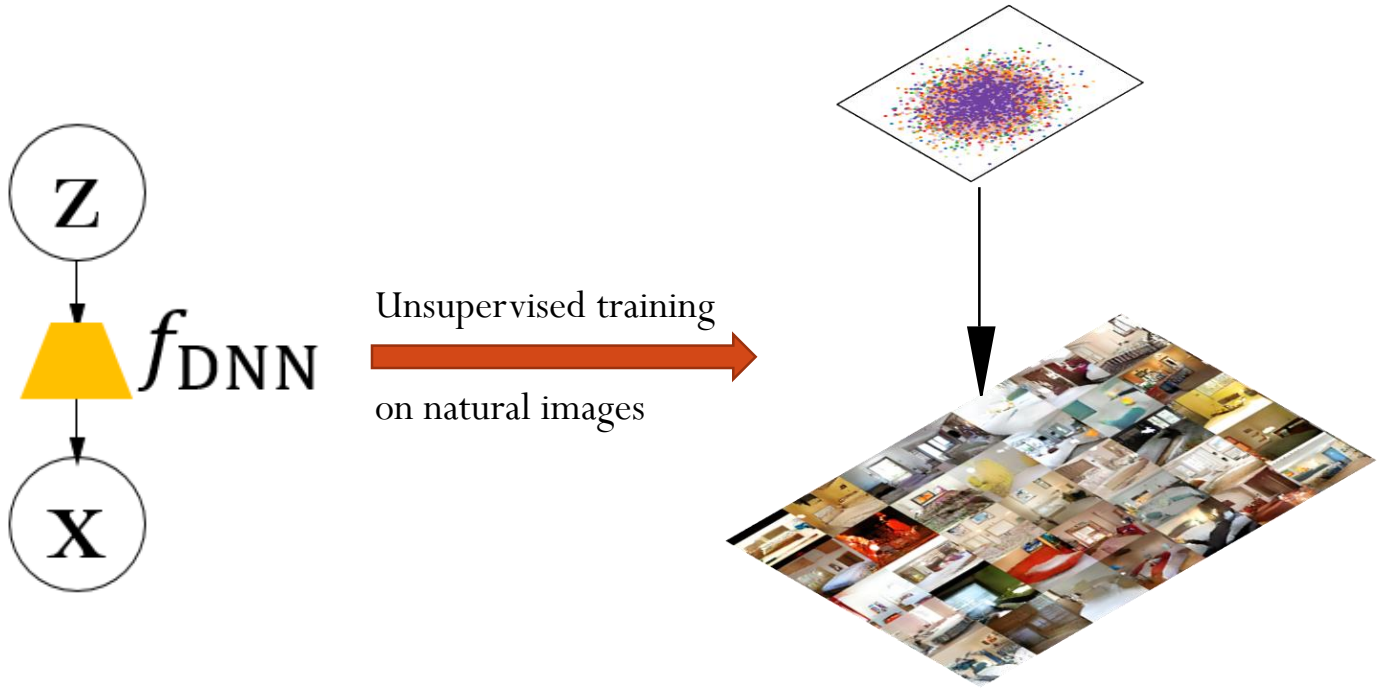
$$p(y) = p(z)\left|\frac{dz}{dy}\right|$$

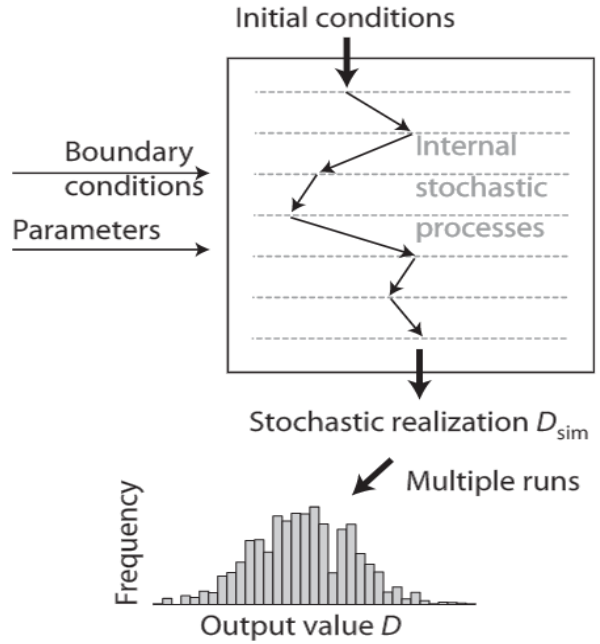
  - where $p(z) = 1$

- This trick is widely used to draw samples from exponential family distributions (e.g., Gaussian, Exponential)

Deep Generative Models with (Differentiable) Neural Networks

- Use (differentiable) DNNs to fit (learn) the complex relationships between random variables

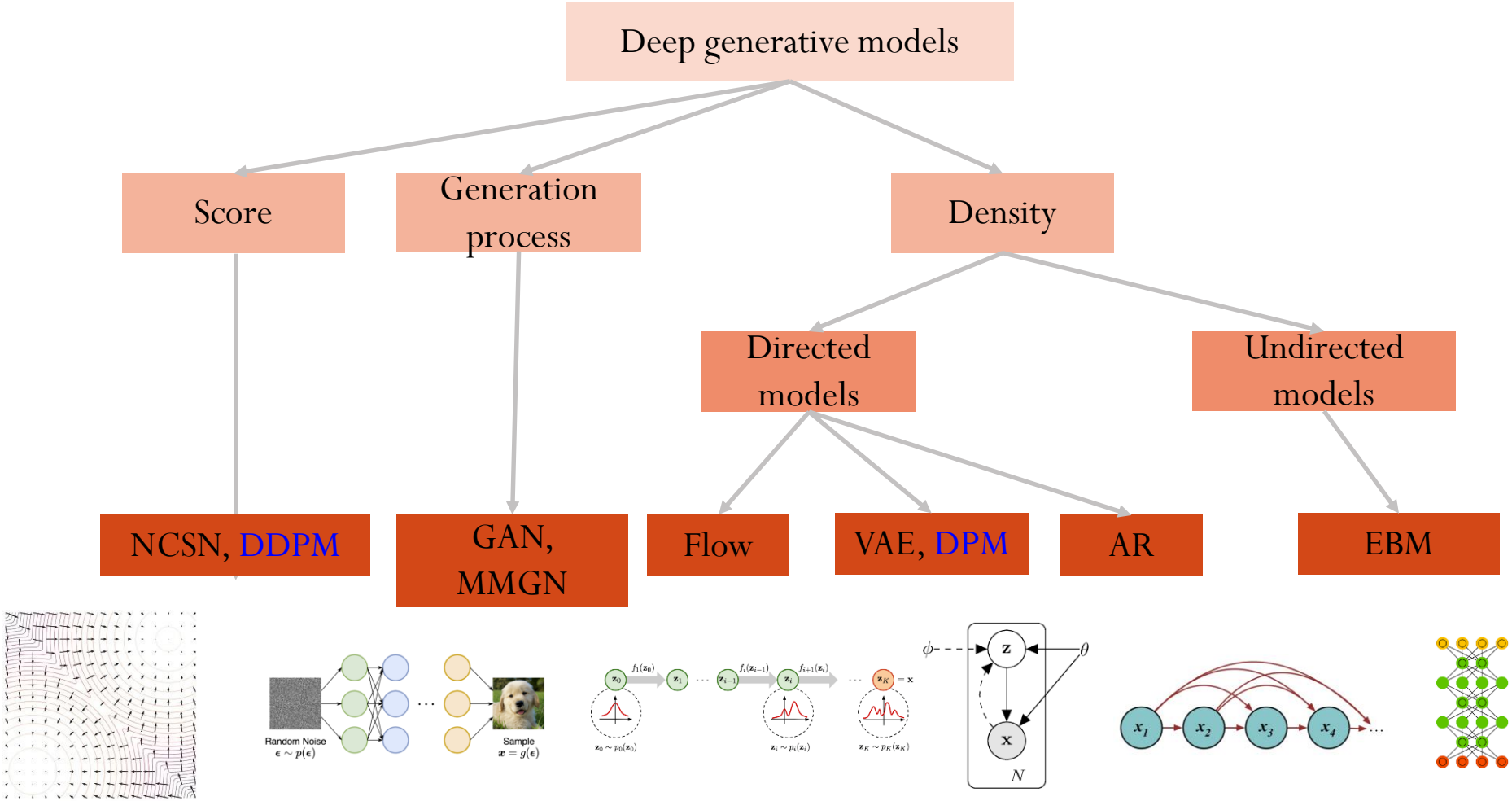




(Hartig et al., 2011)

- Two types:
  - Explicit models (e.g., VAE, Flow-based Models, diffusion probabilistic models)
  - Implicit models (e.g., GAN, Moment-matching generative networks)

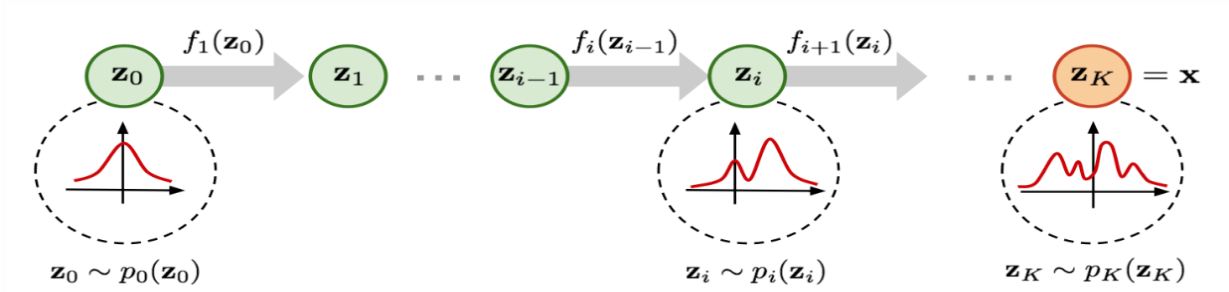
Model Zoo of Deep Generative Models



Images credit: Yang Song. https://yang-song.github.io/blog/2021/score/

Typical Examples – Flow-based Models

- A composition of multiple invertible functions



  - Many variants of invertible functions, including our variational flows (Chen et al., ICML 2020) and implicit flows (Lu et al., ICLR 2021)

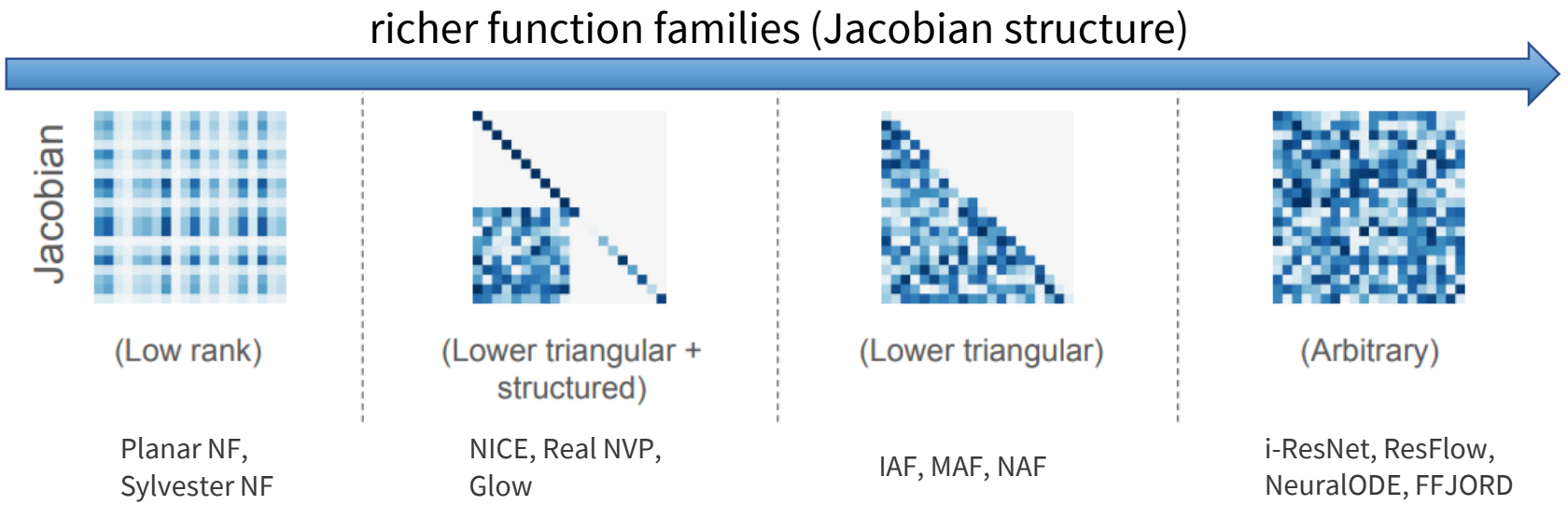
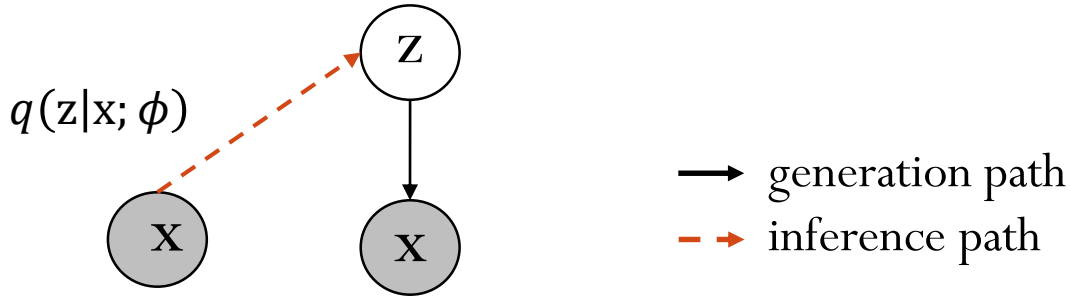


Figure from: http://www.cs.toronto.edu/~rtqichen/pdfs/residual_flows_slides.pdf

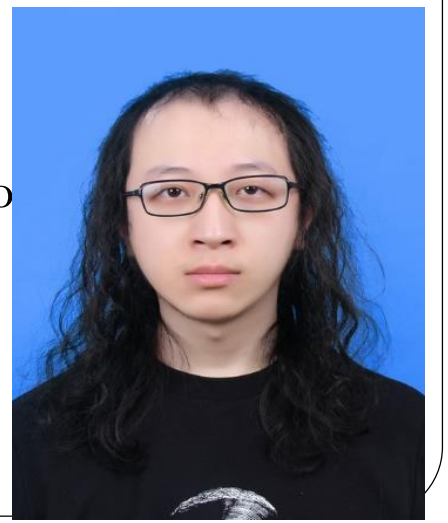
Typical Examples – Variational Auto-Encoder (VAE)

- Variational Bayesian inference for probabilistic models with neural networks (Kingma & Welling, ICLR 2013):

$$q(\mathrm{z}|\mathrm{x};\phi) \approx p(\mathrm{z}|\mathrm{x};\theta)$$

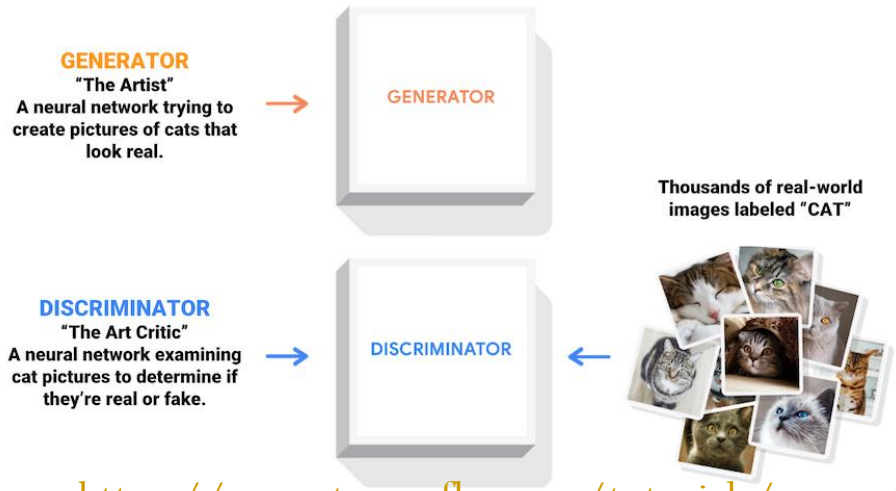


  - All the parameters are learned jointly via SGD with variance reduction
  - Many improvements/extensions to VAEs, including our work on unbiased estimato SUMO (Luo et al., ICLR 2020), implicit variational inference (Shi et al., ICML 2018, Zhou et al., ICML 2020) …



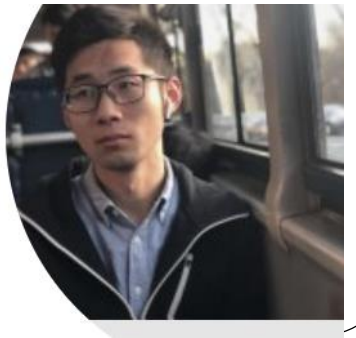
Typical Examples – Generative Adversarial Networks (GANs)

- An implicit generative model learned by a two-player minimax game (Goodfellow et al., NIPS 2014):



$$\min_G \max_D \; \mathbf{E}_{p_{\text{data}}(\mathrm{x})}[\log D(\mathrm{x})] + \mathbf{E}_{p(\mathrm{z})}\left[\log\left(1 - D\left(G(\mathrm{z})\right)\right)\right]$$

https://www.tensorflow.org/tutorials/generative/dcgan



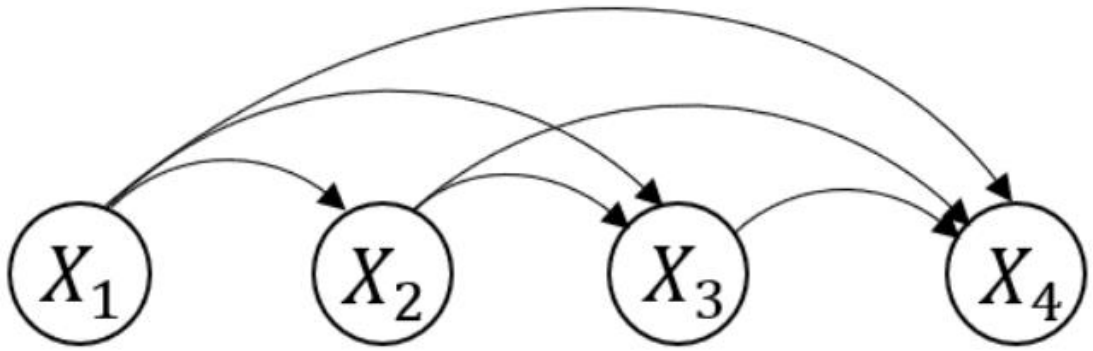
- Many improvements/extensions to GANs, including our work on stabilize the learning of GANs (Xu et al., ICML 2020), triple GAN for semi-supervised learning (Li et al., NIPS 2017, PAMI 2020) …

Basic of Auto-Regressive DGMs

- Consider a sequence data $\mathbf{x} = (x_1, x_2, \ldots, x_d)$
- The joint distribution is factorized as

$$p(\mathbf{x}) = \prod_{i=1}^{d} p(x_i | x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{d} p(x_i | \mathbf{x}_{<i})$$

  - where $\mathbf{x}_{<i} = (x_1, x_2, \ldots, x_{i-1})^\top$
- An auto-regression structure



Basic of Auto-Regressive DGMs

- Directly model each factor $p(x_i|\mathbf{x}_{<i})$ can be exponential!

- Auto-regressive DGMs use a parametric form to characterize each factor in a parameter efficient way!
  - E.g., for binary variance, the distribution can be Bernoulli

$$p(x_i|\mathbf{x}_{<i}) = \text{Bernoulli}(f_i(x_1, x_2, \ldots, x_{i-1}; \boldsymbol{\theta}_i))$$

  - The total parameters are

$$\sum_{i=1}^{d} \dim(\boldsymbol{\theta}_i)$$

Examples of Auto-Regressive DGMs

- Sigmoid belief network (binary variables)

$$f_i = \sigma(\alpha_1^{(i)} x_1 + \alpha_2^{(i)} x_2 + \cdots + \alpha_{i-1}^{(i)} x_{i-1}).$$

  - A logistic regression model that treats $\mathbf{x}_{<i}$ as inputs
  - Total size $\sum_{i=1}^{d} i = O(d^2)$

- MLP

$$\mathbf{h}_i = \sigma(A_i \mathbf{x}_{<i} + \mathbf{b}_i)$$

$$f_i(x_1, \ldots, x_{i-1}) = \sigma(\boldsymbol{\alpha}_i^\top \mathbf{h}_i + c_i),$$

  - Total size $O(d^2 L)$

Examples of Auto-Regressive DGMs

- Neural Auto-regressive Density Estimator (NADE)
  - Share parameters; More efficient than MLP

$$\mathbf{h}_i = \sigma(W_{.,<i}\mathbf{x}_{<i} + \mathbf{b})$$

$$f_i(x_1, \ldots, x_{i-1}) = \sigma(\boldsymbol{\alpha}_i^\top \mathbf{h}_i + c_i),$$

  - W, b is shared by all factors
  - Total size

$$O(dL)$$

Examples of Auto-Regressive DGMs

- Neural Auto-regressive Density Estimator (NADE)
  - Model continuous distribution as MoG

$$p(x_i|\mathbf{x}_{<i}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_{ik}, \sigma_{ik}^2)$$

$$\boldsymbol{\pi} = \text{softmax}\left((V_i^{\pi})^\top \mathbf{h}_i + \mathbf{b}_i^{\pi}\right)$$

$$\boldsymbol{\mu}_i = (V_i^{\mu})^\top \mathbf{h}_i + \mathbf{b}_i^{\mu}$$

$$\boldsymbol{\sigma}_i = \exp\left((V_i^{\sigma})^\top \mathbf{h}_i + \mathbf{b}_i^{\sigma}\right),$$

Examples of Auto-Regressive DGMs

2017,
Transformer
by Google

2018.6, **GPT1**
by OpenAI
(117M params,
8M webpages)

2019.2, **GPT2**
by OpenAI
(1.5B params,
40GB text)

2020.5, **GPT3**
by OpenAI
(175B params,
~700GB text)

2021.12, **CodeX**
by OpenAI
(6B params,
800M lines of
code)

by OpenAI
(build on GPT3.5,
not clear on data)
2022.1, **InstructGPT**

2022.11, **ChatGPT**
by OpenAI
(build on GPT3.5,
not clear on data)

Learn Auto-Regressive Models

- Explicit density!
- Maximize log-likelihood

$$\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \sum_{\mathbf{x}\in\mathcal{D}} \log p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{x}\in\mathcal{D}} \sum_{i=1}^{d} \log p(x_i|\mathbf{x}_{<i}, \boldsymbol{\theta}).$$

- Optimize via SGD

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta_t \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t|B_t)$$

Diffusion in Physics

- Diffusion destroy structures along time

Image source: https://makeagif.com/gif/ink-drip-in-water-stock-footage-red-toobstock-free-stock-video-IR7ZXe

Diffusion in Physics

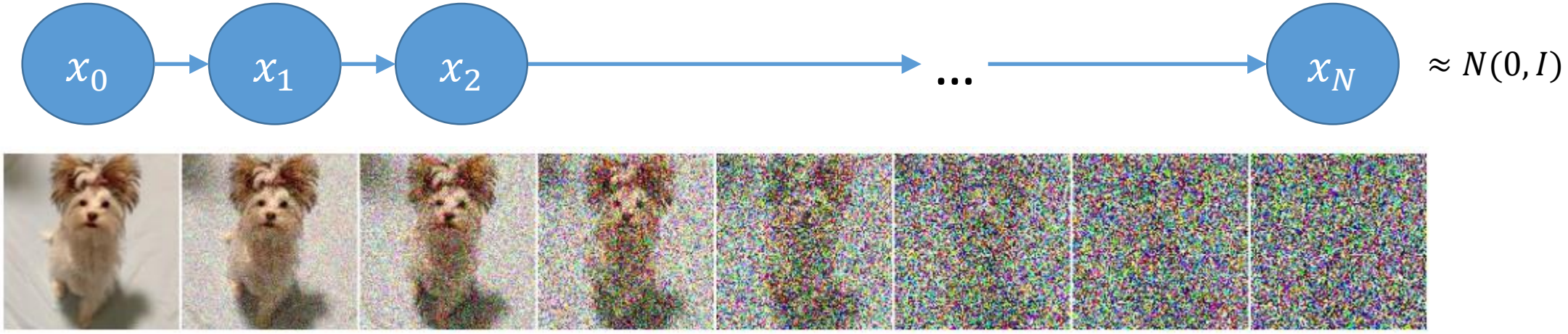
- What if we can reverse time?

Image source: https://makeagif.com/gif/ink-drip-in-water-stock-footage-red-toobstock-free-stock-video-IR7ZXe

Diffusion Probabilistic Models (DPMs)

- Diffusion process gradually injects noise to data
- Described by a Markov chain: $q(x_0, \dots, x_N) = q(x_0)q(x_1|x_0) \dots q(x_N|x_{N-1})$

Transition of diffusion: $q(x_n|x_{n-1}) = N(\sqrt{\alpha_n}x_{n-1}, \beta_n I)$ $\alpha_n = 1 - \beta_n$



Diffusion process: $q(x_0, \dots, x_N) = q(x_0)q(x_1|x_0) \dots q(x_N|x_{N-1})$

Demo Images from *Song et al., ICLR 2021*

[Ho et al. Denoising diffusion probabilistic models (DDPM), Neurips 2020
Song et al. Score-based generative modeling through stochastic differential equations, ICLR 2021]

Diffusion Probabilistic Models (DPMs)

- Diffusion process in the reverse direction ⇔ denoising process
- Reverse factorization: $q(x_0, \dots, x_N) = q(x_0|x_1) \dots q(x_{N-1}|x_N)q(x_N)$

Transition of denoising: $q(x_{n-1}|x_n) = ?$

Diffusion process: $q(x_0, \dots, x_N) = q(x_0)q(x_1|x_0) \dots q(x_N|x_{N-1})$

$= q(x_0|x_1) \dots q(x_{N-1}|x_N)q(x_N)$

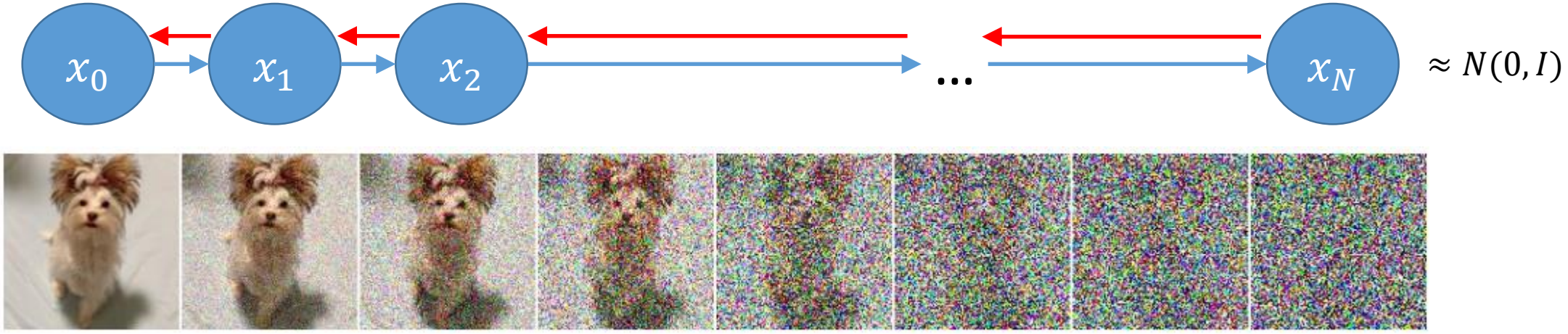
Diffusion Probabilistic Models (DPMs)

- Approximate diffusion process in the reverse direction

Model transition: $p(x_{n-1}|x_n) = N(\mu_n(x_n), \sigma_n^2 I)$

↓ approximate

Transition of denoising: $q(x_{n-1}|x_n) = ?$



Diffusion process: $q(x_0, \dots, x_N) = q(x_0)q(x_1|x_0) \dots q(x_N|x_{N-1})$

$= q(x_0|x_1) \dots q(x_{N-1}|x_N)q(x_N)$

The model: $p(x_0, \dots, x_N) = p(x_0|x_1) \dots p(x_{N-1}|x_N)p(x_N)$

Why Diffusion Models?

Diffusion models are simple but effective

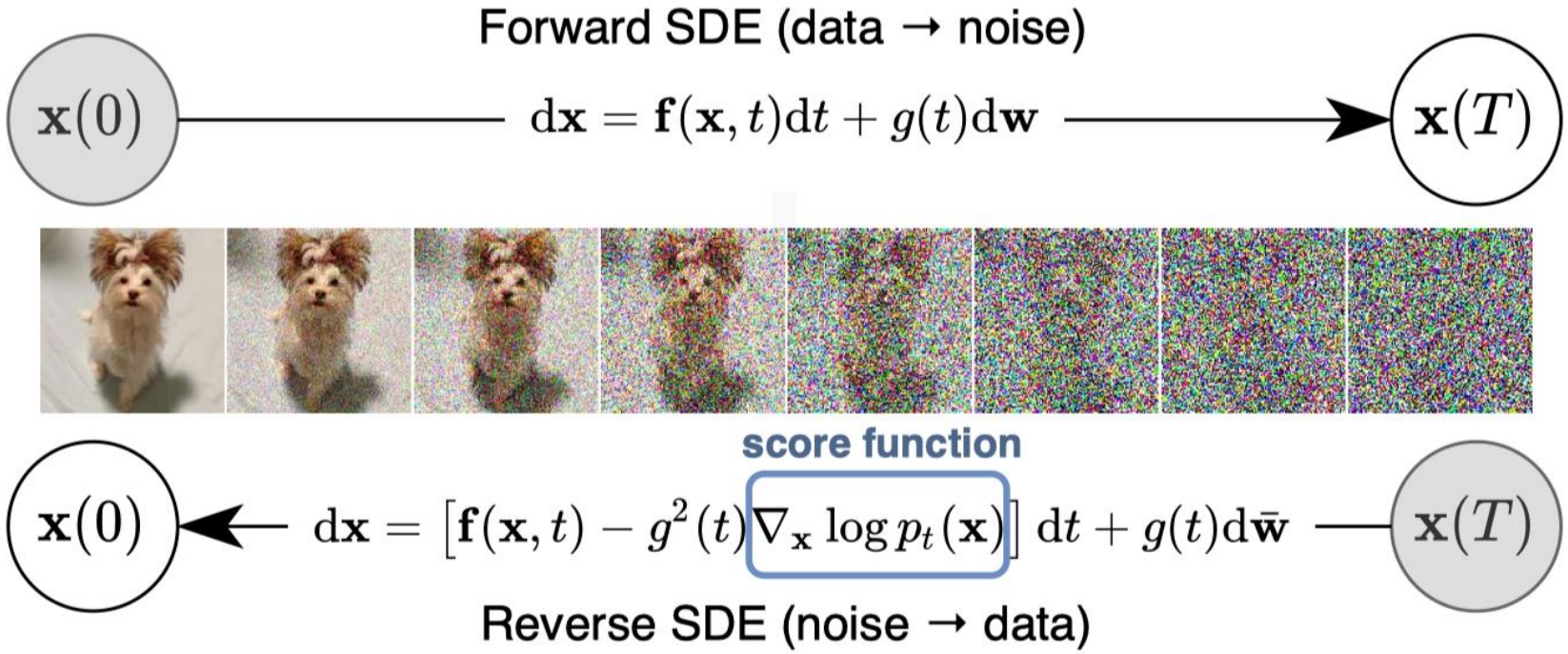
- No need to learn an "encoder" $q(z|x)$.
  - VAEs: Learn models by "**searching**" both p_θ and q_ϕ.
  - Diffusion models: Learn models by "**imitating**" a fixed forward process q.
- Training objective is simple: (MSE loss)

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

- Convergence guarantee:
  - For large enough T, the reverse process is indeed Gaussian!
- Outstanding generation performance

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.

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 :

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

Two Types of Diffusion Probabilistic Models

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

- **Diffusion SDEs:**

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

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

- **Diffusion ODEs:**

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

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

Fast Progress on Generative AI
E.g.: Midjourney V5 – Creates Photorealistic Images

- Prompt: 「Portait of an older man sat in a coffee shop, shot through a window. 」

- Prompt: 「A pair of young Chinese lovers, wearing jackets and jeans, sitting on the roof, the background is Beijing in the 1990s, and the opposite building can be seen --v 5 --s 250 --q 2」

Fast Progress on Generative AI
E.g. : Gen-2 – Text to Video

- Prompt:

「*The late afternoon sun peeking through the window of a NewYork City loft.*」

https://research.runwayml.com/gen2

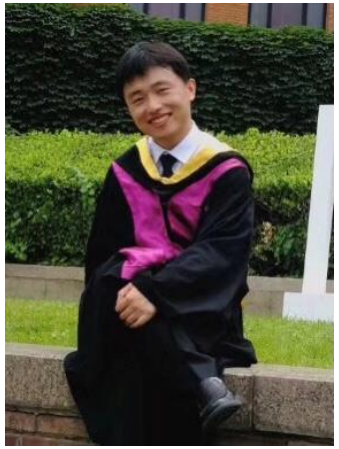
Some Progress on Diffusion Models @ THU TSAIL Group

- Basic theory and algorithms
  - **Score estimate for energy-based LVMs** (ICML 2021)
  - **Analytic-DPM** – optimal variance estimate (ICLR 2022 Outstanding Paper, ICML 2022)
  - **High-order denoising score matching** (ICML 2022)
  - **DPM-Solver** – the fastest inference algorithm (NeurIPS Oral, 2022)
  - **U-ViT** backbone – more scalable (CVPR 2023)
- Novel design of diffusion models for various tasks
  - Energy-guided DPM for **Image-2-Image translation** (NeurIPS, 2022)
  - Equivariant energy-guided DPM for **Molecular design** (ICLR 2023)
  - Generative behavior modeling for **Offline RL** (ICLR 2023)
  - UniDiffuser for **Multimodal inference** (under review)

Analytic-DPM: an Analytic Estimate of the Optimal Reverse Variance in Diffusion Probabilistic Models

https://github.com/baofff/Analytic-DPM

(with Bao, Li, Sun, Zhang, ICLR2022 Outstanding Paper; ICML 2022)



Diffusion Probabilistic Models (DPMs)

- We hope $q(x_0, \dots, x_N) \approx p(x_0, \dots, x_N)$ $\quad p(x_{n-1}|x_n) = N(\mu_n(x_n), \sigma_n^2 I)$
- Achieved by minimizing their KL divergence (i.e., maximizing the ELBO)

min KL $\qquad$ max ELBO $\qquad$ Reweighted ELBO (Score matching)

$$\min_{\mu_n(\cdot),\sigma_n^2} KL(q(x_{0:N})||p(x_{0:N})) \Leftrightarrow \max_{\mu_n(\cdot),\sigma_n^2} \mathrm{E}_q \log \frac{p(x_{0:N})}{q(x_{1:N}|x_0)} \Rightarrow \min_{s_n(\cdot)} \mathrm{E}_n \bar{\beta}_n \mathrm{E}_{q_n(x_n)} \|s_n(x_n) - \nabla \log q_n(x_n)\|^2$$

DDPM only optimizes the model mean
Use handcrafted model variance, e.g., $\sigma_n^2 = \beta_n$

Suboptimal!

Parameterization of $\mu_n(\cdot)$: $\mu_n(x_n) = \frac{1}{\sqrt{\alpha_n}}\left(x_n + \beta_n s_n(x_n)\right)$

Analytical-DPMs

- Can we directly find the optimal solution for $\min_{\mu_n(\cdot),\sigma_n^2} KL(q(x_{0:N})||p(x_{0:N}))$?
- **Yes!!!**

Theorem 1. *(Score representation of the optimal solution to KL minimization)*

The optimal solution to $\min_{\mu_n(\cdot),\sigma_n^2} KL(q(x_{0:N})||p(x_{0:N}))$ *is*

$$\mu_n^*(x_n) = \frac{1}{\sqrt{\alpha_n}}(x_n + \beta_n \nabla \log q_n(x_n)),$$

$$\sigma_n^{*2} = \frac{\beta_n}{\alpha_n}(1 - \beta_n \mathrm{E}_{q_n(x_n)} \frac{\|\nabla \log q_n(x_n)\|^2}{d}).$$

3 key steps in proof:
- Moment matching
- Law of total variance
- Score representation of moments of $q(x_0|x_n)$

See a more general version of Theorem 1 for more general $q(x_{0:N})$ in the full paper
See extension to score-based SDE (Song et al. 2021) in the full paper

(Bao et al., Analytical-DPM, ICLR 2022)

Analytical-DPMs

- Explicitly shows KL and score matching share the same optimal solution

optimal mean to KL : $\mu_n^*(x_n) = \frac{1}{\sqrt{\alpha_n}}(x_n + \beta_n \nabla \log q_n(x_n))$

↑ score matching

parameterization of mean: $\mu_n(x_n) = \frac{1}{\sqrt{\alpha_n}}\left(x_n + \beta_n s_n(x_n)\right)$

- **Training-free** estimate of the optimal variance

optimal variance to KL: $\sigma_n^{*2} = \frac{\beta_n}{\alpha_n}\left(1 - \beta_n \mathrm{E}_{q_n(x_n)} \frac{\|\nabla \log q_n(x_n)\|^2}{d}\right)$

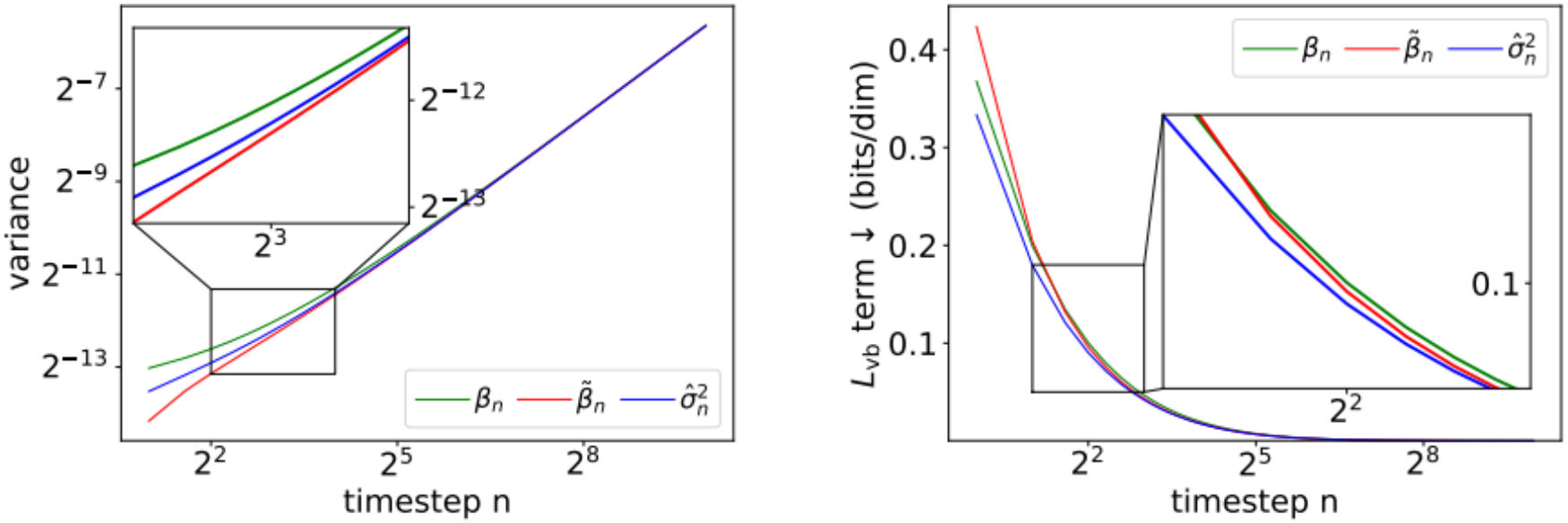
≈

$s_n(x_n) \approx \nabla \log q_n(x_n)$ + Monte Carlo: $\Gamma_n = \frac{1}{M}\sum_{m=1}^{M} \frac{\|s_n(x_{n,m})\|^2}{d}$, $x_{n,m} \sim q_n(x_n)$

analytic estimate of the optimal variance: $\hat{\sigma}_n^{*2} = \frac{\beta_n}{\alpha_n}(1 - \beta_n \Gamma_n)$

(Bao et al., Analytical-DPM, ICLR 2022)

Analytical-DPMs



(a) Variance (b) Terms in L_{vb}

Figure 1: Comparing our analytic estimate $\hat{\sigma}_n^2$ and prior works with handcrafted variances β_n and $\tilde{\beta}_n$. (a) compares the values of the variance for different timesteps. (b) compares the term in L_{vb} corresponding to each timestep. The value of L_{vb} is the area under the corresponding curve.

(Bao et al., Analytical-DPM, ICLR 2022)

Analytical-DPMs

- Byproduct: bounds of the optimal variance σ_n^{*2}

Theorem 2. *(Bounds of the optimal reverse variance)*

σ_n^{*2} has the following lower and upper bounds:

$$\tilde{\beta}_n \leq \sigma_n^{*2} \leq \frac{\beta_n}{\alpha_n}.$$

If we further assume $q(x_0)$ is a bounded distribution in $[a, b]^d$, where d is the dimension of data, then σ_n^{*2} can be further upper bounded by

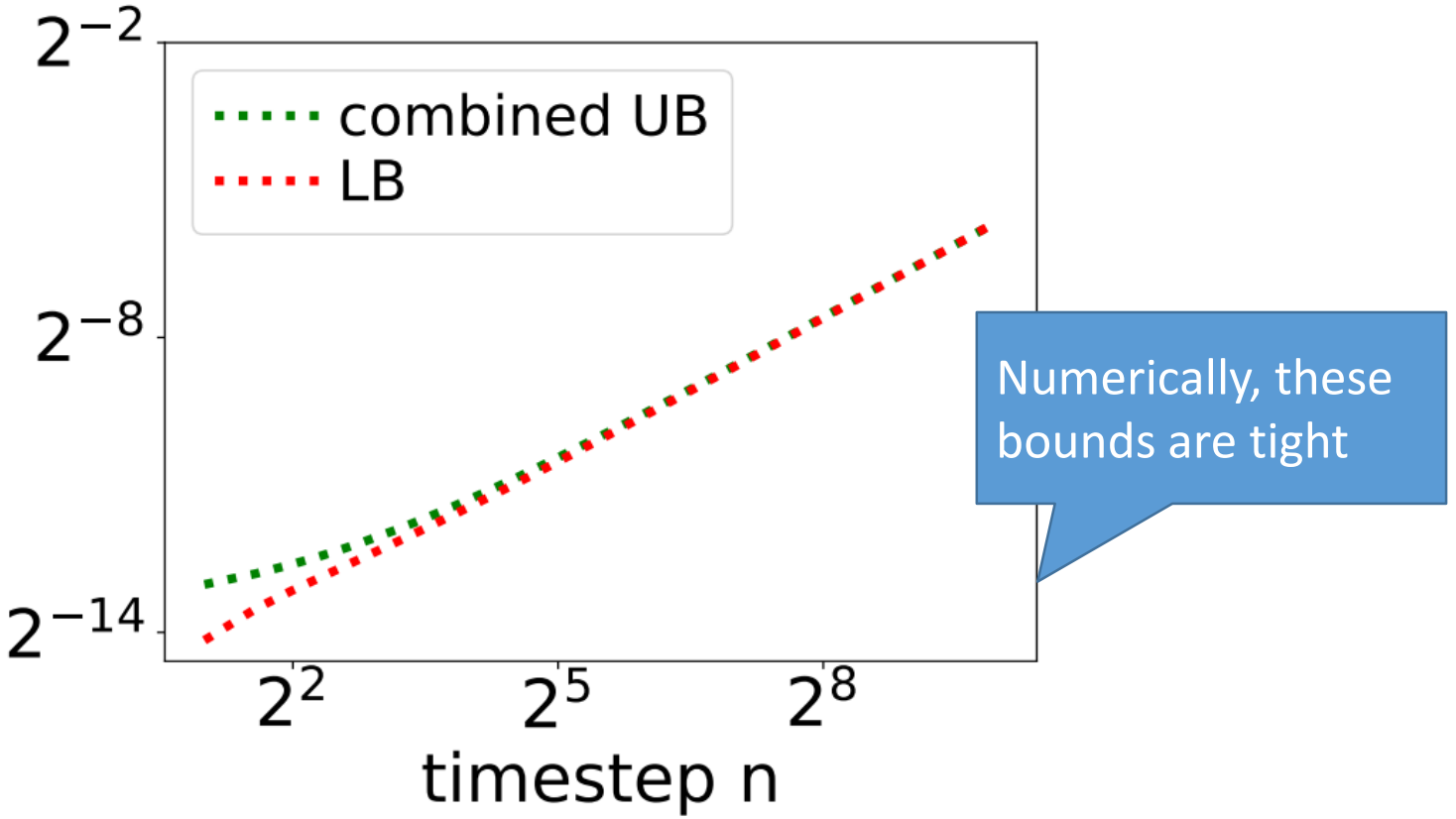
$$\sigma_n^{*2} \leq \tilde{\beta}_n + \frac{\bar{\alpha}_{n-1}\beta_n^2}{\bar{\beta}_n^2}\left(\frac{b-a}{2}\right)^2.$$

See a more general version of Theorem 2 for more general $q(x_{0:N})$ in the full paper

(Bao et al., Analytical-DPM, ICLR 2022)

Analytical-DPMs

- Tightness of bounds of the optimal variance σ_n^{*2}
- Empirically, we clip our estimate using these bounds



(Bao et al., Analytical-DPM, ICLR 2022)

Analytical-DPMs

- Likelihood results
  - Analytic-DPM consistently outperforms DDPM
  - Better efficiency (20 × to 80 × speed up) + better performance

Model \ # timesteps K		10	25	50	100	200	400	1000
CIFAR10 (LS)								
ET	DDPM, $\sigma_n^2 = \tilde{\beta}_n$	74.95	24.98	12.01	7.08	5.03	4.29	3.73
	DDPM, $\sigma_n^2 = \beta_n$	6.99	6.11	5.44	4.86	4.39	4.07	3.75
	Analytic-DDPM	**5.47**	**4.79**	**4.38**	**4.07**	**3.84**	**3.71**	**3.59**
OT	DDPM, $\sigma_n^2 = \beta_n$	5.38	4.34	3.97	3.82	3.77	3.75	3.75
	Analytic-DDPM	**4.11**	**3.68**	**3.61**	**3.59**	**3.59**	**3.59**	**3.59**
CIFAR10 (CS)								
ET	DDPM, $\sigma_n^2 = \tilde{\beta}_n$	75.96	24.94	11.96	7.04	4.95	4.19	3.60
	DDPM, $\sigma_n^2 = \beta_n$	6.51	5.55	4.92	4.41	4.03	3.78	3.54
	Analytic-DDPM	**5.08**	**4.45**	**4.09**	**3.83**	**3.64**	**3.53**	**3.42**
OT	DDPM, $\sigma_n^2 = \beta_n$	5.51	4.30	3.86	3.65	3.57	3.54	3.54
	Analytic-DDPM	**3.99**	**3.56**	**3.47**	**3.44**	**3.43**	**3.42**	**3.42**

Analytical-DPMs

- Likelihood results
  - Analytic-DPM consistently outperforms DDPM
  - Better efficiency (20 × to 80 × speed up) + better performance

CelebA 64x64								
ET	DDPM, $\sigma_n^2 = \tilde{\beta}_n$	33.42	13.09	7.14	4.60	3.45	3.03	2.71
	DDPM, $\sigma_n^2 = \beta_n$	6.67	5.72	4.98	4.31	3.74	3.34	2.93
	Analytic-DDPM	**4.54**	**3.89**	**3.48**	**3.16**	**2.92**	**2.79**	**2.66**
OT	DDPM, $\sigma_n^2 = \beta_n$	4.76	3.58	3.16	2.99	2.94	2.93	2.93
	Analytic-DDPM	**2.97**	**2.71**	**2.67**	**2.66**	**2.66**	**2.66**	**2.66**

Model \ # timesteps K		25	50	100	200	400	1000	4000
ImageNet 64x64								
ET	DDPM, $\sigma_n^2 = \tilde{\beta}_n$	105.87	46.25	22.02	12.10	7.59	5.04	3.89
	DDPM, $\sigma_n^2 = \beta_n$	5.81	5.20	4.70	4.31	4.04	3.81	3.65
	Analytic-DDPM	**4.78**	**4.42**	**4.15**	**3.95**	**3.81**	**3.69**	**3.61**
OT	DDPM, $\sigma_n^2 = \beta_n$	4.56	4.09	3.84	3.73	3.68	3.65	3.65
	Analytic-DDPM	**3.83**	**3.70**	**3.64**	**3.62**	**3.62**	**3.61**	**3.61**

Analytical-DPMs

- Sample quality results (FID)
  - Analytic-DPM outperforms DDPM in cost cases
  - Analytic-DPM consistently improves DDIM
  - Better efficiency ($20\times$ to $80\times$ speed up) + comparable performance

Model \ # timesteps K	10	25	50	100	200	400	1000
CIFAR10 (LS)							
DDPM, $\sigma_n^2 = \tilde{\beta}_n$	44.45	21.83	15.21	10.94	8.23	6.43	5.11
DDPM, $\sigma_n^2 = \beta_n$	233.41	125.05	66.28	31.36	12.96	4.86	†**3.04**
Analytic-DDPM	**34.26**	**11.60**	**7.25**	**5.40**	**4.01**	**3.62**	4.03
DDIM	21.31	10.70	7.74	6.08	5.07	4.61	4.13
Analytic-DDIM	**14.00**	**5.81**	**4.04**	**3.55**	**3.39**	**3.50**	**3.74**
CIFAR10 (CS)							
DDPM, $\sigma_n^2 = \tilde{\beta}_n$	34.76	16.18	11.11	8.38	6.66	5.65	4.92
DDPM, $\sigma_n^2 = \beta_n$	205.31	84.71	37.35	14.81	5.74	**3.40**	**3.34**
Analytic-DDPM	**22.94**	**8.50**	**5.50**	**4.45**	**4.04**	3.96	4.31
DDIM	34.34	16.68	10.48	7.94	6.69	5.78	4.89
Analytic-DDIM	**26.43**	**9.96**	**6.02**	**4.88**	**4.92**	**5.00**	**4.66**

Analytical-DPMs

- Sample quality results (FID)
  - Analytic-DPM outperforms DDPM in cost cases
  - Analytic-DPM consistently improves DDIM
  - Better efficiency ($20\times$ to $80\times$ speed up) + comparable performance

CelebA 64x64							
DDPM, $\sigma_n^2 = \tilde{\beta}_n$	36.69	24.46	18.96	14.31	10.48	8.09	5.95
DDPM, $\sigma_n^2 = \beta_n$	294.79	115.69	53.39	25.65	9.72	**3.95**	**3.16**
Analytic-DDPM	**28.99**	**16.01**	**11.23**	**8.08**	**6.51**	5.87	5.21
DDIM	20.54	13.45	9.33	6.60	4.96	4.15	3.40
Analytic-DDIM	**15.62**	**9.22**	**6.13**	**4.29**	**3.46**	**3.38**	**3.13**
Model \ # timesteps K	25	50	100	200	400	1000	4000
ImageNet 64x64							
DDPM, $\sigma_n^2 = \tilde{\beta}_n$	**29.21**	**21.71**	19.12	17.81	17.48	16.84	16.55
DDPM, $\sigma_n^2 = \beta_n$	170.28	83.86	45.04	28.39	21.38	17.58	16.38
Analytic-DDPM	32.56	22.45	**18.80**	**17.16**	**16.40**	**16.14**	**16.34**
DDIM	26.06	20.10	18.09	17.84	17.74	17.73	19.00
Analytic-DDIM	**25.98**	**19.23**	**17.73**	**17.49**	**17.44**	**17.57**	**18.98**

Analytical-DPMs

The algorithms was adopted by OpenAI in the recently released DALL·E2, a large-scale pre-trained model with 12 Billion parameters for cross-modal image generation, with state-of-the-art performance

Model	FID	Zero-shot FID	Zero-shot FID (filt)
AttnGAN (Xu et al., 2017)	35.49		
DM-GAN (Zhu et al., 2019)	32.64		
DF-GAN (Tao et al., 2020)	21.42		
DM-GAN + CL (Ye et al., 2021)	20.79		
XMC-GAN (Zhang et al., 2021)	9.33		
LAFITE (Zhou et al., 2021)	8.12		
Make-A-Scene (Gafni et al., 2022)	**7.55**		
DALL-E (Ramesh et al., 2021)		$\sim$ 28	
LAFITE (Zhou et al., 2021)		26.94	
GLIDE (Nichol et al., 2021)		12.24	12.89
Make-A-Scene (Gafni et al., 2022)			11.84
unCLIP (AR prior)		10.63	11.08
unCLIP (Diffusion prior)		**10.39**	**10.87**

An astronaut + lounging in a space tropical resort + vaporwave style

Teddy bears + mixing bubbling chemicals like mad scientists + 1990s Saturday morning cartoon style

Extensions of Analytical-DPMs

- We hope $q(x_0, \dots, x_N) \approx p(x_0, \dots, x_N)$ $\quad p(x_{n-1}|x_n) = N(\mu_n(x_n), \Sigma_n(x_n))$
- Achieved by minimizing their KL divergence (i.e., maximizing the ELBO)

min KL $\qquad\qquad$ max ELBO

$$\min_{\mu_n(\cdot),\Sigma_n(\cdot)} KL(q(x_{0:N})||p(x_{0:N})) \Leftrightarrow \max_{\mu_n(\cdot),\Sigma_n(\cdot)} \mathrm{E}_q \log \frac{p(x_{0:N})}{q(x_{1:N}|x_0)}$$

(**Analytic-DPM**) Suppose $\Sigma_n(x_n) = \sigma_n^2$. The optimal solution is

$$\mu_n^*(x_n) = \frac{1}{\sqrt{\alpha_n}}\left(x_n - \frac{\beta_n}{\sqrt{\bar{\beta}_n}}\mathbf{E}_{q(x_0|x_n)}[\epsilon_n]\right),$$

$$\sigma_n^{*2} = \frac{\beta_n}{\alpha_n}\left(1 - \frac{\beta_n}{\bar{\beta}_n}\mathrm{E}_{q_n(x_n)}\frac{\left\|\mathbf{E}_{q(x_0|x_n)}[\epsilon_n]\right\|^2}{d}\right).$$

Noise prediction network:
$\hat{\epsilon}_n(x_n) \approx \mathbf{E}_{q(x_0|x_n)}[\epsilon_n]$

- Isotropic variance is simple
- Irrelevant to state x_n
- Assume the mean is optimal

Extensions of Analytical-DPMs

- What is the optimal diagonal covariance $\Sigma_n(x_n) = \text{diag}(\sigma_n^2(x_n))$?

Theorem 1. Suppose $\Sigma_n(x_n) = \text{diag}(\sigma_n^2(x_n))$. The optimal solution is

$$\mu_n^*(x_n) = \frac{1}{\sqrt{\alpha_n}}\left(x_n - \frac{\beta_n}{\sqrt{\bar{\beta}_n}}\mathbf{E}_{q(x_0|x_n)}[\epsilon_n]\right),$$

$$\sigma_n^*(x_n)^2 = \frac{\bar{\beta}_{n-1}}{\bar{\beta}_n}\beta_n + \frac{\beta_n^2}{\bar{\beta}_n\alpha_n}\Big(\underbrace{\mathbf{E}_{q(x_0|x_n)}[\epsilon_n^2]}_{\approx h_n(x_n)} - \underbrace{\mathbf{E}_{q(x_0|x_n)}[\epsilon_n]^2}_{\approx \hat{\epsilon}_n(x_n)^2}\Big).$$

predict SN: $\min_{h_n} \mathbf{E}_{q(x_0,x_n)}\left\|h_n(x_n) - \epsilon_n^2\right\|^2$

squared noise (SN)

See a more general version of Theorem 1 for more general $q(x_{0:N})$ in the full paper
See extension to score-based SDE (Song et al.) in the full paper

[Estimating the Optimal Covariance with Imperfect Mean in Diffusion Probabilistic Models, Bao et al., ICML 2022]

Extensions of Analytical-DPMs

- In practice, $\hat{\epsilon}_n(x_n)$ and $\mu_n(x_n)$ are not optimal
- What is the optimal diagonal covariance $\Sigma_n(x_n) = \text{diag}(\sigma_n^2(x_n))$?

Theorem 2. Suppose $\Sigma_n(x_n) = \text{diag}(\sigma_n^2(x_n))$. For any mean $\mu_n(x_n)$ parameterized by $\hat{\epsilon}_n(x_n)$, the optimal covariance is

$$\tilde{\sigma}_n^*(x_n)^2 = \frac{\bar{\beta}_{n-1}}{\bar{\beta}_n}\beta_n + \frac{\beta_n^2}{\bar{\beta}_n \alpha_n}\mathbf{E}_{q(x_0|x_n)}\left[\left(\epsilon_n - \hat{\epsilon}_n(x_n)\right)^2\right].$$

$$\approx g_n(x_n)$$

predict NPR: $\min_{g_n} \mathbf{E}_{q(x_0,x_n)} \left\| h_n(x_n) - \left(\epsilon_n - \hat{\epsilon}_n(x_n)\right)^2 \right\|^2$

noise prediction residual (NPR):

[Estimating the Optimal Covariance with Imperfect Mean in Diffusion Probabilistic Models, Bao et al., ICML 2022]

Extensions of Analytical-DPMs

- Likelihood results
  - NPR-DPM (predicting NPR) consistently outperforms Analytic-DPM

		CIFAR10 (LS)						CIFAR10 (CS)					
# TIMESTEPS K		10	25	50	100	200	1000	10	25	50	100	200	1000
ET	DDPM, $\tilde{\beta}_n$	74.95	24.98	12.01	7.08	5.03	3.73	75.96	24.94	11.96	7.04	4.95	3.60
	DDPM, β_n	6.99	6.11	5.44	4.86	4.39	3.75	6.51	5.55	4.92	4.41	4.03	3.54
	A-DDPM	5.47	4.79	4.38	4.07	3.84	3.59	5.08	4.45	4.09	3.83	3.64	3.42
	NPR-DDPM	**5.40**	**4.64**	**4.25**	**3.98**	**3.79**	**3.57**	**5.03**	**4.33**	**3.99**	**3.76**	**3.59**	**3.41**
OT	DDPM, β_n	5.38	4.34	3.97	3.82	3.77	3.75	5.51	4.30	3.86	3.65	3.57	3.54
	A-DDPM	4.11	3.68	3.61	3.59	3.59	3.59	3.99	3.56	3.47	3.44	3.43	3.42
	NPR-DDPM	**3.91**	**3.64**	**3.59**	**3.58**	**3.57**	**3.57**	**3.88**	**3.52**	**3.45**	**3.42**	**3.41**	**3.41**

		CELEBA 64X64						IMAGENET 64X64					
# TIMESTEPS K		10	25	50	100	200	1000	25	50	100	200	400	4000
ET	DDPM, $\tilde{\beta}_n$	33.42	13.09	7.14	4.60	3.45	2.71	105.87	46.25	22.02	12.10	7.59	3.89
	DDPM, β_n	6.67	5.72	4.98	4.31	3.74	2.93	5.81	5.20	4.70	4.31	4.04	3.65
	A-DDPM	4.54	3.89	3.48	3.16	2.92	2.66	4.78	4.42	4.15	3.95	3.81	3.61
	NPR-DDPM	**4.46**	**3.78**	**3.40**	**3.11**	**2.89**	**2.65**	**4.66**	**4.22**	**3.96**	**3.80**	**3.71**	**3.60**
OT	DDPM, β_n	4.76	3.58	3.16	2.99	2.94	2.93	4.56	4.09	3.84	3.73	3.68	3.65
	A-DDPM	2.97	2.71	2.67	2.66	2.66	2.66	3.83	3.70	3.64	3.62	3.62	3.61
	NPR-DDPM	**2.88**	**2.69**	**2.66**	**2.66**	**2.65**	**2.65**	**3.73**	**3.65**	**3.62**	**3.60**	**3.60**	**3.60**

Extensions of Analytical-DPMs

- Sample quality results (FID)
  - Both NPR-DPM & SN-DPM outperform Analytic-DPM

	CIFAR10 (LS)						CIFAR10 (CS)					
# TIMESTEPS K	10	25	50	100	200	1000	10	25	50	100	200	1000
DDPM, $\tilde{\beta}_n$	44.45	21.83	15.21	10.94	8.23	5.11	34.76	16.18	11.11	8.38	6.66	4.92
DDPM, β_n	233.41	125.05	66.28	31.36	12.96	**3.04**	205.31	84.71	37.35	14.81	5.74	**3.34**
A-DDPM	34.26	11.60	7.25	5.40	4.01	4.03	22.94	8.50	5.50	4.45	4.04	4.31
NPR-DDPM	32.35	10.55	6.18	4.52	3.57	4.10	19.94	7.99	5.31	4.52	4.10	4.27
SN-DDPM	**24.06**	**6.91**	**4.63**	**3.67**	**3.31**	3.65	**16.33**	**6.05**	**4.17**	**3.83**	**3.72**	4.07
DDIM	21.31	10.70	7.74	6.08	5.07	4.13	34.34	16.68	10.48	7.94	6.69	4.89
A-DDIM	14.00	5.81	4.04	3.55	3.39	3.74	26.43	9.96	6.02	4.88	4.92	4.66
NPR-DDIM	13.34	5.38	3.95	3.53	3.42	3.72	22.81	9.47	6.04	5.02	5.06	4.62
SN-DDIM	**12.19**	**4.28**	**3.39**	**3.23**	**3.22**	**3.65**	**17.90**	**7.36**	**5.16**	**4.63**	**4.63**	**4.51**

	CELEBA 64X64						IMAGENET 64X64					
# TIMESTEPS K	10	25	50	100	200	1000	25	50	100	200	400	4000
DDPM, $\tilde{\beta}_n$	36.69	24.46	18.96	14.31	10.48	5.95	29.21	21.71	19.12	17.81	17.48	16.55
DDPM, β_n	294.79	115.69	53.39	25.65	9.72	**3.16**	170.28	83.86	45.04	28.39	21.38	16.38
A-DDPM	28.99	16.01	11.23	8.08	6.51	5.21	32.56	22.45	18.80	17.16	16.40	16.34
NPR-DDPM	28.37	15.74	10.89	8.23	7.03	5.33	28.27	20.89	18.06	16.96	**16.32**	16.38
SN-DDPM	**20.60**	**12.00**	**7.88**	**5.89**	**5.02**	4.42	**27.58**	**20.74**	**18.04**	**16.61**	16.37	**16.22**
DDIM	20.54	13.45	9.33	6.60	4.96	3.40	26.06	20.10	18.09	17.84	17.74	19.00
A-DDIM	15.62	9.22	6.13	4.29	3.46	3.13	**25.98**	**19.23**	17.73	17.49	17.44	18.98
NPR-DDIM	14.98	8.93	6.04	4.27	3.59	3.15	28.84	19.62	17.63	17.42	17.30	18.91
SN-DDIM	**10.20**	**5.48**	**3.83**	**3.04**	**2.85**	**2.90**	28.07	19.38	**17.53**	**17.23**	**17.23**	**18.89**

Extensions of Analytical-DPMs

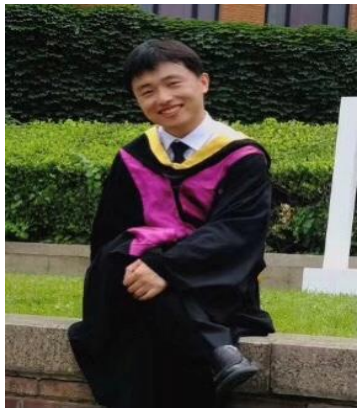
- Sample quality results (FID)
  - Both NPR-DPM & SN-DPM outperform Analytic-DPM when the number of steps is small, leading to higher efficiency

	CIFAR10 (VP SDE)					
# TIMESTEPS K	10	25	50	100	200	1000
EULER-MARUYAMA	292.20	170.17	90.79	47.46	21.92	**2.55**
ANCESTRAL SAMPLING	235.28	129.29	68.52	31.99	12.81	2.72
PROBABILITY FLOW	107.74	21.34	7.78	4.33	3.27	2.82
A-DPM	35.10	11.57	6.54	4.71	3.61	2.98
NPR-DPM	33.70	10.44	5.83	3.97	3.05	3.04
SN-DPM	**25.30**	**7.34**	**4.46**	**3.27**	**2.83**	2.71

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

(with Lu, Zhou, Bao, Chen, Li, NeurIPS 2022 Oral)





Two Types of Diffusion Probabilistic Models

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

- **Diffusion SDEs:**

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

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

- **Diffusion ODEs:**

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

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

(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:

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

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**:

$$\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\frac{(\lambda-\lambda_{t_{i-1}})^n}{n!}\mathrm{d}\lambda = -\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} \frac{(\lambda-\lambda_{t_{i-1}})^n}{n!}\mathrm{d}(e^{-\lambda})$$

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

$$= \cdots$$

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 $\hat{x}_{s_i}$:

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

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

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

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 $k = 2$:

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

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**

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

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

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

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

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

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

end for

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

Comparison with Traditional Runge-Kutta Methods

DPM-Solver is customized for DPMS

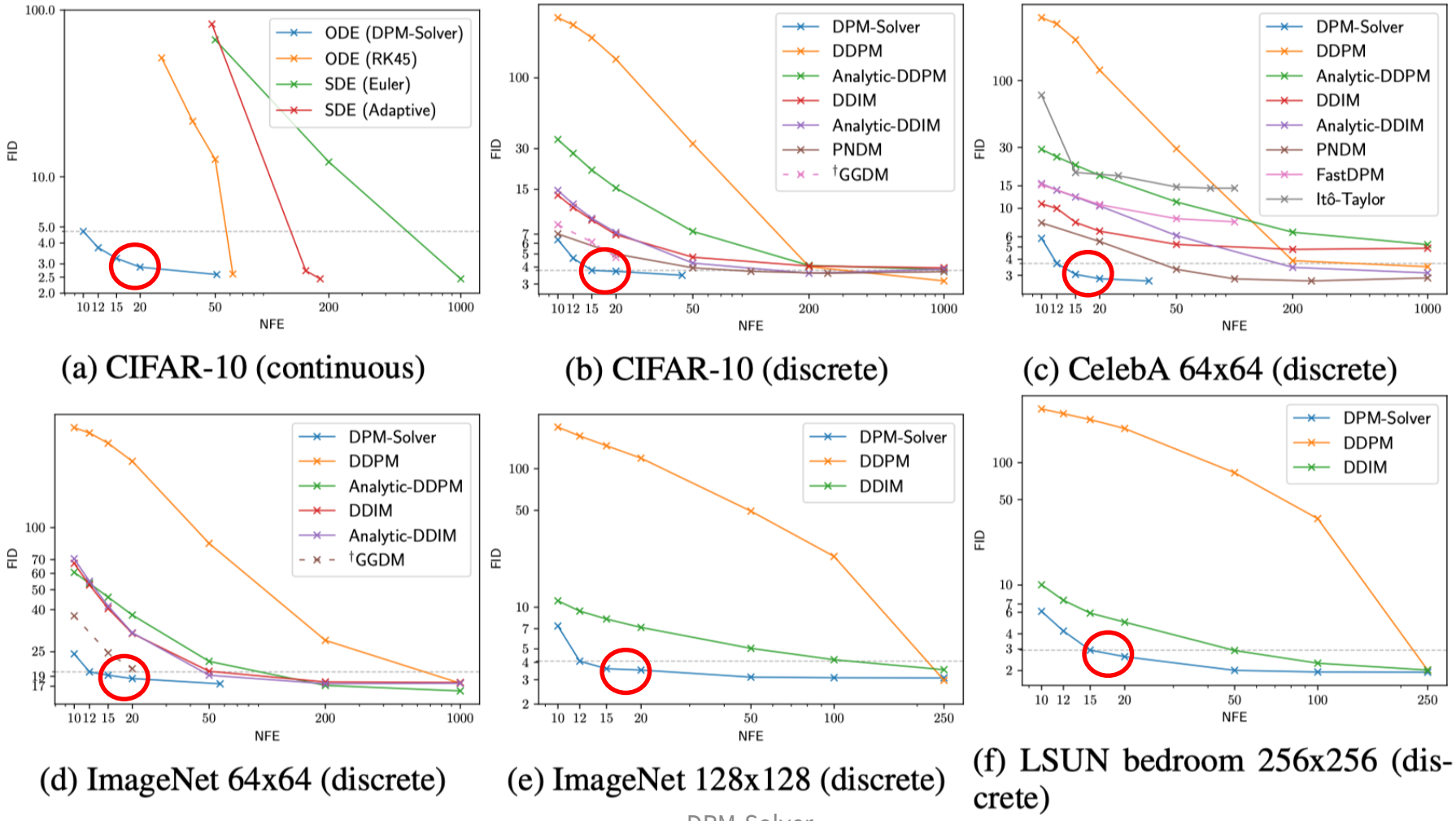
Table 1: FID ↓ on CIFAR-10 for different orders of Runge-Kutta (RK) methods and DPM-Solvers, varying the number of function evaluations (NFE). For RK methods, we evaluate diffusion ODEs w.r.t. both t (Eq. (2.7)) and λ (Eq. (E.1)). We use uniform step size in t for RK (t), and uniform step size in λ for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

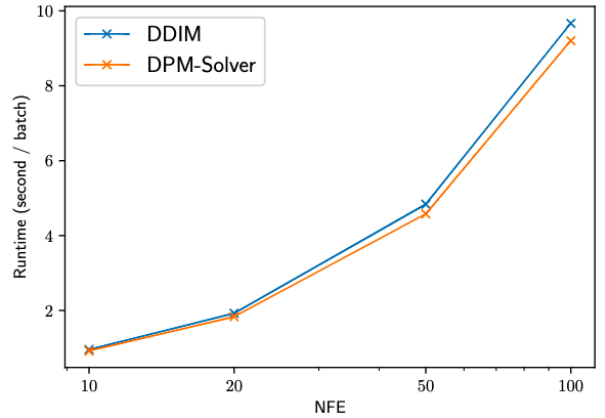
(d) ImageNet 64x64 (discrete)

(e) ImageNet 128x128 (discrete)

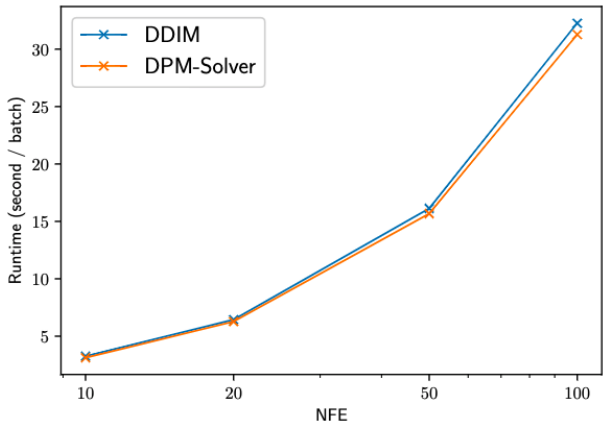
(f) LSUN bedroom 256x256 (discrete)

Additional Computation Costs are Neglectable

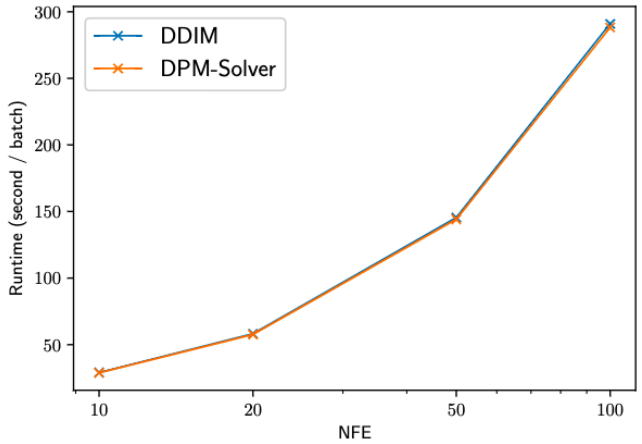
Because we analytically computes all the known terms



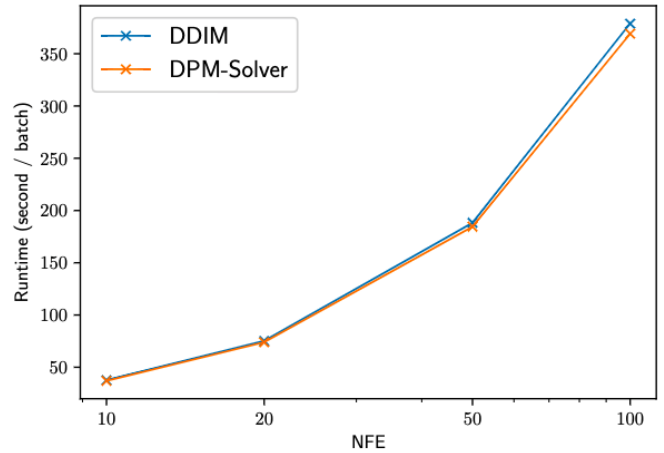
CIFAR10 32x32



CelebA 64x64



ImageNet 128x128



LSUN bedroom 256x256

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

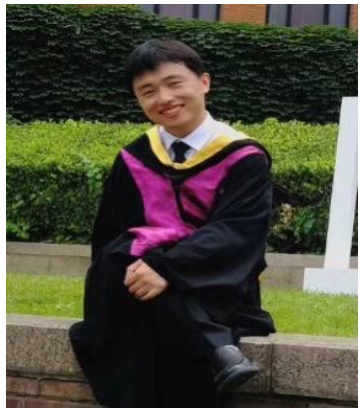
Under the same NFE, The computation costs of DDIM and DPM-Solver are almost the same.

(Our implementation is even slightly faster than the original DDIM.)

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

(with Lu, Zhou, Bao, Chen, Li, arXiv)





(Conditional) Guided Sampling by DPMs

Only need to modify the noise prediction model

Classifier guidance:

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

Classifier-free guidance:

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

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

Challenges for Guided Sampling with Large Guidance Scale

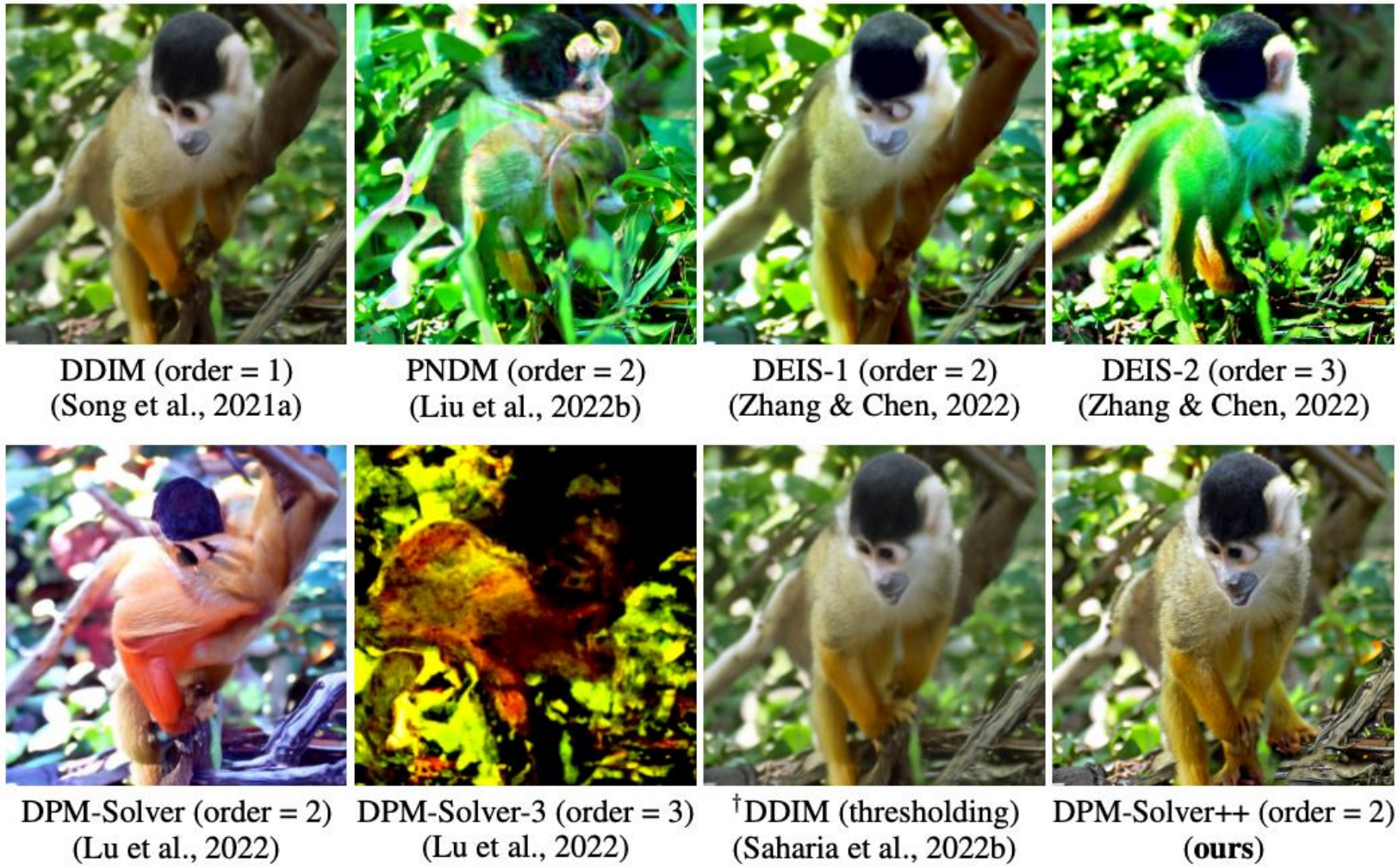
Challenge 1: "train-test mismatch" (Saharia et al., 2022)

Image data is bounded in $[0, 255]^D$, but large guidance scale causes **out-of-bound data**.

(a) No thresholding. (b) Static thresholding. (c) Dynamic thresholding.

Challenges for Guided Sampling with Large Guidance Scale

Challenge 2: unstable high-order solvers



ImageNet 256x256.

Guidance scale is 8.0.

15 function evaluations.

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

Foundation of DPM-Solver++

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

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

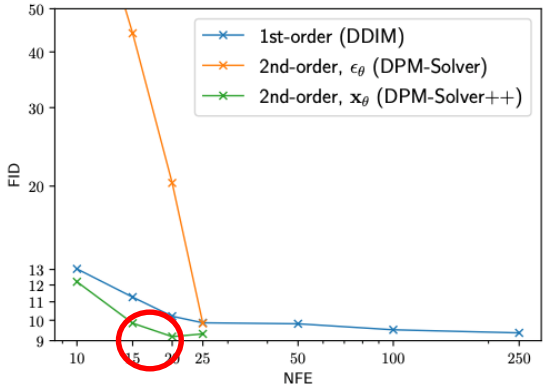
Algorithm 2 DPM-Solver++(2M).

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

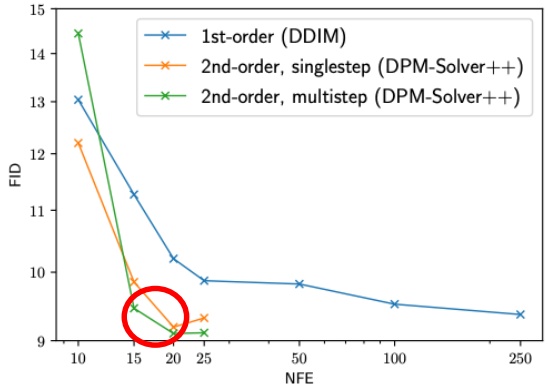
Denote $h_i := \lambda_{t_i} - \lambda_{t_{i-1}}$ for $i = 1, \ldots, M$.
$\tilde{\boldsymbol{x}}_{t_0} \leftarrow \boldsymbol{x}_T$. Initialize an empty buffer Q.
$Q \xleftarrow{\text{buffer}} \boldsymbol{x}_\theta(\tilde{\boldsymbol{x}}_{t_0}, t_0)$
$\tilde{\boldsymbol{x}}_{t_1} \leftarrow \frac{\sigma_{t_1}}{\sigma_{t_0}}\tilde{\boldsymbol{x}}_0 - \alpha_{t_1}\left(e^{-h_1} - 1\right)\boldsymbol{x}_\theta(\tilde{\boldsymbol{x}}_{t_0}, t_0)$
$Q \xleftarrow{\text{buffer}} \boldsymbol{x}_\theta(\tilde{\boldsymbol{x}}_{t_1}, t_1)$
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

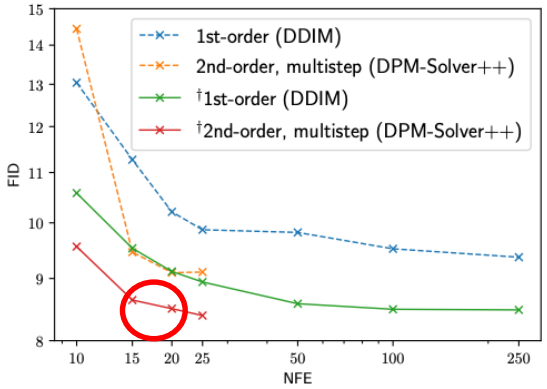
DPM-Solver++ can greatly improve the sample quality for large guidance scale



(a) From $\boldsymbol{\epsilon}_\theta$ to $\boldsymbol{x}_\theta$.



(b) From singlestep to multistep.



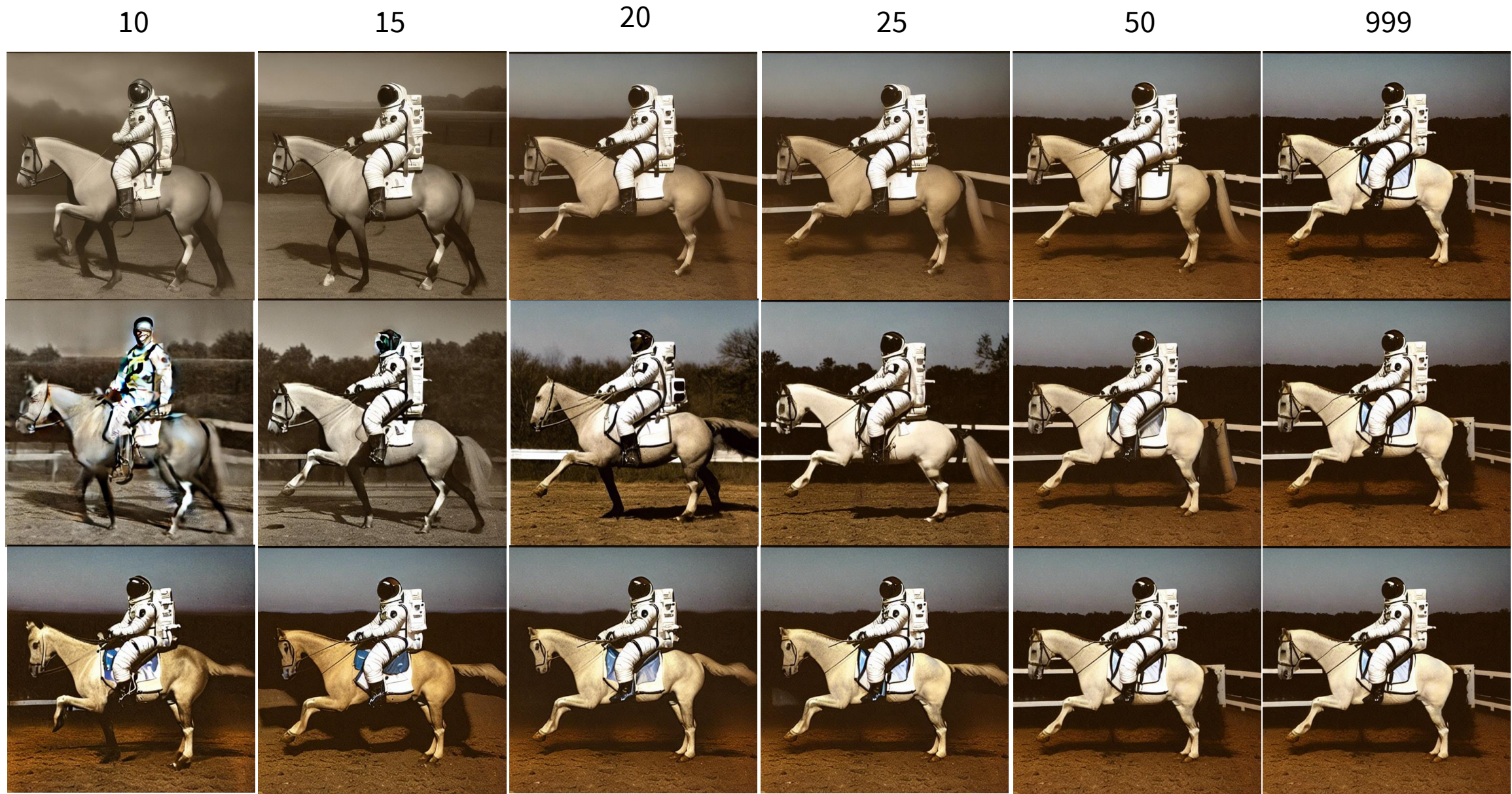
(c) Thresholding.

DPM-Solver-2
($\boldsymbol{\epsilon}_\theta$, singelstep)

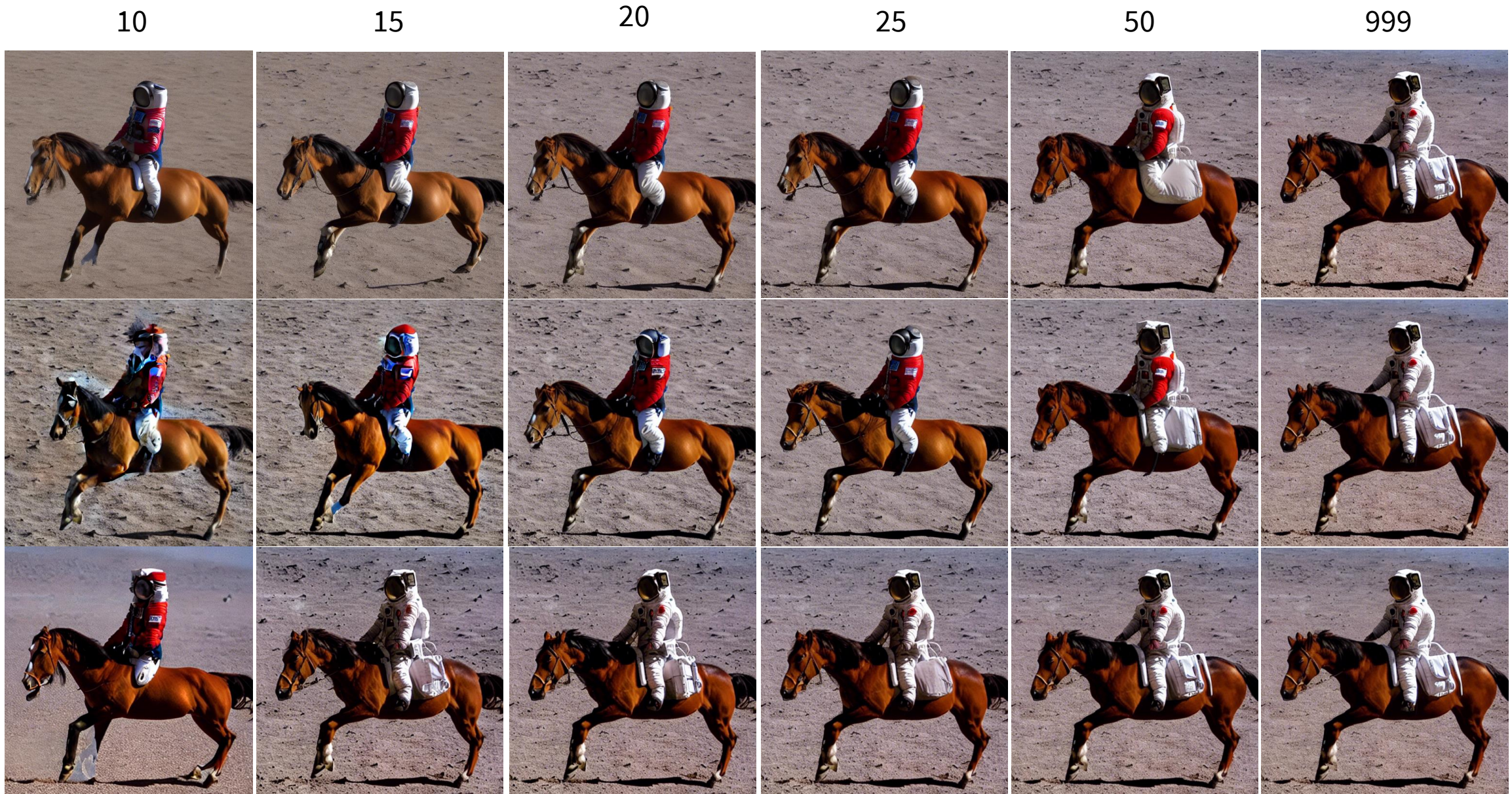
DPM-Solver++(2S)
($\boldsymbol{x}_\theta$, singlestep)

DPM-Solver++(2M)
($\boldsymbol{x}_\theta$, multistep, thresholdin)

Example: Stable-Diffusion with DPM-Solver++



Example: Stable-Diffusion with DPM-Solver++



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=3, method="singlestep",
                      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	$E_{x_0,\epsilon,t}\left[\omega_1(t)\|\|\epsilon_\theta(x_t,t)-\epsilon\|\|_2^2\right]$	DDPM, Stable-Diffusion
"x_start": data prediction model x_θ	$E_{x_0,\epsilon,t}\left[\omega_2(t)\|\|x_\theta(x_t,t)-x_0\|\|_2^2\right]$	DALL·E 2
"v": velocity prediction model v_θ	$E_{x_0,\epsilon,t}\left[\omega_3(t)\|\|v_\theta(x_t,t)-(\alpha_t\epsilon-\sigma_t x_0)\|\|_2^2\right]$	Imagen Video
"score": marginal score function s_θ	$E_{x_0,\epsilon,t}\left[\omega_4(t)\|\|\sigma_t s_\theta(x_t,t)+\epsilon\|\|_2^2\right]$	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

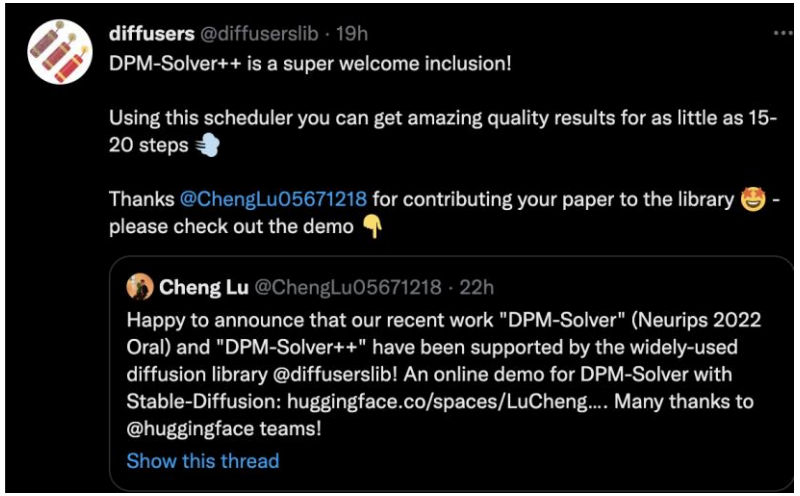
Very easy to use in stable-diffusion

```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:



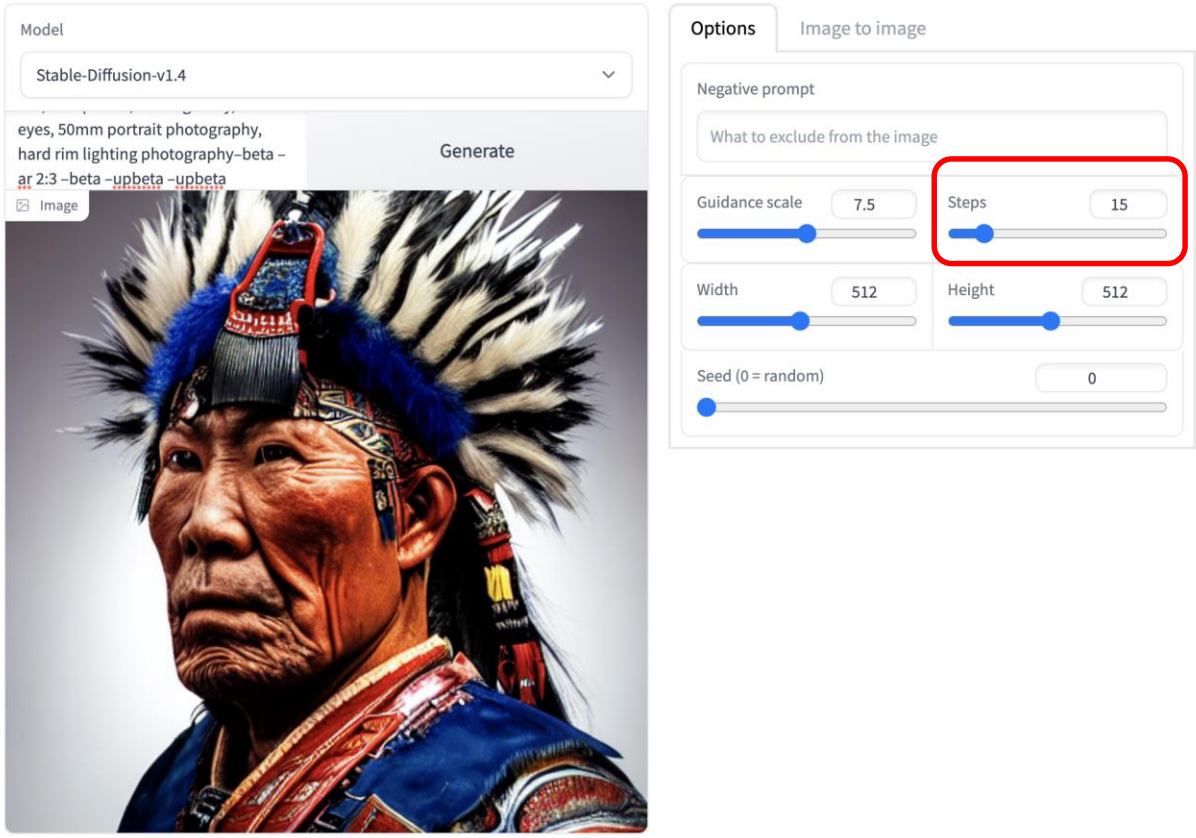
10-step diffusion models on CPU:

Stable-diffusion-WebUI for:

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.

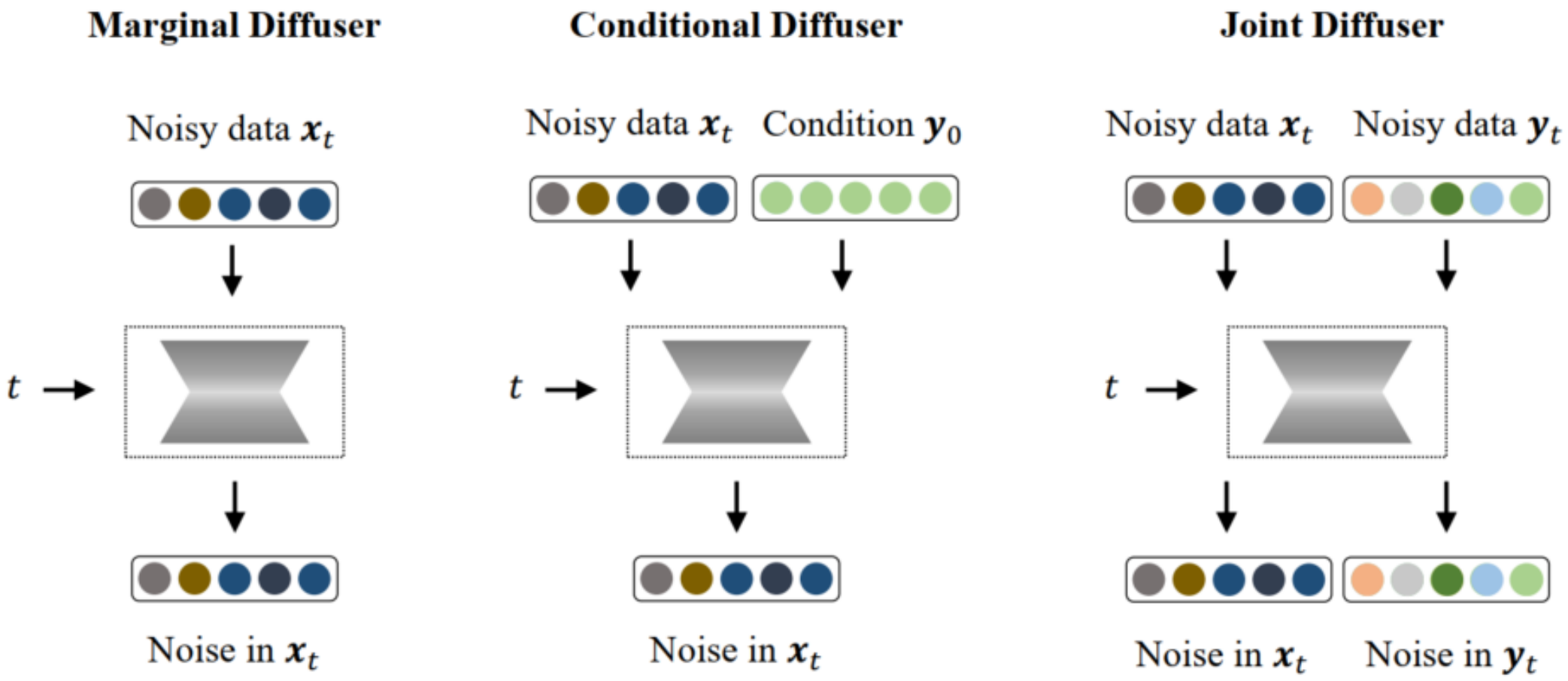


Diffusion Models for Multimodal Inference

- An ideal multimodal diffuser should be able to perform inference in various manners!

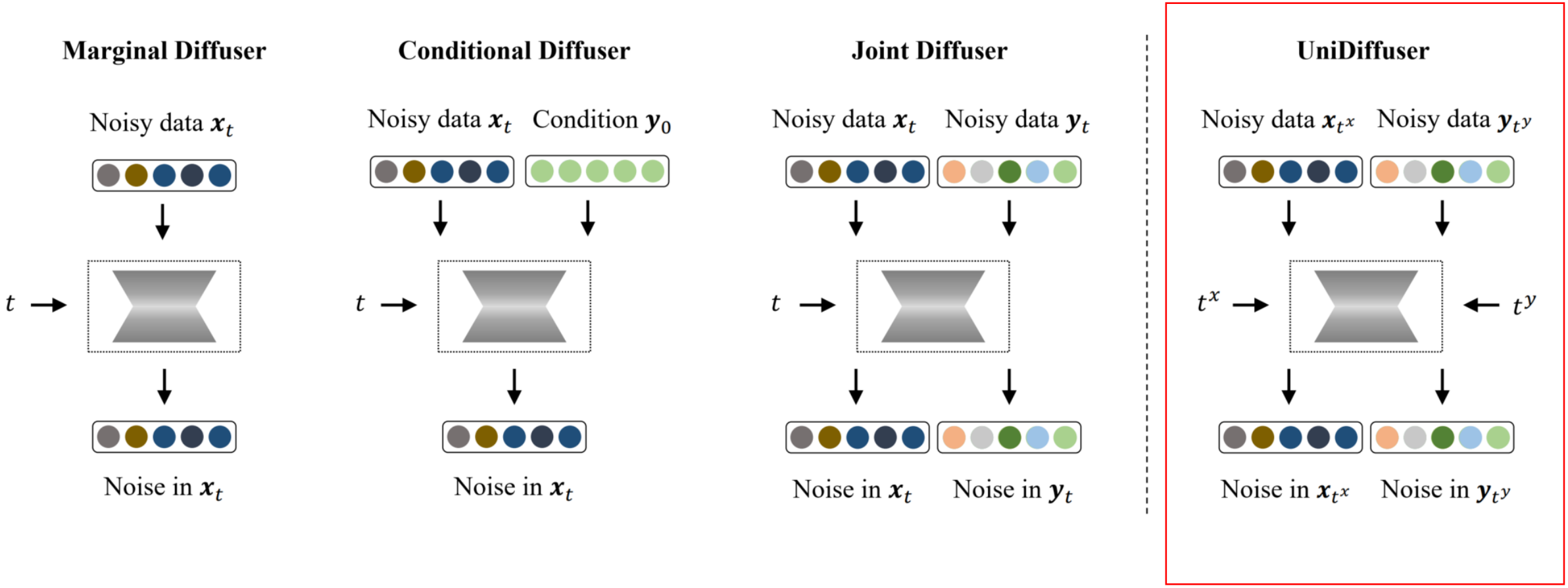
Diffusion Models for Multimodal Inference

- Traditional approaches learn a single diffusion model for each task!



Diffusion Models for Multimodal Inference

- We design UniDiffuser – a single model to fit all distributions in multimodel learning!



Diffusion Models for Multimodal Inference

- Key Insight: All distributions lead to an estimate of a conditional mean!

- Marginal distribution q(x0)-> $\mathbb{E}[\epsilon^x | x_t]$
- Conditional distribution q(x0|y0)-> $\mathbb{E}[\epsilon^x | x_t, y_0]$
- Joint distribution q(x0, y0) -> $\mathbb{E}[\epsilon^x, \epsilon^y | x_t, y_t]$

$\mathbb{E}[\epsilon^x, \epsilon^y | x_t, y_t]$

$\mathbb{E}[\epsilon^x, \epsilon^y | x_{t^x}, y_{t^y}]$

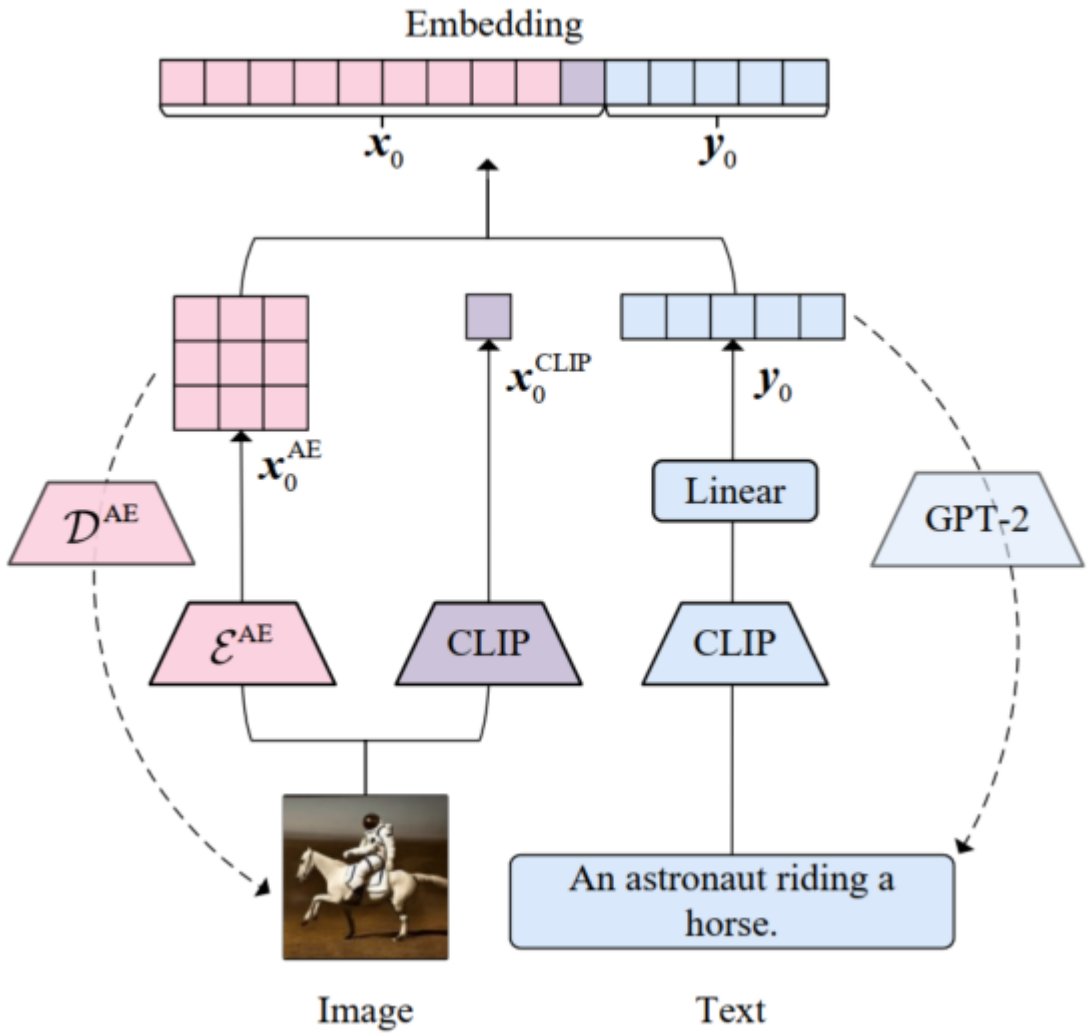
$t^y = T$ → $\mathbb{E}[\epsilon^x | x_t]$

$t^y = 0$ → $\mathbb{E}[\epsilon^x | x_t, y_0]$

$t^x = t^y = t$ → $\mathbb{E}[\epsilon^x, \epsilon^y | x_t, y_t]$

Diffusion Models for Multimodal Inference

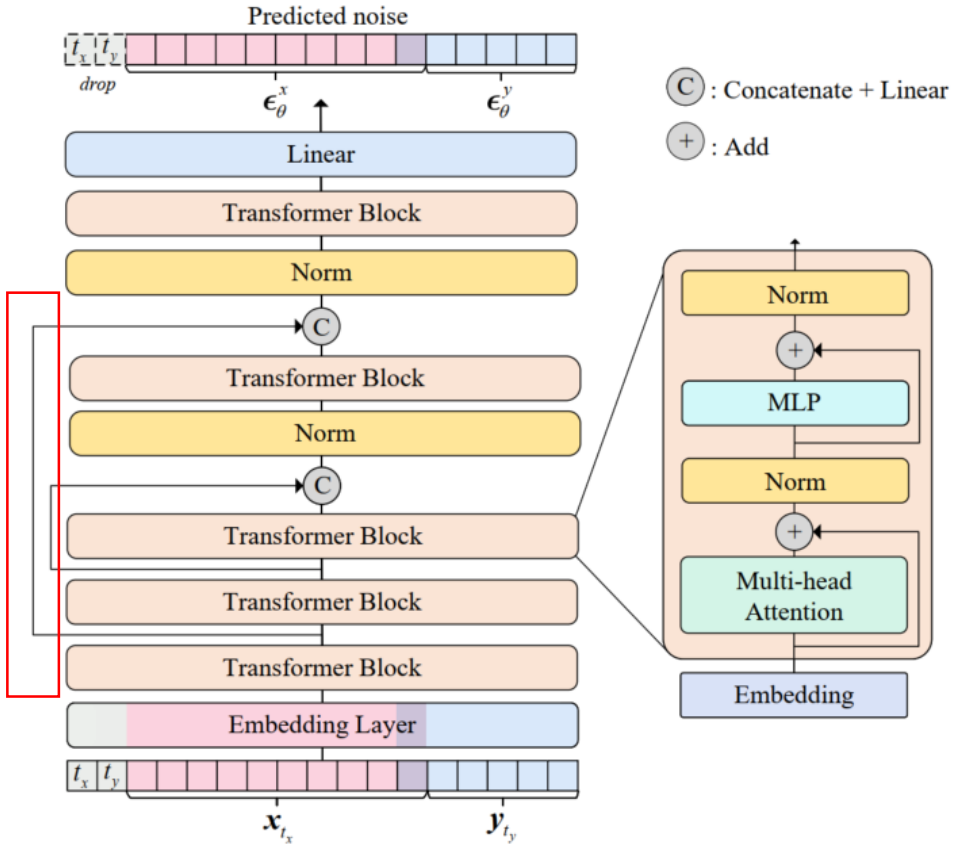
- More details on the backbone network
  - Consider image & text
    - Image is continuous
    - Text is discrete
  - Model in the latent space!



Diffusion Models for Multimodal Inference

- More details on the backbone network
  - Then we can use a transformer to model latent embeddings
  - We use the **U-ViT**
    - Everything as tokens
    - Long skip connections
    - All transformers except the last layer

Long skip connections between shallow and deep layers greatly improve the performance



Bao et al., All are Worth Words: A ViT Backbone for Diffusion Models, CVPR 2023

Experimental Results

- Comparison with multi-modal diffusion models

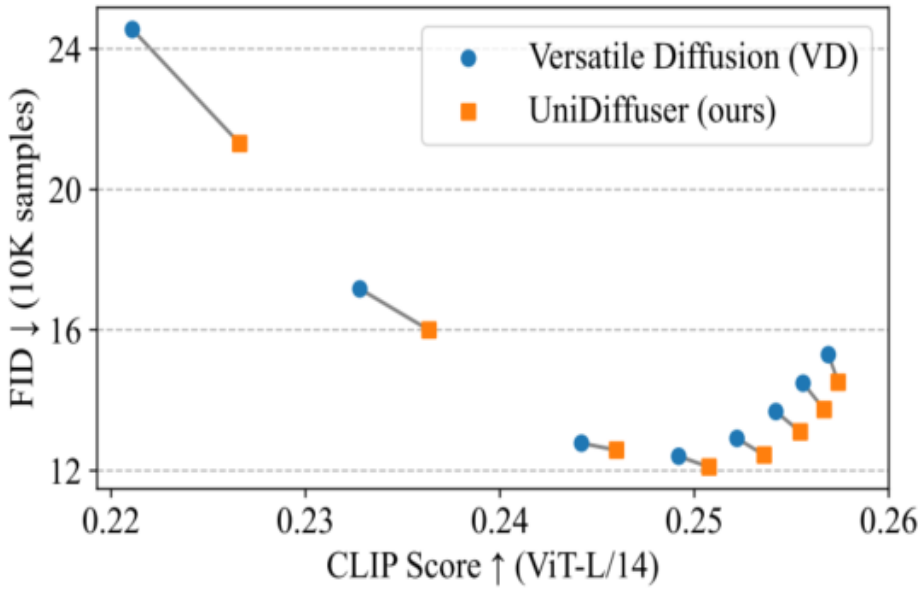


Figure 5. **Comparing UniDiffuser and VD in text-to-image generation.** We connect the results with the same scale in CFG. UniDiffuser consistently outperforms VD in all settings w.r.t. both the CLIP score ↑ (horizontal axis) and FID ↓ (vertical axis).

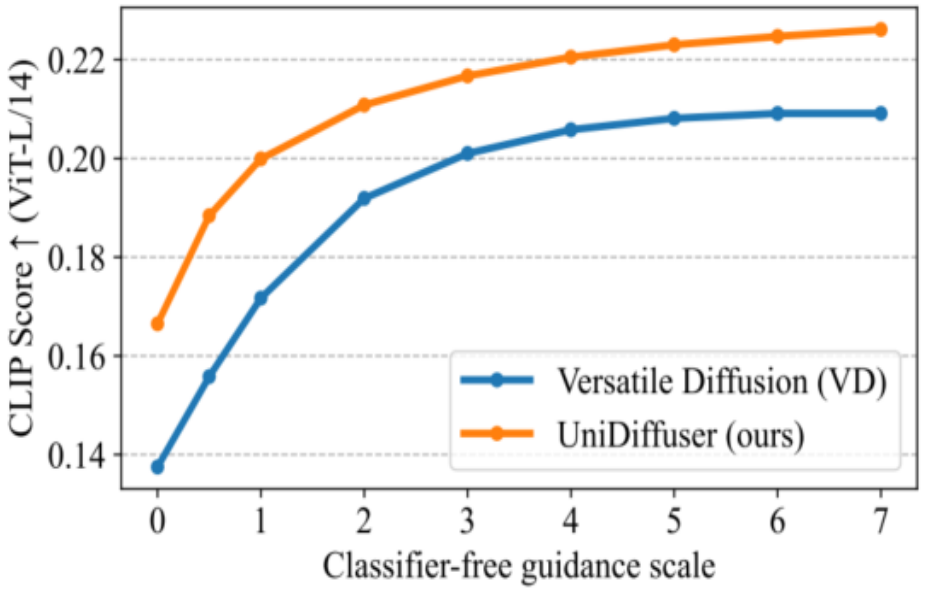


Figure 6. **Comparing UniDiffuser and VD in image-to-text generation.** UniDiffuser consistently outperforms VD with the same CFG scale (horizontal axis) w.r.t. the CLIP score ↑ (vertical axis).

Experimental Results

- Comparison with multi-modal diffusion models

Figure 7. **Random samples of UniDiffuser and VD on text-to-image generation.** UniDiffuser produces semantically correct images given representative prompts while VD does not.

Experimental Results

- Comparison with bespoken models for text-2-image generation

Table 1. **Zero-shot FID ↓ on the MS-COCO validation set.** † marks results produced by us upon official implementation and other results are taken from the corresponding references. We report the results of UniDiffuser and VD with a scale of 3 in CFG, which is the best choice for both models according to Figure 5.

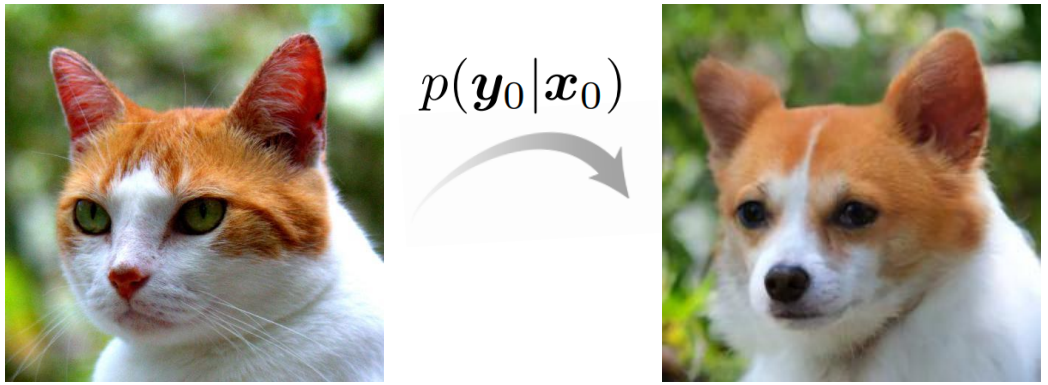
Model	FID ↓
Bespoken models	
GLIDE (Nichol et al., 2022)	12.24
Make-A-Scene (Gafni et al., 2022)	11.84
DALL·E 2 (Ramesh et al., 2022)	10.39
Stable Diffusion† (Rombach et al., 2022)	8.59
Imagen (Saharia et al., 2022)	7.27
Parti (Yu et al., 2022)	**7.23**
General-purpose models	
Versatile Diffusion† (Xu et al., 2022)	10.09
UniDiffuser (**ours**)	**9.71**

Other Applications with Novel Design of Diffusion Models

- Energy-guided SDE for Unpaired Image-2-Image translation (NeurIPS 2022)

source target

Training

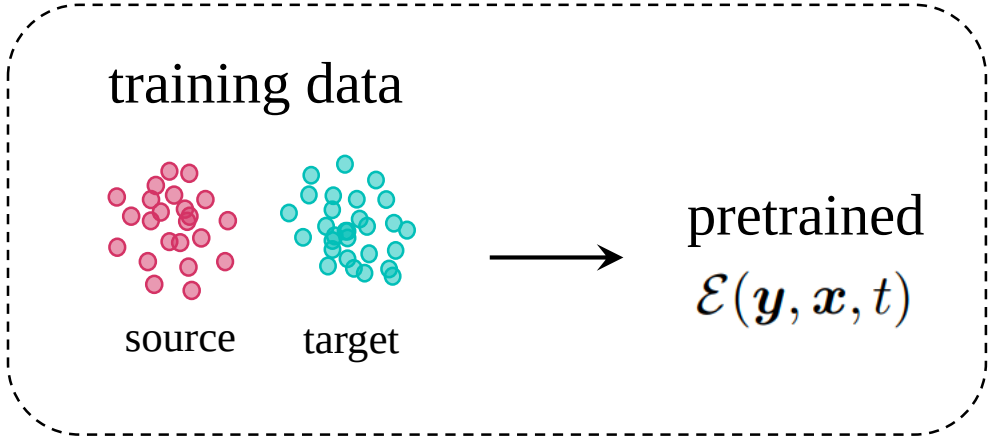


source image $\boldsymbol{x}_0$ translated image

Inference

Energy-Guided Stochastic Differential Equations (EGSDE)

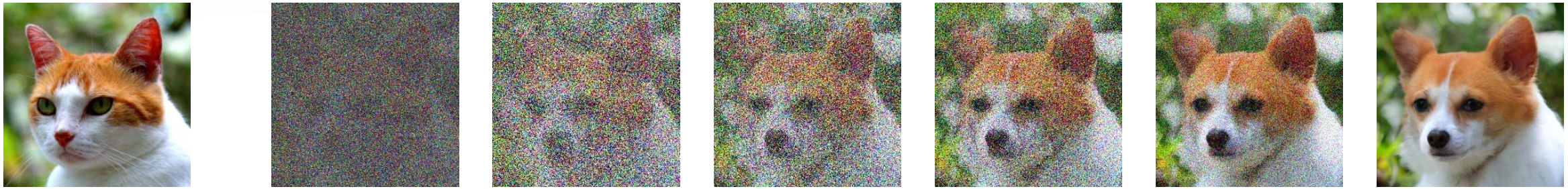
Zhao et al, NeurIPS 2022



$q_{M|0}(\boldsymbol{x}_M|\boldsymbol{x}_0)$

$\boldsymbol{y}_M = \boldsymbol{x}_M$

$p(\boldsymbol{y}_0|\boldsymbol{x}_0)$

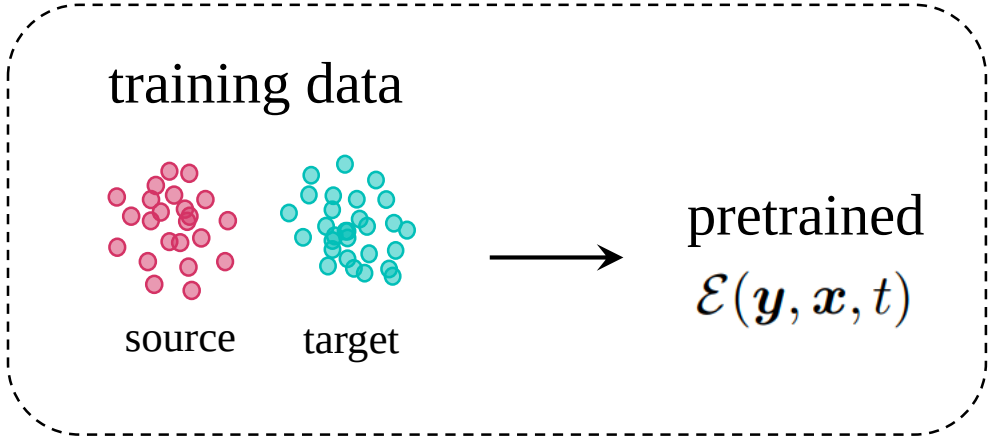


source image $\boldsymbol{x}_0$

$$\boldsymbol{y}_M \text{---} \mathrm{d}\boldsymbol{y} = [\boldsymbol{f}(\boldsymbol{y}, t) - g(t)^2(s(\boldsymbol{y}, t) - \lambda_s \nabla_{\boldsymbol{y}} \mathcal{E}_s(\boldsymbol{y}, \boldsymbol{x}_0, t) - \lambda_i \nabla_{\boldsymbol{y}} \mathcal{E}_i(\boldsymbol{y}, \boldsymbol{x}_0, t))]\mathrm{d}t + g(t)\mathrm{d}\overline{\boldsymbol{w}} \rightarrow \boldsymbol{y}_0$$

Energy-Guided Stochastic Differential Equations (EGSDE)

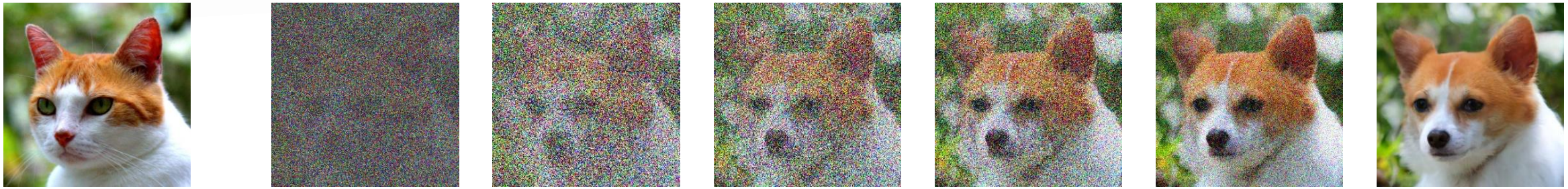
Zhao et al, NeurIPS 2022



$q_{M|0}(\boldsymbol{x}_M|\boldsymbol{x}_0)$

$\boldsymbol{y}_M = \boldsymbol{x}_M$

$p(\boldsymbol{y}_0|\boldsymbol{x}_0)$



source image $\boldsymbol{x}_0$

$$\boldsymbol{y}_M \text{---} \mathrm{d}\boldsymbol{y} = [\boldsymbol{f}(\boldsymbol{y}, t) - g(t)^2(s(\boldsymbol{y}, t) - \lambda_s \nabla_{\boldsymbol{y}} \mathcal{E}_s(\boldsymbol{y}, \boldsymbol{x}_0, t) - \lambda_i \nabla_{\boldsymbol{y}} \mathcal{E}_i(\boldsymbol{y}, \boldsymbol{x}_0, t))]\mathrm{d}t + g(t)\mathrm{d}\overline{\boldsymbol{w}} \rightarrow \boldsymbol{y}_0$$

known

Results

Zhao et al, NeurIPS 2022

Realistic **Faithful** ***Human Evaluation, Both***

Model	FID ↓	L2 ↓	PSNR ↑	SSIM ↑	AMT ↑
		Cat → Dog			
CycleGAN* [54]	85.9	-	-	-	-
MUNIT* [17]	104.4	-	-	-	-
DRIT* [25]	123.4	-	-	-	-
Distance* [3]	155.3	-	-	-	-
SelfDistance* [3]	144.4	-	-	-	-
GCGAN* [10]	96.6	-	-	-	-
LSeSim* [52]	72.8	-	-	-	-
ITTR (CUT)* [53]	68.6	-	-	-	-
StarGAN v2 [8]	54.88 ± 1.01	133.65 ± 1.54	10.63 ± 0.10	0.27 ± 0.003	-
CUT* [34]	76.21	59.78	17.48	**0.601**	79.6%
ILVR [7]	74.37 ± 1.55	56.95 ± 0.14	17.77 ± 0.02	0.363 ± 0.001	75.4%
SDEdit [30]	74.17 ± 1.01	47.88 ± 0.06	19.19 ± 0.01	**0.423 ± 0.001**	65.2%
EGSDE	**65.82 ± 0.77**	**47.22 ± 0.08**	**19.31 ± 0.02**	0.415 ± 0.001	-
EGSDE†	**51.04 ± 0.37**	62.06 ± 0.10	17.17 ± 0.02	0.361 ± 0.001	-

$$\text{EGSDE} : \lambda_s = 500, \lambda_i = 2, M = 0.5\text{T}$$
$$\text{EGSDE}^{\dagger}: \lambda_s = 700, \lambda_i = 0.5, M = 0.6\text{T}$$

Results

Zhao et al, NeurIPS 2022

Model	FID ↓	L2 ↓	PSNR ↑	SSIM ↑	AMT ↑
		Cat → Dog			
CycleGAN* [54]	85.9	-	-	-	-
MUNIT* [17]	104.4	-	-	-	-
DRIT* [25]	123.4	-	-	-	-
Distance* [3]	155.3	-	-	-	-
SelfDistance* [3]	144.4	-	-	-	-
StarGAN v2 [8]	54.88 ± 1.01	133.65 ± 1.54	10.63 ± 0.10	0.27 ± 0.003	-
CUT* [34]	76.21	59.78	17.48	**0.601**	79.6%
ILVR [7]	74.37 ± 1.55	56.95 ± 0.14	17.77 ± 0.02	0.363 ± 0.001	75.4%
SDEdit [30]	74.17 ± 1.01	47.88 ± 0.06	19.19 ± 0.01	**0.423 ± 0.001**	65.2%
EGSDE	**65.82 ± 0.77**	**47.22 ± 0.08**	**19.31 ± 0.02**	0.415 ± 0.001	-
EGSDE†	**51.04 ± 0.37**	62.06 ± 0.10	17.17 ± 0.02	0.361 ± 0.001	-

EGSDE outperforms the SBDMs-based methods in almost all metrics

$$\text{EGSDE}: \lambda_s = 500, \lambda_i = 2, M = 0.5\text{T}$$
$$\text{EGSDE}^{\dagger}: \lambda_s = 700, \lambda_i = 0.5, M = 0.6\text{T}$$

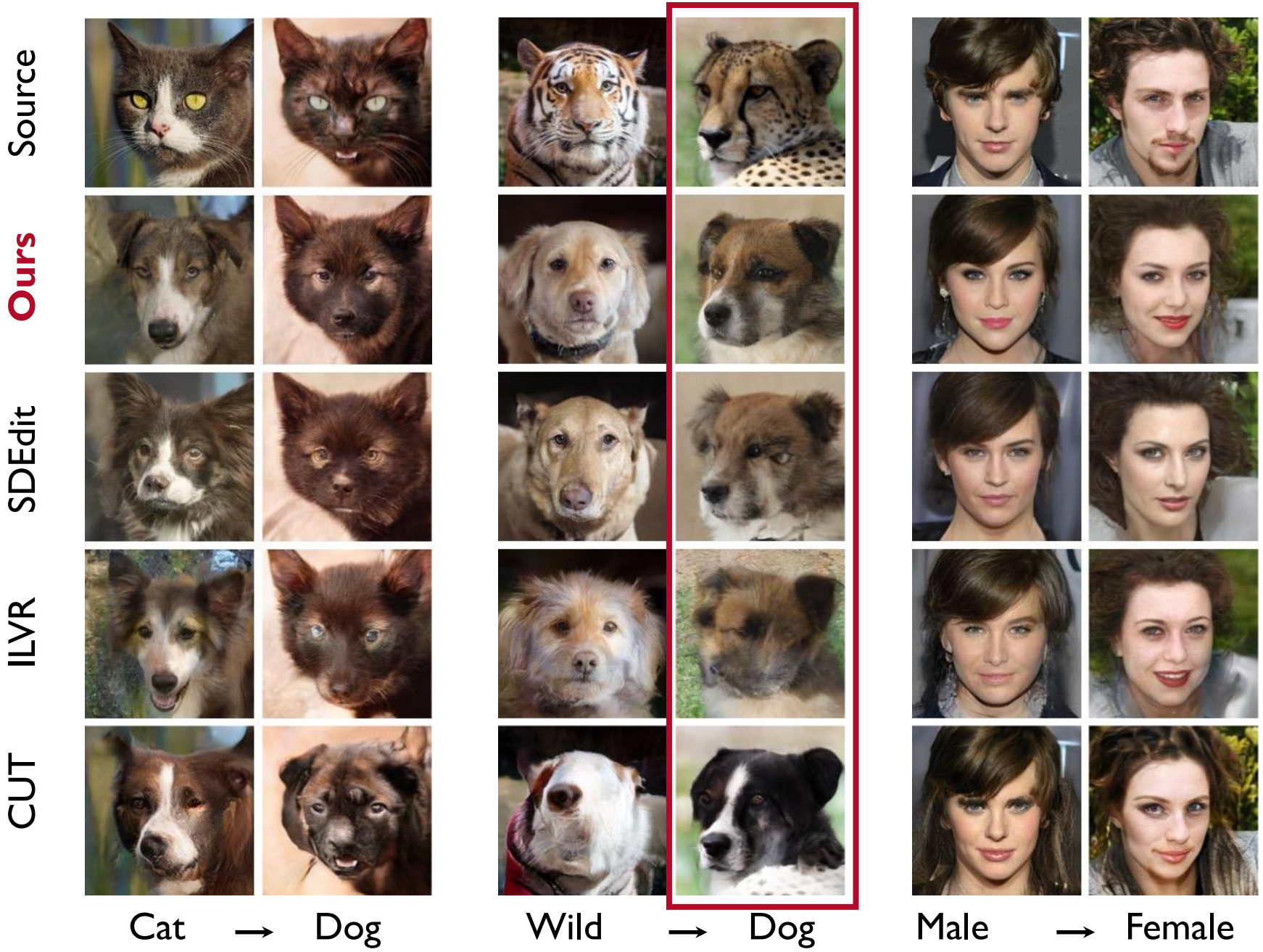
Results

Zhao et al, NeurIPS 2022

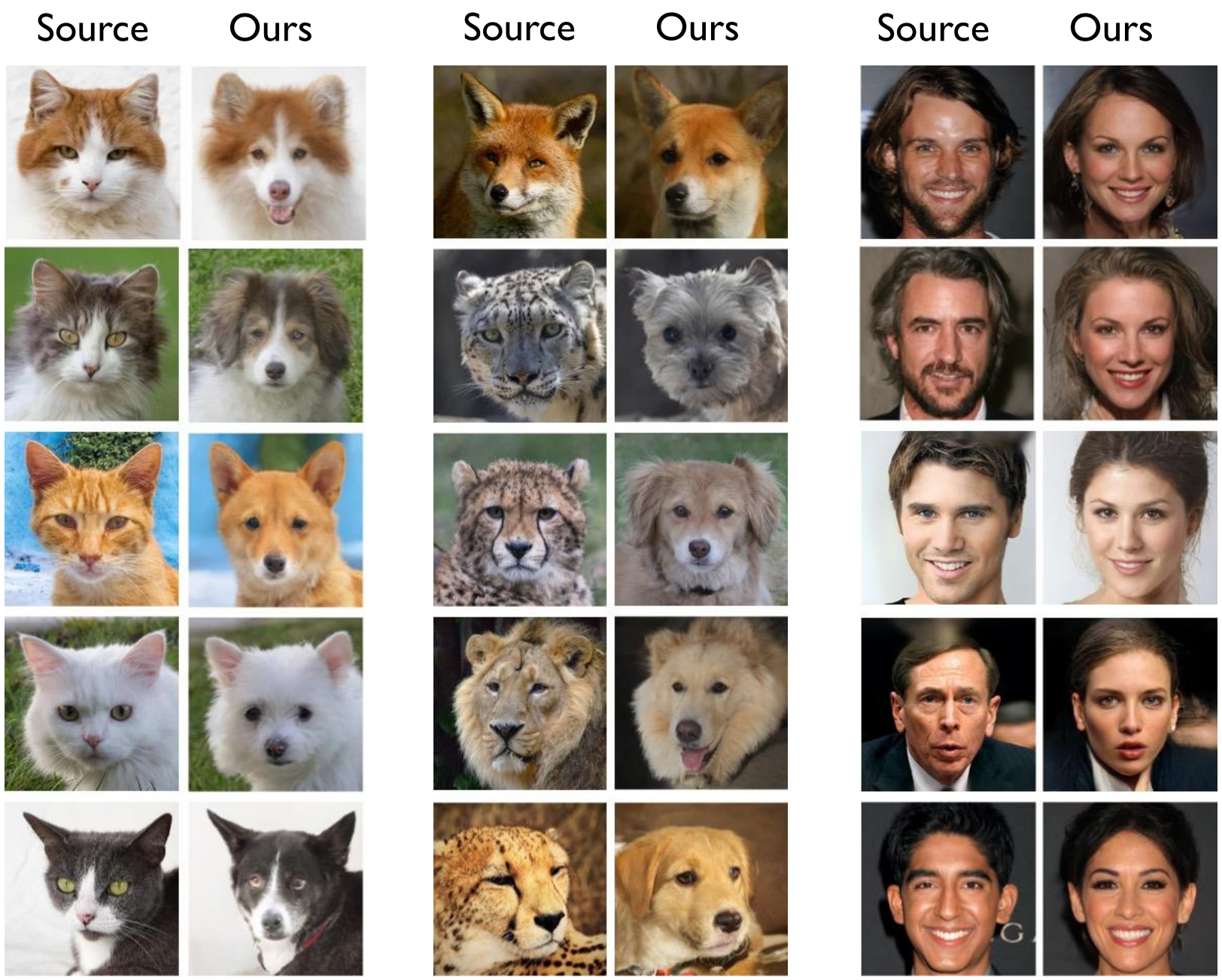
Model	FID ↓	L2 ↓	PSNR ↑	SSIM ↑	AMT ↑
		Cat → Dog			
CycleGAN* [54]	85.9	-	-	-	-
MUNIT* [17]	104.4	-	-	-	-
DRIT* [25]	123.4	-	-	-	-
Distance* [3]	155.3	-	-	-	-
SelfDistance* [3]	144.4	-	-	-	-
GCGAN* [10]	96.6	-	-	-	-
LSeSim* [52]	72.8	-	-	-	-
ITTR (CUT)* [53]	68.6	-	-	-	-
StarGAN v2 [8]	54.88 ± 1.01	133.65 ± 1.54	10.63 ± 0.10	0.27 ± 0.003	-
CUT* [34]	76.21	59.78	17.48	**0.601**	79.6%
ILVR [7]	74.37 ± 1.55	56.95 ± 0.14	17.77 ± 0.02	0.363 ± 0.001	75.4%
SDEdit [30]	74.17 ± 1.01	47.88 ± 0.06	19.19 ± 0.01	**0.423 ± 0.001**	65.2%
EGSDE	**65.82 ± 0.77**	**47.22 ± 0.08**	**19.31 ± 0.02**	0.415 ± 0.001	-
EGSDE†	**51.04 ± 0.37**	62.06 ± 0.10	17.17 ± 0.02	0.361 ± 0.001	-

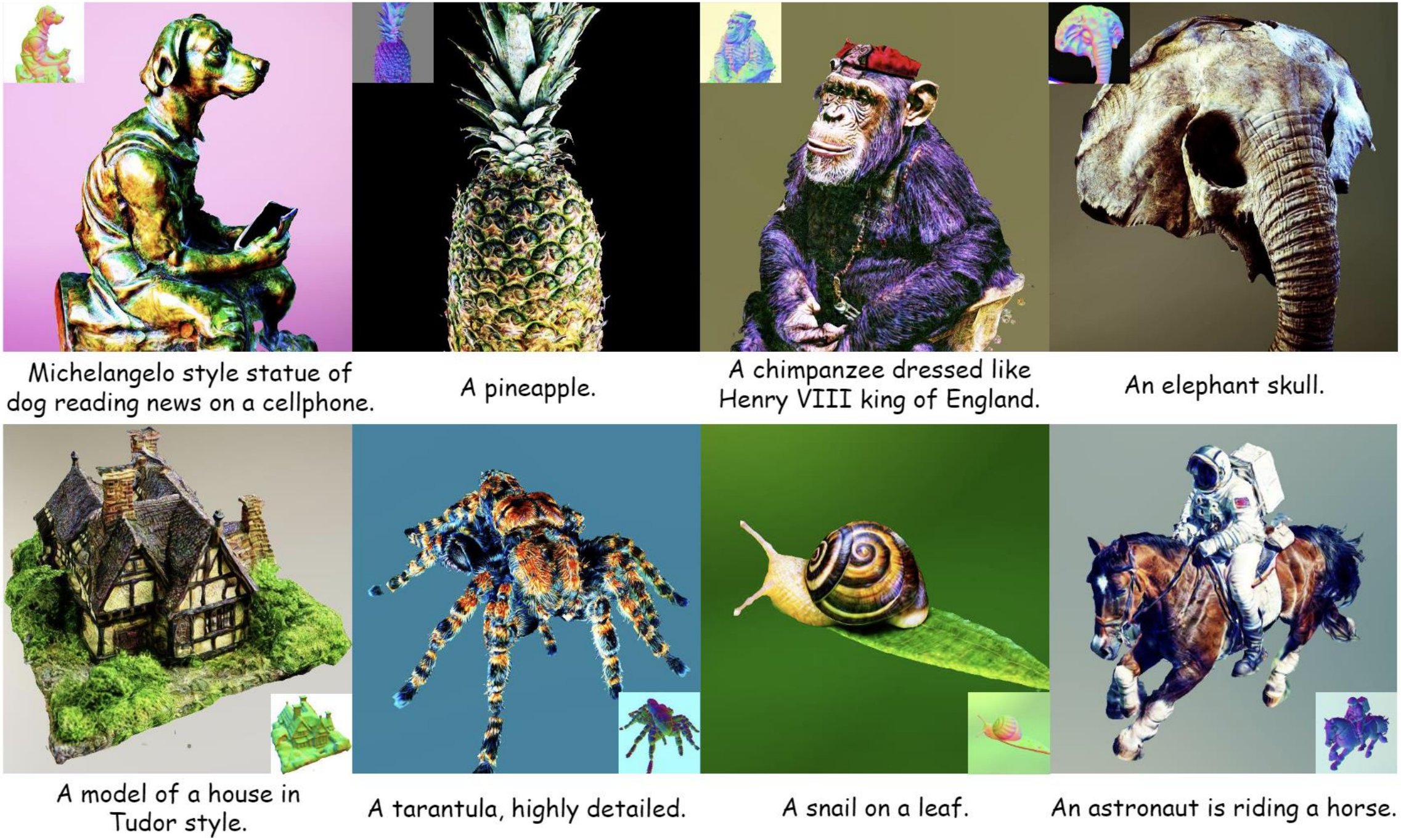
EGSDE outperforms the current state-of-art GANs-based methods

Results



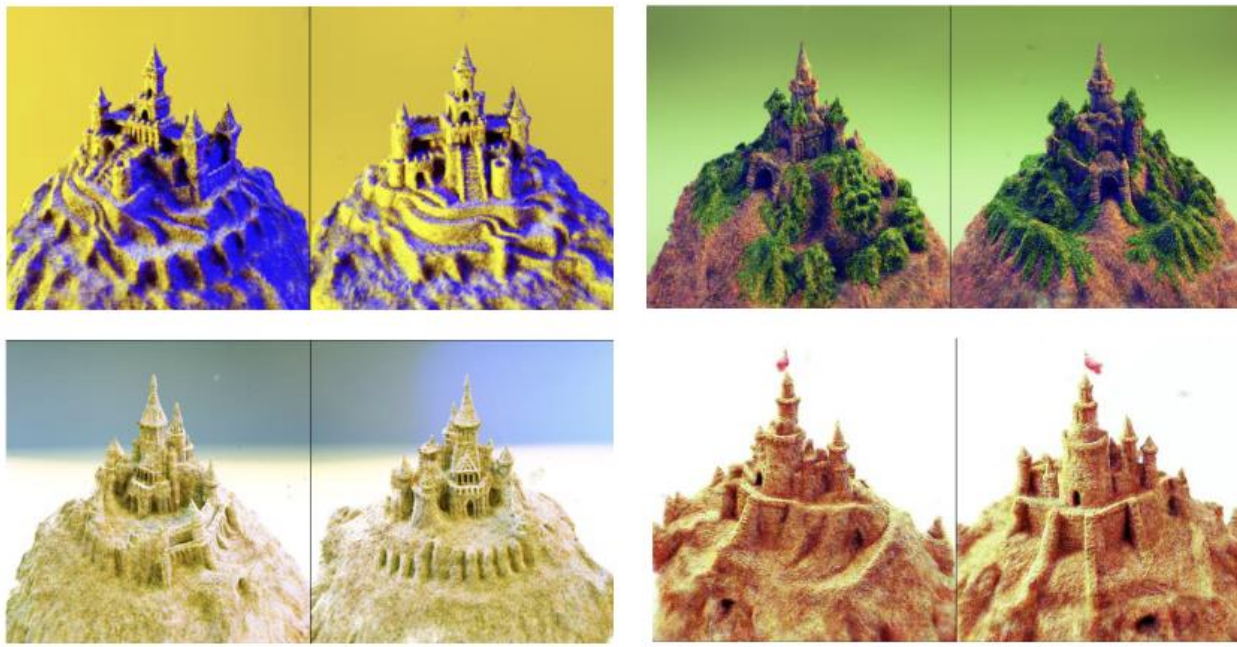
Results

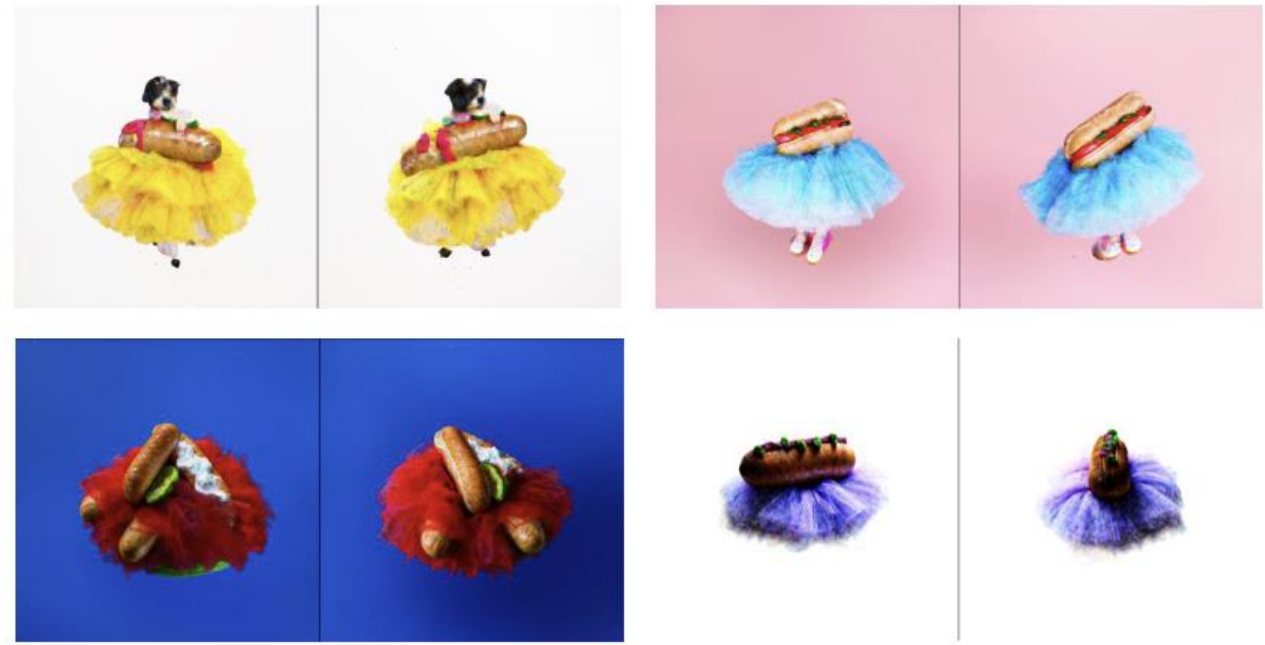

Source
Ours
Source
Ours
Source
Ours
Cat → Dog
Wild → Dog
Male → Female



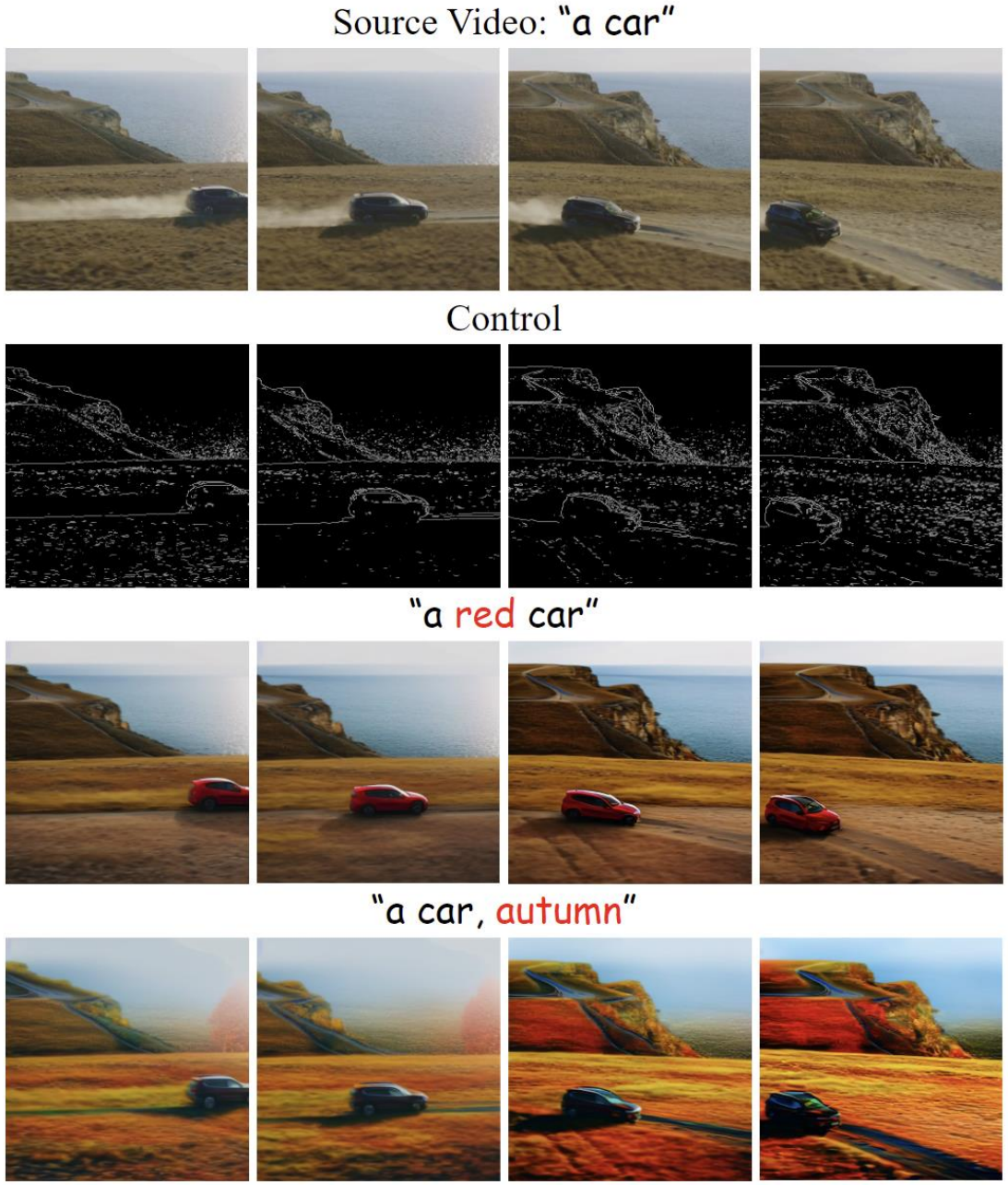
(a) ProlificDreamer can generate meticulously detailed and photo-realistic 3D textured meshes.

(b) ProlificDreamer can generate high rendering resolution (i.e., 512×512) and high-fidelity NeRF with rich structures and complex effects. Besides, the bottom results show that ProlificDreamer can generate complex scenes with 360° views because of our *scene initialization* (see Sec. 4.2).





(c) ProlificDreamer can generate diverse and semantically correct 3D scenes given the same text.



(a) Canny Edge Map Control

(b) HED Boundary Control

Source Video: "the back view of a woman"

Control

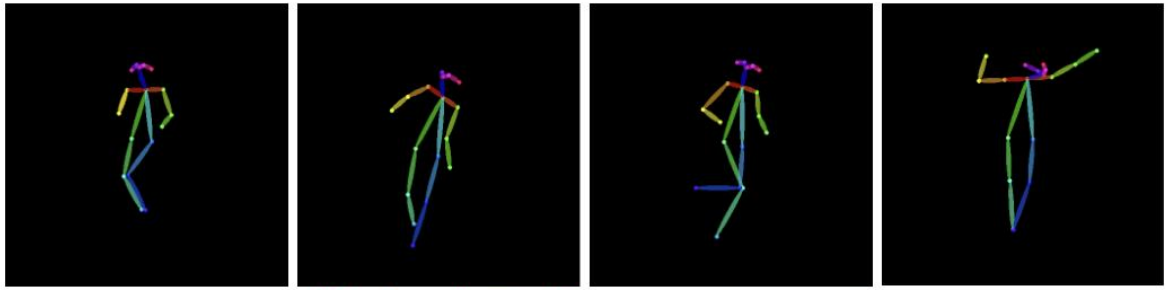
"+ A. J. Casson style"

"+ Vivian Maier style, on a street beside a building"

(c) Depth Map Control

Source Video : "a person is dancing"

Control



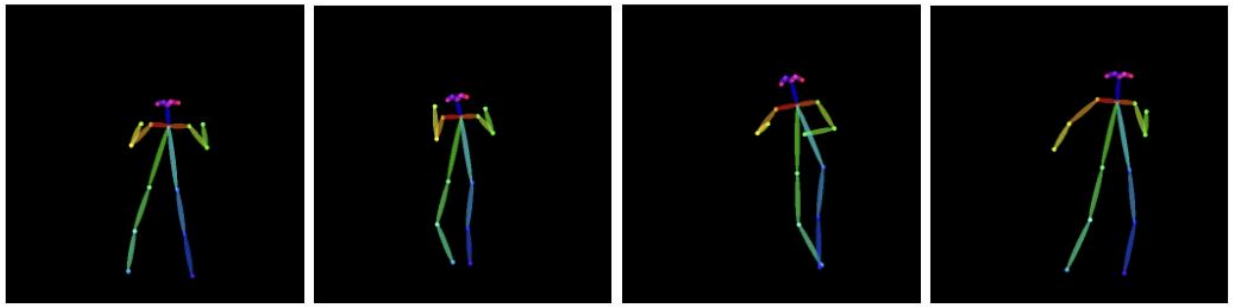
"Michael Jackson is dancing"

"Sherlock Holmes is dancing, on the street of London, raining"

(d) Pose Control

Source Video: "a person is dancing"

Control 1



Control 2

"a panda is dancing"

(e) Multiple Control

Summary

- Diffusion probabilistic models are powerful in image synthesis, image-2-image translation, crossmodal inference, reinforcement learning, etc
- We propose Analytic-DPM that proves analytical solutions of the mean and variance, with **20 to 80** $\times$ speed up
- We propose DPM-Solver based on a highly simplified formulation of the exact solutions of diffusion ODEs; it can generate high-quality samples in around **10** steps and almost converge in **20** steps.
- We propose a unified diffusion model for crossmodal inference, and an energy-guided SDE for image-2-image translation

- Check out https://ml.cs.tsinghua.edu.cn/diffusion/RelatedWork for other progress on molecular design, RL, backbone architecture, etc