Diffusion Probabilistic Models: Foundations, Fast Inference and Controllable Generation

Chongxuan Li

ML Group @ RUC

Renmin University of China

Fan Bao

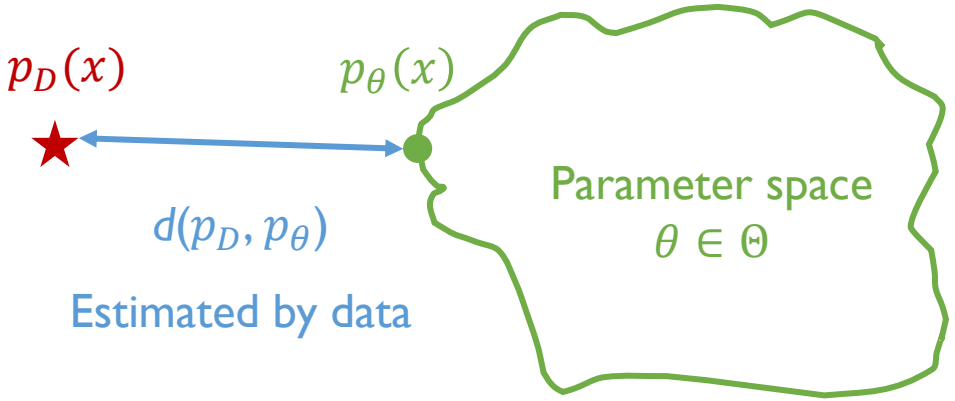
TSAIL

Tsinghua University

Generative modeling

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

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



Goal of generative modeling: $p_\theta(x) \approx p_D(x)$.

Image credit: Fisher Yu, https://www.yf.io/p/lsun

Diffusion in Physics

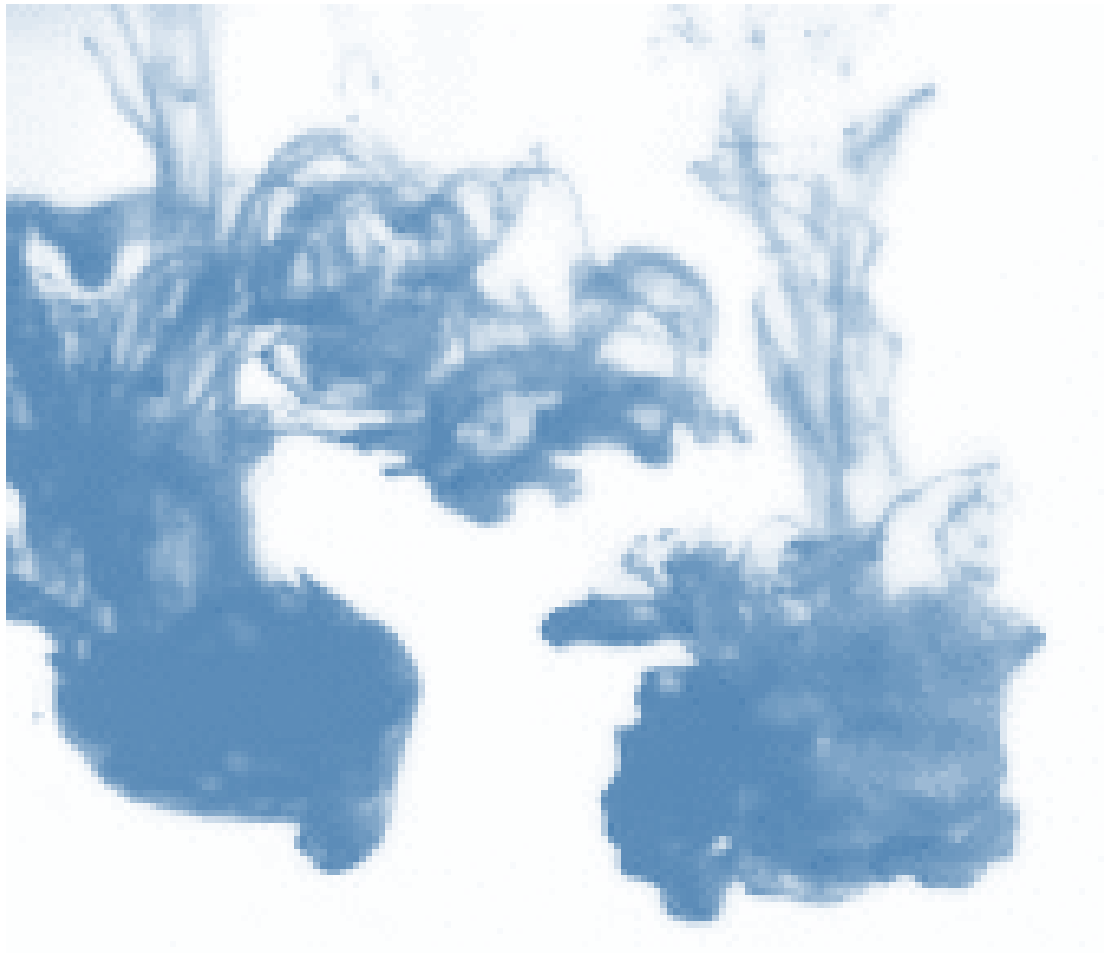
Diffusion destroy structures along time

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

Diffusion in Physics

Diffusion destroy structures along time

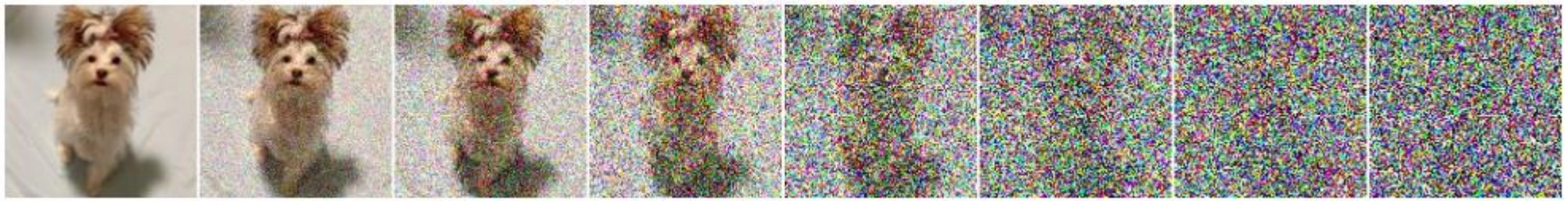
What if we can reverse time?



Diffusion-based score models

Add Markovian Gaussian noise

Data distribution → Gaussian noise



Model distribution ← Gaussian prior

Learn to denoise/generate data

Image credit: *Sohl-Dickstein*

Text-to-image generation

Nichol & Dhariwal et al, ICML 2022

"a hedgehog using a calculator"

"a corgi wearing a red bowtie and a purple party hat"

"robots meditating in a vipassana retreat"

"a fall landscape with a small cottage next to a lake"

"a surrealist dream-like oil painting by salvador dalí of a cat playing checkers"

"a professional photo of a sunset behind the grand canyon"

"a high-quality oil painting of a psychedelic hamster dragon"

"an illustration of albert einstein wearing a superhero costume"

Real-world applications

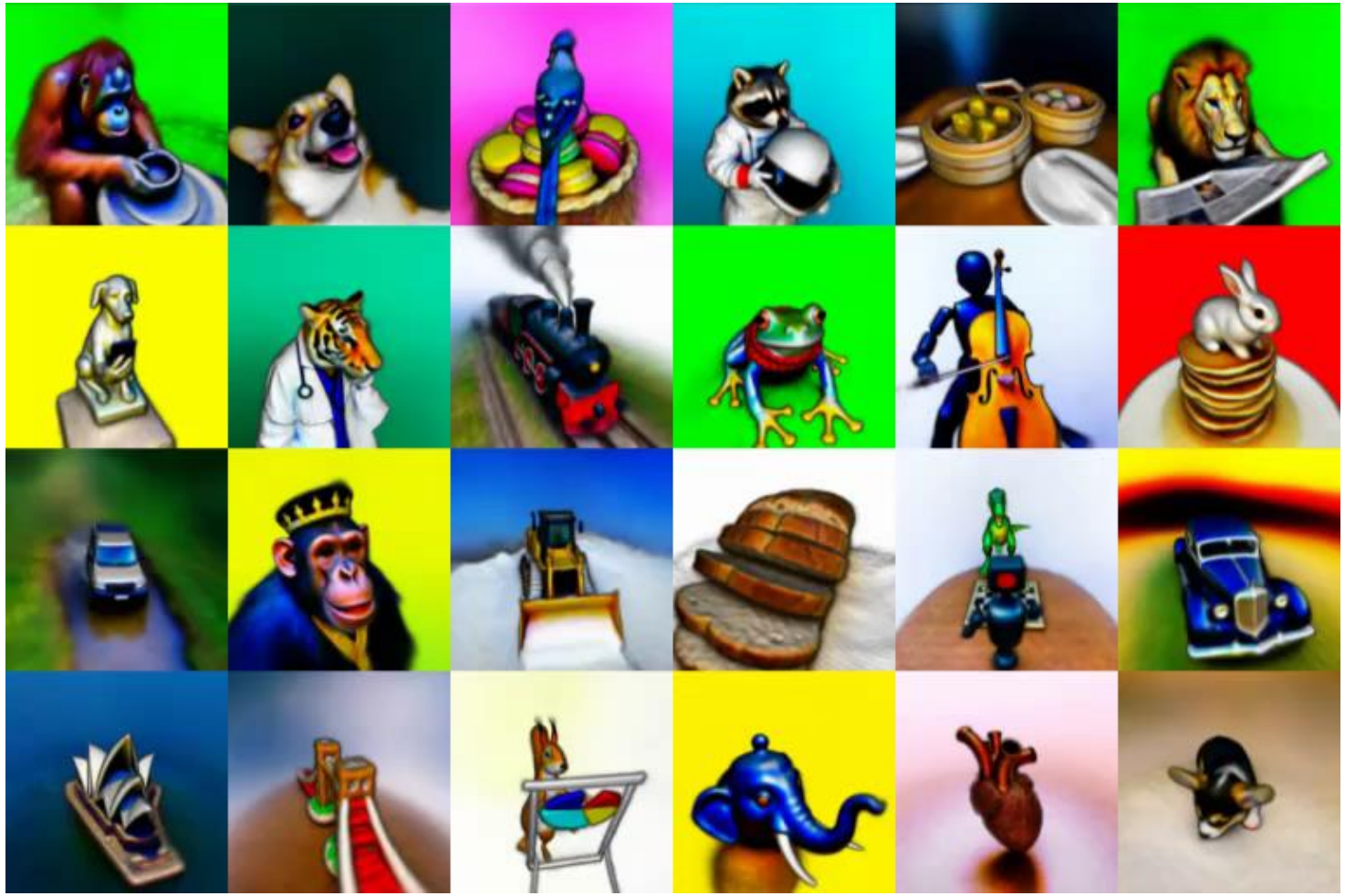
Video: https://twitter.com/karenxcheng/status/1541438655327133697

Video generation

Jonathon et al, Arxiv Preprint 2022

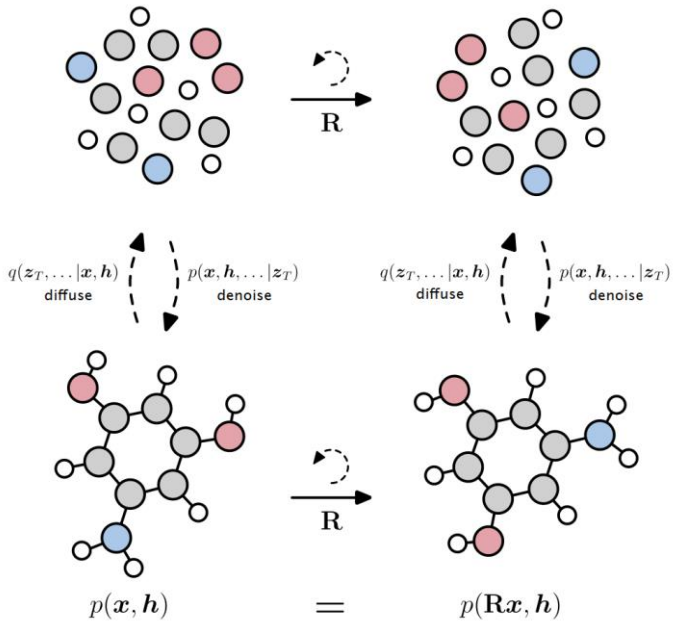
3D scene generation

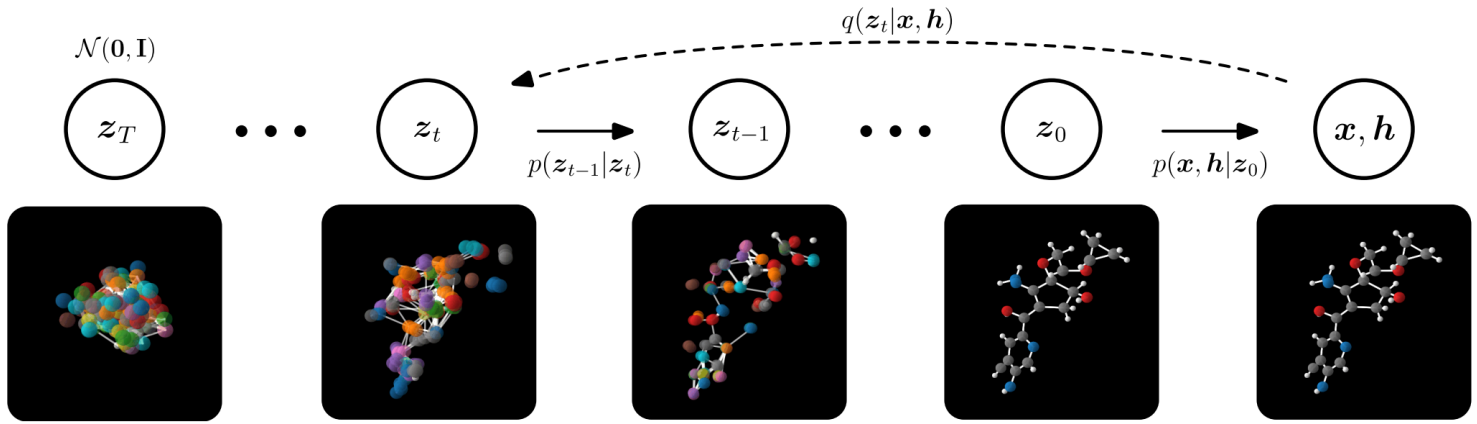
Ben et al, Arxiv Preprint 2022



Molecule generation

Hoogeboom et al, ICML 2021





Foundation of Diffusion Models

Diffusion probabilistic models

Sohl-Dickstein et al, ICML 2015

Forward diffusion: a Markov chain with Gaussian kernel with 1000 steps typically

Data distribution → Adding noise → Gaussian noise

$$q\left(\mathbf{x}^{(0)}\right)$$

$$q\left(\mathbf{x}^{(T)}\right) \approx \mathcal{N}\left(\mathbf{x}^{(T)}; 0, \mathbf{I}\right)$$

Simple forward kernel

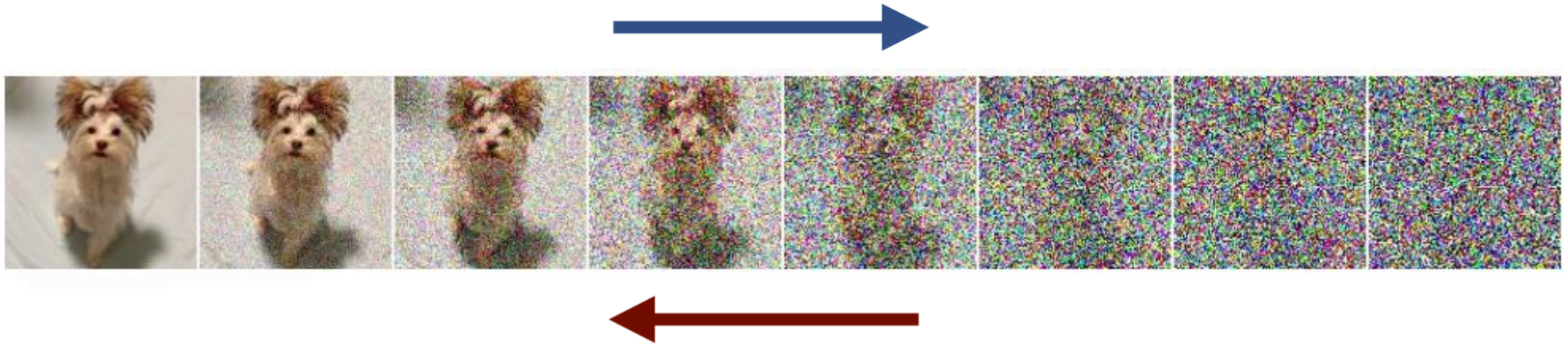
$$q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \mathbf{x}^{(t-1)}\sqrt{1-\beta_t}, \mathbf{I}\beta_t\right)$$

Decay towards origin Add small noise

Image credit: *Sohl-Dickstein*

Diffusion probabilistic models

Sohl-Dickstein et al, ICML 2015



Core idea: learn to map noise to data by reversing the time

- We can get noise from any data distribution
- The forward process is reversible
- The backward process has the same functional form

Image credit: *Sohl-Dickstein*

Diffusion probabilistic models

Sohl-Dickstein et al, ICML 2015

Backward diffusion: a Markov chain with learnable Gaussian kernel

Model distribution

$$p\left(\mathbf{x}^{(0)}\right) \approx q\left(\mathbf{x}^{(0)}\right)$$

Learn to denoise

Gaussian prior

$$p\left(\mathbf{x}^{(T)}\right) = \mathcal{N}\left(\mathbf{x}^{(T)}; 0, \mathbf{I}\right)$$

Learnable reverse kernel

$$p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right) = \mathcal{N}\left(\mathbf{x}^{(t-1)}; f_\mu\left(\mathbf{x}^{(t)}, t\right), f_\Sigma\left(\mathbf{x}^{(t)}, t\right)\right)$$

Learned drift and covariance functions

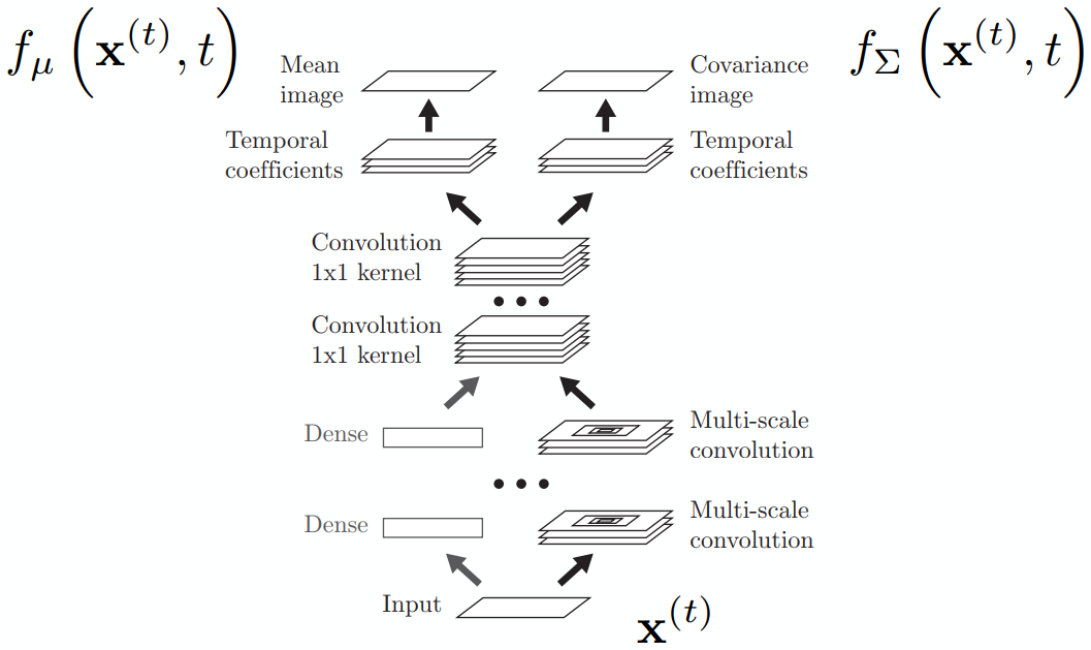
Image credit: *Sohl-Dickstein*

Diffusion probabilistic models

Sohl-Dickstein et al, ICML 2015

$$p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right) = \mathcal{N}\left(\mathbf{x}^{(t-1)}; f_\mu\left(\mathbf{x}^{(t)}, t\right), f_\Sigma\left(\mathbf{x}^{(t)}, t\right)\right)$$

Learned drift and covariance functions



Parameterized by a time-conditioned NN

Image credit: *Sohl-Dickstein*

Training DPMs by Maximum Likelihood

Sohl-Dickstein et al, ICML 2015

likelihood ELBO

$$\mathrm{E}_{q(x_0)}[\log p_\theta(x_0)] = \mathrm{E}_{q(x_0)}\left[\log \int p_\theta(x_{0:T})dx_{1:T}\right] \geq \mathrm{E}_{q(x_0)}\mathrm{E}_{q(x_{1:T}|x_0)} \log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}$$

Jensen's inequality: equality holds when $q(x_{1:T}|x_0) = p(x_{1:T}|x_0)$

Remember that The backward process has the same functional form as the forward one.

DPM can be understood as a hierarchical latent variable model with fixed variational posterior.

Training DPMs by Maximum Likelihood

Sohl-Dickstein et al, ICML 2015

$$\mathrm{E}_{q(x_{0:T})} \log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} = \mathrm{E}_q \left[L_T + L_0 - \sum KL(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) \right]$$

ELBO

Decomposition for efficiency

- Naïve approach
  - $O(T)$ time complexity for a single update; single stochasticity
- Decomposition approach
  - Treat the summation as expectation and random sample a time step
  - $O(1)$ time complexity for a single update; double stochasticity
  - We will explain the $O(1)$ time later

Training DPMs by Maximum Likelihood

Sohl-Dickstein et al, ICML 2015

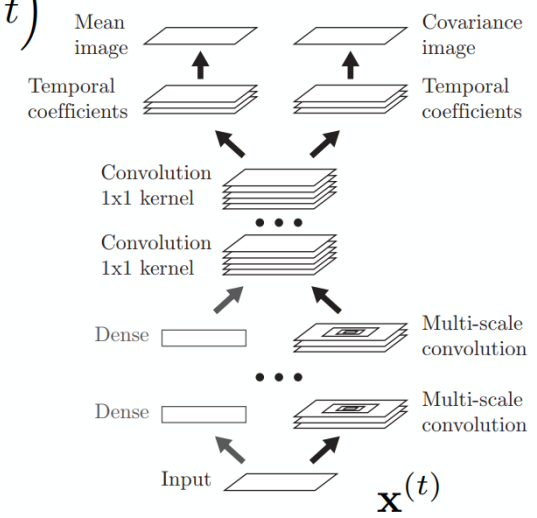
$$\mathrm{E}_{q(x_{0:T})} \log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} = \mathrm{E}_q \left[L_T + L_0 - \sum KL(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t)) \right]$$

$$q(x_{t-1}|x_t, x_0) = N(\tilde{\mu}_t(x_0, x_t), \tilde{\beta}_t I)$$

Closed-form

$$p_\theta(x_{t-1}|x_t) = N\left(f_\mu(x_t, t), f_\Sigma(x_t, t)\right)$$

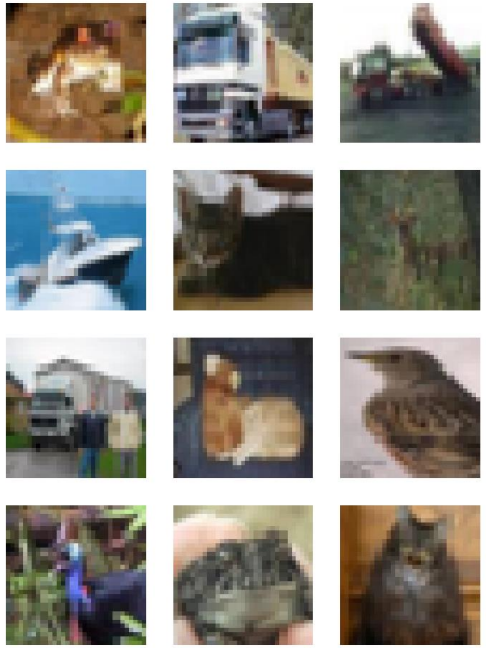
$f_\mu\left(\mathbf{x}^{(t)}, t\right)$ $f_\Sigma\left(\mathbf{x}^{(t)}, t\right)$



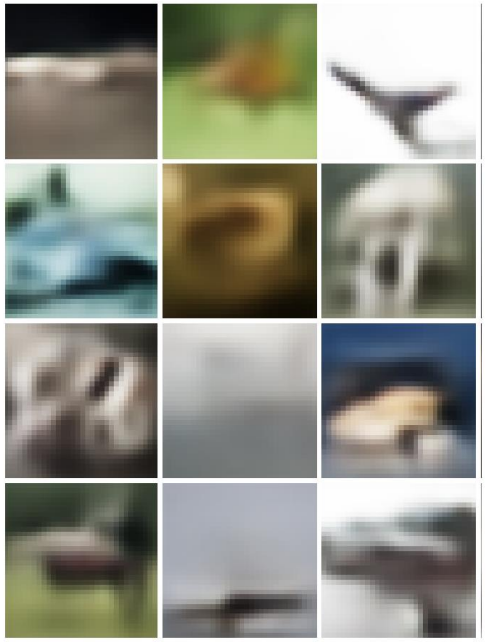
Learning as regression

Results for DPM

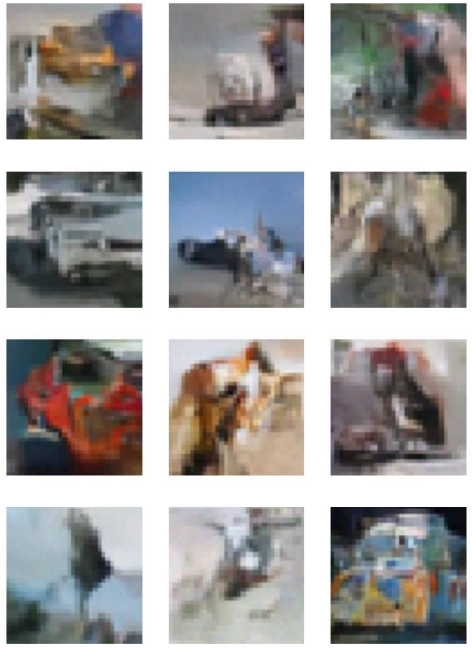
Sohl-Dickstein et al, ICML 2015



Training Data



Samples from
DRAW
[Gregor *et al*, 2015]



Samples from
diffusion model

MLE for a hierarchical latent variable model is equivalent to

learning score functions for data distributions with various noise

Denoising diffusion probabilistic models

Jonathon et al., NeurIPS 2021

$$\mathrm{E}_{q(x_0)}[\log p_\theta(x_0)] \geq \mathrm{E}_{q(x_{0:T})} \log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} = \mathrm{E}_q \left[L_T + L_0 - \sum KL(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) \right]$$

Regress mean with fixed variance

$$\|\mu_\theta(x_t, t) - \tilde{\mu}_t(x_0, x_t)\|^2$$

Denoising diffusion probabilistic models

Jonathon et al., NeurIPS 2021

$$\mathrm{E}_{q(x_0)}[\log p_\theta(x_0)] \geq \mathrm{E}_{q(x_{0:T})} \log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} = \mathrm{E}_q \left[L_T + L_0 - \sum KL(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) \right]$$

$$\mu_\theta(x^{(t)}, t) \rightarrow \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x^{(0)} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x^{(t)}$$
$$= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \frac{1}{\sqrt{\bar{\alpha}_t}} (x^{(t)} - \sqrt{1-\bar{\alpha}_t}\epsilon_t) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x^{(t)}$$
$$= \frac{1}{\sqrt{\alpha_t}} \left(\frac{\beta_t}{1-\bar{\alpha}_t} x^{(t)} + \frac{\alpha_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x^{(t)} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t \right)$$
$$= \frac{1}{\sqrt{\alpha_t}} \left(\frac{\beta_t + \alpha_t - \bar{\alpha}_t}{1-\bar{\alpha}_t} x^{(t)} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t \right) = \frac{1}{\sqrt{\alpha_t}} \left(x^{(t)} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t \right)$$

Regress mean with fixed variance $\|\mu_\theta(x_t, t) - \tilde{\mu}_t(x_0, x_t)\|^2$

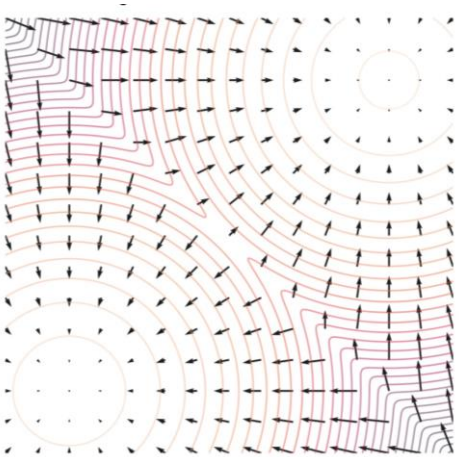
Reparameterization: $x_t = \sqrt{\alpha_t} x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$

Regress Gaussian noise $\|\epsilon_\theta(x_t, t) - \epsilon\|^2$

Denoising diffusion probabilistic models

Jonathon et al., NeurIPS 2021

$$\mathrm{E}_{q(x_0)}[\log p_\theta(x_0)] \geq \mathrm{E}_{q(x_{0:T})} \log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} = \mathrm{E}_q \left[L_T + L_0 - \sum KL(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) \right]$$



$p_\theta(\mathbf{x})$ vs. $s_\theta(\mathbf{x})$

Image credit: *Yang Song*

Regress mean with fixed variance $\|\mu_\theta(x_t, t) - \tilde{\mu}_t(x_0, x_t)\|^2$

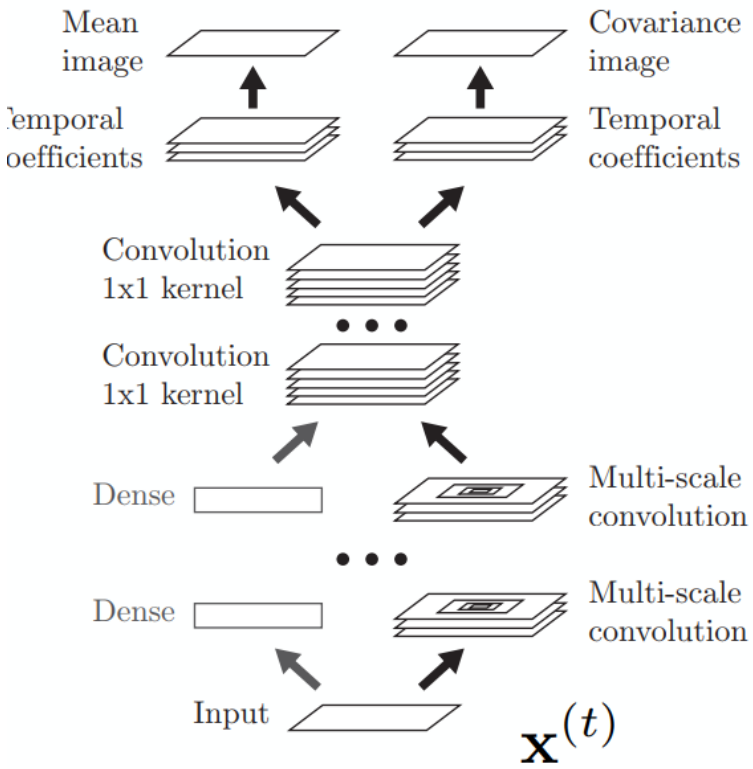
Reparameterization: $x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$

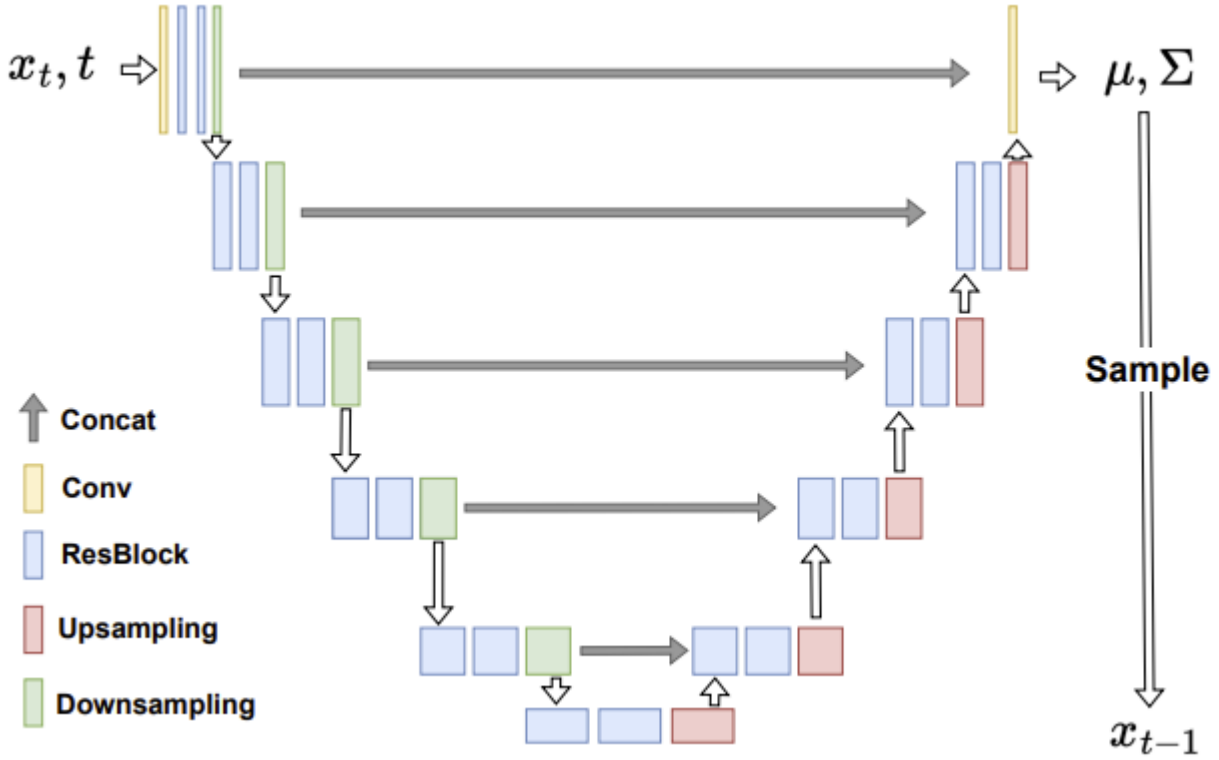
Regress Gaussian noise $\|\epsilon_\theta(x_t, t) - \epsilon\|^2$

Equivalent to DSM (*Vincent, 2011*) $\|s_\theta(x_t, t) - \nabla \log q_t(x_t)\|^2$

Skip connections in the model

Jonathon et al., NeurIPS 2021



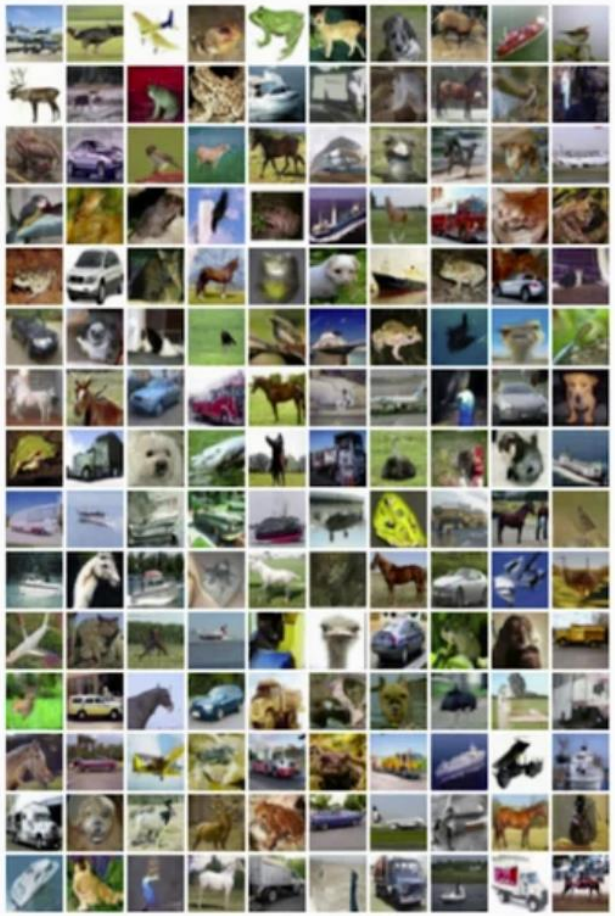


Similar architectures have been investigated in *Song & Ermon, NeurIPS 2019*

Results of DDPM

Jonathon et al., NeurIPS 2021

CelebA-HQ 256x256



CIFAR-10 FID = 3.17 (SOTA)

From finite steps to infinite steps

Song et al, ICLR 2021

$$q(x_i|x_{i-1}) = \mathcal{N}(\sqrt{1-\beta_i}x_{i-1}, \beta_i I)$$

Reparameterization

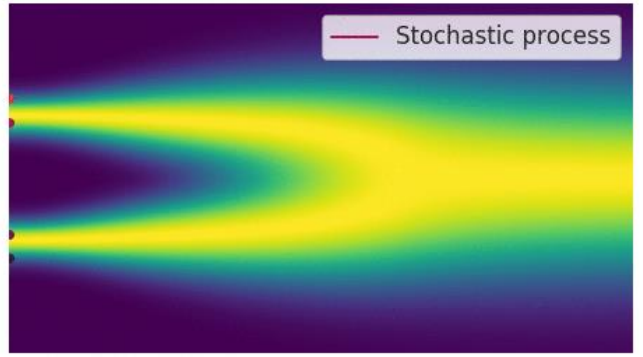
$$\mathbf{x}_i = \sqrt{1-\beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N, \quad \mathbf{z}_{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

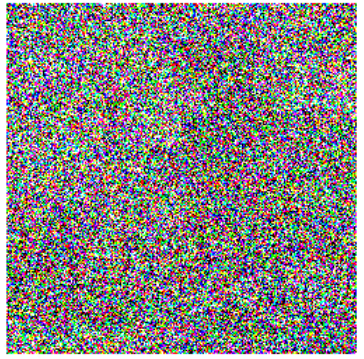
Rescaling by N $(\Delta t = \frac{1}{N}$ $)$

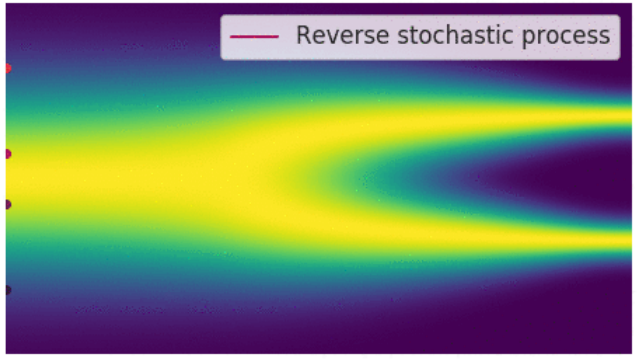
$$\mathbf{x}(t+\Delta t) = \sqrt{1-\beta(t+\Delta t)\Delta t}\,\mathbf{x}(t) + \sqrt{\beta(t+\Delta t)\Delta t}\,\mathbf{z}(t)$$

Limit of $\Delta t \to 0$

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w}$$







From finite steps to infinite steps

Song et al, ICLR 2021

- Reverse time

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

- Training in a continuous manner

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

Results of SDE

Song et al, ICLR 2021

High quality samples

Table 2: NLLs and FIDs (ODE) on CIFAR-10.

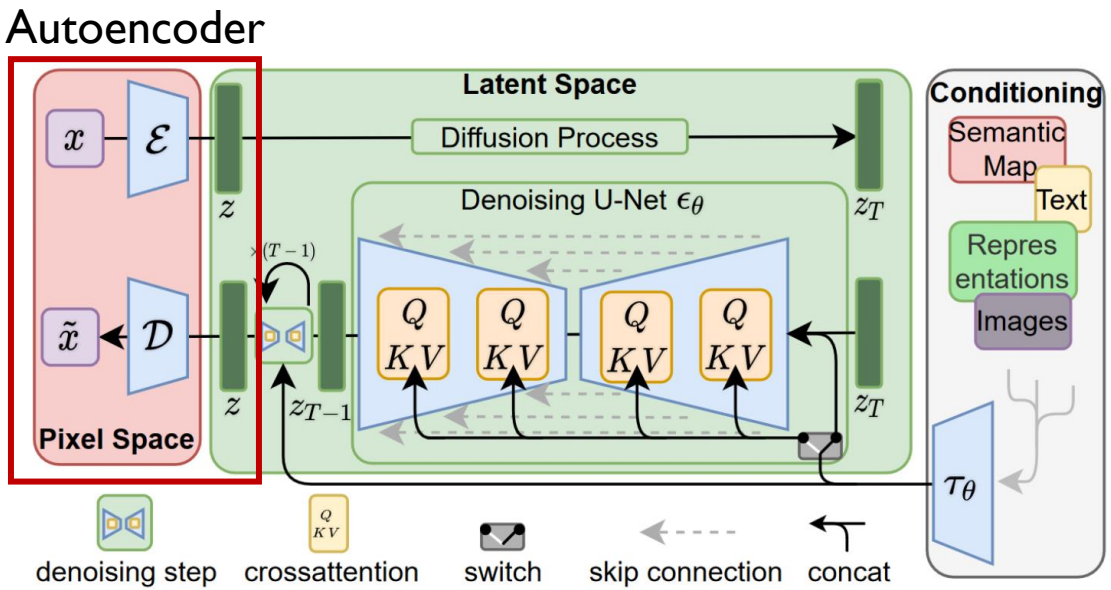
Model	NLL Test ↓	FID ↓
RealNVP (Dinh et al., 2016)	3.49	-
iResNet (Behrmann et al., 2019)	3.45	-
Glow (Kingma & Dhariwal, 2018)	3.35	-
MintNet (Song et al., 2019b)	3.32	-
Residual Flow (Chen et al., 2019)	3.28	46.37
FFJORD (Grathwohl et al., 2018)	3.40	-
Flow++ (Ho et al., 2019)	3.29	-
DDPM (L) (Ho et al., 2020)	$\leqslant 3.70^{*}$	13.51
DDPM (L_{simple}) (Ho et al., 2020)	$\leqslant 3.75^{*}$	3.17
DDPM	3.28	3.37
DDPM cont. (VP)	3.21	3.69
DDPM cont. (sub-VP)	3.05	3.56
DDPM++ cont. (VP)	3.16	3.93
DDPM++ cont. (sub-VP)	3.02	3.16
DDPM++ cont. (deep, VP)	3.13	3.08
DDPM++ cont. (deep, sub-VP)	**2.99**	**2.92**

Latent Diffusion Models

Rombach et al, CVPR 2022

- Two stage learning:
  - Use an autoencoder to get the latent representation of an image
  - Learn the latent representation using a diffusion model

- Latent space has low dimension
  - Easy to learn
  - Efficient forward and backward



Latent Diffusion Models

Rombach et al, CVPR 2022

- A famous application: Stable Diffusion

Elucidating the Design Space of Diffusion-Based Generative Models

Karras et al, NeurIPS 2022

- Express different schedule in a single framework:
  - $x_t = x_0 + \sigma(t)\epsilon$
  - Predict x_0 given x_t using a denoising function $D(x;\sigma)$
  - Parameterize $D(x;\sigma) = c_{skip}(\sigma)x + c_{out}(\sigma)F(c_{in}(\sigma)x; c_{noise}(\sigma))$ based on numerical stability

Elucidating the Design Space of Diffusion-Based Generative Models

Karras et al, NeurIPS 2022

	CIFAR-10 [29] at 32×32				FFHQ [27] 64×64		AFHQv2 [7] 64×64	
	Conditional		Unconditional		Unconditional		Unconditional	
Training configuration	VP	VE	VP	VE	VP	VE	VP	VE
A Baseline [49] (*pre-trained)	2.48	3.11	3.01*	3.77*	3.39	25.95	2.58	18.52
B + Adjust hyperparameters	2.18	2.48	2.51	2.94	3.13	22.53	2.43	23.12
C + Redistribute capacity	2.08	2.52	2.31	2.83	2.78	41.62	2.54	15.04
D + Our preconditioning	2.09	2.64	2.29	3.10	2.94	3.39	2.79	3.81
E + Our loss function	1.88	1.86	2.05	1.99	2.60	2.81	2.29	2.28
F + Non-leaky augmentation	**1.79**	**1.79**	**1.97**	**1.98**	**2.39**	**2.53**	**1.96**	**2.16**
NFE	35	35	35	35	79	79	79	79

Fast Inference

Motivation

DPM

- Learning
  - MLE (tractable posterior) / SM
- Sampling
  - 1000 number of function evaluations
- Performance
  - SOTA generation and likelihood results

GAN, VAE, FLOW

- Learning
  - Adversarial / MLE + traditional VI / det of Jacobian
- Sampling
  - Single function evaluation
- Performance
  - Competitive to SOTA

Motivation

DPM

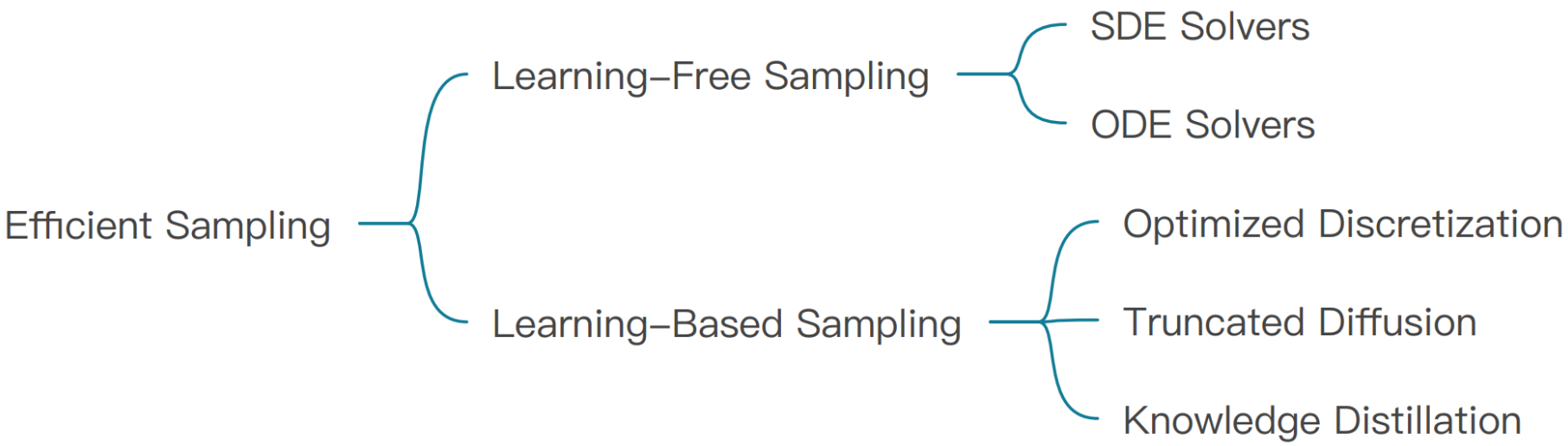
- Learning
  - MLE (tractable posterior) / SM
- Sampling
  - 1000 number of function evaluations
- Performance
  - SOTA generation and likelihood results

GAN, VAE, FLOW

- Learning
  - Adversarial / MLE + traditional VI / det of Jacobian
- Sampling
  - Single function evaluation
- Performance
  - Competitive to SOTA

Fast inference

Review of diffusion models: Yang et al, arxiv 2022



SDE-based Solver

The variance matters

$$\mathrm{E}_{q(x_0)}[\log p_\theta(x_0)] \geq \mathrm{E}_{q(x_{0\ldots T})} \log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} = \mathrm{E}_q \left[L_T + L_0 - \sum KL(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) \right]$$

The variance of $p_\theta(x_{t-1}|x_t)$ are set manually by considering extreme case of $q(x_0)$.

- Standard Gaussian $q(x_0)$ corresponds to β_t
- Single point $q(x_0)$ corresponds to the $\widetilde{\beta}_t$

OpenAI learns the variance using a neural network. *Diffusion Models Beat GANs on Image Synthesis, NeurIPS 2021*

Can we find the optimal covariance w.r.t. the ELBO with a minimal assumption on data?

Score representation of the optimal mean and covariance

Bao et al, ICLR 2022

Main Theorem. The optimal mean and covariance w.r.t. the ELBO can be written as:

$$\mu_t^*(x_t) = \frac{1}{\sqrt{1-\beta_t}}(x_t + \beta_t \nabla \log q_t(x_t)), \qquad \sigma_t^{*2} = \frac{\beta_t}{1-\beta_t}\left(1 - \beta_t \mathrm{E}_{q_t(x_t)} \frac{\|\nabla \log q_t(x_t)\|^2}{d}\right).$$

Key steps in the proof:

- Moment matching: $\min_{p \text{ is Gaussian}} KL(q||p)$ is equivalent to matching the moments of q to p
- Low of total variance: conditional expectation (moments) of $q(x_{t-1}|x_t)$ can be represented conditional expectation of $q(x_0|x_t)$
- The conditional expectation of $q(x_0|x_t)$ can be represented by score of $q_t(x_t)$ if $q(x_t|x_0)$ is Gaussian.

Score representation of the optimal mean and covariance

Bao et al, ICLR 2022

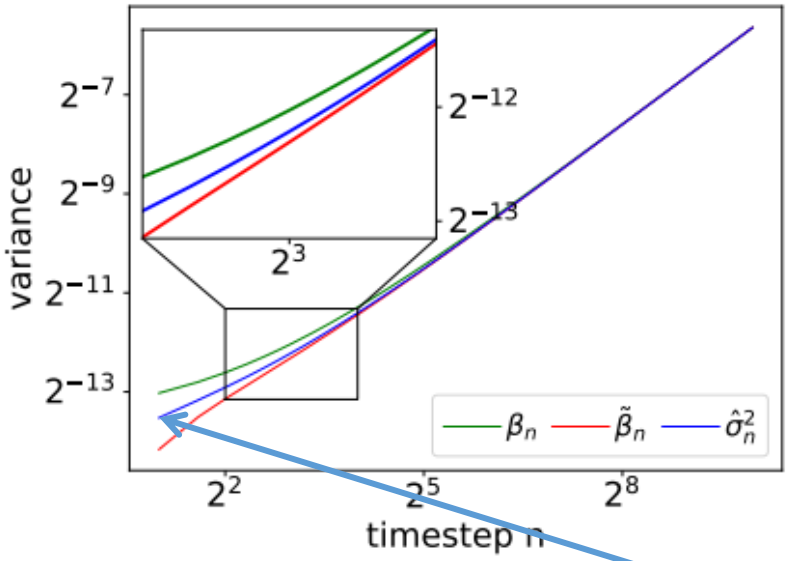
Main Theorem. The optimal mean and covariance w.r.t. the ELBO can be written as:

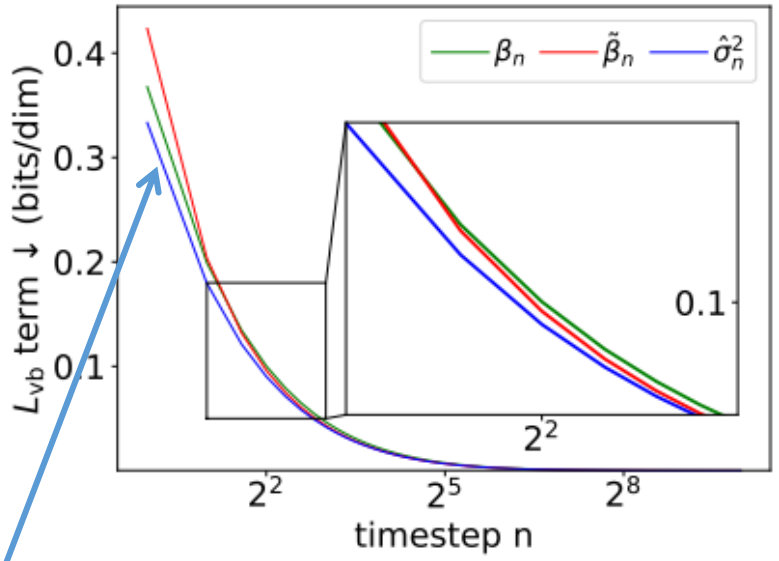
$$\mu_t^*(x_t) = \frac{1}{\sqrt{1-\beta_t}}(x_t + \beta_t \nabla \log q_t(x_t)), \qquad \sigma_t^{*2} = \frac{\beta_t}{1-\beta_t}(1 - \beta_t \mathrm{E}_{q_t(x_t)} \frac{\|\nabla \log q_t(x_t)\|^2}{d}).$$

- The optimal mean representation coincides with existing work
- Theory: the optimal covariance representation has an analytic from w.r.t. the score function
- Algorithm: we can estimate the optimal covariance without additional training to improve the results

Differences between Analytic-DPM and DDPM

Bao et al, ICLR 2022





Comparing the variances

Very different near data

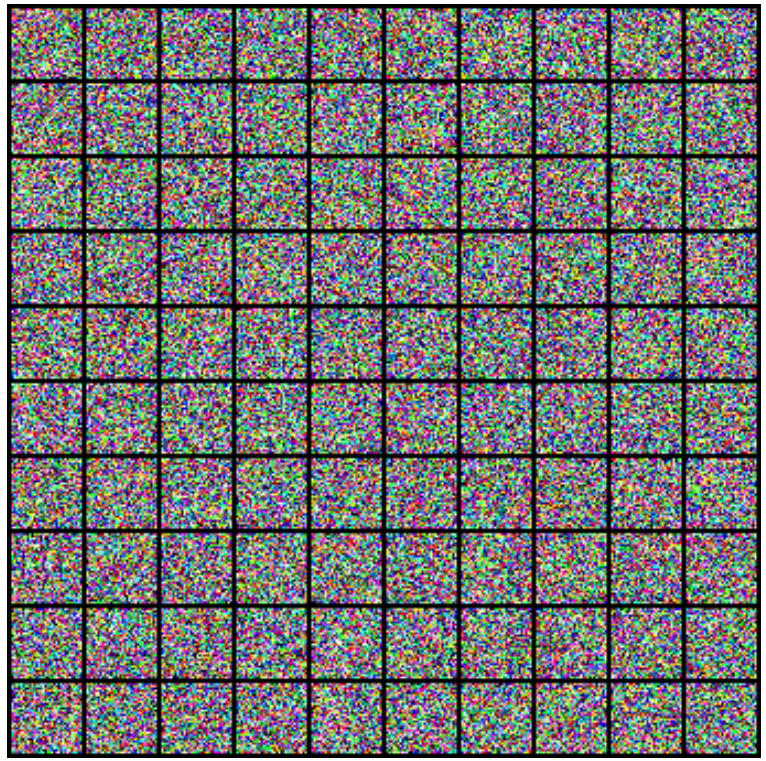
Analytic-DPM

Comparing KL divergence

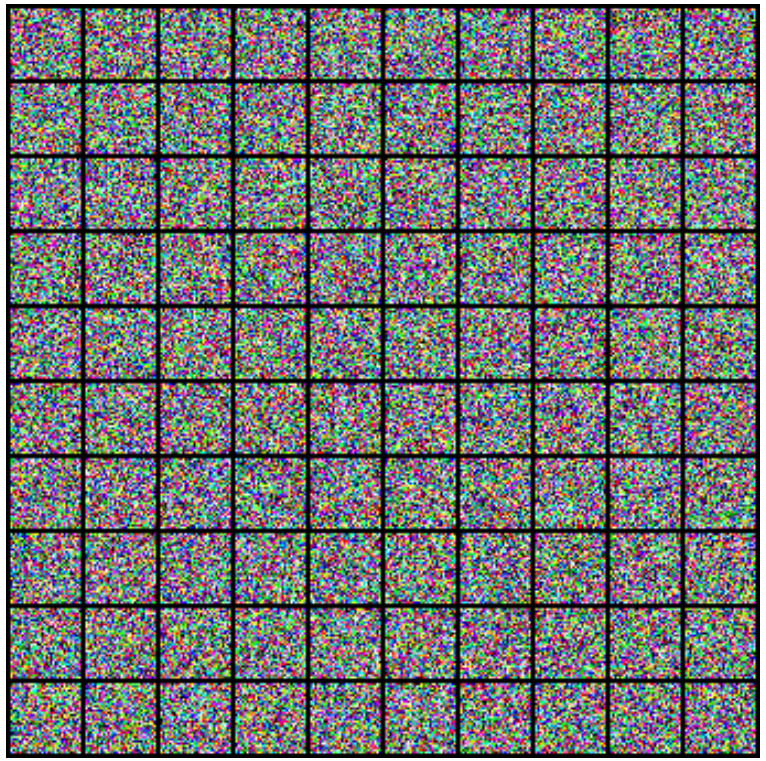
A tighter ELBO

Quality-efficiency trade-off: $20\times$ to $80\times$ speed up with the same sample quality

Bao et al, ICLR 2022



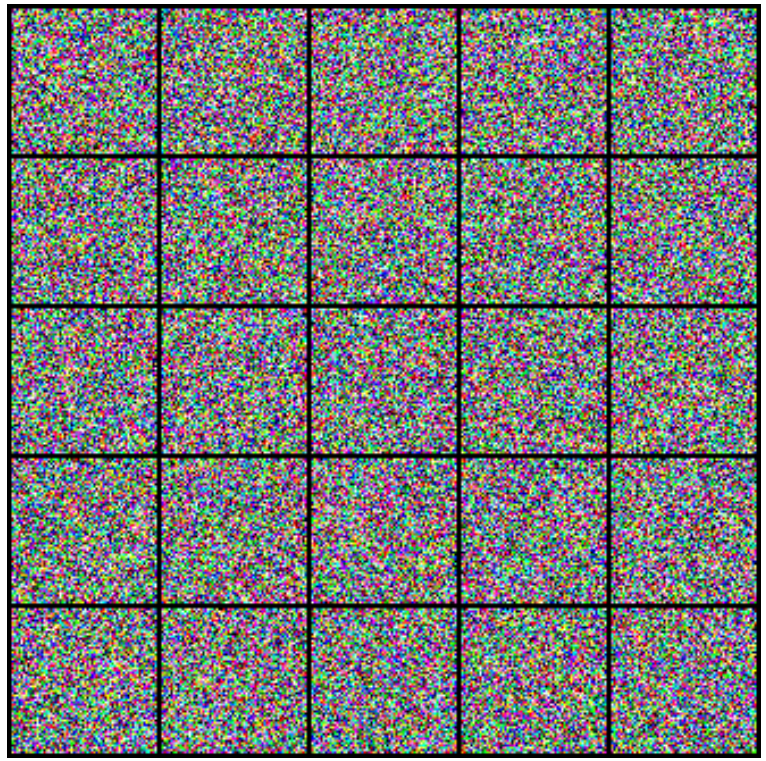
Original DDPM in 1000 steps



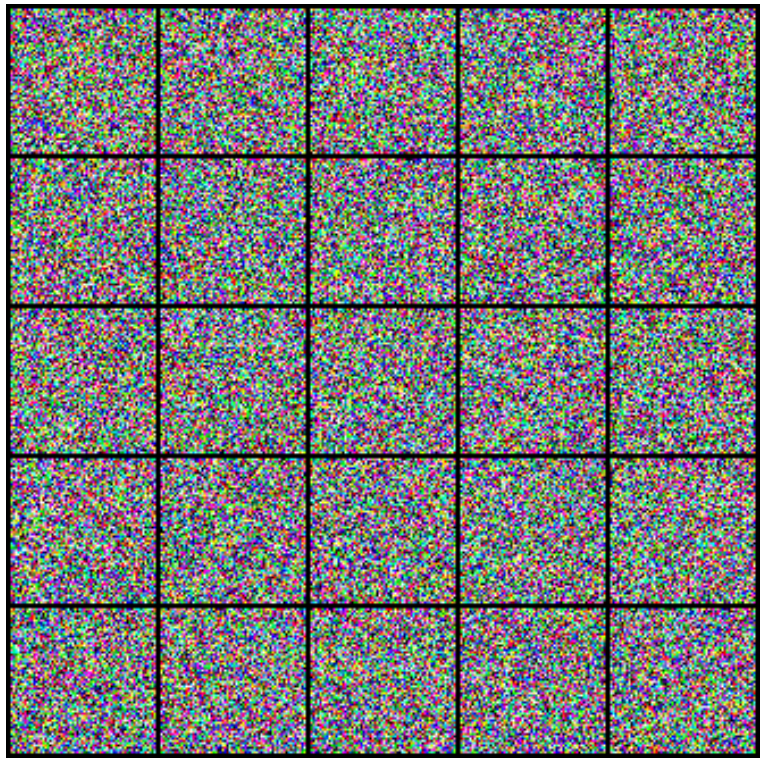
Analytic-DDPM in 50 steps

Quality-efficiency trade-off: $20\times$ to $80\times$ speed up with the same sample quality

Bao et al, ICLR 2022



Original DDPM in 1000 steps



Analytic-DDPM in 50 steps

Contribute to DALL·E 2

To obtain a full generative model of images, we combine the CLIP image embedding *decoder* with a *prior* model, which generates possible CLIP image embeddings from a given text caption. We compare our text-to-image system with other systems such as DALL-E [40] and GLIDE [35], finding that our samples are comparable in quality to GLIDE, but with greater diversity in our generations. We also develop methods for training diffusion priors in latent space, and show that they achieve comparable performance to autoregressive priors, while being more compute-efficient. We refer to our full text-conditional image generation stack as *unCLIP*, since it generates images by inverting the CLIP image encoder.

For the AR prior, we use a Transformer text encoder with width 2048 and 24 blocks and a decoder with a causal attention mask, width 1664, and 24 blocks. For the diffusion prior, we use a Transformer with width 2048 and 24 blocks, and sample with Analytic DPM [2] with 64 strided sampling steps. To reuse hyperparameters tuned for diffusion noise schedules on images from Dhariwal and Nichol [11], we scale the CLIP embedding inputs by 17.2 to match the empirical variance of RGB pixel values of ImageNet images scaled to $[-1, 1]$.

	AR prior	Diffusion prior	64	$64 \rightarrow 256$	$256 \rightarrow 1024$
Diffusion steps	-	1000	1000	1000	1000
Noise schedule	-	cosine	cosine	cosine	linear
Sampling steps	-	64	250	27	15
Sampling variance method	-	analytic [2]	learned [34]	DDIM [47]	DDIM [47]
Crop fraction	-	-	-	0.25	0.25
Model size	1B	1B	3.5B	700M	300M
Channels	-	-	512	320	192
Depth	-	-	3	3	2
Channels multiple	-	-	1,2,3,4	1,2,3,4	1,1,2,2,4,4
Heads channels	-	-	64	-	-
Attention resolution	-	-	32,16,8	-	-
Text encoder context	256	256	256	-	-
Text encoder width	2048	2048	2048	-	-
Text encoder depth	24	24	24	-	-
Text encoder heads	32	32	32	-	-

Slightly better FID results & much faster sampling speed than an autoregressive approach

ODE-based Solver

Two types of diffusion probabilistic models

Lu et al, NeurIPS 2022

- **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.
  - Sampling by **Analytic-DPM** is also equivalent to discretization of diffusion SDEs.

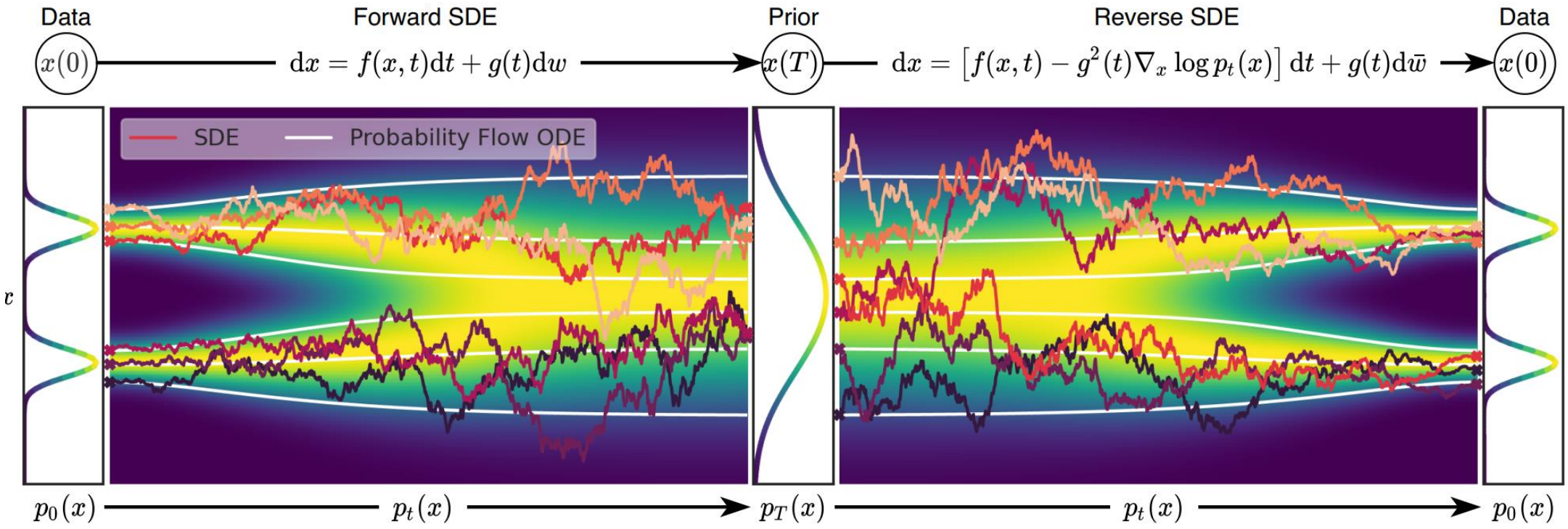
- **Diffusion ODEs:**

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

  - Sampling by **DDIM** is equivalent to a first-order discretization of diffusion ODEs.

Two types of diffusion probabilistic models

Lu et al, NeurIPS 2022



ODE: Deterministic & Invertible

DDIM as a first-order discretization of diffusion ODEs

Song et al, ICLR 2021

- Update rule of DDIM:
  - $x_{t-\Delta t} = \alpha_{t-\Delta t}\hat{x}(x_t, t) + \sigma_{t-\Delta t}\epsilon_\theta(x_t, t)$
  - $(x_{t-\Delta t} - x_t)/\Delta t = (\alpha_{t-\Delta t}\hat{x}(x_t, t) + \sigma_{t-\Delta t}\epsilon_\theta(x_t, t) - x_t)/\Delta t$

- Let $\Delta t \to 0$
  - DDIM reduces to $\frac{dx_t}{dt} = \frac{d\alpha_t}{dt}\hat{x}(x_t, t) + \frac{d\sigma_t}{dt}\epsilon_\theta(x_t, t) = f(t)x_t + \frac{g(t)^2}{2\sigma_t}\epsilon_\theta(x_t, t)$

Diffusion ODE

Solving diffusion ODEs by traditional Runge-Kutta methods

Lu et al, NeurIPS 2022

Solve the diffusion ODEs directly by Runge-Kutta methods (second order or higher)

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

Approximated as a whole black box, causes large discretization error

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

Observation 1: exactly computing the linear part

Lu et al, NeurIPS 2022

- 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

Lu et al, NeurIPS 2022

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

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

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

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**

Designing high-order solvers for diffusion ODEs

Lu et al, NeurIPS 2022

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

Lu et al, NeurIPS 2022

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

Lu et al, NeurIPS 2022

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}:

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

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

DPM-Solver: customized solver for diffusion ODEs

Lu et al, NeurIPS 2022

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

Linear term — **Exactly Computed**

Derivatives — **Approximated**

Coefficients — **Exactly computed**

High-order errors — **Omitted**

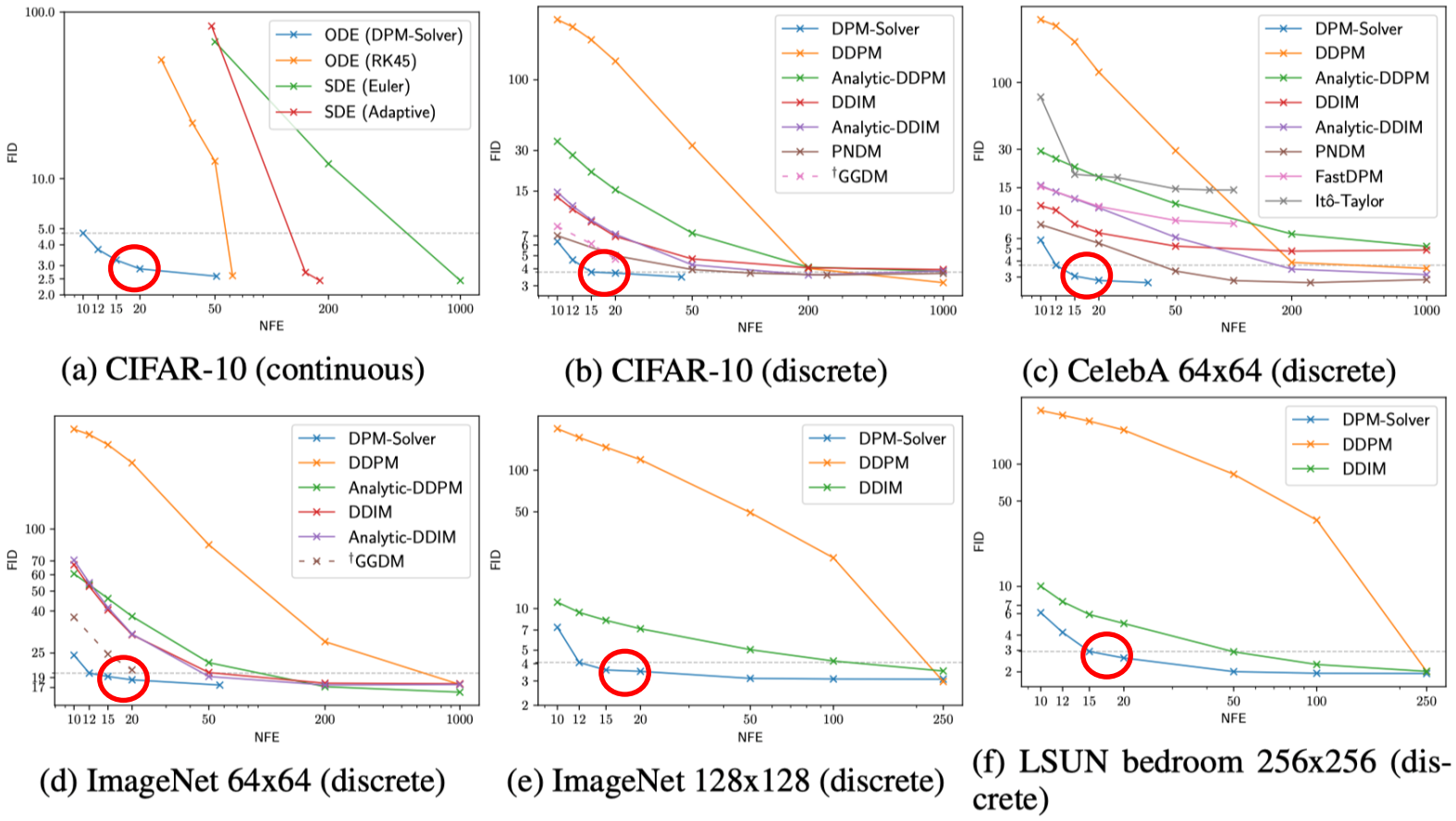
Runge-Kutta $$\boldsymbol{x}_t = \boldsymbol{x}_s + \int_s^t \left(f(\tau)\boldsymbol{x}_\tau + \frac{g^2(\tau)}{2\sigma_\tau}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\tau, \tau)\right)\mathrm{d}\tau$$

Approximated as a whole black box

Better quality-efficiency trade-off than RK and DDIM

Lu et al, NeurIPS 2022

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

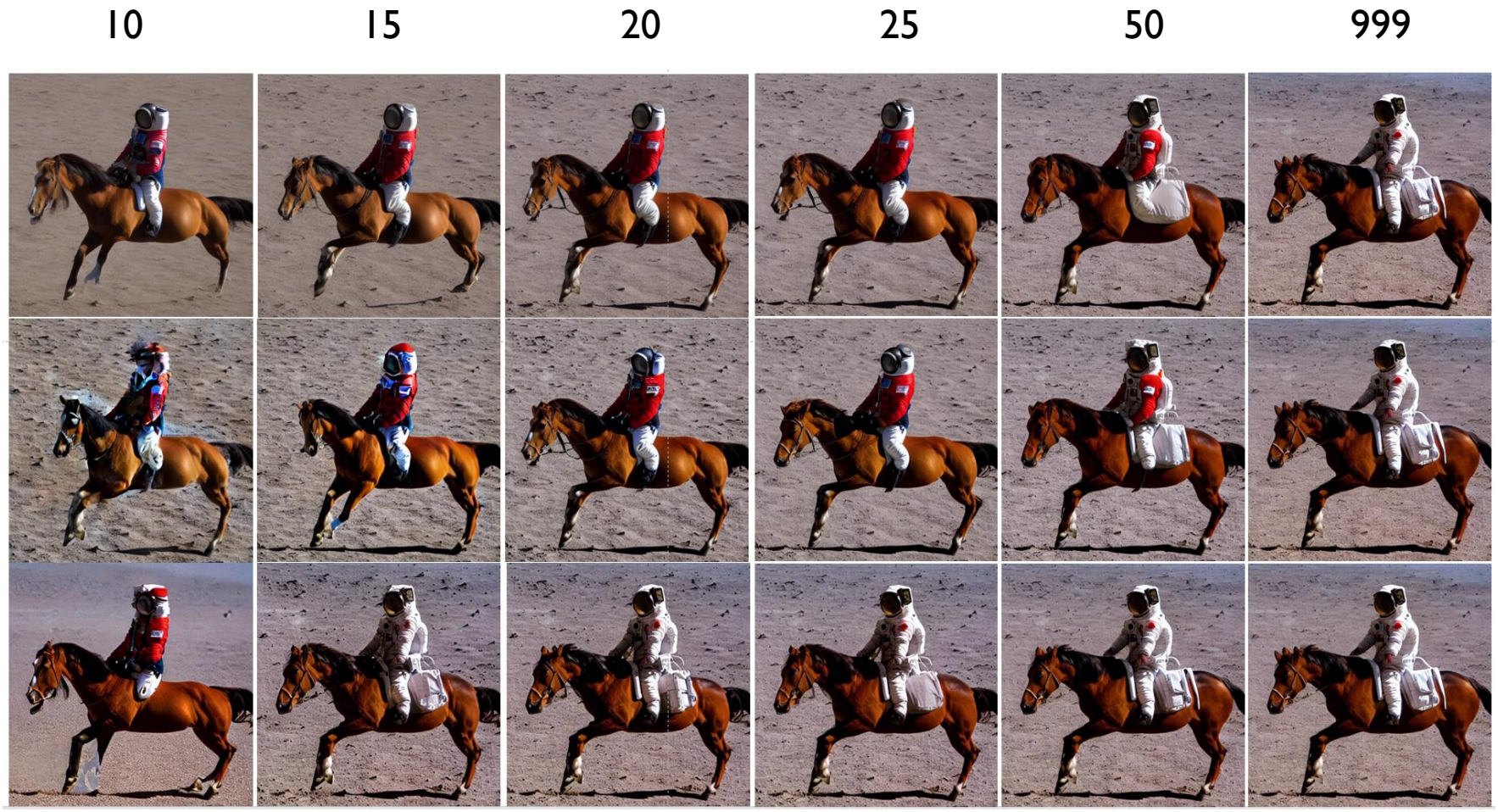


(a) CIFAR-10 (continuous) (b) CIFAR-10 (discrete) (c) CelebA 64x64 (discrete)

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

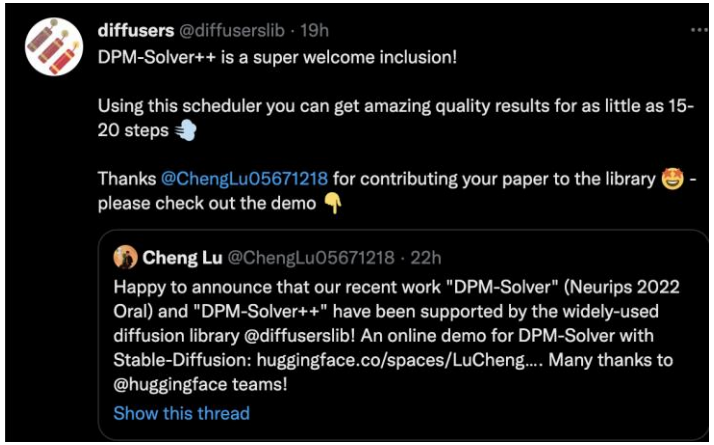
Example: stable-diffusion with DPM-Solver

Lu et al, Arxiv Preprint 2022

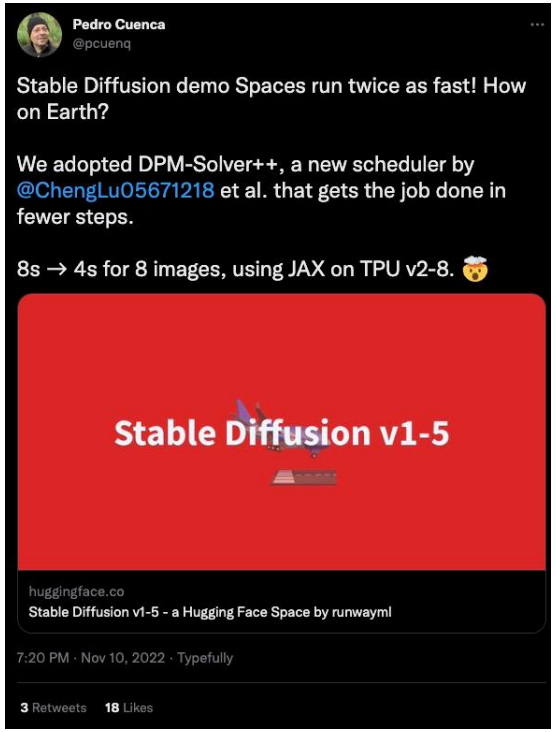


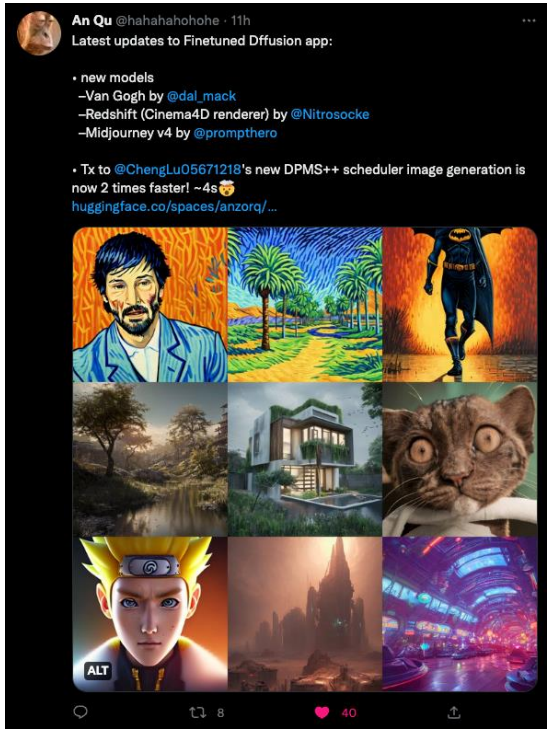
Contribute to many open-source large models

Diffusers official account:



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



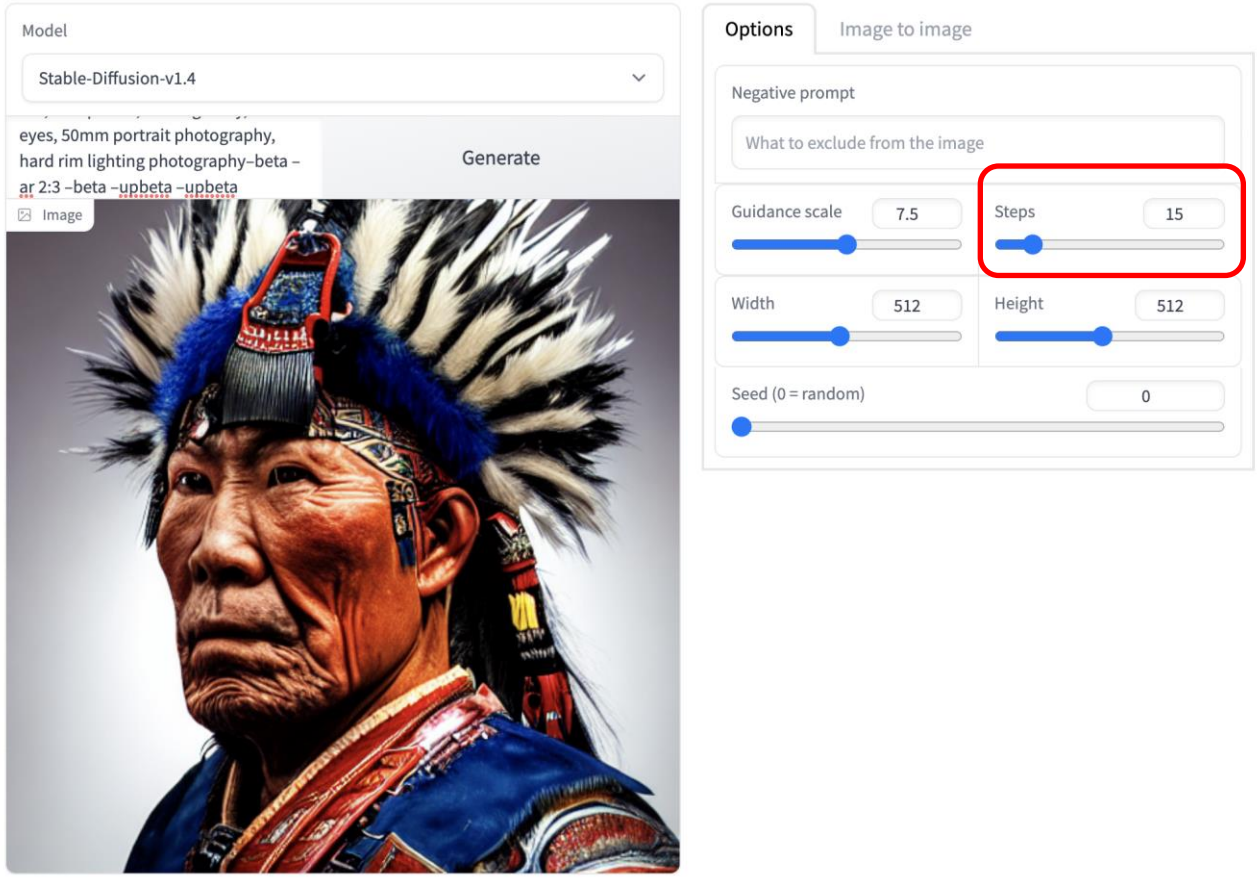


Stable-diffusion-WebUI for:

(the name with “DPM” or “DPM++”)

Online demo for stable-diffusion with DPM-Solver

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



Controllable Generation

Generative models with conditions

vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

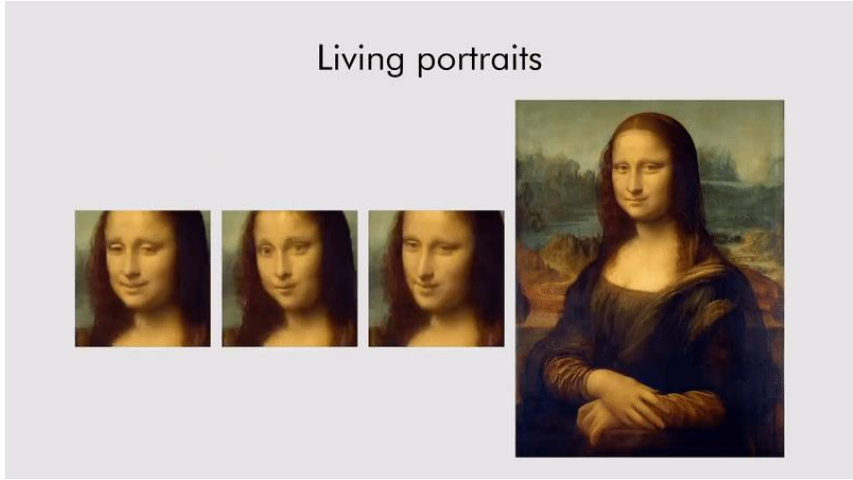
a close up of a handpalm with leaves growing from it

an espresso machine that makes coffee from human souls, artstation

panda mad scientist mixing sparkling chemicals, artstation

a corgi's head depicted as an explosion of a nebula

Naïvely, train a conditional distribution $p(x \mid c)$ on paired data, quite like supervised learning



Guided sampling for conditional sampling

Classifier guidance

A Nichol et al., ICML 2021

Original motivation: what if we do not have paired data during training?

Can we turn an unconditional model to a conditional one?

Likelihood Prior (our model)

Bayes' rule $$p(x \mid c) = \frac{p(c \mid x)\, p(x)}{p(c)}$$

Posterior (our goal) Evidence (a constant)

Classifier guidance

A Nichol et al., ICML 2021

Bayes' rule $p(x \mid c) = \dfrac{p(c \mid x)\, p(x)}{p(c)} \quad \Rightarrow \quad \nabla_x \log p(x \mid c) = \nabla_x \log p(x) + \nabla_x \log p(c \mid x)$

Classifier guidance

A Nichol et al., ICML 2021

Bayes' rule $p(x \mid c) = \frac{p(c \mid x)\, p(x)}{p(c)} \quad \Rightarrow \quad \nabla_x \log p(x \mid c) = \nabla_x \log p(x) + \nabla_x \log p(c \mid x)$

Temperature/scale

Classifier guidance: special case with a classifier **multiplied by a factor**

$$\nabla_x \log p_s(x \mid c) = \nabla_x \log p(x) + s\, \nabla_x \log p(c \mid x)$$

Sample direction Pretrained model Pretrained classifier

Classifier guidance

A Nichol et al., ICML 2021

Bayes' rule $p(x \mid c) = \frac{p(c \mid x)\, p(x)}{p(c)} \quad \Rightarrow \quad \nabla_x \log p(x \mid c) = \nabla_x \log p(x) + \nabla_x \log p(c \mid x)$

Temperature/scale

Classifier guidance: special case with a classifier multiplied by a factor

$$\tilde{\epsilon}_{\theta,\phi}(x_t, t, c) \coloneqq \epsilon_\theta(x_t) - s\ \sigma_t \nabla_{x_t} \log p_\phi(c|x_t, t)$$

Sample direction Pretrained model Pretrained classifier

Iterative version: consider noise in the classifier and introduce approximate (Taylor expansion)

Classifier guidance

A Nichol et al., ICML 2021

Bayes' rule $p(x \mid c) = \frac{p(c \mid x)\, p(x)}{p(c)} \quad \Rightarrow \quad \nabla_x \log p(x \mid c) = \nabla_x \log p(x) + \nabla_x \log p(c \mid x)$

Temperature/scale

Classifier guidance: special case with a classifier multiplied by a factor

$$\tilde{\epsilon}_{\theta,\phi}(x_t, t, c) \coloneqq \epsilon_\theta(x_t, c) - s\ \sigma_t \nabla_{x_t} \log p_\phi(c|x_t, t)$$

Sample direction Pretrained model Pretrained classifier

Magic things happen: it trades off the controllability and quality well with a large s, **even when the pretrained diffusion model is already a conditional one!**

Classifier free guidance

Ho and Salimans, Arxiv preprint 2022

Classifier guidance $\nabla_x \log p_s(x \mid c) = \nabla_x \log p(x) + s \nabla_x \log p(c \mid x)$

Motivation: can we achieve a similar trade-off without training the classifier?

Bayes' rule (opposite way) $p(c \mid x) = \frac{p(x \mid c)\, p(c)}{p(x)} \Rightarrow \nabla_x \log p(c \mid x) = \nabla_x \log p(x \mid c) - \nabla_x \log p(x)$

Classifier free guidance

Ho and Salimans, Arxiv preprint 2022

Classifier guidance

$$\nabla_x \log p_s(x \mid c) = \nabla_x \log p(x) + s\, \nabla_x \log p(c \mid x)$$

Bayes' rule (opposite way)

$$p(c \mid x) = \frac{p(x \mid c)\, p(c)}{p(x)} \quad \Rightarrow \quad \nabla_x \log p(c \mid x) = \nabla_x \log p(x \mid c) - \nabla_x \log p(x)$$

Substitute this into the formula for classifier guidance

Classifier free guidance

$$\nabla_x \log p_s(x \mid c) = (1 - s)\nabla_x \log p(x) + s\, \nabla_x \log p(x \mid c)$$

Sample direction — Unconditional score — Conditional score

Classifier free guidance

Ho and Salimans, Arxiv preprint 2022

Classifier guidance

$$\nabla_x \log p_s(x \mid c) = \nabla_x \log p(x) + s\, \nabla_x \log p(c \mid x)$$

Classifier free guidance

$$\nabla_x \log p_s(x \mid c) = (1 - s)\nabla_x \log p(x) + s\, \nabla_x \log p(x \mid c)$$

Sample direction Unconditional score Conditional score

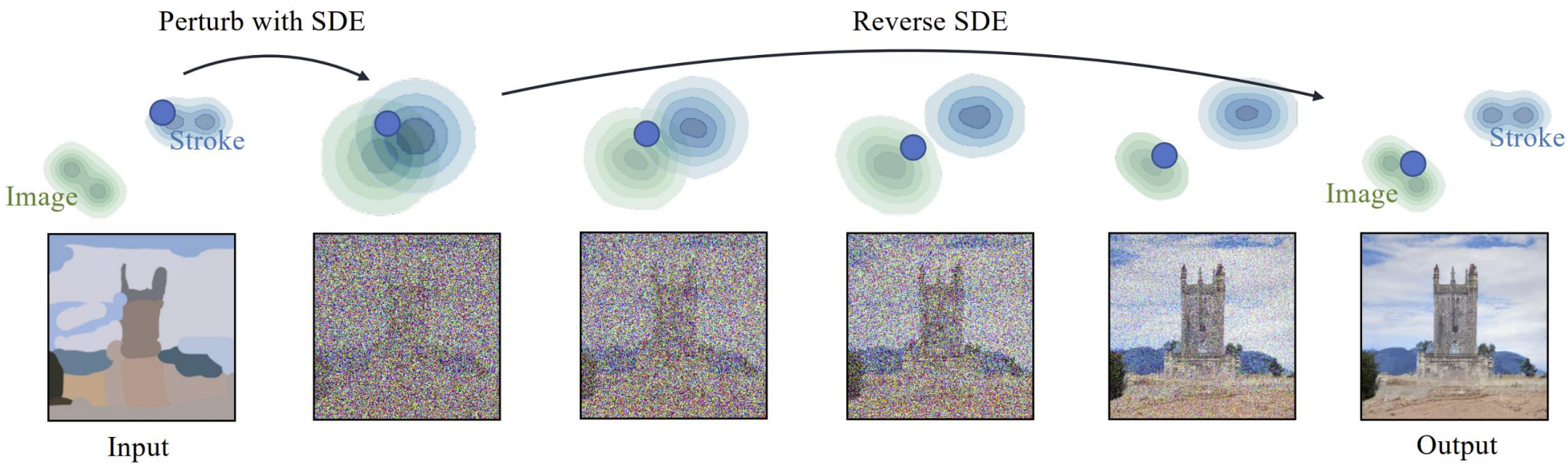
- Train both scores models by the original denoise score matching with shared parameters
- Sample from a "very conditional model" with $s > 1$
- It requires paired data during training but is parameter efficient and less sensitive
- A very popular trick for text-to-image and label-conditional image generation

Conditions are more than classifier labels

SDEdit: guided image synthesis and editing with SDE

Meng et al. ICLR 2022

Given a pre-trained SDE model, the input will be transferred to the trained domain



Run forward 500 steps to get a noisy image

Denoise from the noisy image by the pretrained SDE

Results of SDEdit

Meng et al. ICLR 2022

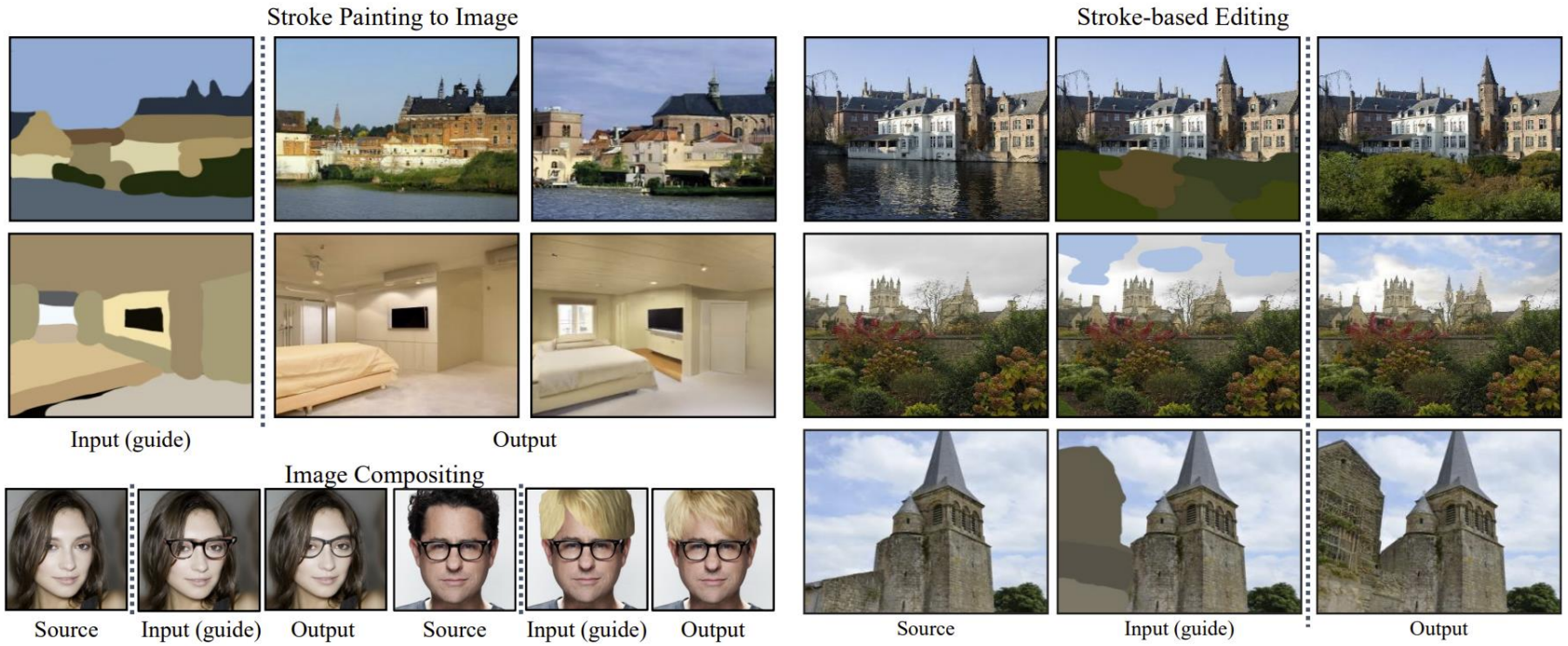
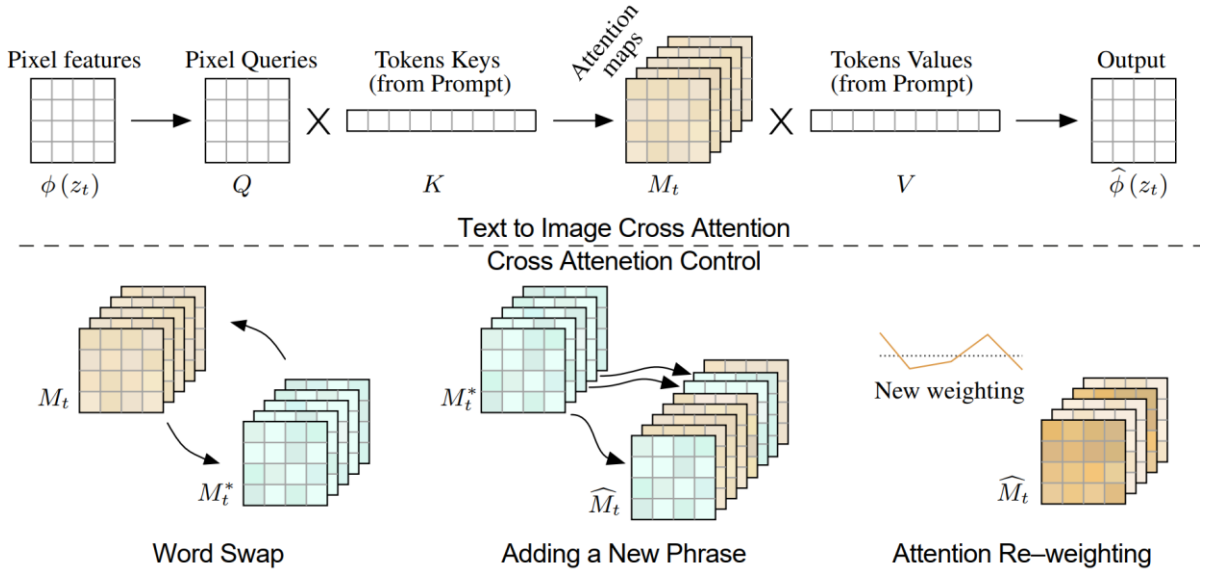


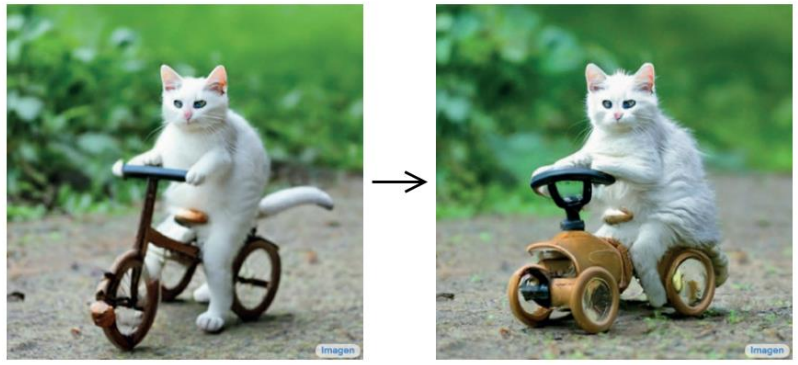
Figure 1: Stochastic Differential Editing (SDEdit) is a **unified** image synthesis and editing framework based on stochastic differential equations. SDEdit allows stroke painting to image, image compositing, and stroke-based editing **without** task-specific model training and loss functions.

Prompt-to-prompt image editing with cross attention control

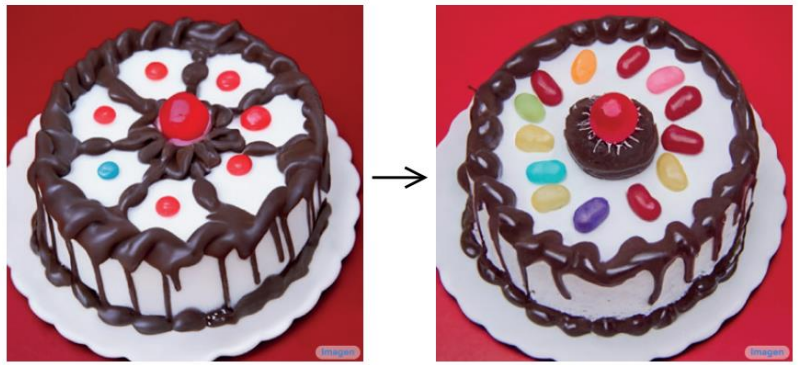
Amir et al., arxiv, preprint 2022

Control the semantics of a modified prompt by manipulating the cross attention module





"Photo of a cat riding on a bicycle."
car



"a cake with decorations."
jelly beans

Personalizing text-to-image generation using textual inversion

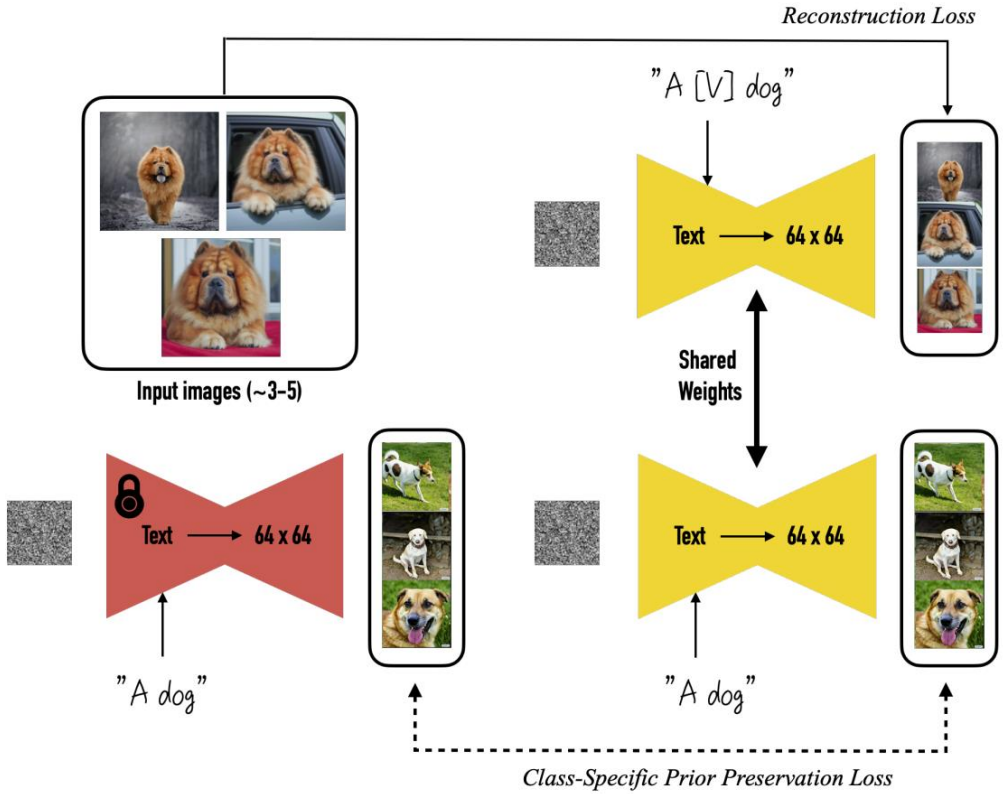
Gal et al., NeurIPS 2022

Find the optimal text embedding that minimizes the noise prediction loss for a pretrained model

DreamBooth: fine tuning text-to-image diffusion models for subject-driven generation

Ruiz et al, arxiv preprint 2022

Fine-tine a large pre-trained model by a reconstruction loss conditioned on a special identifier while preserve the ability to generate diverge images within the same class



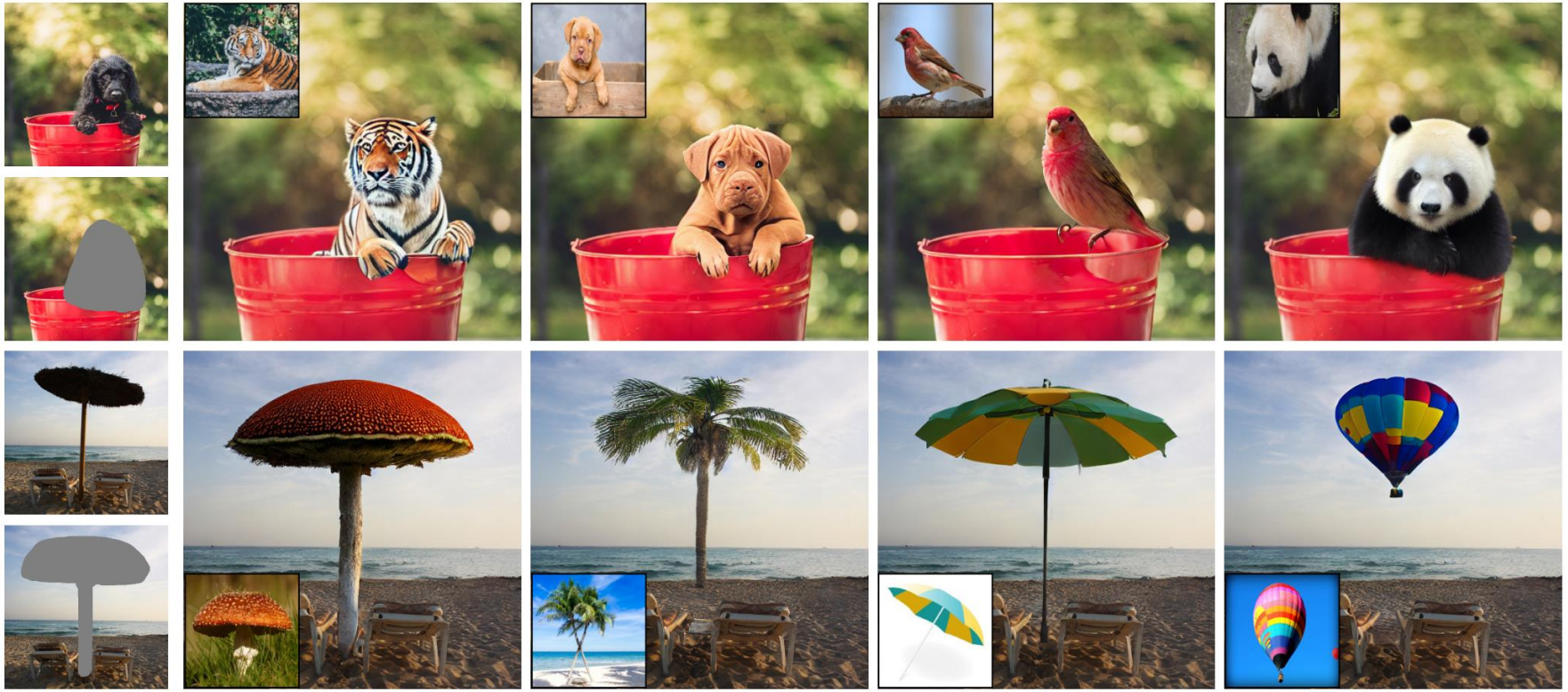

Input images


Input images

Paint by example: exemplar-based image editing with diffusion models

Yang et al, Arxiv preprint 2022

Using masks from real images to produces a training dataset. Some tricks: CLIP class tokens as conditions, strong data augmentation for masked part and augmentation for mask shape.

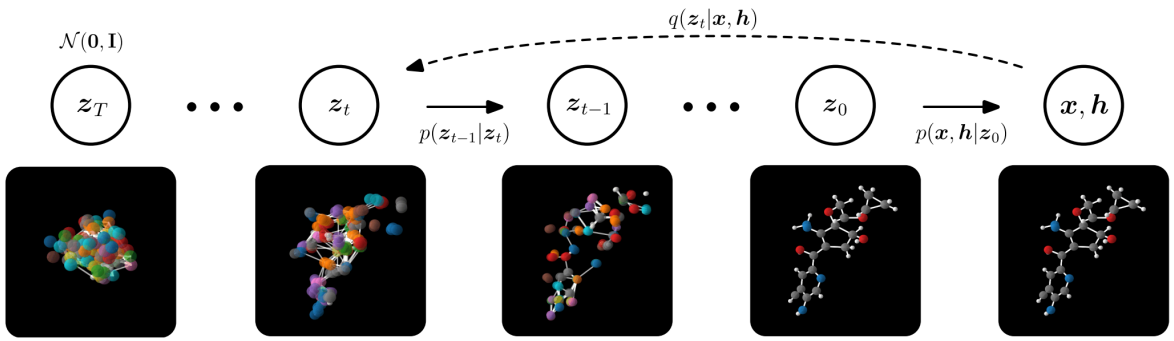


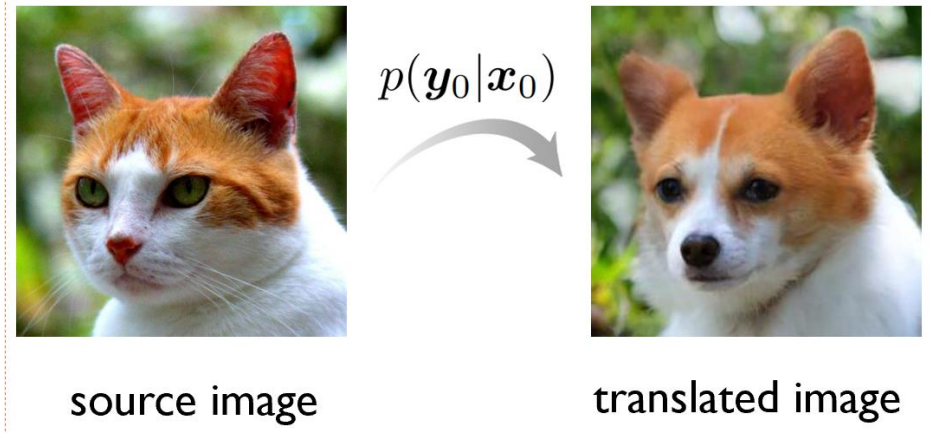
Energy guidance: a guided sampling framework for general conditions

Controllable generation is much more than classifier guidance

Zhao et al, NeurIPS 2022

- Examples that cannot be handled by classifiers
  - General loss functions: norms, cosine similarity and triplet loss
  - Domain specific properties: geometry-equivalence
  - Combination of multiple controllers





The energy guidance framework

Zhao et al, NeurIPS 2022

- A general framework to encode domain knowledge in the sampling of a pre-trained score model

Energy guidance $\tilde{\epsilon}_{\theta,\phi}(x_t, t, c) \coloneqq \epsilon_\theta(x_t) - s\,\nabla_{x_t}\varepsilon_\phi(x_t, t, c)$

Sample direction Pretrained model Energy function

- Energy guidance only requires that ε is differentiable and regular
- Classifier guidance is a special case with a classifier, e.g. $\varepsilon(x_t, t, c) \coloneqq \log p_\phi(c|x_t, t)$
- Energy guidance can be a linear combination of multiple "sub-energy functions"

Energy guidance as generalized classifier guidance

Zhao et al, NeurIPS 2022

- The energy function defines a probability distribution as:

$$q_\phi(x|c) = \frac{1}{Z(\phi)} e^{-\varepsilon_\phi(x_t,t,c)}, \qquad Z(\phi) = \int e^{-\varepsilon_\phi(x_t,t,c)} dx.$$

- Similarly to classifier guidance, with some approximation, energy guidance is proven to get samples from the product of experts (PoE) as follows:

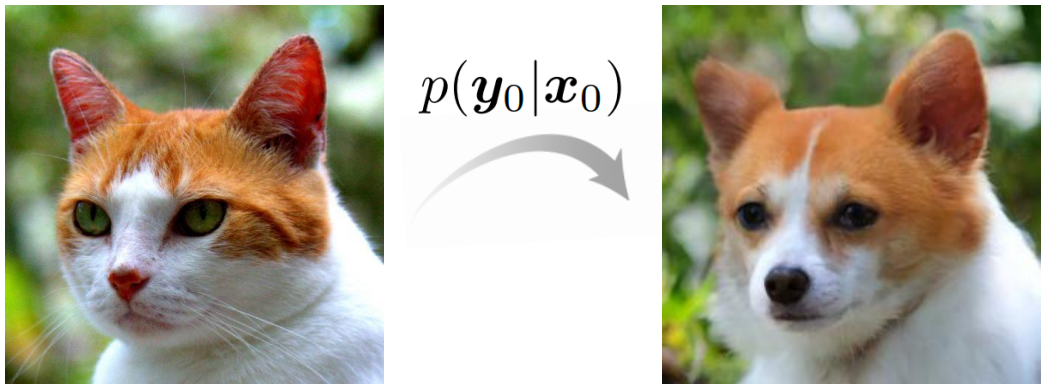
$$p_{\theta,\phi}(x \mid c) \propto p_\theta(x)\, q_\phi(x \mid c)$$

- Bayes' rule is also in the form of PoE. Namely, classifier guidance is a special case.

Application I: unpaired Image-to-Image translation

source target

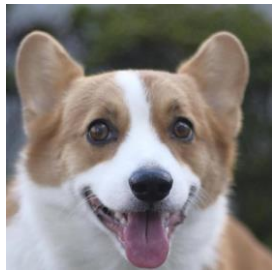
Training



source image translated image

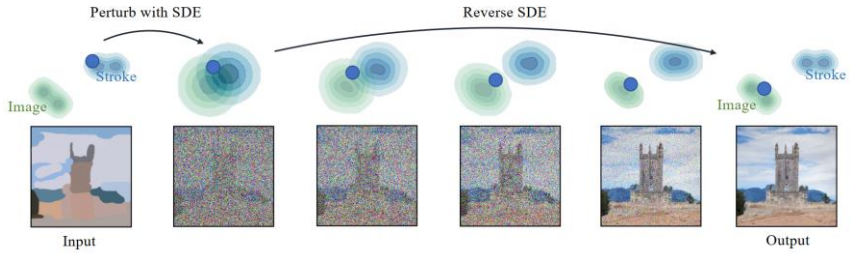
Inference

Motivation



target

source image



SDE
$s(y,t)$

Did not leverage the training data in the source domain at all

SDE
$s(y,t)$

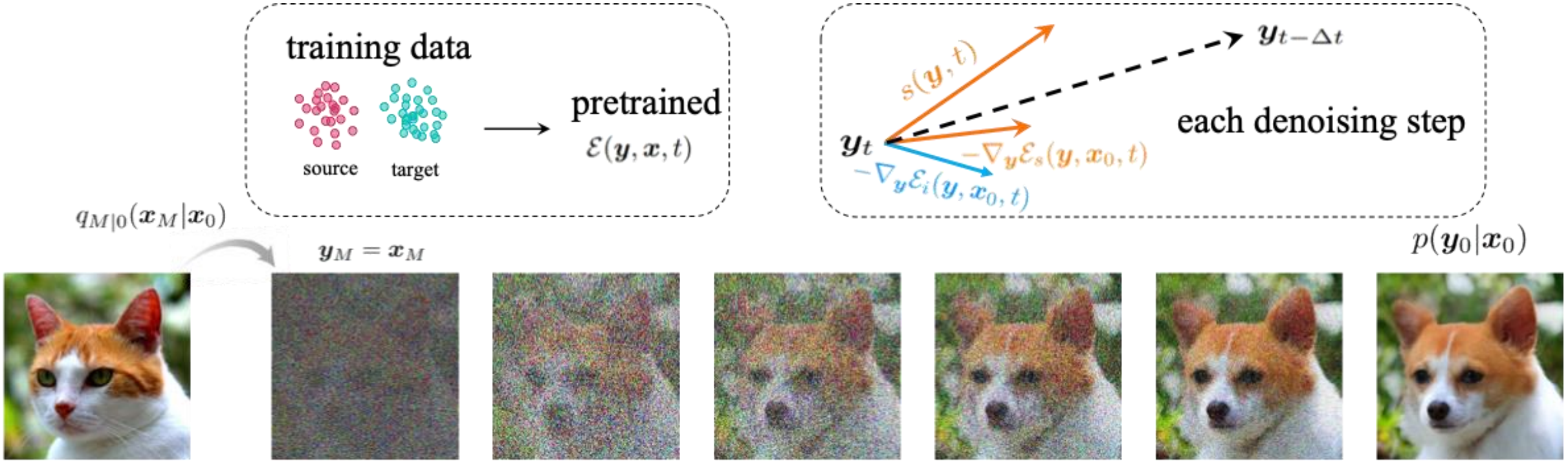
translated image

Training

Translation only in inference

Energy-Guided Stochastic Differential Equations (EGSDE)

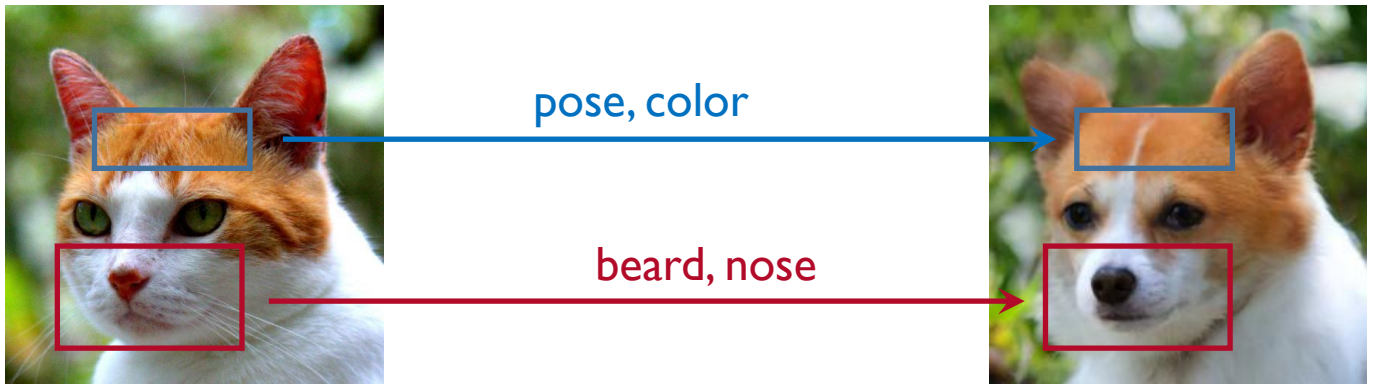
Zhao et al, NeurIPS 2022



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

Introduce a pretrained energy function that uses source data and guides sampling

What is good translation?



source image

x_0

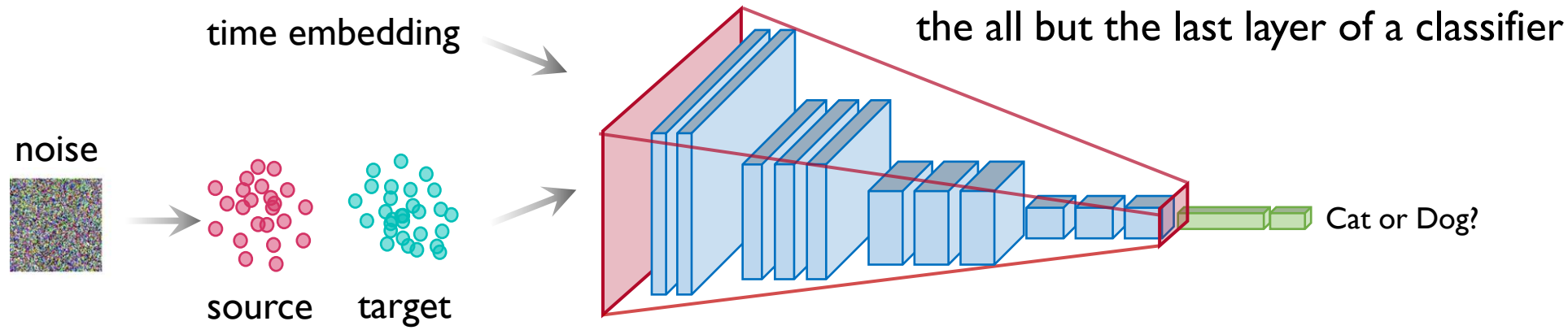
translated image

- Be ***realistic*** for the target domain by changing the domain-specific features
- Be ***faithful*** for the source image by preserving the domain-independent features

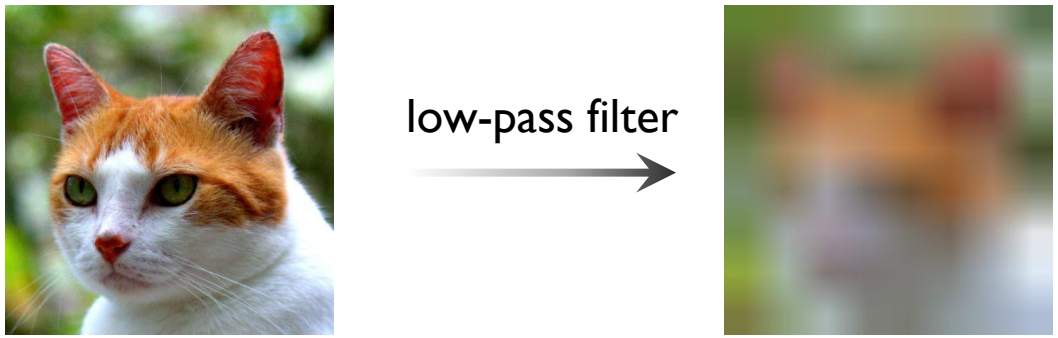
Choice of energy

Zhao et al, NeurIPS 2022

- ***Realistic:*** dissimilarity on features extracted by a domain-specific feature extractor



- ***Faithful:*** similarity on features extracted by a domain-independent feature extractor



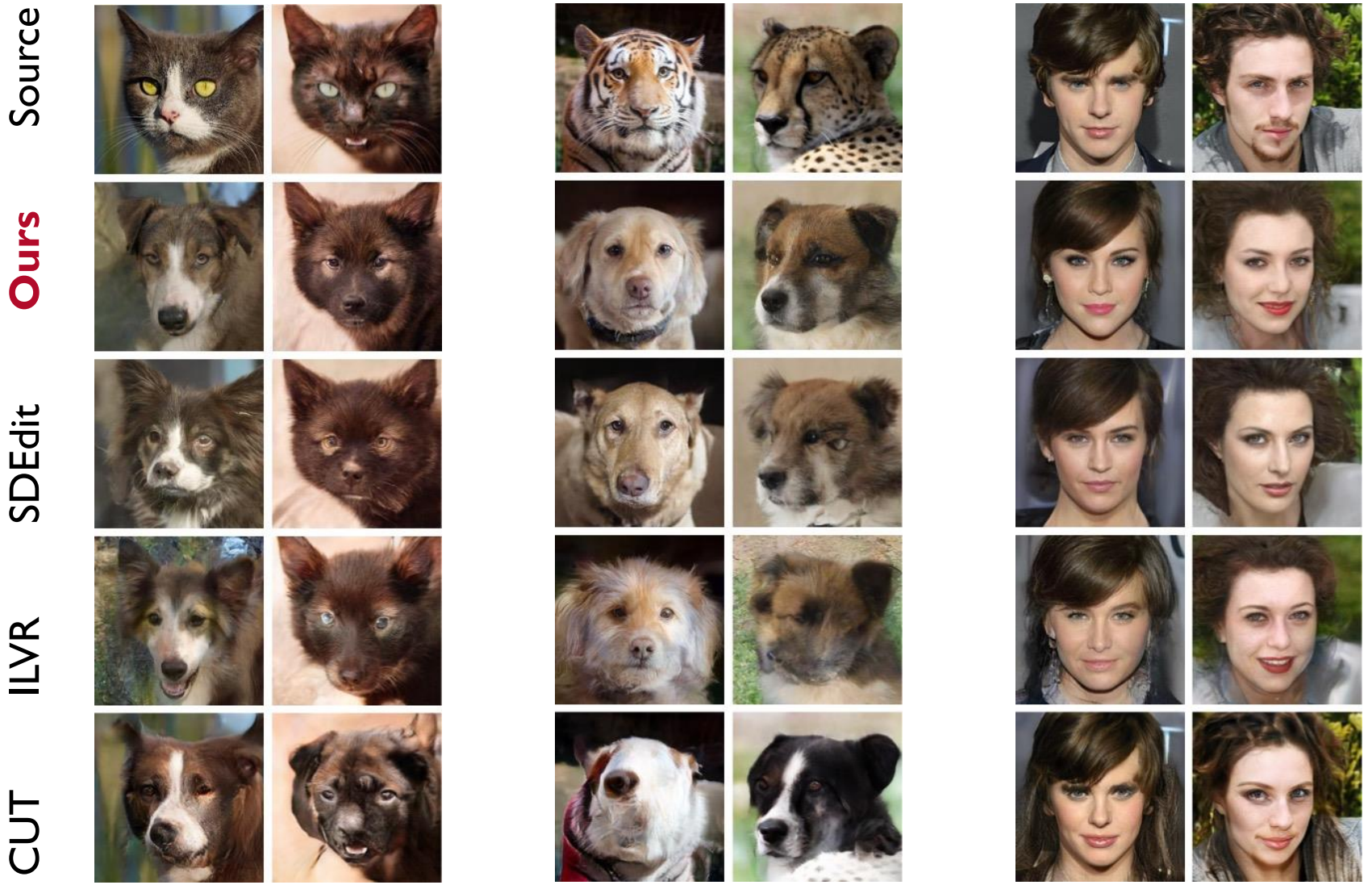
Results of EGSDE

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	-

Results of EGSDE



The function of each expert

Zhao et al, NeurIPS 2022

Input

$\boldsymbol{x}_0$

realistic expert $\mathcal{E}_s(\boldsymbol{y}, \boldsymbol{x}_0, t)$

realistic

$\lambda_s = 0$

$\lambda_s = 1500$

faithful expert $\mathcal{E}_i(\boldsymbol{y}, \boldsymbol{x}_0, t)$

faithful

$\lambda_i = 0$

$\lambda_i = 150$

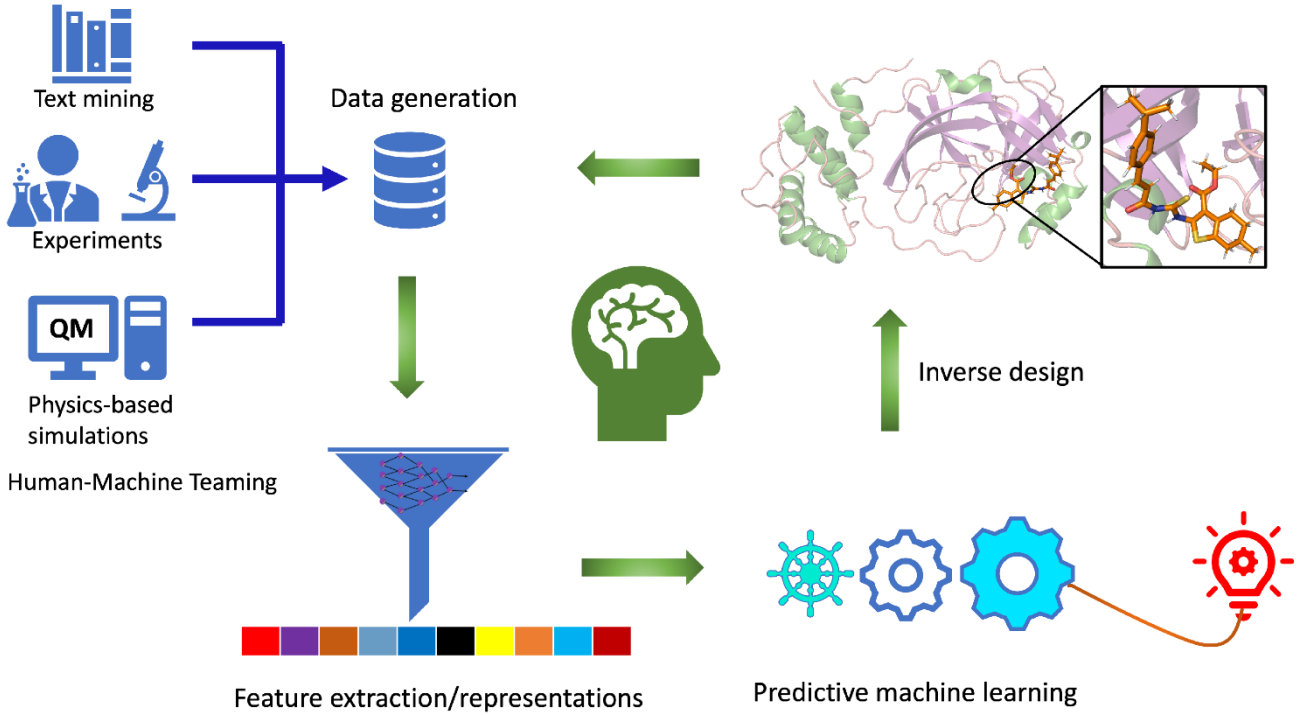
Output

$$\mathcal{E}(\boldsymbol{y}, \boldsymbol{x}_0, t)$$
$$\|$$
$$\lambda_s \mathcal{E}_s(\boldsymbol{y}, \boldsymbol{x}_0, t) + \lambda_i \mathcal{E}_i(\boldsymbol{y}, \boldsymbol{x}_0, t)$$

Application II: controllable inverse molecular design

Bao & Zhao et al., Arixv 2022

- Inverse molecular design: generated molecules should satisfy certain desirable properties

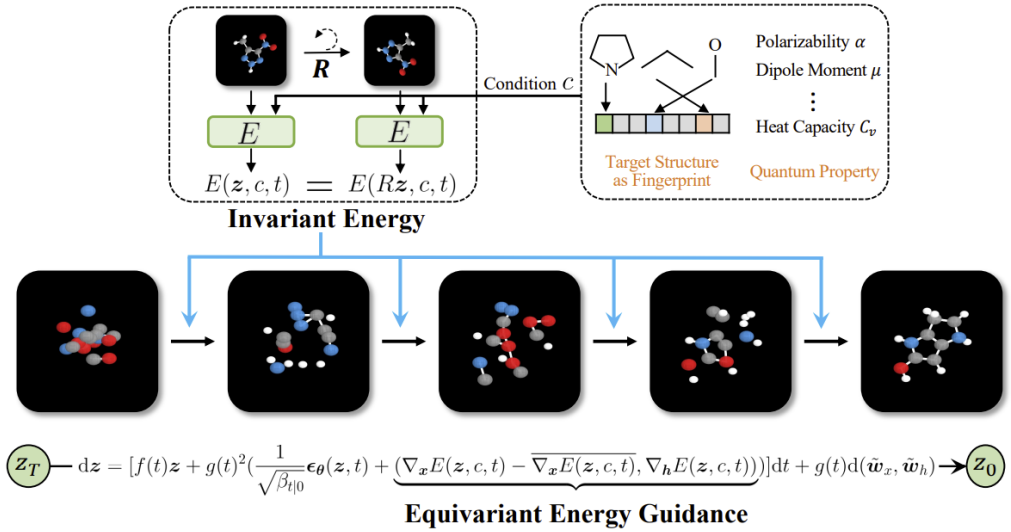


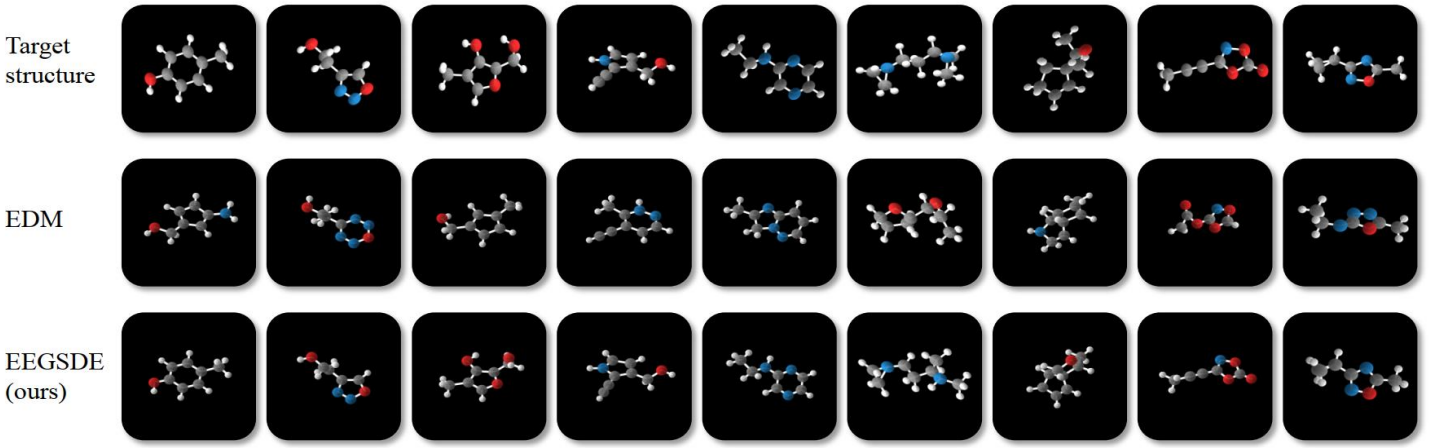
- Energy guidance: enhance alignment to conditions with domain-specific energy functions

EEGSDE for controllable inverse molecular design

Bao & Zhao et al., Arixv 2022

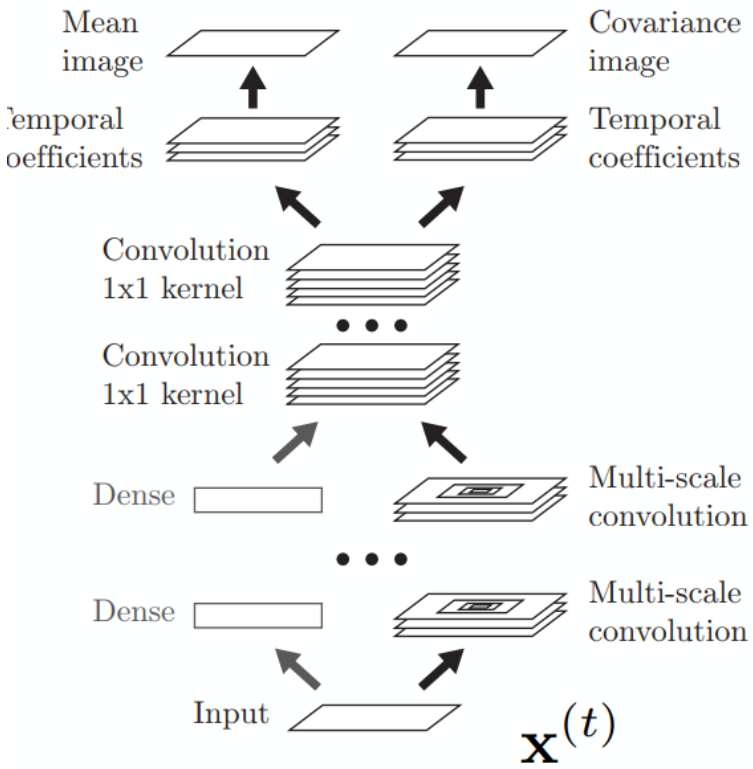
- Equivariant energy guidance for controllable generation with geometry invariance
- A combination of multiple conditions including quantum properties, functional groups and others

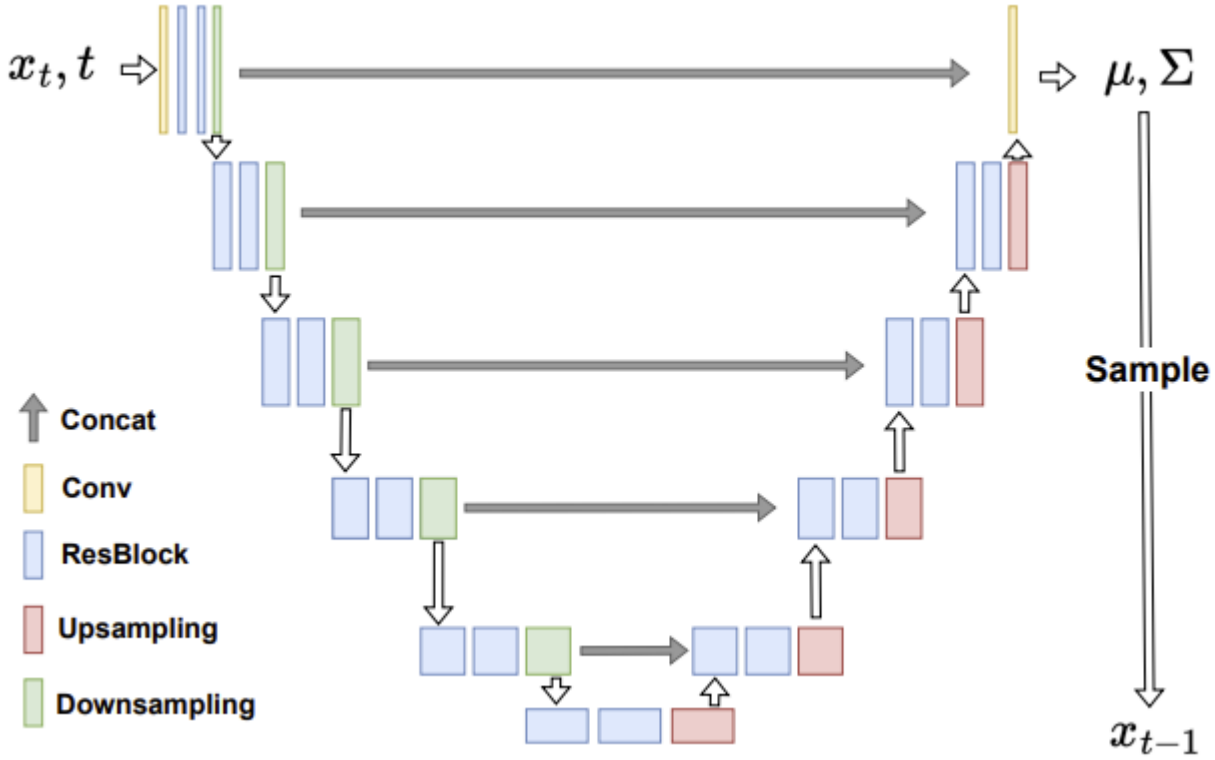




Backbones in conditional diffusion models

Backbones in DPMs





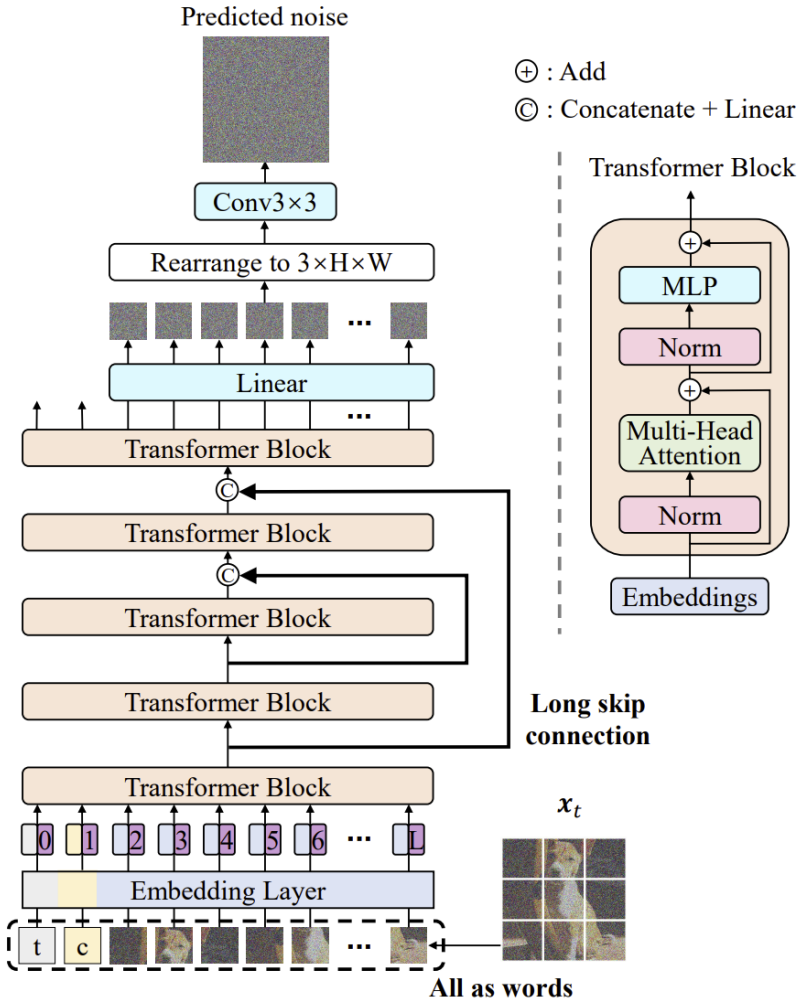
U-Net dominates DPMs

All are worth words: a ViT backbone for diffusion models

Bao et al., Arixv 2022

- U-Net dominants diffusion models. Is every component necessary or not?

- U-ViT:
  - Everything as tokens
  - Long skip connections
  - All transformers except the last layer



All are worth words: a ViT backbone for diffusion models

Bao et al., Arixv 2022

- U-ViT is comparable to U-Net on conditional, unconditional and text-to-image generation:
  - ✓ Long skip connections
  - × Convolution, up-sampling, down-sampling, and special design for conditions
- U-ViT potentially benefits large scale multi-modal generative models

Model	FID	Type	Training datasets
Generative model trained on external large dataset (zero-shot)			
DALL-E [48]	∼ 28	Autoregressive	DALL-E dataset (250M)
CogView [13]	27.1	Autoregressive	Internal dataset (30M)
LAFITE [76]	26.94	GAN	CC3M (3M)
GLIDE [44]	12.24	Diffusion	DALL-E dataset (250M)
Make-A-Scene [18]	11.84	Autoregressive	Union datasets (without MS-COCO) (35M)
DALL-E 2 [47]	10.39	Diffusion	DALL-E dataset (250M)
Imagen [51]	7.27	Diffusion	Internal dataset (460M) + LAION (400M)
Parti [71]	7.23	Autoregressive	LAION (400M) + FIT (400M) + JFT (4B)
Re-Imagen [8]	**6.88**	Diffusion	KNN-ImageText (50M)
Generative model trained on external large dataset with access to MS-COCO			
VQ-Diffusion† [19]	13.86	Discrete diffusion	Conceptual Caption Subset (7M)
Make-A-Scene [18]	7.55	Autoregressive	Union datasets (with MS-COCO) (35M)
Re-Imagen‡ [8]	5.25	Diffusion	KNN-ImageText (50M)
Parti† [71]	**3.22**	Autoregressive	LAION (400M) + FIT (400M) + JFT (4B)
Generative model trained on MS-COCO			
AttnGAN [69]	35.49	GAN	MS-COCO (83K)
DM-GAN [77]	32.64	GAN	MS-COCO (83K)
VQ-Diffusion [19]	19.75	Discrete diffusion	MS-COCO (83K)
DF-GAN [64]	19.32	GAN	MS-COCO (83K)
XMC-GAN [73]	9.33	GAN	MS-COCO (83K)
Friro [17]	8.97	Diffusion	MS-COCO (83K)
LAFITE [76]	8.12	GAN	MS-COCO (83K)
U-Net*	7.32	Latent diffusion	MS-COCO (83K)
~~U-ViT-Small (ours)~~	~~5.95~~	~~Latent diffusion~~	~~MS-COCO (83K)~~
U-ViT-Small-Deep (ours)	**5.48**	Latent diffusion	MS-COCO (83K)

Outlook

Future directions: better diffusion models

- Theoretical understanding and better training
  - How to characterizes the approximation, optimization and generalization behavior?
  - What is the optimal parameterization and discrete time schedule?
- Faster inference (with guidance)
  - Fewer steps: is it possible to use three steps or even a single step to get samples?
  - Faster single steps: quantized, compressed models for edge devices
- Backbones
  - Do score models require special architectures?

Future directions: use diffusion models better

- Controllable and explainable generation
  - Labels, text and energy are far from there
  - Algorithms to support a user-friendly UI
- Larger models
  - How to obtain a specialized model efficiently from large models?
- Applications for generative models
  - Can diffusion models beat GANs everywhere?

Summary of the talk

- **Foundations of diffusion models**
  - Sohl-Dickstein J, Weiss E, Maheswaranathan N, et al. Deep unsupervised learning using nonequilibrium thermodynamics ICML, 2015.
  - Song Y, Ermon S. Generative modeling by estimating gradients of the data distribution[J]. NeurIPS, 2019.
  - Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models[J]. Advances in Neural Information Processing Systems, 2020.
  - Song Y, Sohl-Dickstein J, Kingma D P, et al. Score-based generative modeling through stochastic differential equations[J]. ICLR 2021.
- **Fast inference**
  - Bao F, Li C, Zhu J, et al. Analytic-DPM: an analytic estimate of the optimal reverse variance in diffusion probabilistic models[J]. ICLR 2022.
  - Bao F, Li C, Sun J, et al. Estimating the Optimal Covariance with Imperfect Mean in Diffusion Probabilistic Models[J]. ICML 2022.
  - Lu C, et al. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. NeurIPS 2022.
  - Lu C, et al. DPM-Solver++: Fast Solver for Guided Sampling of Diffusion Probabilistic Models. Arxiv preprint 2022.

Summary of the talk

- **Controllable generation**
  - Alexander Quinn Nichol, Prafulla Dhariwal Improved Denoising Diffusion Probabilistic Models, ICML 2021.
  - Jonathan Ho, Tim Salimans, Classifier-Free Diffusion Guidance Arxiv preprint 2022.
  - Gal R, et al. An image is worth one word: Personalizing text-to-image generation using textual inversion, arXiv 2022.
  - Ruiz N, Li Y, Jampani V, et al. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation, arXiv 2022.
  - Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, Daniel Cohen-Or. Prompt-to-Prompt Image Editing with Cross Attention Control, arXiv 2022.
  - Binxin Yang et al. Paint by Example: Exemplar-based Image Editing with Diffusion Models. arXiv 2022.
  - Zhao M, Bao F, Li C, Zhu J. EGSDE: Unpaired Image-to-Image Translation via Energy-Guided Stochastic Differential Equations[J]. NeurIPS 2022.
  - Bao F, Zhao M, Hao Z, Li P, Li C, Zhu J. Equivariant Energy-Guided SDE for Inverse Molecular Design. Arxiv preprint 2022.
  - Bao F et al. All are Worth Words: A ViT Backbone for Diffusion Models. Arxiv preprint 2022.

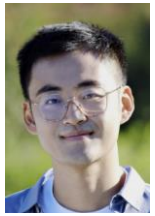
Collaborators

Bo Zhang

Jun Zhu

Hang Su

Jianfei Chen

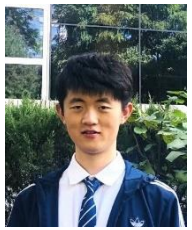


Yue Cao

Jiacheng Sun

Cheng Lu

Min Zhao



Shen Nie

Kaiwen Xue

Peiyao Li

Zhongkai Hao

Source code

- Fast inference
  - Analytic-DPM: https://github.com/baofff/Analytic-DPM
  - Analytic-DPM++: https://github.com/baofff/Extended-Analytic-DPM
  - DPM-Solver(++): https://github.com/LuChengTHU/dpm-solver

- Controllable generation
  - EGSDE: https://github.com/ML-GSAI/EGSDE
  - U-ViT: https://github.com/baofff/U-ViT

Thank you!

Email: chongxuanli@ruc.edu.cn

Homepage: https://zhenxuan00.github.io/