Score-Based Diffusion Models

LECTURE 16

CS236: Deep Generative Models

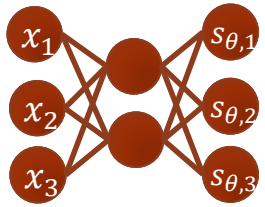
Plan for today

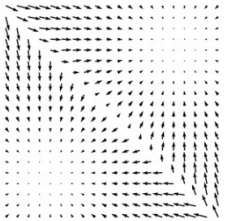
1. Recap on score-based models
2. Diffusion models as score-based models
3. Diffusion models as (hierarchical) VAEs
4. Diffusion models as normalizing flow models
5. Efficient sampling strategies
6. Controllable generation

Score-based models

- A model that represents the score function

$$s_\theta(\mathbf{x})$$





- **Score estimation:** training the score-based model from datapoints
- Score matching

$$\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}[\|\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x})\|_2^2]$$
$$=\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}\left[\frac{1}{2}\|\boldsymbol{s}_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}}\boldsymbol{s}_\theta(\mathbf{x}))\right] + \text{const.}$$

- Not scalable for deep score-based models and high dimensional data

Denoising score matching

$\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$

Data distribution

$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$

Perturbation distribution/kernel

$\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}})$

Noise-perturbed data distribution

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim p_{\text{data}}} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) \|_2^2]$$

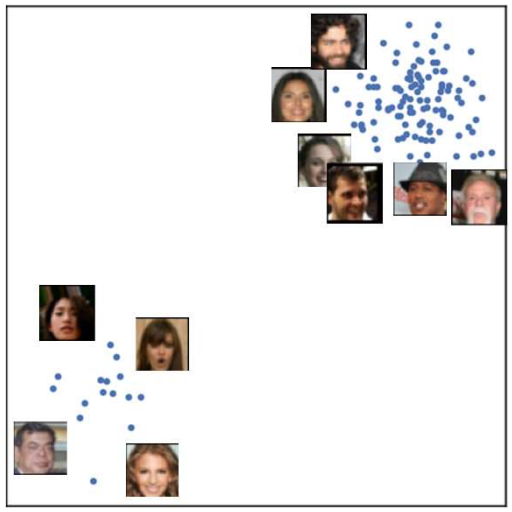
$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2] + \text{const.}$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\left\| \boldsymbol{s}_\theta(\mathbf{x} + \sigma \mathbf{z}) + \frac{\mathbf{z}}{\sigma} \right\|_2^2 \right] + \text{const.}$$

- **Pros:**
  - Much more scalable than score matching
  - Reduces score estimation to a denoising task
- **Con:** estimates score of noise-perturbed data

$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x}) \neq \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

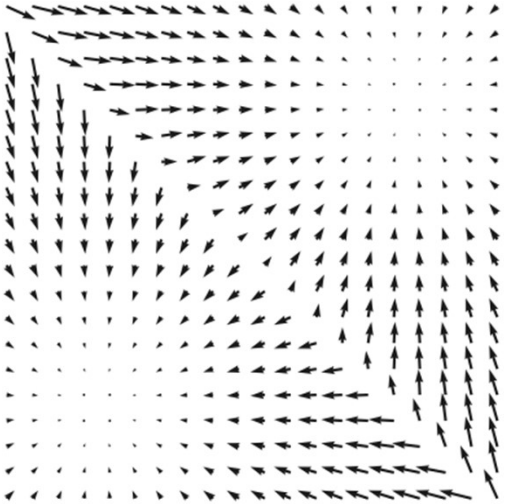
Score-based generative modeling



Data samples

$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$$

score matching



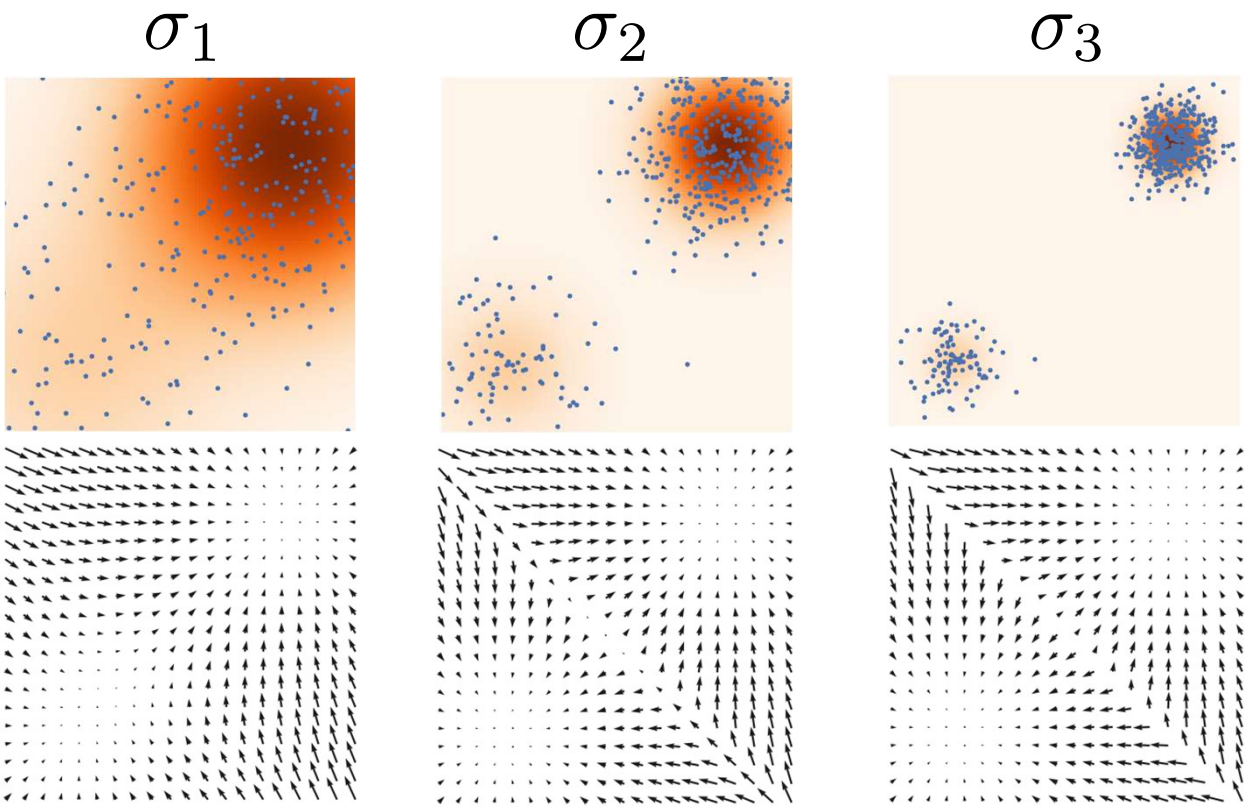
Scores

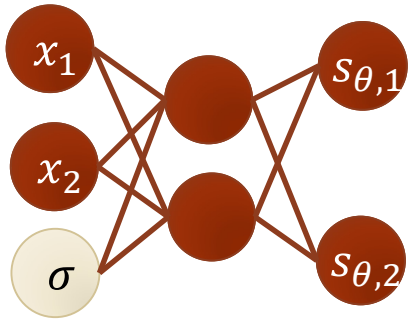
$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

Langevin dynamics

New samples

Joint Score Estimation via Noise Conditional Score Networks



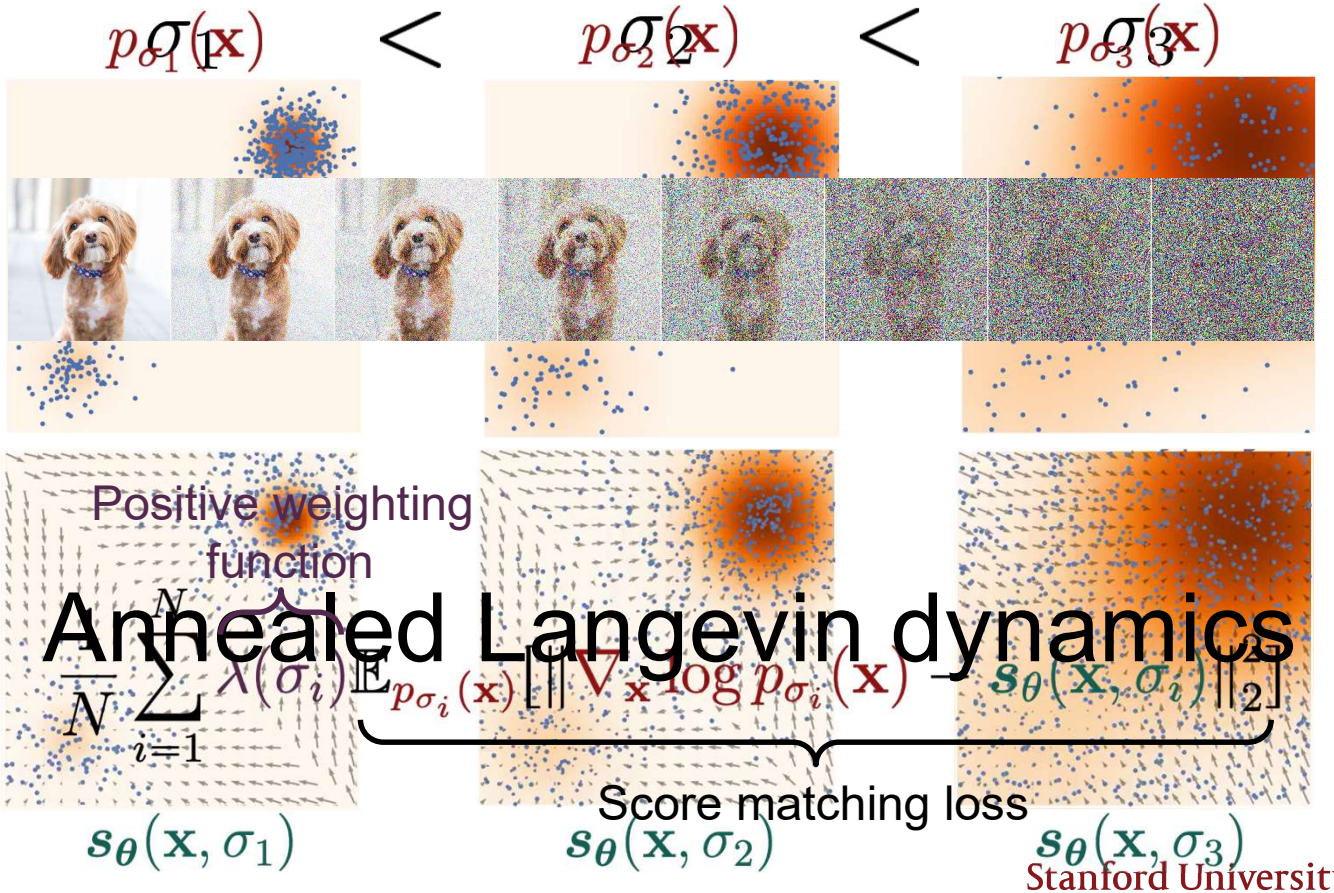


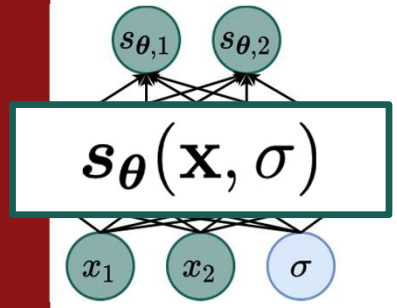
Noise Conditional Score Network (NCSN)

Stanford University

Using multiple noise levels

Data

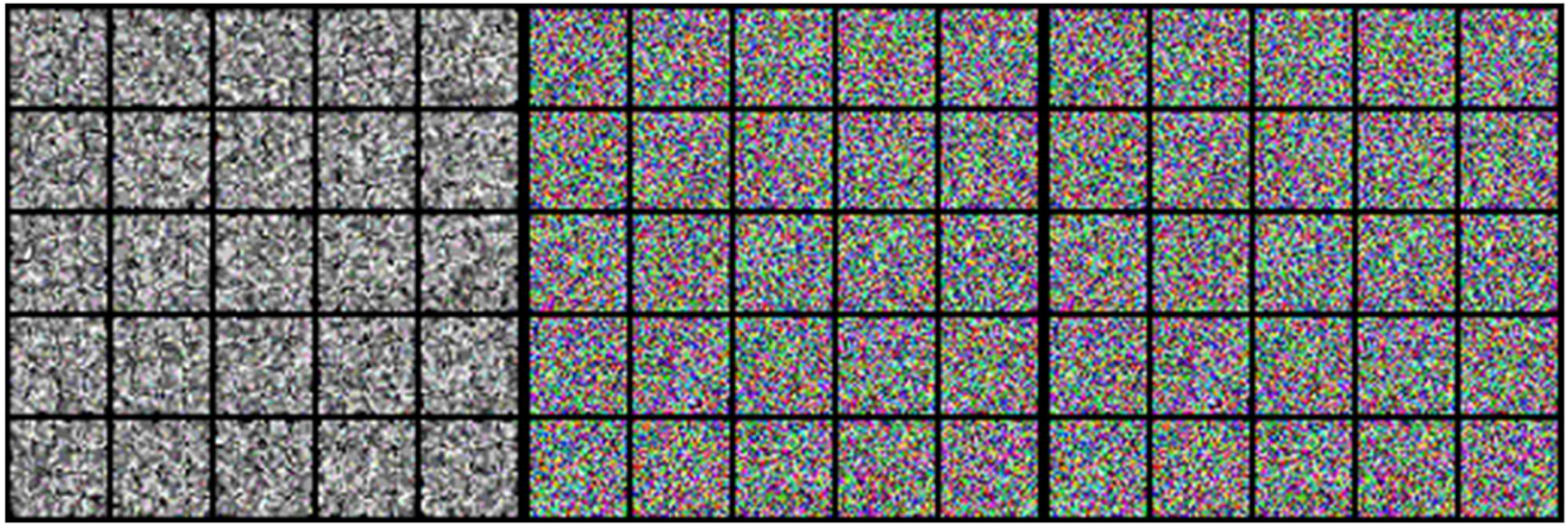




Noise Conditional Score Model

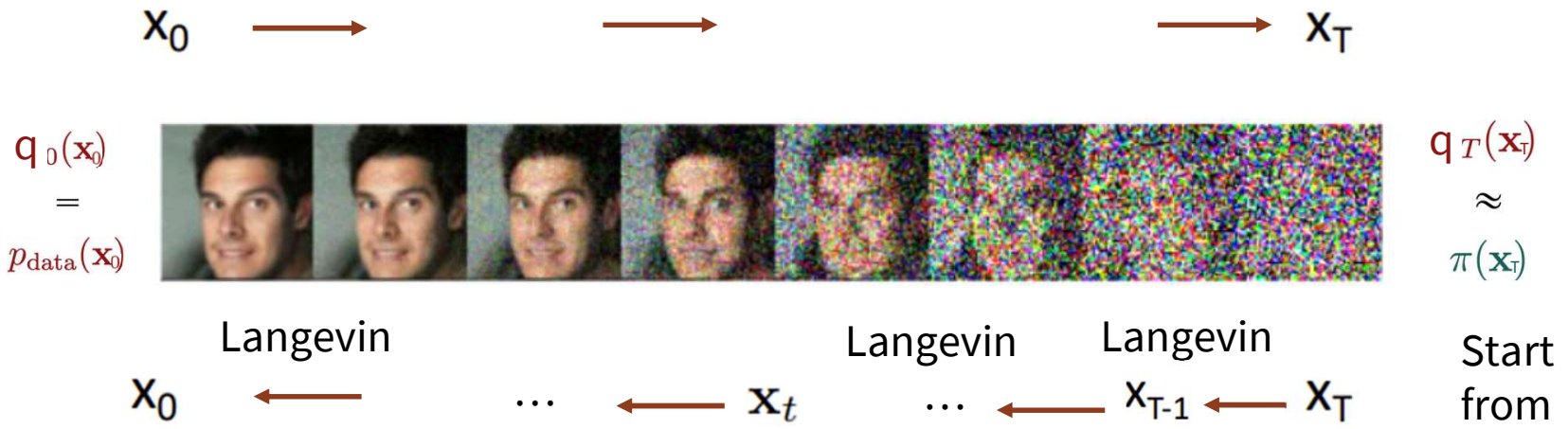
Annealed Langevin dynamics

Experiments: Sampling



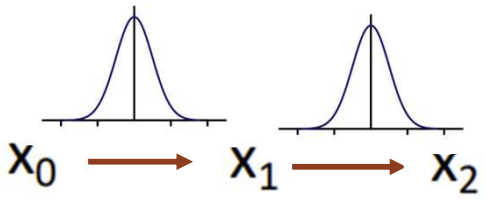
Sampling as Iterative Denoising

Inverse process: iteratively add Gaussian noise



Iteratively remove noise

Iterative noising process



Progressively add noise

$\longrightarrow$ x_T

$q_0(\mathbf{x}_0)$
$=$
$p_{\text{data}}(\mathbf{x}_0)$

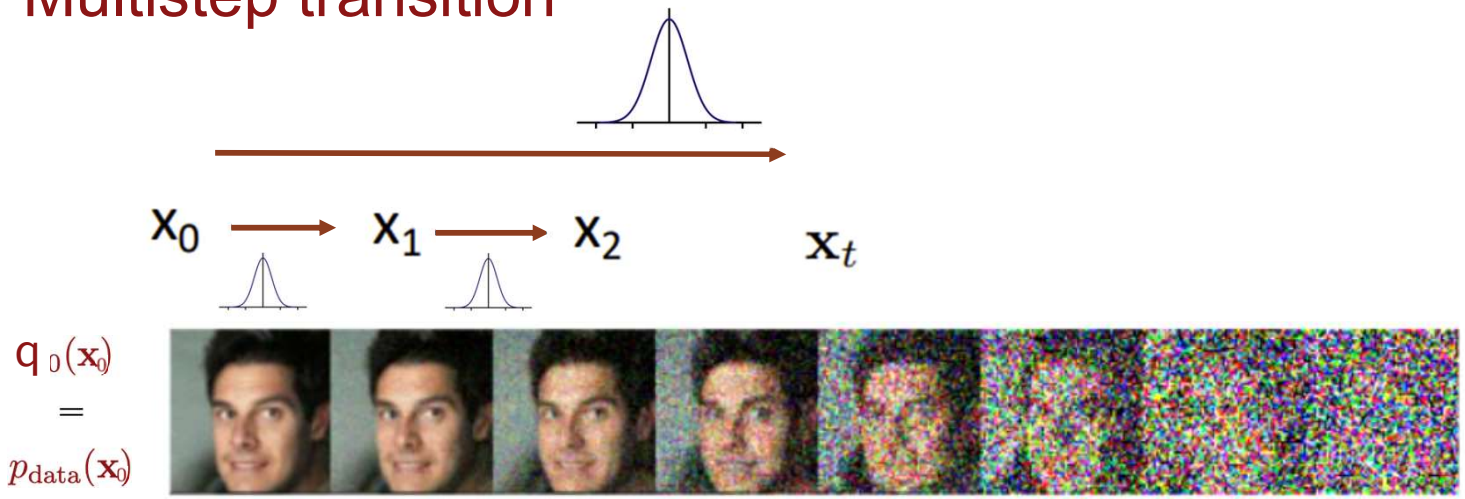
$q_T(\mathbf{x}_T)$
$\approx$
$\pi(\mathbf{x}_T)$

Noise perturbed densities are obtained by adding Gaussian noise

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

Defines a joint distribution $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$

Multistep transition



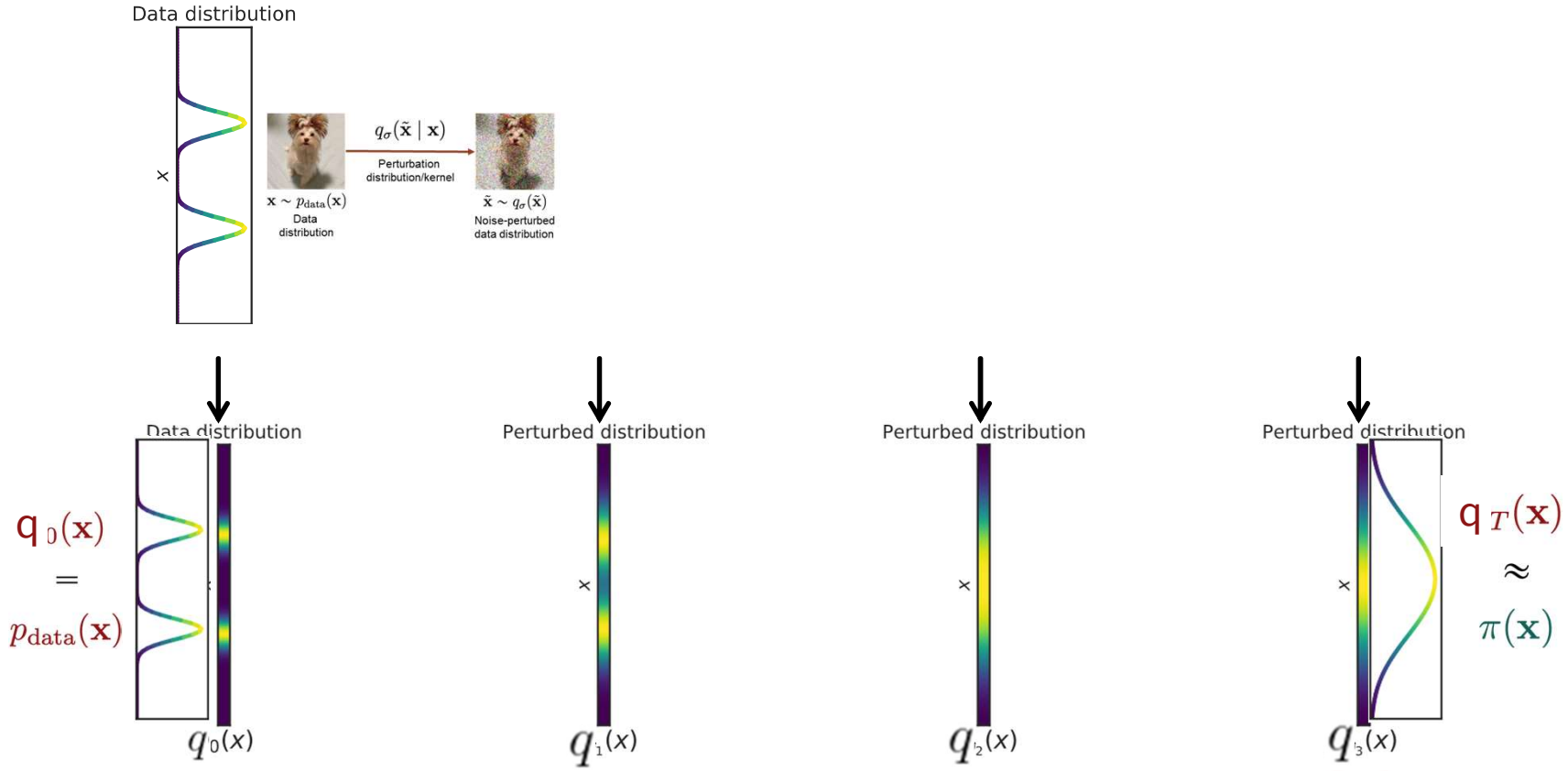
Multi-step transitions are also Gaussian and can be computed in closed form

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$$

$$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$$

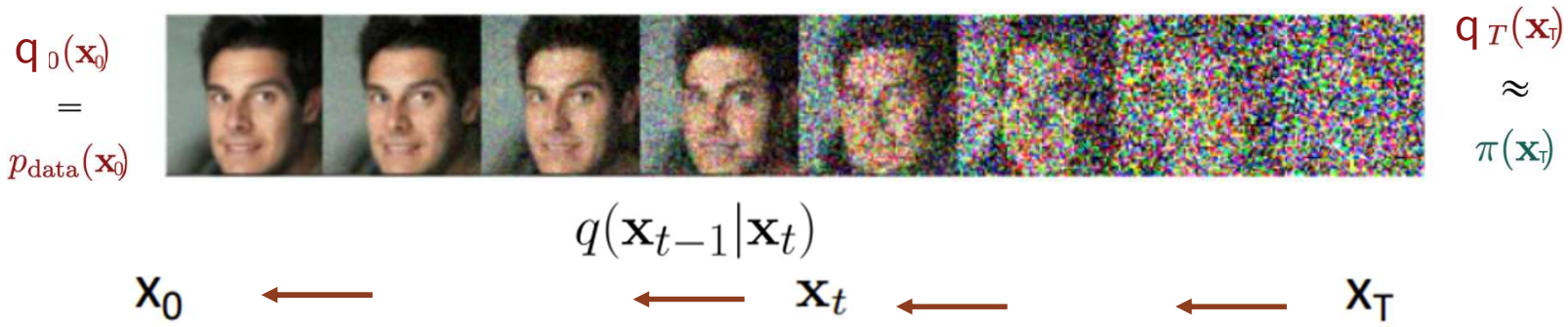
1) Same as noise-perturbed data distributions in score-based models
2) Efficient sampling at any t

Diffusion perspective



Stanford University

Iterative Denoising



Ideal sampling process:

1. Sample $\mathbf{x}_T$ om $\pi(\mathbf{x}_T)$ i.e. from pure noise
2. Iteratively sample from the true denoising distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

Issue: Exact denoising distribution is unknown!

Solution: Learn a variational approximation

Iterative Denoising

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

$q_T(\mathbf{x}_T) \approx \pi(\mathbf{x}_T)$

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) \approx p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

$\mathbf{x}_0 \leftarrow \cdots \leftarrow \mathbf{x}_t \leftarrow \cdots \leftarrow \mathbf{x}_T$

Sample $\mathbf{x}_T$ from $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}) = \pi$

Iteratively sample from $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

Joint distribution: $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

Diffusion model as a hierarchical VAE

Encoder (**fixed**): $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$\mathbf{x}_0$ → → → $\mathbf{x}_T$

$q_T(\mathbf{x}_T) \approx \pi(\mathbf{x}_T)$

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

$p(\mathbf{x}_T)$

$\mathbf{x}_0$ ← ← ← $\mathbf{x}_T$

Prior over latents

Decoder (**learnable**): $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

From VAE to Hierarchical VAE

Basic VAE:

A mixture of an infinite number of Gaussians:

1. $\mathbf{z} \sim \mathcal{N}(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$ where μ_θ,Σ_θ are neural networks

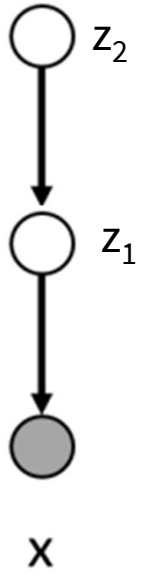
ELBO training:

$$E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q_\phi(\mathbf{z}|\mathbf{x}))]$$

From VAE to Hierarchical VAE

Hierachical VAE (decoder): $p(x, z_1, z_2) = p(z_2)\, p(z_1 \mid z_2)\, p(x \mid z_1)$

1. Sample z_2 from a simple prior $N(0, I)$
2. Sample z_1 from a decoder $p(z_1 \mid z_2)$
3. Sample x from a decoder $p(x \mid z_1)$



Encoder: $q(z_1, z_2 \mid x)$

ELBO training: $E_{q(z_1, z_2 \mid x)}\left[\log\left(\frac{p(x, z_1, z_2)}{q(z_1, z_2 \mid x)}\right)\right]$

Training a denoising diffusion probabilistic model

Encoder (q) has no learnable parameters!

Encoder (**fixed**): $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$\mathbf{x}_0$ → $\mathbf{x}_T$

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

$p(\mathbf{x}_T)$

Decoder: $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

$\mathbf{x}_0$ ← $\mathbf{x}_T$

ELBO loss (negative ELBO averaged over data distribution):

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Diffusion models as score-based models

The ELBO objective is

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Decoder parameterization: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(\mathbf{x}_t, t)\right)$$

Up to scaling, predict noise that was added and subtract it

$$L = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0), t\sim\mathcal{U}\{1,T\}, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\lambda_t||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon, t)||^2\right]$$

ELBO loss reduces to denoising score matching!

Training and inference

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Denoising score matching training

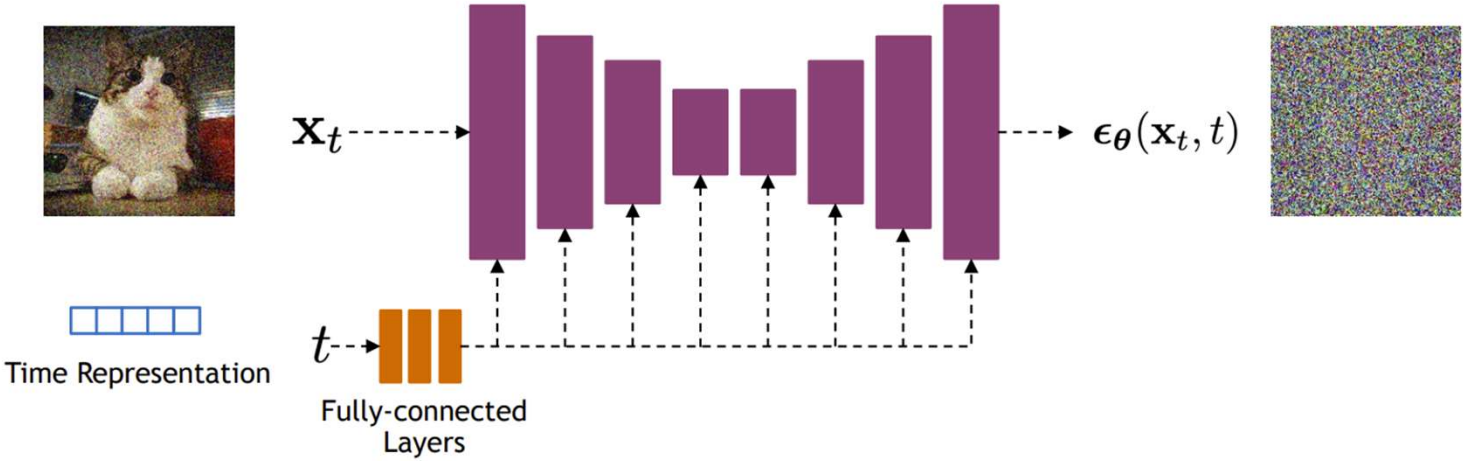
$$\frac{1}{L} \sum_{i=1}^{L} E_{\mathbf{x} \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\left\| \boldsymbol{\epsilon}_\theta(\mathbf{x} + \sigma_i \mathbf{z}, \sigma_i) + \mathbf{z} \right\|_2^2 \right]$$

Iteratively Sample from decoders $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ (Annealead Langevin Dynamics)

$$[\ \boldsymbol{\epsilon}_\theta(\cdot, \sigma_i) := \sigma_i \boldsymbol{s}_\theta(\cdot, \sigma_i)\ ]$$

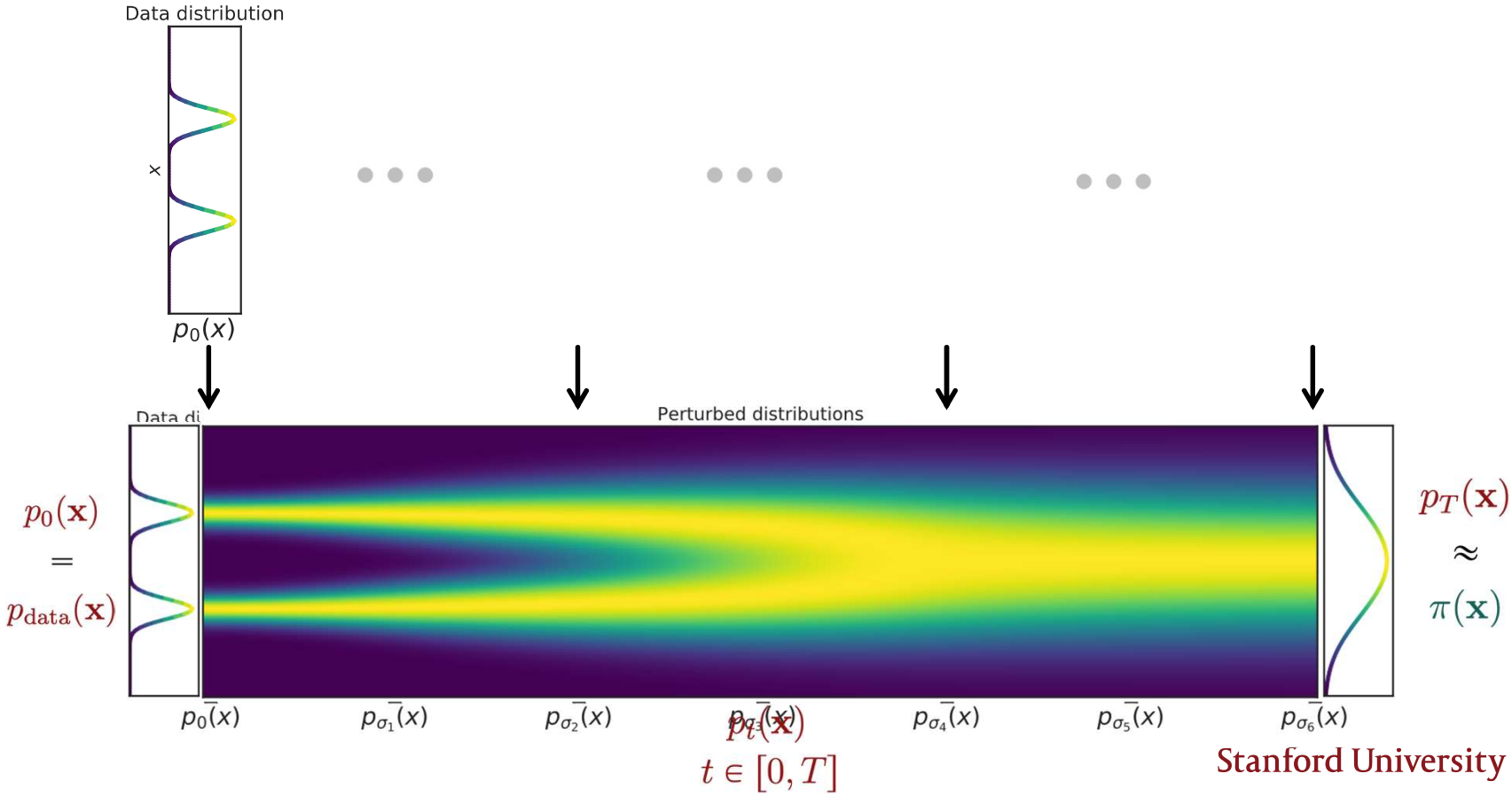
Architecture for the denoiser

Unet architecture used in practice



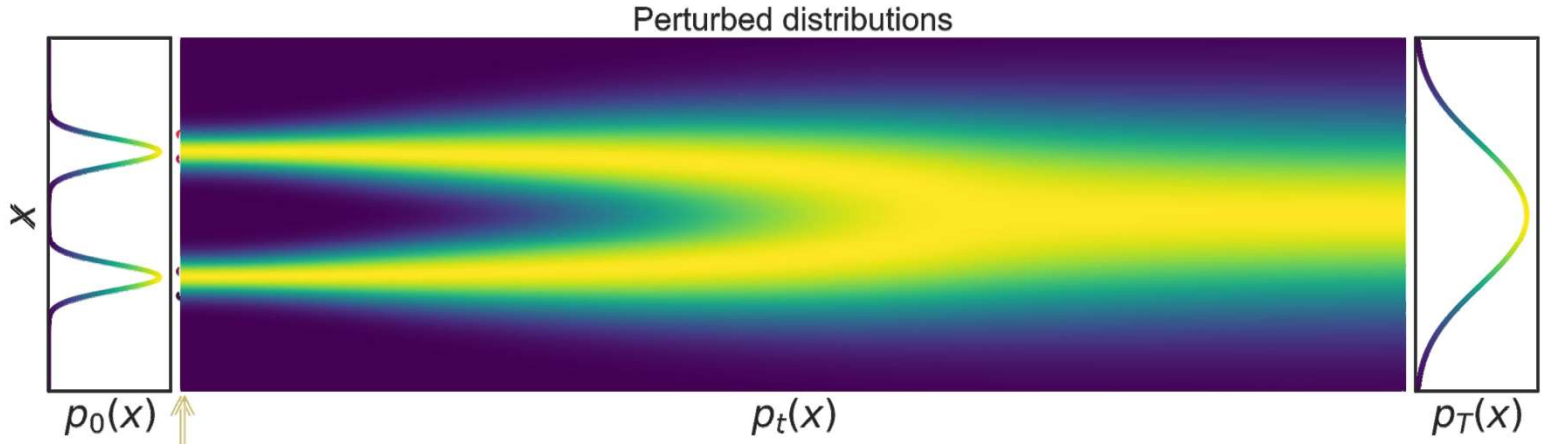
Same as the noise-condtional score network (t represents the time index or equivalently the noise level)

Infinite noise levels

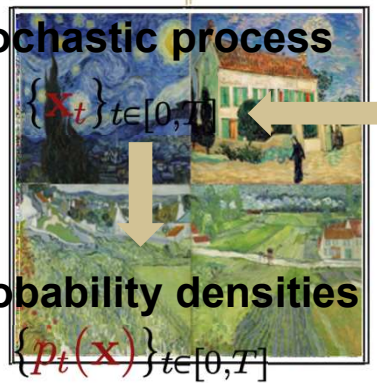


Stanford University

Perturbing data with stochastic processes



Stochastic process



Probability densities

Stochastic differential equation (SDE)

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

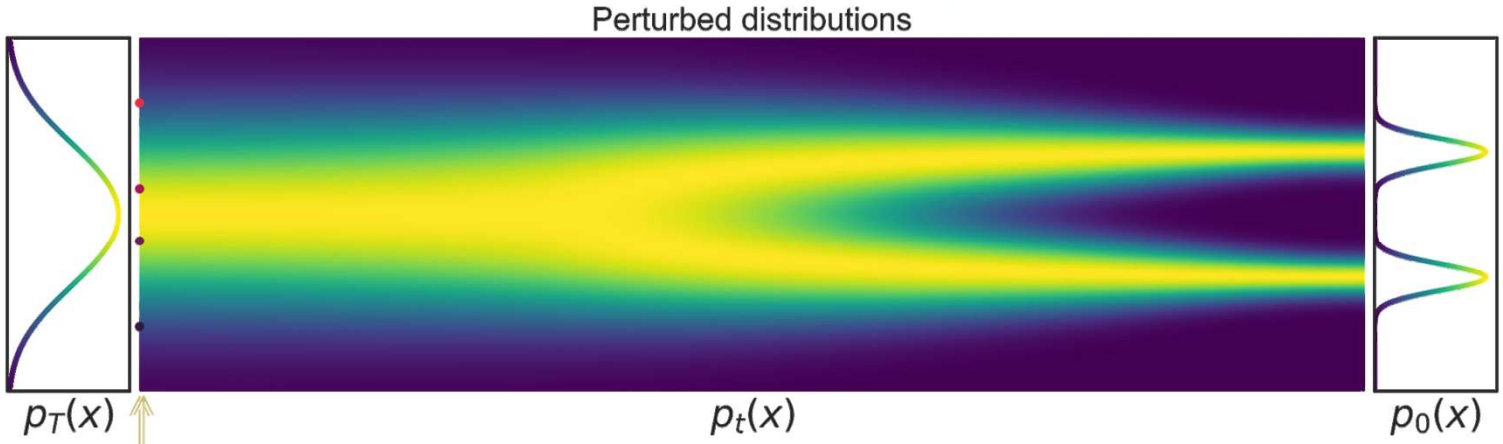
Deterministic drift — Infinitesimal noise

WLOG: Toy SDE

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

$$p_T(\mathbf{x}) \approx \pi(\mathbf{x})$$

Generation via reverse stochastic processes



$\pi(\mathbf{x}) \approx p_T(\mathbf{x})$

Forward SDE (t: 0→T)

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

Reverse SDE (t: T→0)

$$\mathrm{d}\mathbf{x}_t = -\sigma(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\bar{\mathbf{w}}_t$$

Infinitesimal noise in the reverse time direction

Score function!

Score-based generative modeling via SDEs

- Time-dependent score-based model

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

- Training:

$$\mathbb{E}_{t \in \mathcal{U}(0,T)}[\lambda(t) \mathbb{E}_{p_t(\mathbf{x})}[\| \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \|_2^2]]$$

- Reverse-time SDE

$$\mathrm{d}\mathbf{x} = -\sigma^2(t) \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \mathrm{d}t + \sigma(t) \mathrm{d}\bar{\mathbf{w}}$$

- Euler-Maruyama (analgous to Euler for ODEs)

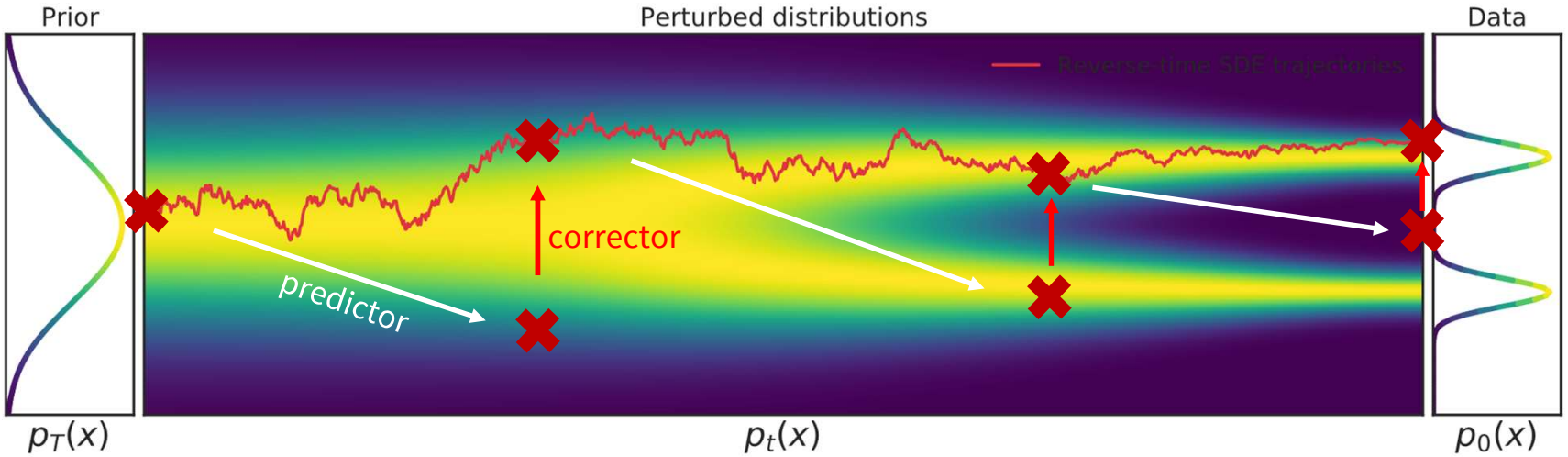
$$\mathbf{x} \leftarrow \mathbf{x} - \sigma(t)^2 \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \Delta t + \sigma(t)\ \mathbf{z} \quad (\mathbf{z} \sim \mathcal{N}(\mathbf{0}, |\Delta t|\ \mathbf{I}))$$

$$t \leftarrow t + \Delta t$$

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.

Predictor-Corrector sampling methods

- Predictor-Corrector sampling.
  - **Predictor:** Numerical SDE solver
  - **Corrector:** Score-based MCMC



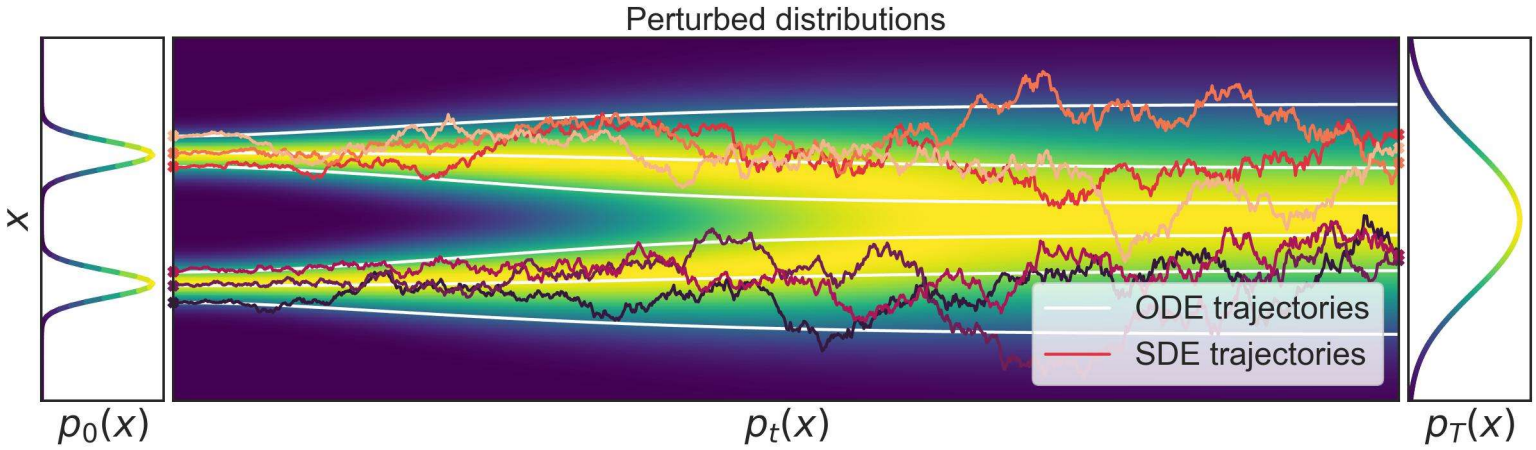
Results on predictor-corrector sampling

Model	FID↓	IS↑
Conditional		
BigGAN (Brock et al., 2018)	14.73	9.22
StyleGAN2-ADA (Karras et al., 2020a)	**2.42**	**10.14**
Unconditional		
StyleGAN2-ADA (Karras et al., 2020a)	2.92	9.83
NCSN (Song & Ermon, 2019)	25.32	8.87 ± .12
NCSNv2 (Song & Ermon, 2020)	10.87	8.40 ± .07
DDPM (Ho et al., 2020)	3.17	9.46 ± .11
DDPM++	2.78	9.64
DDPM++ cont. (VP)	2.55	9.58
DDPM++ cont. (sub-VP)	2.61	9.56
DDPM++ cont. (deep, VP)	2.41	9.68
DDPM++ cont. (deep, sub-VP)	2.41	9.57
NCSN++	2.45	9.73
NCSN++ cont. (VE)	2.38	9.83
NCSN++ cont. (deep, VE)	**2.20**	**9.89**

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.

High-Fidelity Generation for 1024x1024 Images

Converting the SDE to an ODE



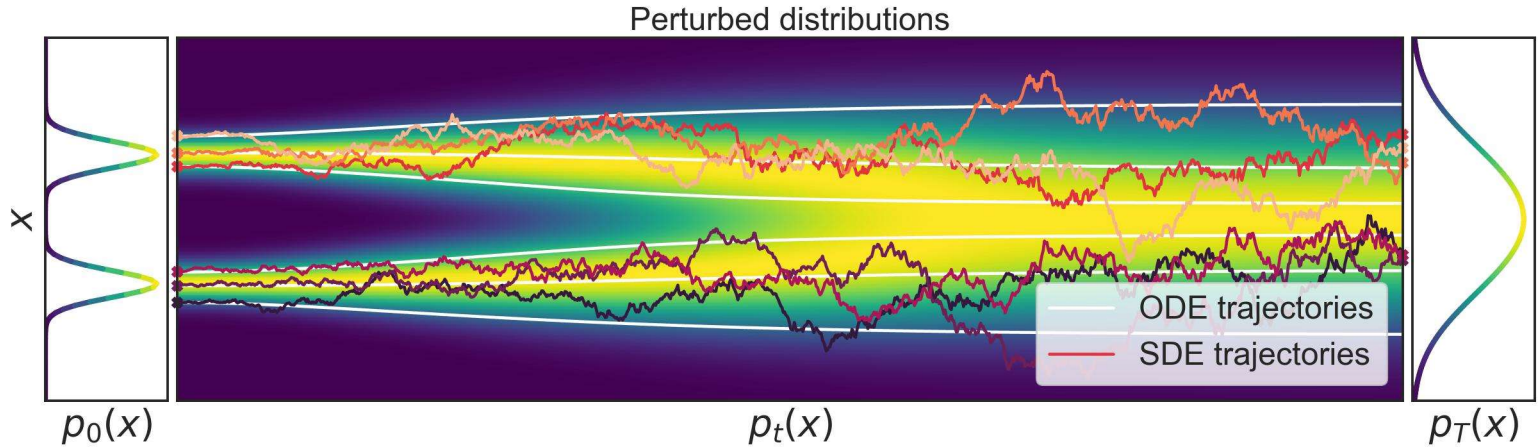
SDE **Ordinary differential equation (ODE)**

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t \quad \Longleftrightarrow \quad \frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -\frac{1}{2}\sigma(t)^2 \boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)}$$

Score function

$$\approx \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$$

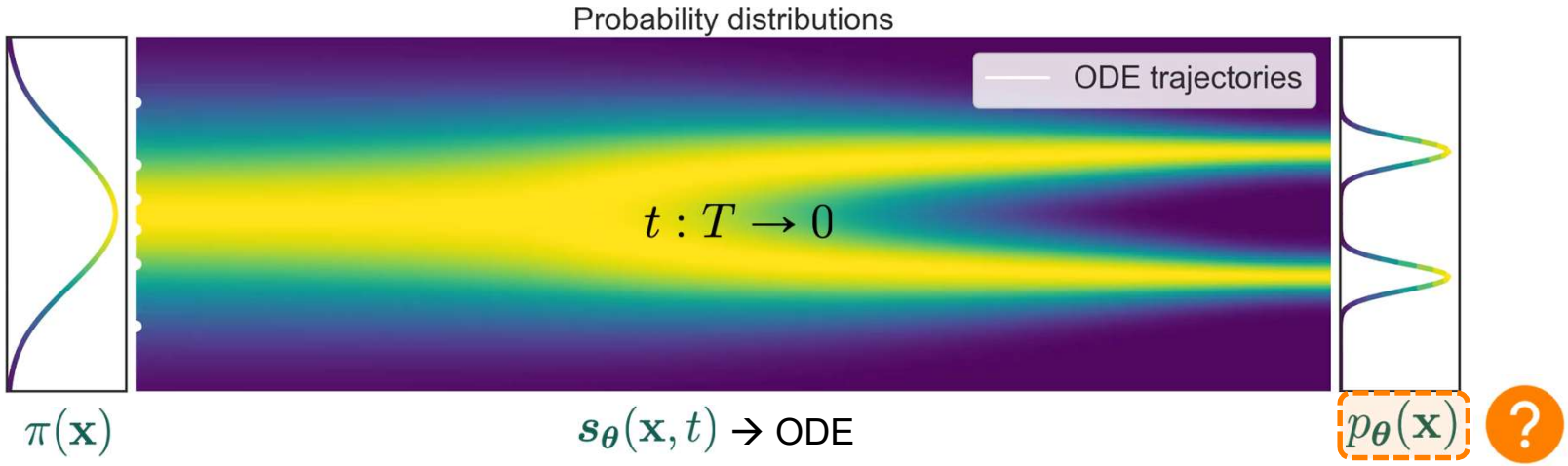
Converting the SDE to an ODE



We can think of this as a (continuous time, infinite depth) normalizing flow

1. Unique ODE solution → Invertible mapping
2. To invert, solve ODE backwards from T to 0

Evaluating likelihoods with ODEs (flow model)



Computing the probability density function (change of variables formula)

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_0) = \log \pi(\mathbf{x}_T) - \frac{1}{2}\int_0^T \sigma(t)^2 \operatorname{trace}\left(\nabla_{\mathbf{x}} \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\right) \mathrm{d}t$$

ODE solver

Computable in polynomial time

Unbiased estimator

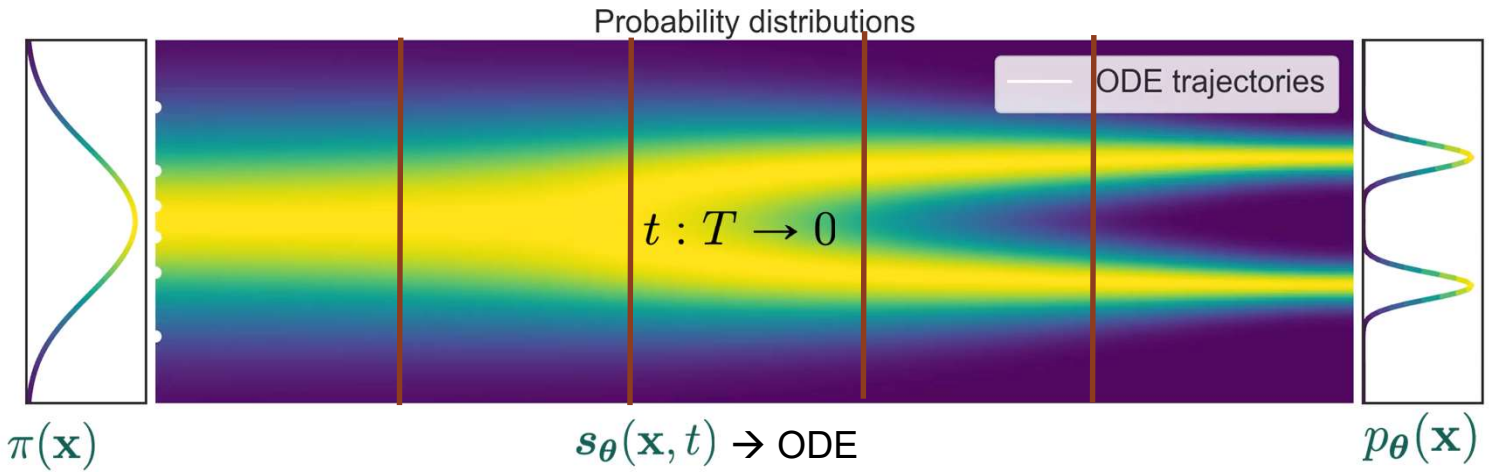
Competitive likelihoods on test data

Negative log-probability ↓ (**bits/dim**)

Method	CIFAR-10	ImageNet 32x32
PixelSNAIL [Chen et al. 2018]	2.85	3.80
Delta-VAE [Razavi et al. 2019]	2.83	3.77
Sparse Transformer [Child et al. 2019]	2.80	–

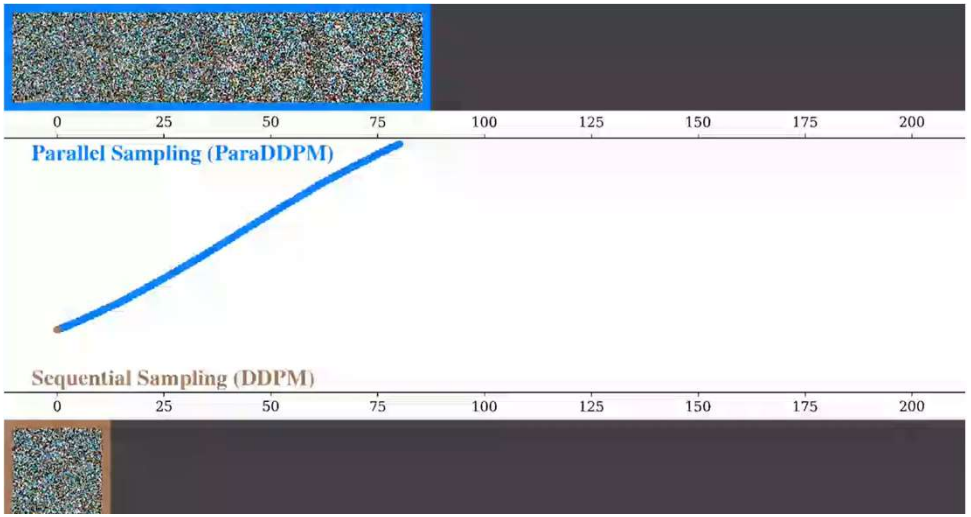
Challenges years of dominance of autoregressive models and VAEs

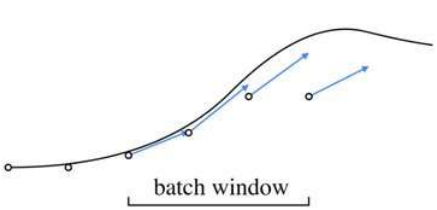
Accelerated sampling



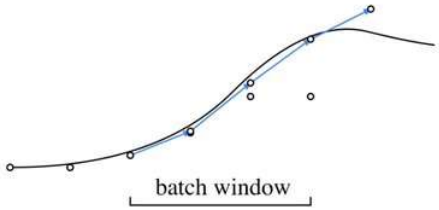
- Numerical methods + ODE formulation to accelerate sampling
- DDIM [Song and Ermon, 2021]:
  - Coarsely discretize the time axis, take big steps
  - Corresponds to exponential integrator (semi-linear ODE) [Lu et al, 2022; Zhang and Chen, 2022]
  - 10x-50x speedups, comparable sample quality

Parallel ODE solving

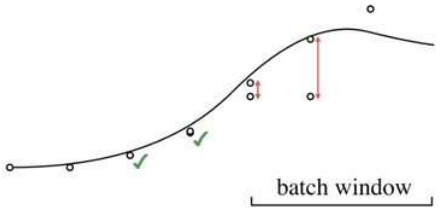




(a) Compute the drift of $x^k_{t:t+p}$ on a batch window of size $p = 4$, in parallel



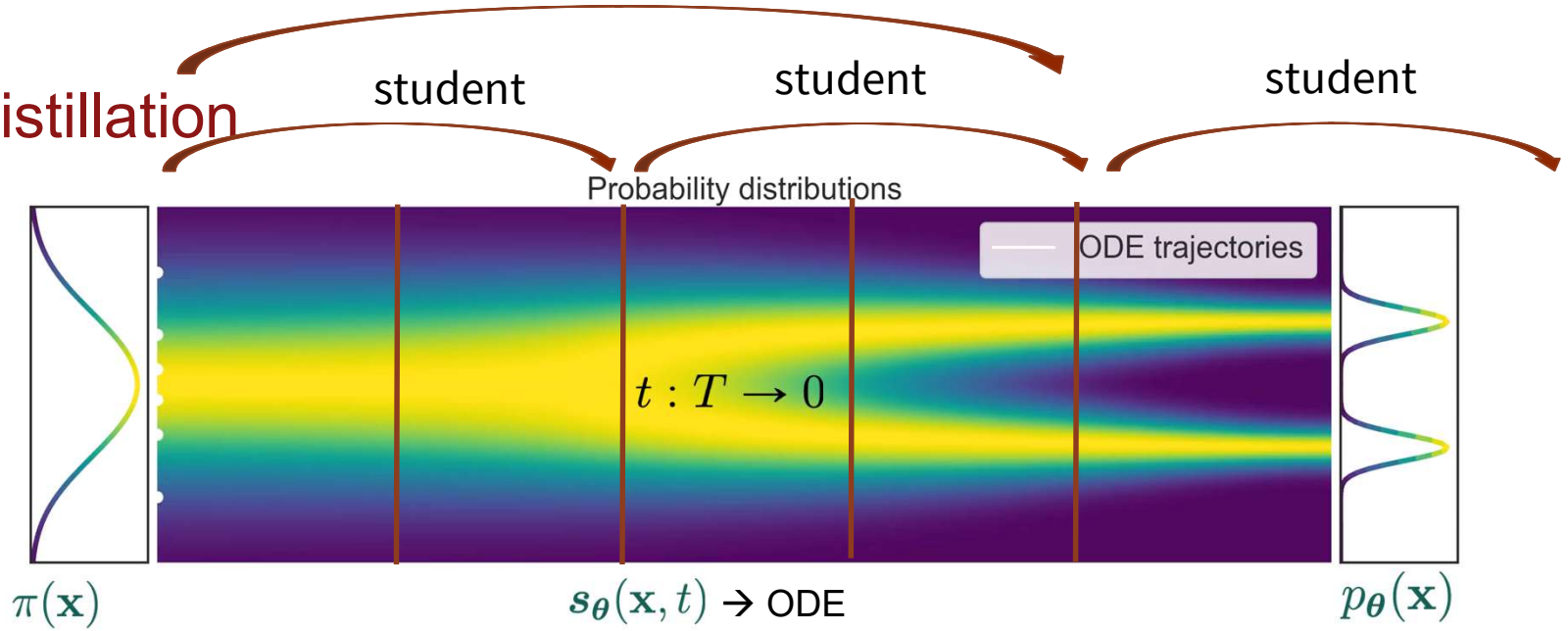
(b) Update the values to $x^{k+1}_{t:t+p}$ using the cumulative drift of points in the window



(c) Determine how far to slide the window forward, based on the error $\|x_i^{k+1} - x_i^k\|^2$.

versity

Distillation



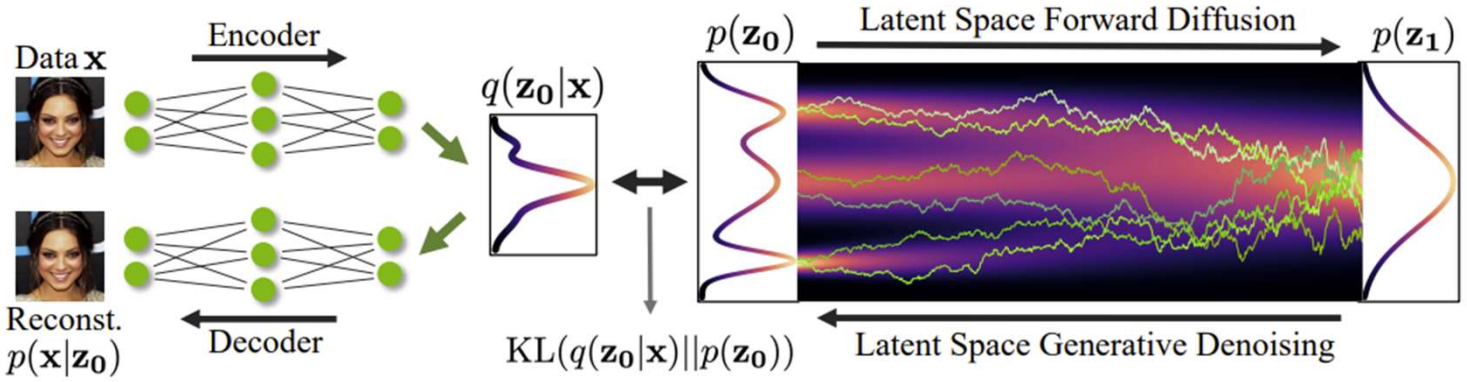
- Progressive distillation [Salimans, Ho 2022]
  - DDIM sampler as a teacher model
  - Student model trained to do in 1 step what DDIM achieves in 2 steps
  - Applied recursively to drastically reduce the number of steps required

On distillation of guided diffusion models

(a) 2 denoising steps (b) 4 denoising steps (c) 8 denoising steps

Meng, Rombach, Gao, Kingma, Ermon, Ho, Salimans "On distillation of guided diffusion models." 2023.

Latent diffusion model

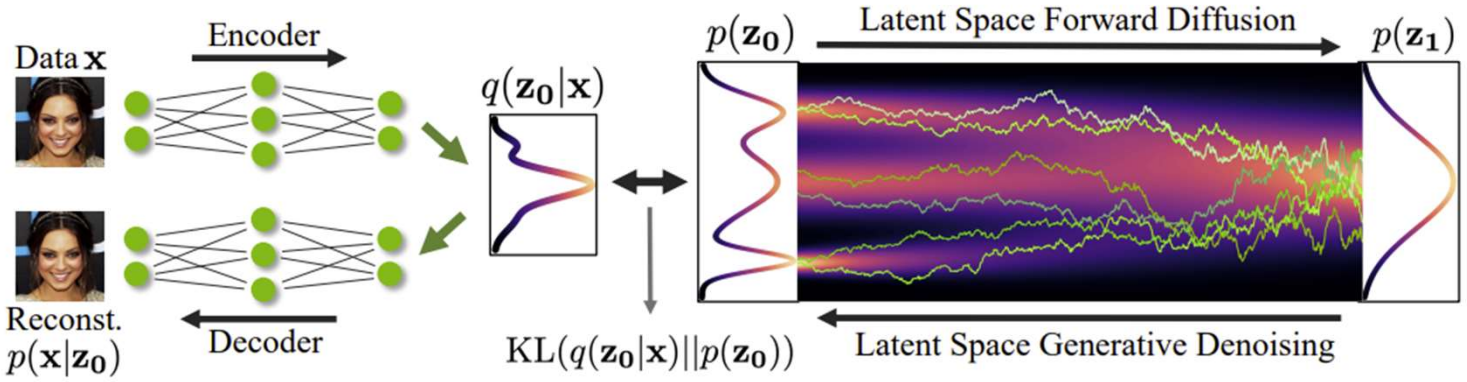


VAE mapping data to lower dimensional space

1. **Faster**
2. **Can be applied to any data type (e.g. discrete)**

Diffusion model prior over the latent space of the autoencoder

Stable diffusion text2image model



VAE mapping data to lower dimensional space

1. Pre-trained, focus on reconstruction (autoencoder)

Diffusion model prior over the latent space of the autoencoder

2. Trained in the second stage, keeping initial autoencoder fixed

Large scale, open source model, widely adopted

Conditional generation

User input:

An astronaut riding a horse

Conditional generation

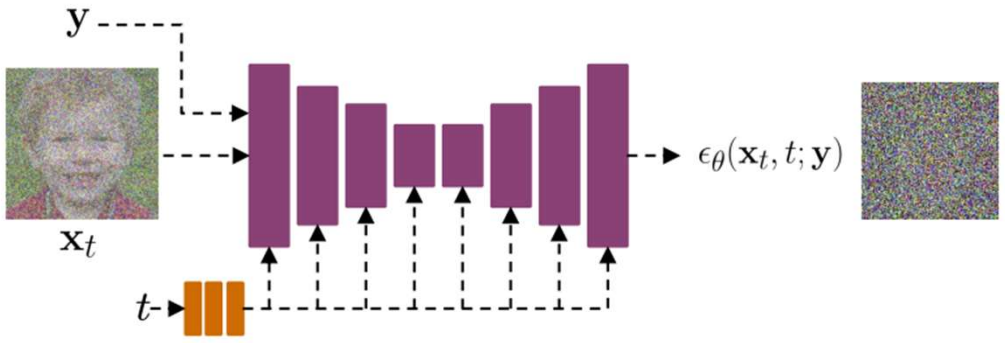
Let (x,y) denote (image,caption) pairs

Training a conditional generative model involves learning p(x | y)

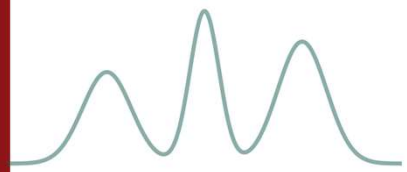
Train score model for the image x conditional on caption y

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}(\mathbf{x},\mathbf{y})}\mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\mathbb{E}_{t\sim\mathcal{U}[0,T]}\ ||\epsilon_\theta(\mathbf{x}_t, t; \mathbf{y}) - \epsilon||_2^2$$

Need a suitable architecture



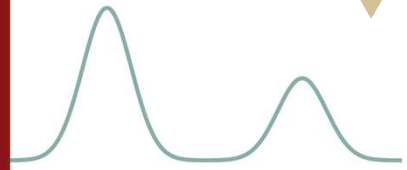
Control the generation process



$p(\mathbf{x})$

Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”



$p(\mathbf{x} \mid \mathbf{y})$

Inverse distribution

Bayes’ rule:

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x})\, p(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y})}$$

Bayes’ rule for score functions:

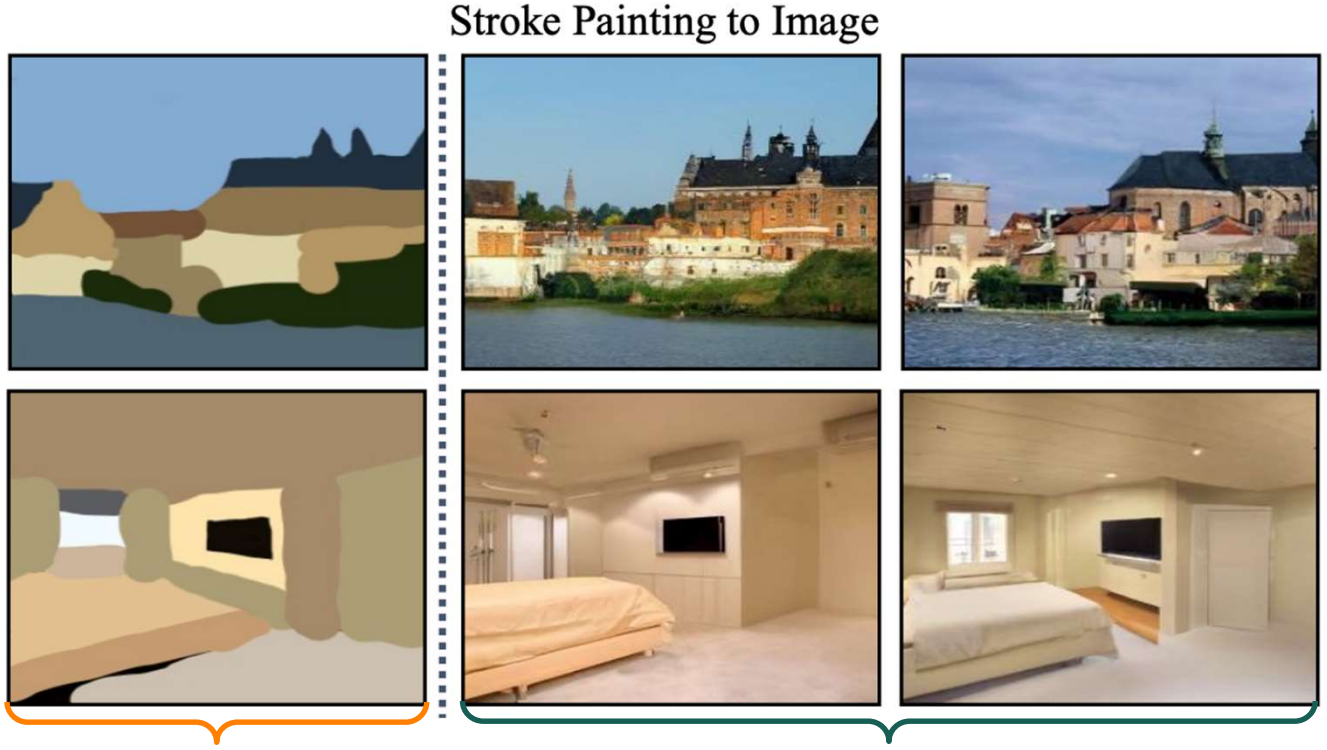
$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{y})$$

$$= \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$$

$\nabla_{\mathbf{x}} \log p(\mathbf{y})$ → 0

Plug in different forward models for the same score model

Stroke to image synthesis



Stroke paintings

$\mathbf{y}$

Sampled images

$\mathbf{x} \mid \mathbf{y}$

Forward model

$p(\mathbf{y} \mid \mathbf{x})$

can be specified.

[Meng, He, Song, Song, Wu, Zhu, Ermon. ICLR 2022]

Language-guided image generation

$\mathbf{y}$

$\mathbf{x} \mid \mathbf{y}$

(Prompt)
Treehouse in the style of Studio Ghibli animation

Forward model

$p(\mathbf{y} \mid \mathbf{x})$

is an image captioning neural network.

[Work by @danielrussruss]

Controllable generation: Text-guided generation

An abstract painting of computer science:

A painting of the starry night by van Gogh

https://colab.research.google.com/drive/1FuOobQOmDJuG7rGsMWfQa883A9r4HxEO?usp=sharing

Classifier-free guidance

Bayes' rule:

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x})\, p(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y})}$$

Bayes' rule for score functions:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{y})$$

$$= \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$$

Conditional score

0

Unconditional score

Classifier obtained as the difference

Classifier-free guidance

Train both a conditional and an unconditional score model (by randomly dropping the caption during training)

Combine the two models as follows

$$(1+w)\nabla_{\mathbf{x}_t}\log p(\mathbf{y}|\mathbf{x}_t)+\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)=(1+w)\big(\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{y})-\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)\big)+\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)$$

$$=(1+w)\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{y})-w\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)$$

w is the classifier-guidance strength

Effect of classifier guidance

Increased classifier guidance strength (w)

Summary

Discrete time diffusion model as a hierarchical VAE

- Connections with score-based models (ELBO equivalence to score matching)

Continuous time diffusion models

- SDE perspective: VAE with an infinite number of latent variables
- ODE perspective: normalizing flow (exact likelihoods!)

Accelerated sampling

- Advanced numerical methods for solving ODE/SDEs
- Distillation

Controllable generation

- Classifier guidance
- Classifier-free guidance