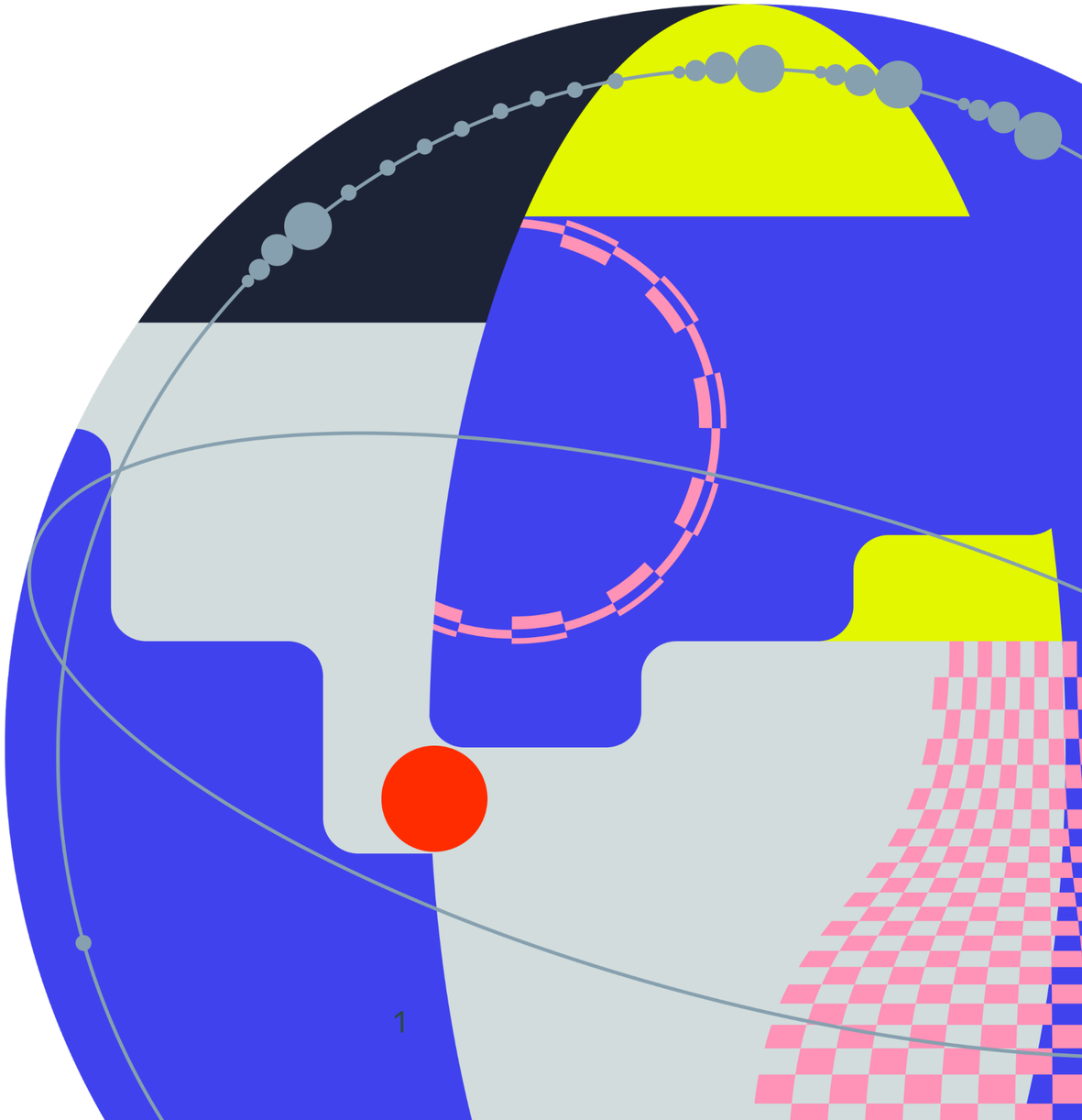
Introduction to Diffusion Models

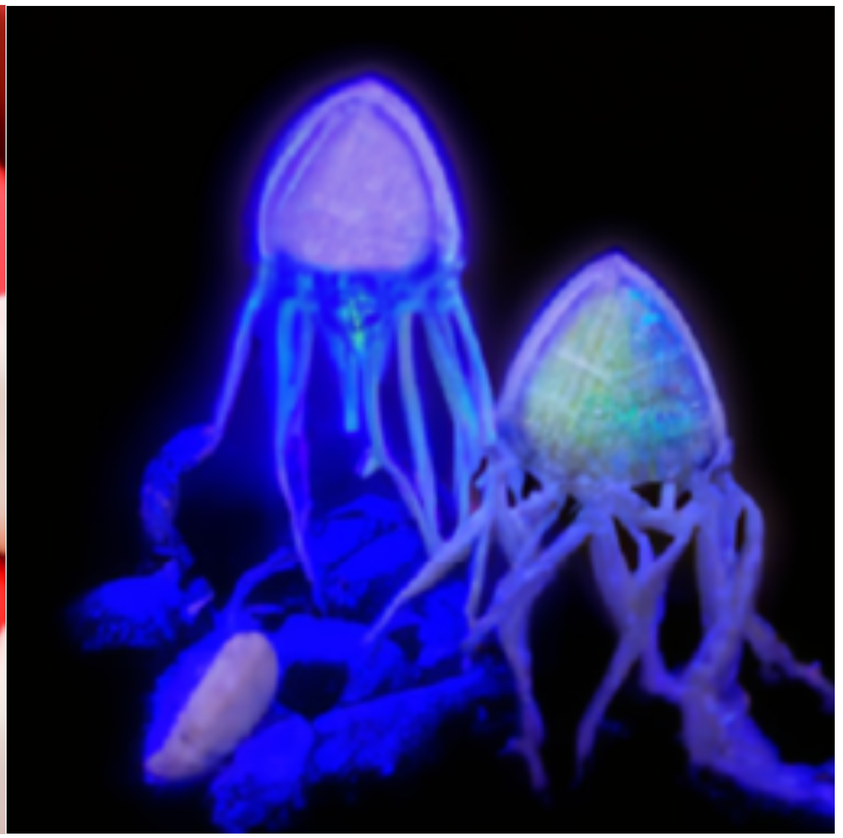


Dmitry Baranchuk
Generative AI Researcher at Yandex Research

Text-to-image 2021

humanoid android, covered in white porcelain skin, blue eyes, white wispy ghost wearing ornate armour

happy marshmallows, in style of adventure time, intricate detail, concept art

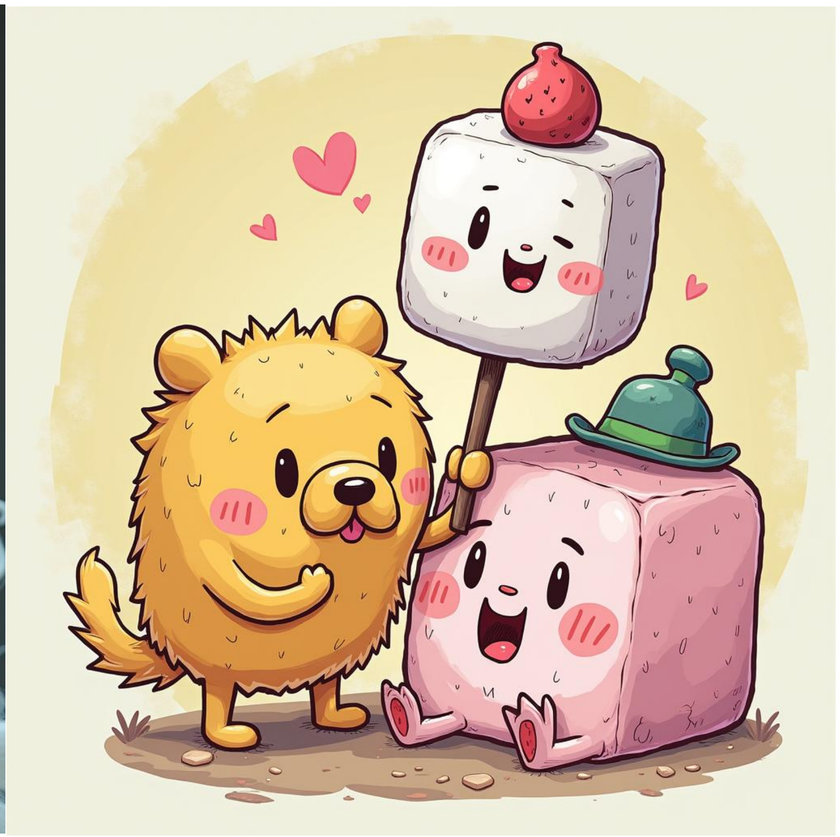


sci-fi cosmic diarama of a quasar and jellyfish in a resin cube, volumetric lighting, high resolution, hdr, sharpen, Photorealism

GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. 2021

Text-to-image 2024

humanoid android, covered in white porcelain skin, blue eyes, white wispy ghost wearing ornate armour



happy marshmallows, in style of adventure time, intricate detail, concept art

sci-fi cosmic diarama of a quasar and jellyfish in a resin cube, volumetric lighting, high resolution, hdr, sharpen, Photorealism

Black Forest Labs. Flux.1. 2024

Text-to-video

A fluffy koala bear surfs.

It has a grey and white coat and a round nose.

The surfboard is yellow. The koala bear is holding onto the surfboard with its paws.

The koala bear's facial expression is focused.

The sun is shining.

Movie Gen: A Cast of Media Foundation Models. 2024

Text-to-3D

Gao et al. CAT3D: Create Anything in 3D with Multi-View Diffusion Models, 2024

Course motivation and goals

01 Get comprehensive understanding of diffusion generative models

02 Go beyond the basic formulations and discuss the recent advances in the field

03 Discuss challenges in developing production-grade image generative models

Course outline

01	***Lecture 1***	*Introduction to diffusion models*
	Seminar 1	*Basic diffusion implementation*
02	***Lecture 2***	*DPM formulation via SDE and ODE*
	Seminar 2	*Implementing an efficient sampler*
03	***Lecture 3***	*Diffusion architectures. Training and sampling techniques. Text-to-image formulation*
	Seminar 3	*Text-to-image generation.*
04	***Lecture 4***	*Diffusion distillation. ODE-based methods*
	Lecture 5	*ODE-free diffusion distillation*
	Seminar 4	*Implementing text-to-image consistency models*
05	***Lecture 6***	*RL tuning for diffusion models*
	Lecture 7	*YandexART - a production-grade diffusion model*

Lecture 1 outline

01 Generative modeling background

02 Denoising Diffusion Probabilistic Models

03 Denoising Score Matching

Useful materials

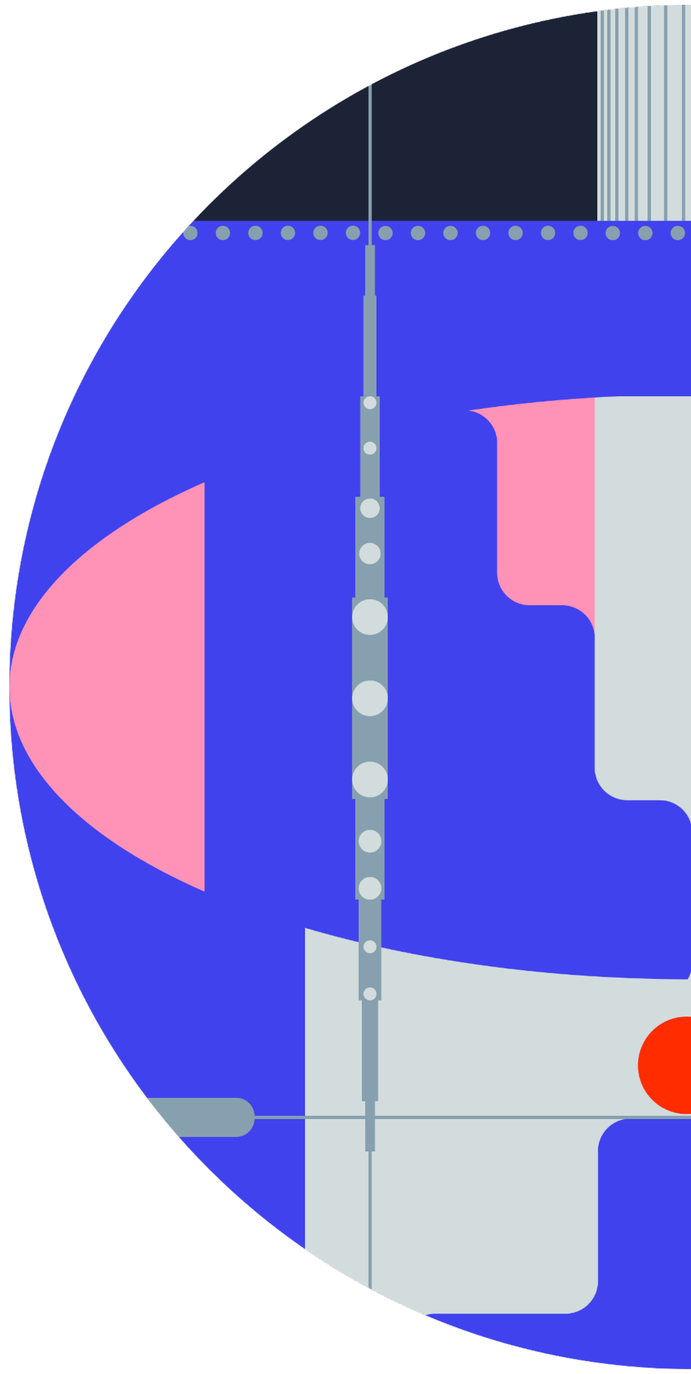
01 Stanford CS236 Deep Generative Models on YouTube

02 Deep Generative Models at YSDA by Roma Isachenko

03 Various valuable blog posts and notes

https://deepgenerativemodels.github.io/notes
https://github.com/r-isachenko/2024-DGM-AIMasters-course
https://lilianweng.github.io/posts/2021-07-11-diffusion-models

Generative modeling background



Generative modeling

Given

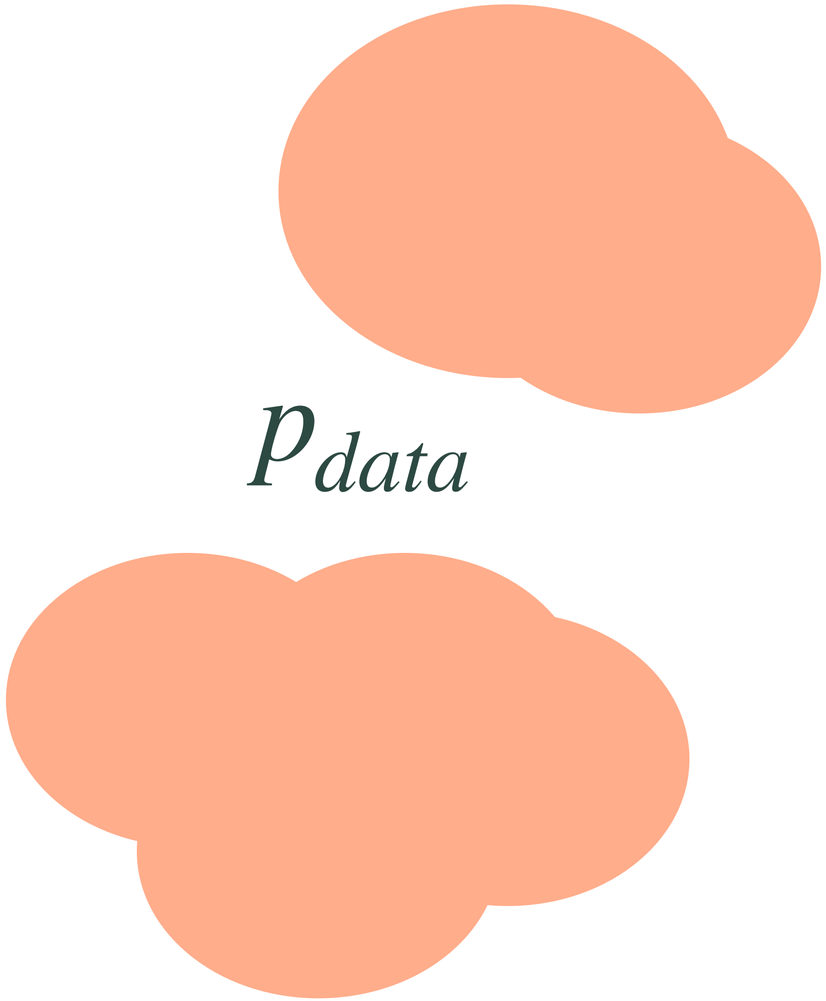
$D = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ from p_{data}

Goals

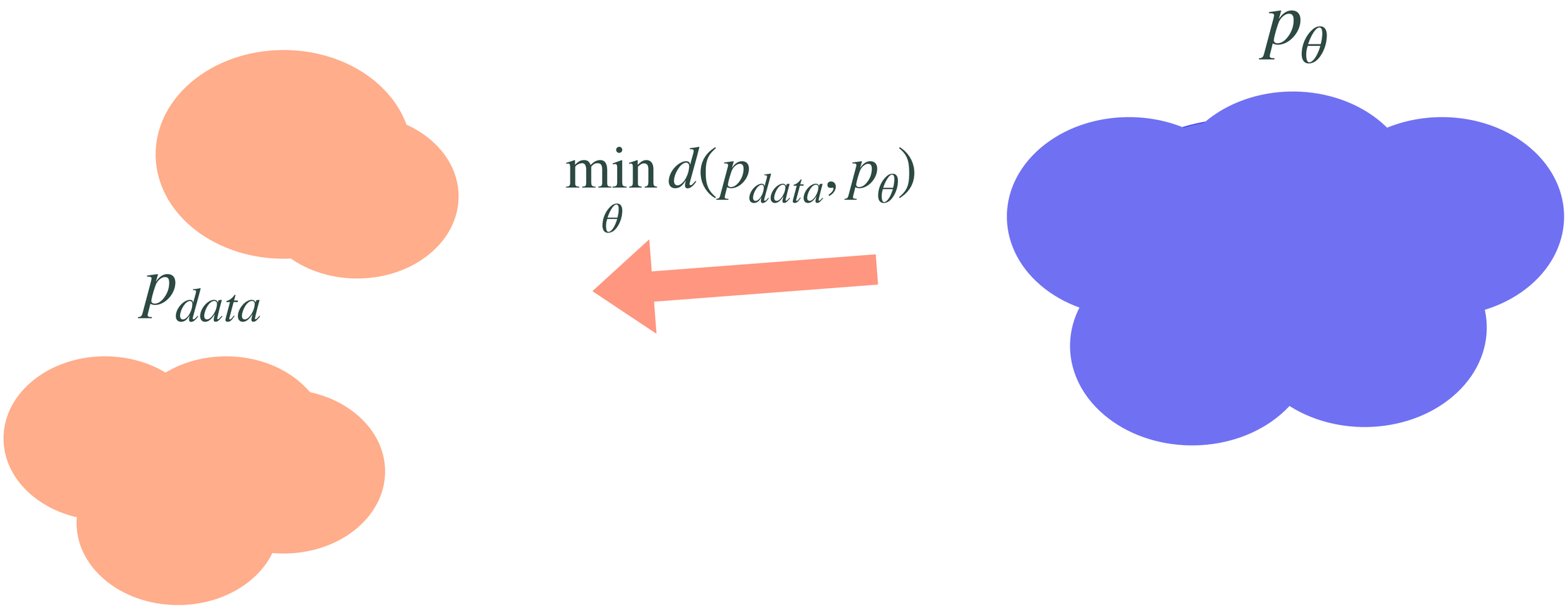
1. Produce new samples from p_{data} Top priority

2. Obtain meaningful data representations

3. Estimate density for data points



Generative modeling

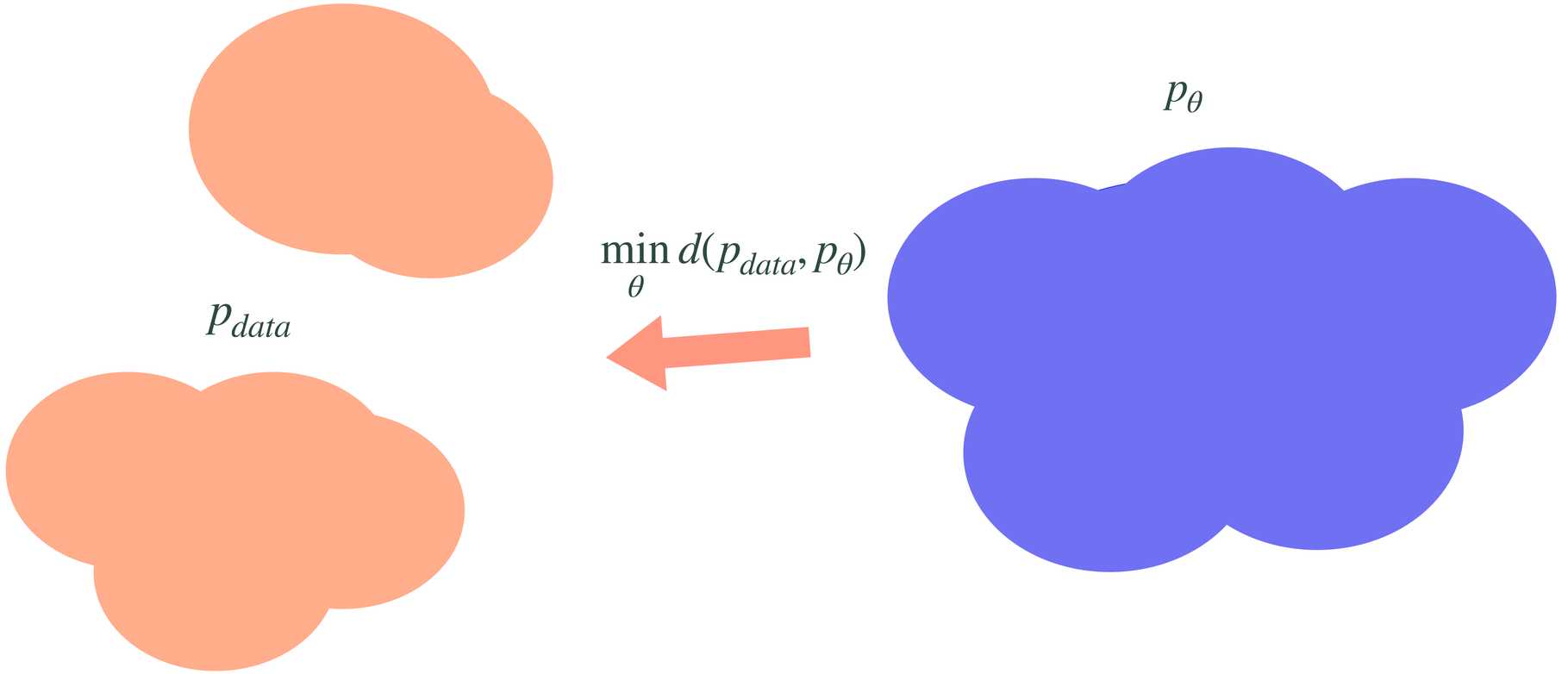


Generative modeling

How to represent p_θ?

What is $d(\cdot)$?

How to minimize $d(\cdot)$?



Useful math tricks

Marginalization

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p(\mathbf{z}) p(\mathbf{x} \mid \mathbf{z}) d\mathbf{z}$$

Monte Carlo estimation

$$\mathop{\mathbb{E}}_{p(\mathbf{x})} f(\mathbf{x}) = \int p(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \approx \frac{1}{K} \sum_{k=1}^{K} f(\mathbf{x}_k), \; \mathbf{x}_k \sim p(\mathbf{x})$$

Jensen's inequality

$$f(\mathop{\mathbb{E}}_{p(\mathbf{x})} [\mathbf{x}]) \geq \mathop{\mathbb{E}}_{p(\mathbf{x})} \left[f(\mathbf{x}) \right], \text{ if } f(\,\cdot\,) \text{ is concave}$$

Chain rule (probability)

$$p(\mathbf{x}_0, \ldots, \mathbf{x}_d) = \prod_{i=1}^{d} p(\mathbf{x}_i \mid \mathbf{x}_{<i})$$

Markov chain

$$p(\mathbf{x}_0, \ldots, \mathbf{x}_d) = \prod_{i=1}^{d} p(\mathbf{x}_i \mid \mathbf{x}_{i-1})$$

Log-derivative trick

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

How to represent probability distribution?

$$p_\theta(\mathbf{x}) - ?$$

How to represent probability distribution?

$$p_\theta(\mathbf{x}) - ?$$

Autoregressive models

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{d} p_\theta(\mathbf{x}_i \,|\, \mathbf{x}_{<i})$$

Variational autoencoders

$$p_\theta(\mathbf{x}) = \int p(\mathbf{z}) p_\theta(\mathbf{x} \,|\, \mathbf{z}) d\mathbf{z}$$

Normalizing flows

$$p_\theta(\mathbf{x}) = p(\mathbf{z}) \,|\, det(J_{f_\theta}(\mathbf{x})) \,|\,, \mathbf{z} = f_\theta(\mathbf{x})$$

Generative adversarial networks (GANs)

$$\mathbf{z} \sim p(\mathbf{z})$$
$$\mathbf{x} = g_\theta(\mathbf{z})$$

Similarity measures on distributions

1. Kullback-Leibrer divergence: $D_{KL}(P\|Q) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})}$

2. Jensen-Shannon divergence: $JSD(P\|Q) = \frac{1}{2} D_{KL}(P\|\frac{P+Q}{2}) + \frac{1}{2} D_{KL}(Q\|\frac{P+Q}{2})$

3. Fisher divergence: $D_F(P\|Q) = \int p(\mathbf{x}) \|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \nabla_x \log q(\mathbf{x})\|_2^2$

4. Wasserstein distance: $W_p(P,Q) = (\inf_{J \in J(P,Q)} \int \|\mathbf{x} - \mathbf{y}\|^p dJ(X,Y))^{\frac{1}{p}}$

5. Maximum mean discrepancy: $MMD(\mathbb{F}, X, Y) := \sup_{f \in \mathbb{F}} (\frac{1}{m} \sum_{i=1}^{m} f(\mathbf{x}_i) - \frac{1}{n} \sum_{j=1}^{n} f(\mathbf{y}_j))$

Similarity measures on distributions

1. Kullback-Leibrer divergence: $D_{KL}(P\|Q) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})}$ Denoising diffusion probabilistic models

2. Jensen-Shannon divergence: $JSD(P\|Q) = \frac{1}{2} D_{KL}(P\|\frac{P+Q}{2}) + \frac{1}{2} D_{KL}(Q\|\frac{P+Q}{2})$

3. Fisher divergence: $D_F(P\|Q) = \int p(\mathbf{x}) \|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \nabla_{\mathbf{x}} \log q(\mathbf{x})\|_2^2$ Denoising score matching

4. Wasserstein distance: $W_p(P,Q) = (\inf_{J \in J(P,Q)} \int \|\mathbf{x} - \mathbf{y}\|^p dJ(X,Y))^{\frac{1}{p}}$

5. Maximum mean discrepancy: $MMD(\mathbb{F}, X, Y) := \sup_{f \in \mathbb{F}} (\frac{1}{m} \sum_{i=1}^{m} f(\mathbf{x}_i) - \frac{1}{n} \sum_{j=1}^{n} f(\mathbf{y}_j))$

KL divergence

Properties

$$D_{KL}(p_{data} \| p_\theta) \geq 0$$

$$D_{KL}(p_{data} \| p_\theta) = 0 \text{ if and only if } p_{data} = p_\theta$$

$$D_{KL}(p_{data} \| p_\theta) \neq D_{KL}(p_\theta \| p_{data})$$

Forward KL *Reverse KL*

Forward KL

$$D_{KL}(p_{data}\|p_\theta) = \int p_{data}(\mathbf{x}) \log \frac{p_{data}(\mathbf{x})}{p_\theta(\mathbf{x})} = \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \log \frac{p_{data}(\mathbf{x})}{p_\theta(\mathbf{x})} = \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \log p_{data}(\mathbf{x}) - \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \log p_\theta(\mathbf{x})$$

Negative entropy $-H(p_{data})$

Cross entropy $H(p_{data}, p_\theta)$

Forward KL

$$D_{KL}(p_{data}\|p_\theta) = \int p_{data}(\mathbf{x}) \log \frac{p_{data}(\mathbf{x})}{p_\theta(\mathbf{x})} = \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \log \frac{p_{data}(\mathbf{x})}{p_\theta(\mathbf{x})} = \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \log p_{data}(\mathbf{x}) - \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \log p_\theta(\mathbf{x})$$

Negative entropy $-H(p_{data})$

Cross entropy $H(p_{data}, p_\theta)$

Const

$$\arg\min_\theta D_{KL}(p_{data}\|p_\theta) = \arg\min_\theta \left[H(p_{data}, p_\theta) - H(p_{data})\right] = \arg\min_\theta \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} - \log p_\theta(\mathbf{x}) = \arg\max_\theta \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \log p_\theta(\mathbf{x})$$

Maximum likelihood estimation (MLE)

Forward vs Reverse KL

Forward KL

Monte Carlo estimation

$$\arg\min_\theta D_{KL}(p_{data}\|p_\theta) = \arg\min_\theta \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} -\log p_\theta(\mathbf{x}) = \arg\max_\theta \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \log p_\theta(\mathbf{x}) \approx \arg\max_\theta \sum_{\mathbf{x}_i \in D} \left[\log p_\theta(\mathbf{x}_i)\right]$$

D is available

Forward vs Reverse KL

Forward KL

Monte Carlo estimation

$$\arg\min_\theta D_{KL}(p_{data}\|p_\theta) = \arg\min_\theta \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} -\log p_\theta(\mathbf{x}) = \arg\max_\theta \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \log p_\theta(\mathbf{x}) \approx \arg\max_\theta \sum_{\mathbf{x}_i \in D} \left[\log p_\theta(\mathbf{x}_i)\right]$$

D is available

Reverse KL

$$\arg\min_\theta D_{KL}(p_\theta\|p_{data}) = \arg\min_\theta \left[\mathop{\mathbb{E}}_{p_\theta(\mathbf{x})} \log p_\theta(\mathbf{x}) - \mathop{\mathbb{E}}_{p_\theta(\mathbf{x})} \log p_{data}(\mathbf{x})\right] \approx \arg\min_\theta \left[\sum_{\mathbf{x}\sim p_\theta(\mathbf{x})} \left[\log p_\theta(\mathbf{x}) - \log p_{data}(\mathbf{x})\right]\right]$$

Sampling from $p_\theta(\mathbf{x})$

Usually unavailable

Latent variable models

Monte Carlo estimation

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}) d\mathbf{z} = \mathop{\mathbb{E}}_{p(\mathbf{z})} p_\theta(\mathbf{x} \mid \mathbf{z}) \approx \frac{1}{K} \sum_{k=1}^{K} p_\theta(\mathbf{x} \mid \mathbf{z}^{(k)}), \mathbf{z}^{(k)} \sim p(\mathbf{z})$$

Prior distribution, e.g., $\mathcal{N}(0, I)$

Can be parameterized with a NN: $f_\theta(\mathbf{z}) \to \mathbf{x}$

Latent variable models

Monte Carlo estimation

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}) d\mathbf{z} = \mathop{\mathbb{E}}_{p(\mathbf{z})} p_\theta(\mathbf{x} \mid \mathbf{z}) \approx \frac{1}{K} \sum_{k=1}^{K} p_\theta(\mathbf{x} \mid \mathbf{z}^{(k)}), \mathbf{z}^{(k)} \sim p(\mathbf{z})$$

Prior distribution, e.g., $\mathcal{N}(0, I)$

Can be parameterized with a NN: $f_\theta(\mathbf{z}) \rightarrow \mathbf{x}$

$$\arg\max_\theta \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \log p_\theta(\mathbf{x}) \approx \arg\max_\theta \sum_{\mathbf{x}_i \in D} \log p_\theta(\mathbf{x}_i) \approx \arg\max_\theta \sum_{\mathbf{x}_i \in D} \log \left[\frac{1}{K} \sum_{k=1}^{K} p_\theta(\mathbf{x}_i \mid \mathbf{z}^{(k)}) \right]$$

High variance for high-dimensional $z \rightarrow$ does not scale :(

Evidence lower bound

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \log \int \frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{q_\phi(\mathbf{z} \mid \mathbf{x})} p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \geq \text{Jensen's inequality}$$

$$\geq \int q_\phi(\mathbf{z} \mid \mathbf{x}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} d\mathbf{z} = \mathop{\mathbb{E}}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} =: ELBO(\mathbf{x}, \theta, \phi)$$

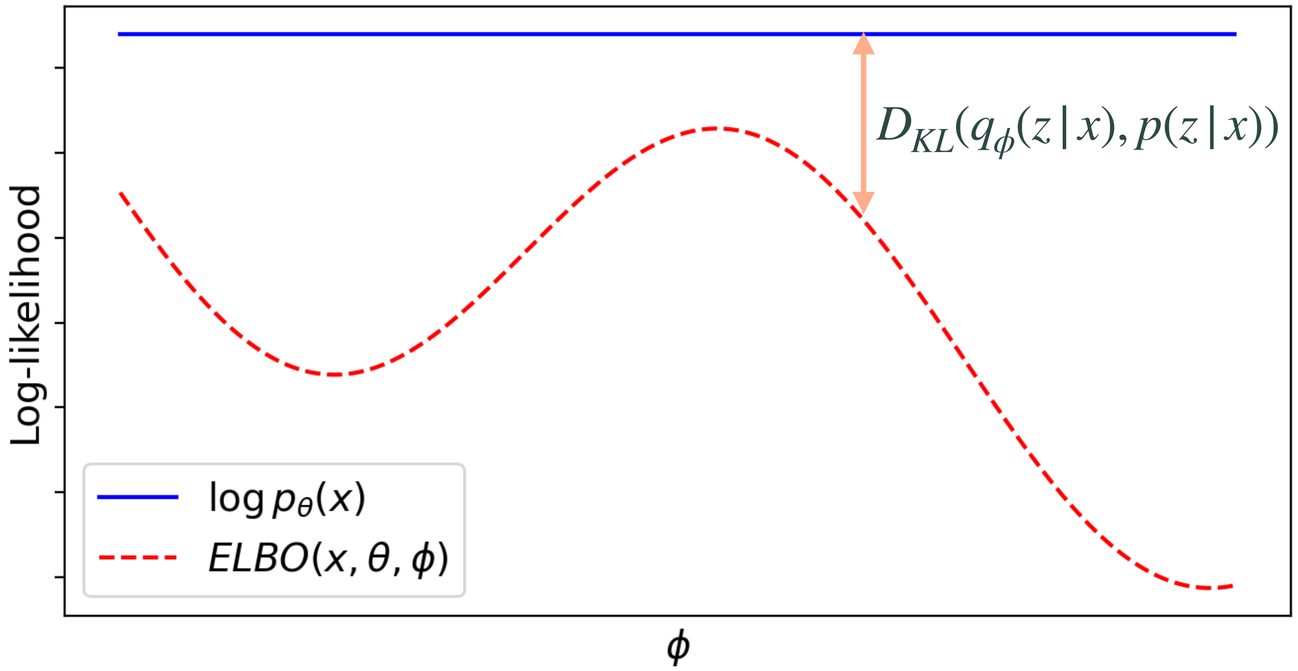
$q_\phi(\mathbf{z} \mid \mathbf{x})$ — posterior distribution, parametrized with a neural network $g_\phi(\mathbf{x}) \to \mathbf{z}$

How close is $ELBO(\mathbf{x}, \theta, \phi)$ to $\log p_\theta(\mathbf{x})$?

Evidence lower bound

$$\log p_\theta(\mathbf{x}) = \underbrace{\int q_\phi(\mathbf{z}\,|\,\mathbf{x}) \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x})} d\mathbf{z}}_{ELBO(\mathbf{x},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(\mathbf{z}\,|\,\mathbf{x}), p(\mathbf{z}\,|\,\mathbf{x}))}_{\text{Gap between } \log p_\theta(\mathbf{x}) \text{ and } ELBO(\mathbf{x},\theta,\phi)}$$

Unknown true posterior



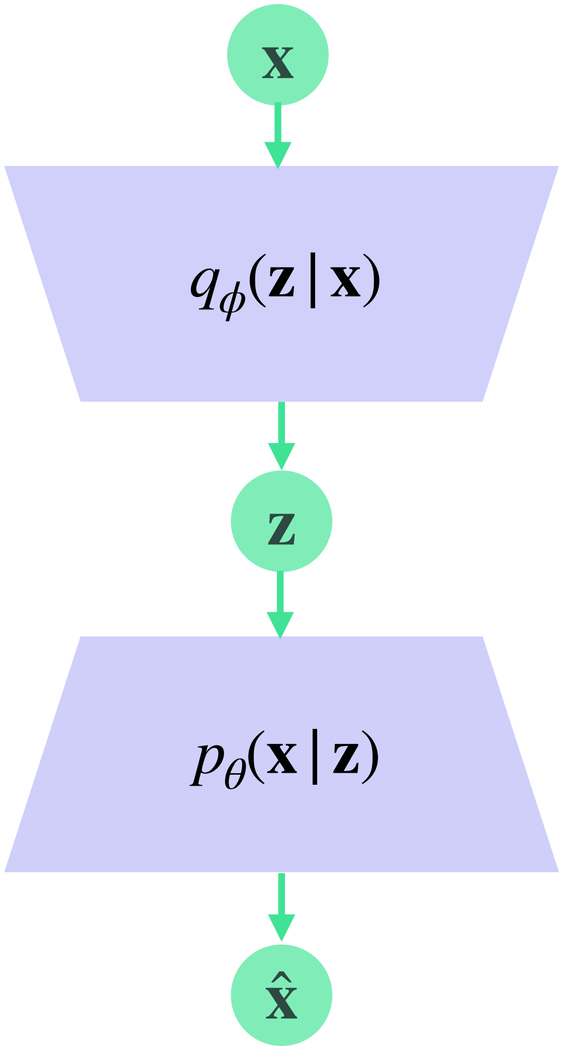
Variational autoencoders

Step 1: Update ϕ to approximate $\log p_\theta(\mathbf{x})$ better

$$\phi \leftarrow \phi + \nabla_\phi ELBO(\mathbf{x}, \theta, \phi) = \phi + \nabla_\phi \sum_{\mathbf{x} \in D} \mathop{\mathbb{E}}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} \right]$$

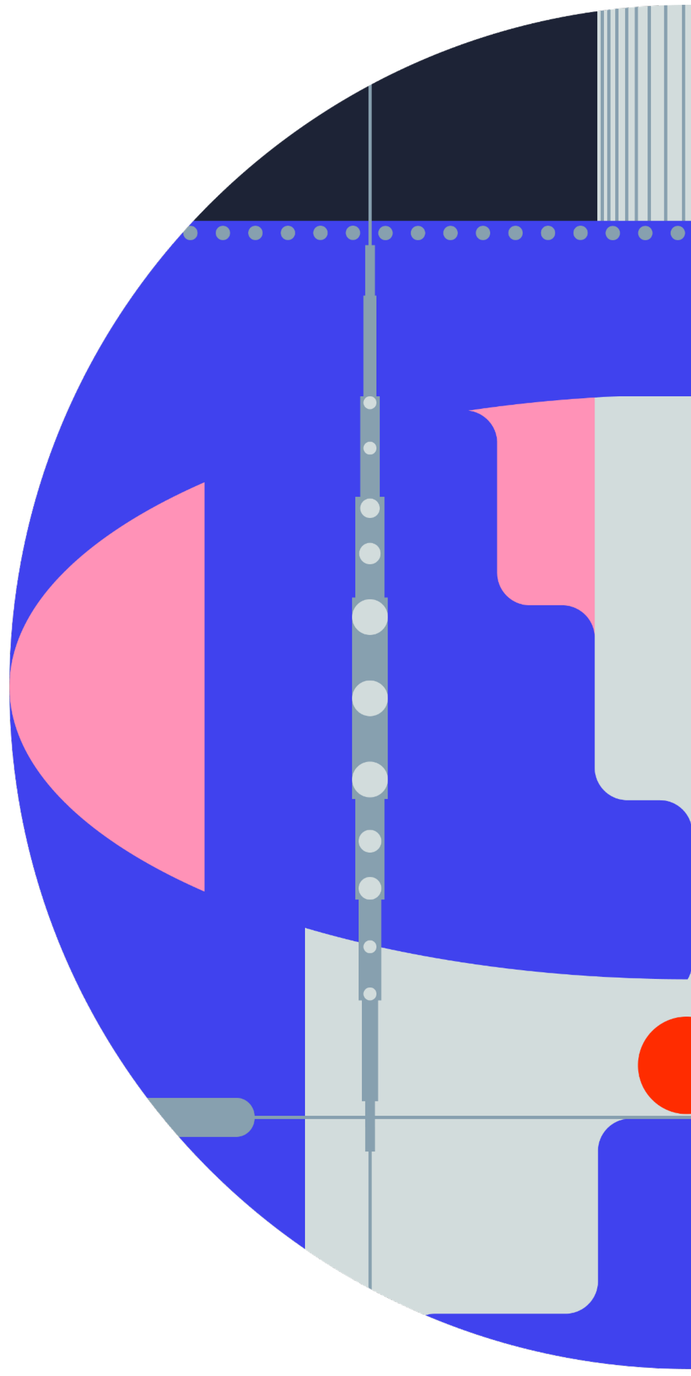
Step 2: Update θ to maximize $\log p_\theta(\mathbf{x})$

$$\theta \leftarrow \theta + \nabla_\theta ELBO(\mathbf{x}, \theta, \phi) = \theta + \nabla_\theta \sum_{\mathbf{x} \in D} \mathop{\mathbb{E}}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} \right]$$



D. Kingma & M. Welling, Auto-Encoding Variational Bayes. 2013

Denoising diffusion probabilistic models



Denoising diffusion probabilistic models

Latent variable model perspective

$$p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \qquad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$ Markov chain

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

Denoising diffusion probabilistic models

Latent variable model perspective

$$p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \qquad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$ Markov chain

$$\log p_\theta(\mathbf{x}_0) = \log \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} = \log \int \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \geq$$ Jensen's inequality

not parameterized

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

Denoising diffusion probabilistic models

Latent variable model perspective

$$p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \qquad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$ Markov chain

$$\log p_\theta(\mathbf{x}_0) = \log \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} = \log \int \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \geq$$ Jensen's inequality

$$\geq \int q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}) d\mathbf{x}_{1:T} = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} =: ELBO(\mathbf{x}_0, \theta)$$

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

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM vs VAE

DDPM

$$ELBO_{DDPM} = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}\,|\,\mathbf{x}_0)}$$

$T = 1$

$$\mathop{\mathbb{E}}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log \frac{p_\theta(\mathbf{x}_0, \mathbf{x}_1)}{q(\mathbf{x}_1\,|\,\mathbf{x}_0)}$$

$T \gg 1; \quad q(\mathbf{x}_1\,|\,\mathbf{x}_0); \quad d_{x_1} = d_{x_0}$

VAE

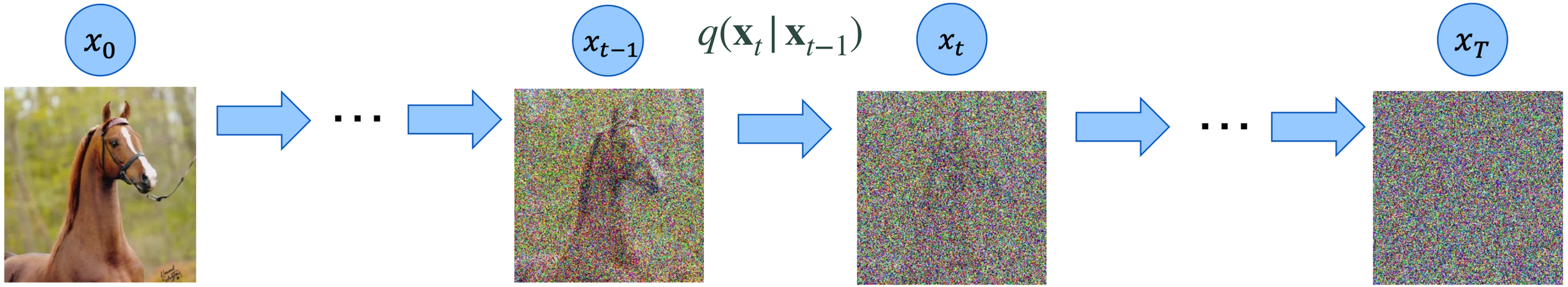
$$ELBO_{VAE} = \mathop{\mathbb{E}}_{q_\phi(\mathbf{z}|\mathbf{x}_0)} \log \frac{p_\theta(\mathbf{x}_0, \mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x}_0)}$$

$T = 1; \quad q_\phi(\mathbf{z}\,|\,\mathbf{x}_0); \quad d_z < d_{x_0}$

DDPM Overview

Forward process

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



J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM Overview

Forward process

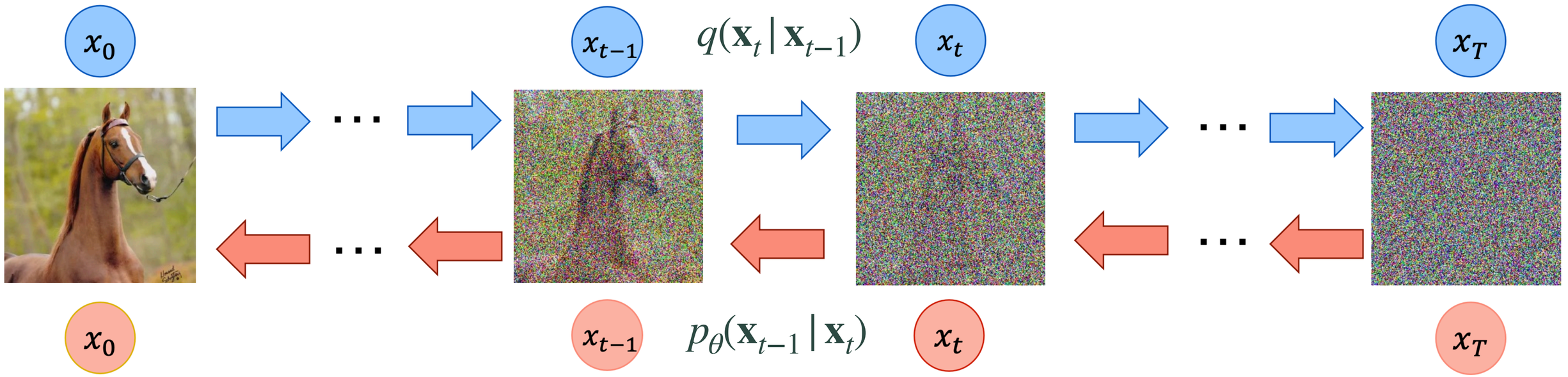
$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

Revert it and approximate

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

Reverse process

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



J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

Forward process

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

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

$$\mathbf{x}_t = \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0,I)$$

$\{\beta_1, \ldots, \beta_T\}$ — variance schedule, $\beta_t \in (0,1)$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

Forward process

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

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

$$\mathbf{x}_t = \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0,I)$$

$\{\beta_1, \ldots, \beta_T\}$ — variance schedule, $\beta_t \in (0,1)$

Useful property

Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

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

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0,I)$$

Derive x_t directly from x_0

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

Reverse process

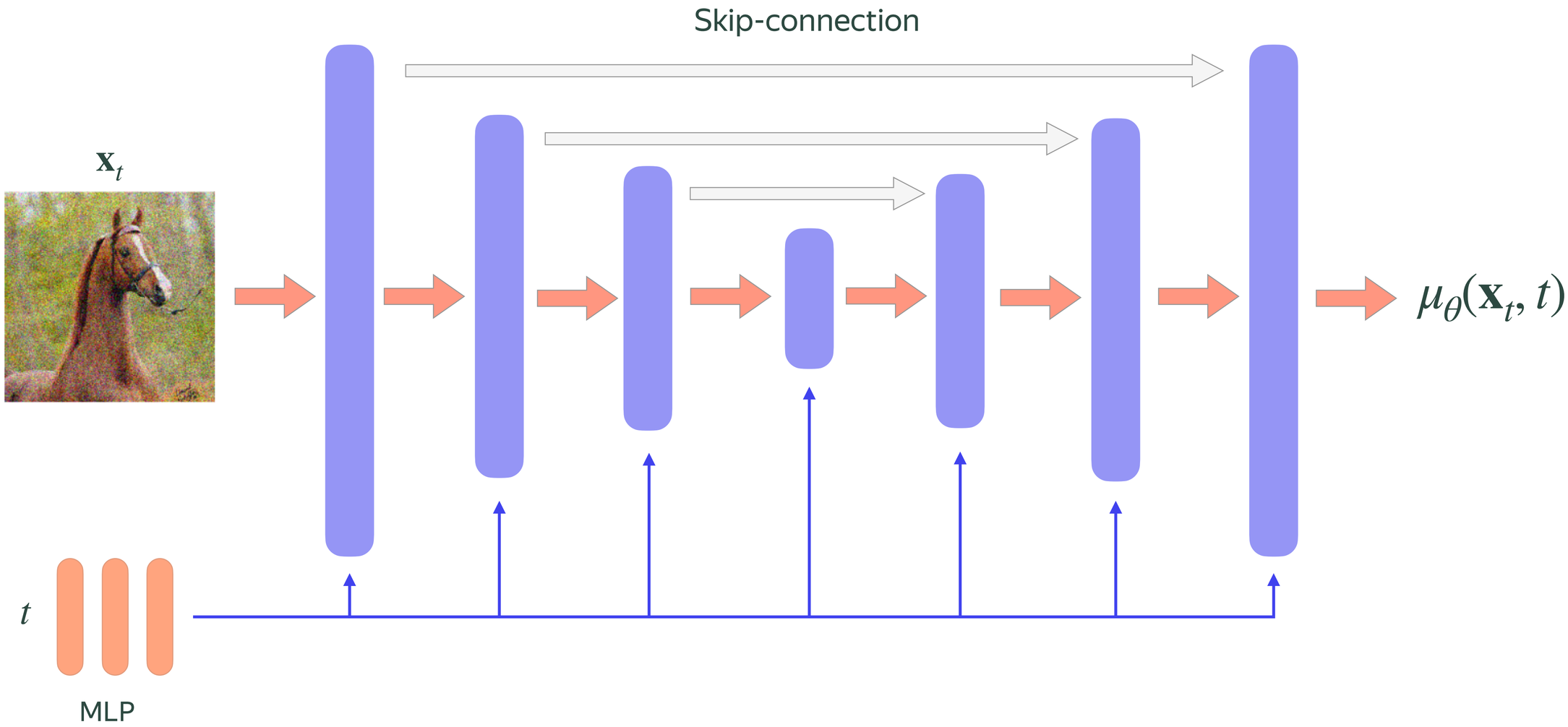
$$\mathbf{x}_T \sim p(\mathbf{x}_T) = \mathcal{N}(0, I)$$

Usually non-learnable, e.g., $\beta_t I$

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

$$\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \sqrt{\beta_t} \cdot \epsilon_t, \text{ where } \epsilon_t \sim \mathcal{N}(0, I)$$

Model parametrization



DDPM training objective

$$ELBO(\mathbf{x}_0, \theta) = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}$$

Note: we derive the objective for a single $\mathbf{x}_0 \sim p_{data}(\mathbf{x}_0)$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM training objective

$$ELBO(\mathbf{x}_0, \theta) = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_T) + \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} \right] =$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM training objective

$$ELBO(\mathbf{x}_0, \theta) = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_T) + \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} \right] =$$

$$= \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_T) + \sum_{t > 1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} + \log \frac{p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)}{q(\mathbf{x}_1 \mid \mathbf{x}_0)} \right] = \dots$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM training objective

$$ELBO(\mathbf{x}_0, \theta) = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_T) + \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} \right] =$$

$$= \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_T) + \sum_{t > 1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} + \log \frac{p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)}{q(\mathbf{x}_1 \mid \mathbf{x}_0)} \right] = \dots$$

Hocus-pocus: $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0) = \dfrac{q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) q(\mathbf{x}_t \mid \mathbf{x}_0)}{q(\mathbf{x}_{t-1} \mid \mathbf{x}_0)}$

$q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ is Markovian

Bayes' theorem

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM training objective

$$\ldots = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_T) + \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t)}{q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t, \mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_0)}{q(\mathbf{x}_t \,|\, \mathbf{x}_0)} + \log \frac{p_\theta(\mathbf{x}_0 \,|\, \mathbf{x}_1)}{q(\mathbf{x}_1 \,|\, \mathbf{x}_0)} \right] =$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM training objective

$$\ldots = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_T) + \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1} \mid \mathbf{x}_0)}{q(\mathbf{x}_t \mid \mathbf{x}_0)} + \log \frac{p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)}{q(\mathbf{x}_1 \mid \mathbf{x}_0)} \right] =$$

$$= \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T \mid \mathbf{x}_0)} + \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)} + \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1) \right] =$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM training objective

$$\ldots = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_T) + \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}\,|\,\mathbf{x}_t)}{q(\mathbf{x}_{t-1}\,|\,\mathbf{x}_t, \mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}\,|\,\mathbf{x}_0)}{q(\mathbf{x}_t\,|\,\mathbf{x}_0)} + \log \frac{p_\theta(\mathbf{x}_0\,|\,\mathbf{x}_1)}{q(\mathbf{x}_1\,|\,\mathbf{x}_0)} \right] =$$

$$= \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T\,|\,\mathbf{x}_0)} + \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}\,|\,\mathbf{x}_t)}{q(\mathbf{x}_{t-1}\,|\,\mathbf{x}_t, \mathbf{x}_0)} + \log p_\theta(\mathbf{x}_0\,|\,\mathbf{x}_1) \right] = \quad \text{Expand a bracket}$$

$$= \mathop{\mathbb{E}}_{q(\mathbf{x}_T|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T\,|\,\mathbf{x}_0)} + \sum_{t>1} \mathop{\mathbb{E}}_{q(\mathbf{x}_t,\mathbf{x}_{t-1}|\mathbf{x}_0)} \log \frac{p_\theta(\mathbf{x}_{t-1}\,|\,\mathbf{x}_t)}{q(\mathbf{x}_{t-1}\,|\,\mathbf{x}_t, \mathbf{x}_0)} + \mathop{\mathbb{E}}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0\,|\,\mathbf{x}_1) = \ldots$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM training objective

$$= \underbrace{\mathop{\mathbb{E}}_{q(\mathbf{x}_T|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T \mid \mathbf{x}_0)}}_{-D_{KL}(q(\mathbf{x}_T \mid \mathbf{x}_0)\|p(\mathbf{x}_T))} + \sum_{t>1} \underbrace{\mathop{\mathbb{E}}_{q(\mathbf{x}_t,\mathbf{x}_{t-1}|\mathbf{x}_0)}}_{\substack{\mathbb{E} \\ q(\mathbf{x}_t|\mathbf{x}_0)} \quad \substack{\mathbb{E} \\ q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)} + \mathop{\mathbb{E}}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1) =$$

$$= \underbrace{- D_{KL}(q(\mathbf{x}_T \mid \mathbf{x}_0)\|p(\mathbf{x}_T))}_{-L_T} + \sum_{t>1} \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} \underbrace{- D_{KL}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t))}_{-L_t} + \underbrace{\mathop{\mathbb{E}}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)}_{-L_0} =: -L$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM training objective

$$L_{DDPM} = \underbrace{D_{KL}(q(\mathbf{x}_T \mid \mathbf{x}_0) \| p(\mathbf{x}_T))}_{L_T \text{ constant}} + \sum_{t>1} \underbrace{\mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} D_{KL}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t))}_{L_t} - \underbrace{\mathop{\mathbb{E}}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)}_{L_0}$$

Can be omitted for simplicity

Let's focus on L_t

$$L_t = \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} D_{KL}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t))$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM training objective

$$L_t = \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} D_{KL}(q(\mathbf{x}_{t-1}\,|\,\mathbf{x}_t, \mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1}\,|\,\mathbf{x}_t))$$

$$q(\mathbf{x}_{t-1}\,|\,\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t\,|\,\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}\,|\,\mathbf{x}_0)}{q(\mathbf{x}_t\,|\,\mathbf{x}_0)} = \sqrt{\frac{(1-\bar{\alpha}_t)}{2\pi\beta_t(1-\bar{\alpha}_{t-1})}} \cdot e^{-\frac{(\mathbf{x}_t - \sqrt{1-\beta_t}\mathbf{x}_{t-1})^2}{2\beta_t} - \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{2(1-\bar{\alpha}_{t-1})} + \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{2(1-\bar{\alpha}_t)}}$$

$$\mathcal{N}(\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t I)$$

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \cdot \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \cdot \mathbf{x}_0; \quad \tilde{\beta}_t = \frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM training objective

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t I); \qquad p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \tilde{\beta}_t I)$$

$$L_t = \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} D_{KL}\left(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\right) =$$

$$= \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} D_{KL}\left(\mathcal{N}(\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t I) \| \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \tilde{\beta}_t I)\right) = D_{KL} \text{ between normal distributions}$$

$$= \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\tilde{\beta}_t} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|_2^2\right]$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

Reparameterized training objective

$$L_t = \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\tilde{\beta}_t} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|_2^2 \right]$$

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon \rightarrow \mathbf{x}_0 = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t} \cdot \epsilon}{\sqrt{\bar{\alpha}_t}}$$

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \cdot \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \cdot \mathbf{x}_0 = \boxed{\frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \epsilon \right)}$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

Reparameterized training objective

$$L_t = \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\tilde{\beta}_t} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|_2^2 \right]$$

$$\frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \epsilon \right)$$

Reparameterize: $$\frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \epsilon_\theta(\mathbf{x}_t, t) \right)$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

Reparameterized training objective

$$L_t = \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\tilde{\beta}_t} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|_2^2 \right]$$

$$\frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \epsilon \right)$$

Reparameterize: $$\frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \epsilon_\theta(\mathbf{x}_t, t) \right)$$

$$L_t = \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{\beta_t^2}{2\tilde{\beta}_t \alpha_t (1-\bar{\alpha}_t)} \|\epsilon_\theta(\mathbf{x}_t, t) - \epsilon\|_2^2 \right] = \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}(0,I)} \left[\frac{\beta_t^2}{2\tilde{\beta}_t \alpha_t (1-\bar{\alpha}_t)} \|\epsilon_\theta(\sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon, t) - \epsilon\|_2^2 \right]$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

Final training objective

Simplify

$$L_t = \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}(0,I)} \left[\frac{\beta_t^2}{2\tilde{\beta}_t \alpha_t (1-\bar{\alpha}_t)} \| \epsilon_\theta(\sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon, t) - \epsilon \|_2^2 \right]$$

$$L_{simple}^t = \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}(0,I)} \| \epsilon_\theta(\sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon, t) - \epsilon \|_2^2$$

$$L_{simple} = \mathop{\mathbb{E}}_{\mathbf{x}_0 \sim p_{data}(\mathbf{x})} \mathop{\mathbb{E}}_{t \sim U(2,T)} \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}(0,I)} \| \epsilon_\theta(\sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon, t) - \epsilon \|_2^2$$

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

Alternative objectives

ϵ-prediction $\mathop{\mathbb{E}}_{\mathbf{x}_0 \sim p_{data}(\mathbf{x})} \mathop{\mathbb{E}}_{t \sim U(2,T)} \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}(0,I)} \|\epsilon_\theta(\sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon, t) - \epsilon\|_2^2$

μ-prediction $\mathop{\mathbb{E}}_{\mathbf{x}_0 \sim p_{data}(\mathbf{x})} \mathop{\mathbb{E}}_{t \sim U(2,T)} \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|_2^2$

$\mathbf{x}_0$-prediction $\mathop{\mathbb{E}}_{\mathbf{x}_0 \sim p_{data}(\mathbf{x})} \mathop{\mathbb{E}}_{t \sim U(2,T)} \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}(0,I)} \|\mathbf{f}_\theta(\sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon, t) - \mathbf{x}_0\|_2^2$

ϵ-prediction — a default choice in practice.

Nevertheless, loss weighting and parameterizations have been widely explored in later works.

J. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS2020

DDPM summary

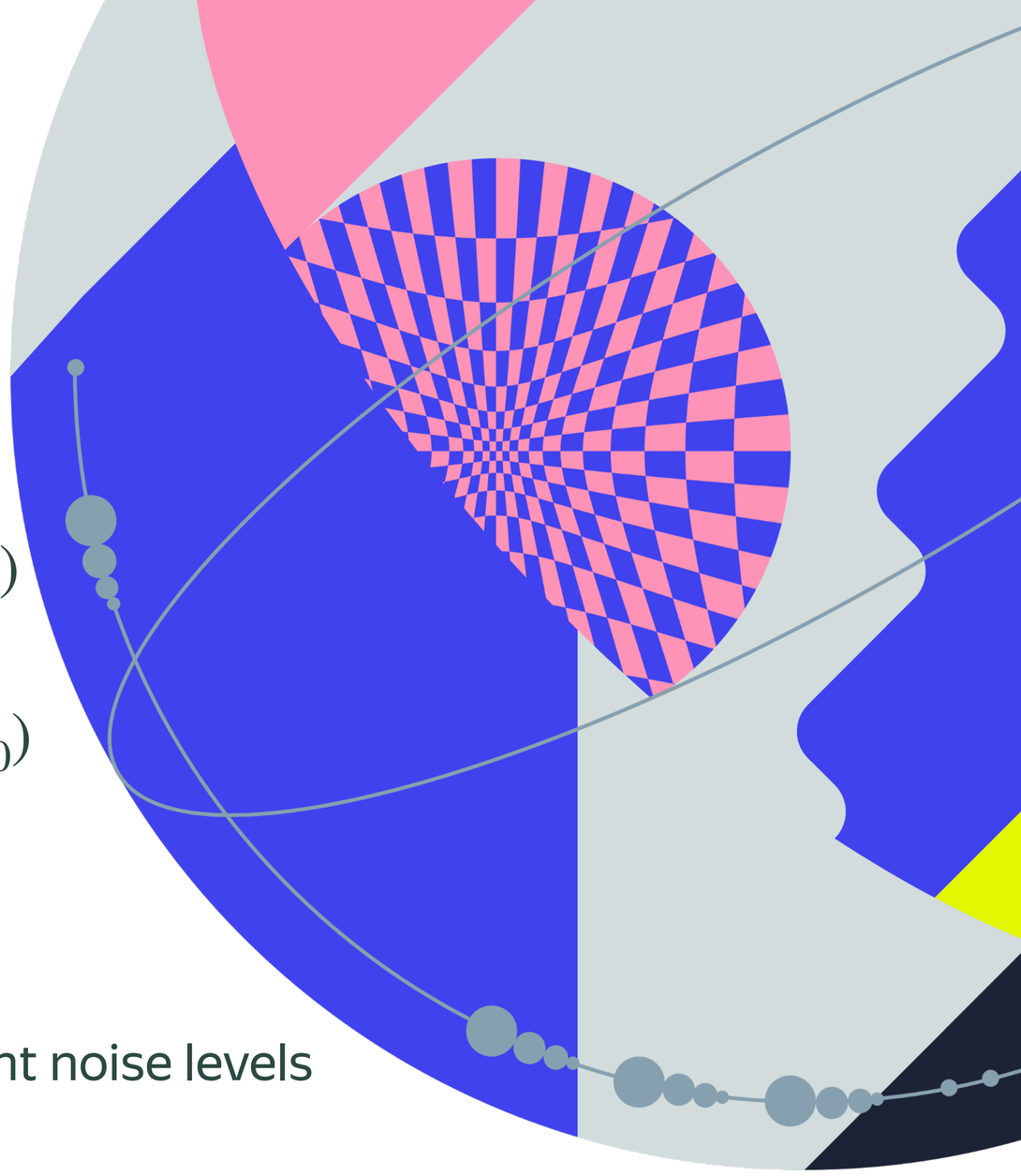
DDPMs are latent variable models, similar to VAE

Forward process — Markov chain $p(\mathbf{x}_0) \rightarrow p(\mathbf{x}_T) = \mathcal{N}(0,I)$

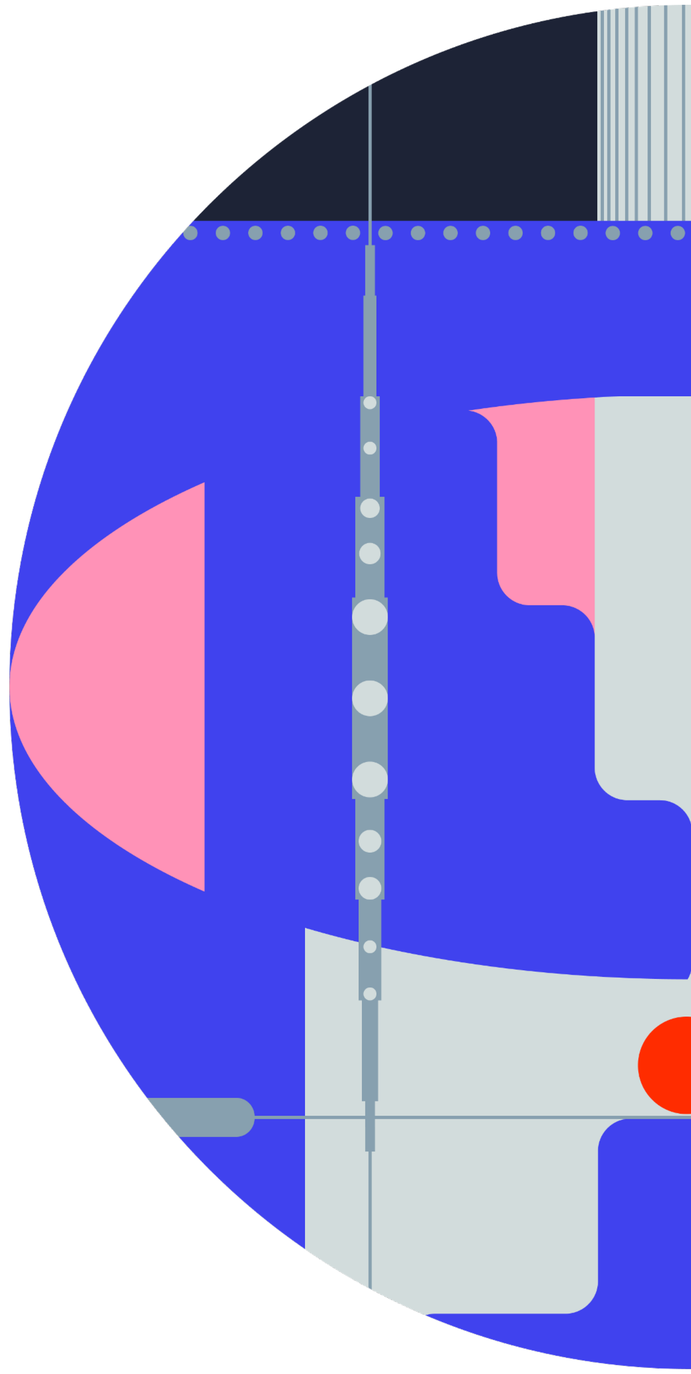
Reverse process reverts the forward process $p(\mathbf{x}_T) \rightarrow p(\mathbf{x}_0)$

Costly iterative sampling with $T \approx 1000$ steps

Loss — MSE between predicted and added noise at different noise levels



Denoising score matching



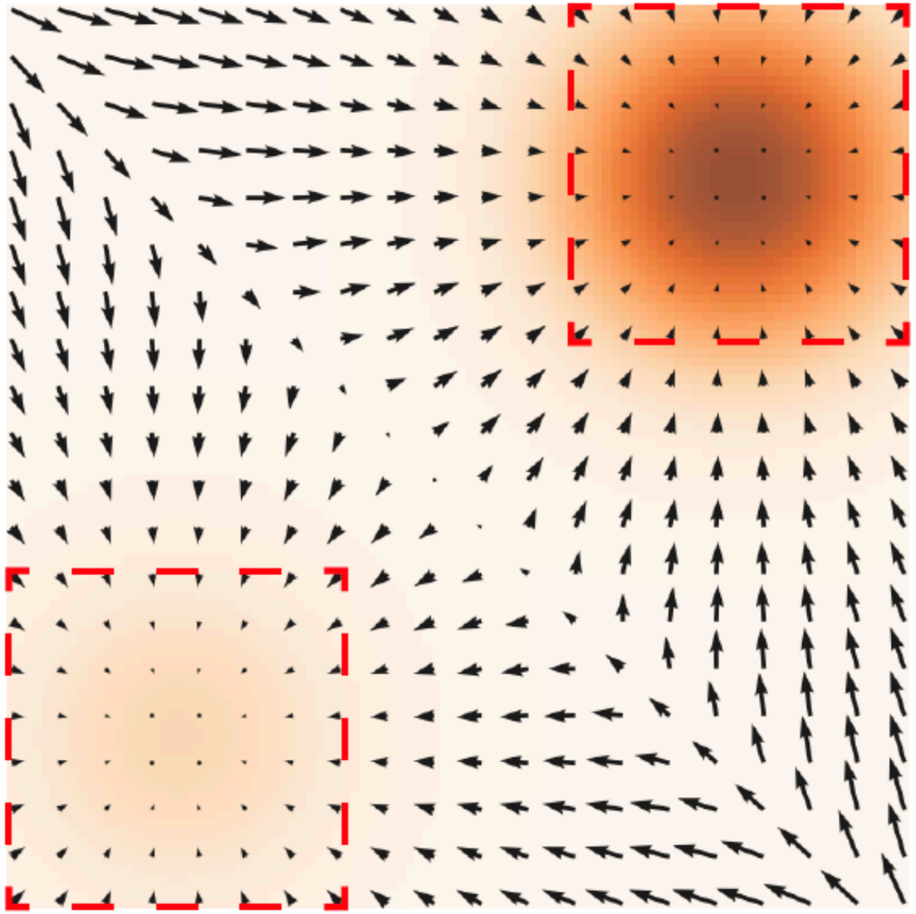
How to represent probability distribution?

$$p_\theta(\mathbf{x}) - ?$$

How to represent probability distribution?

$$p_\theta(\mathbf{x}) - ?$$

Score function: $\nabla_{\mathbf{x}} \log p_{data}(\mathbf{x})$



Y. Song et al., Generative Modeling by Estimating Gradients of the Data Distribution, NeurIPS2019

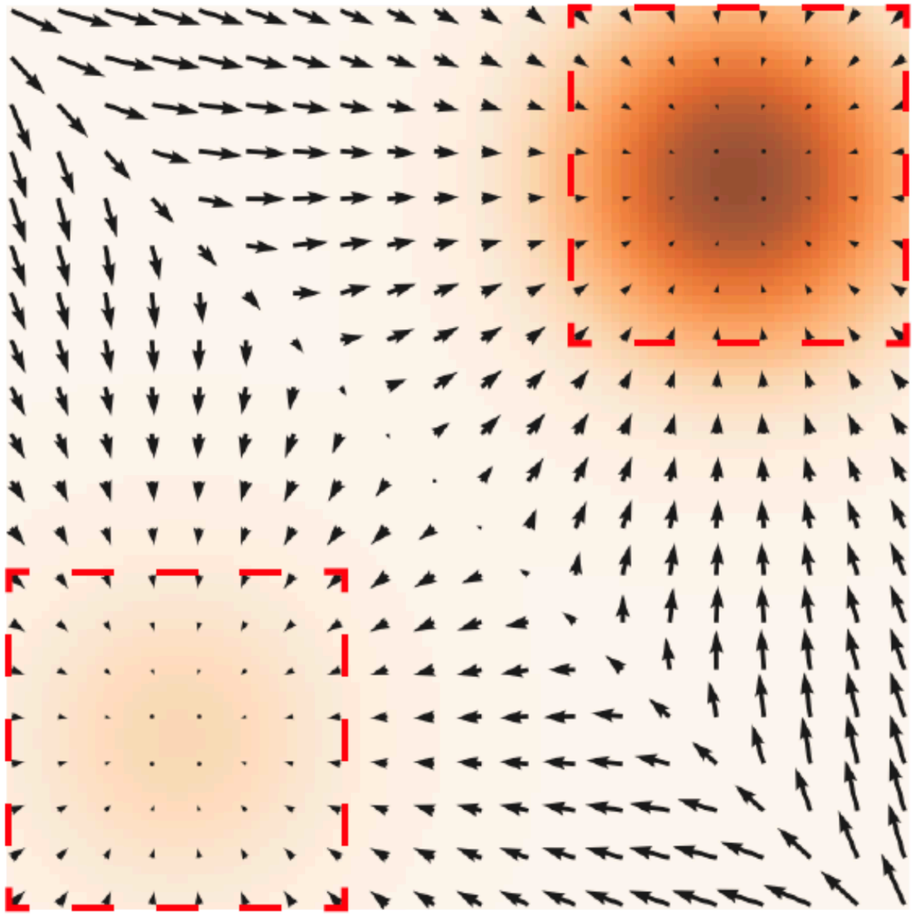
How to represent probability distribution?

$$p_\theta(\mathbf{x}) - ?$$

Score function: $\nabla_{\mathbf{x}} \log p_{data}(\mathbf{x})$

Score-based model: $s_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$

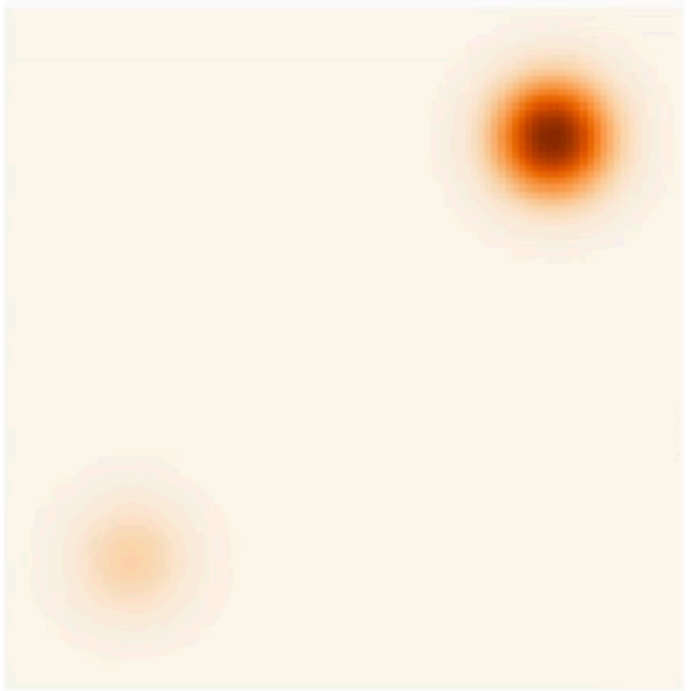
$s_\theta(\mathbf{x})$ — a NN that directly predicts $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$



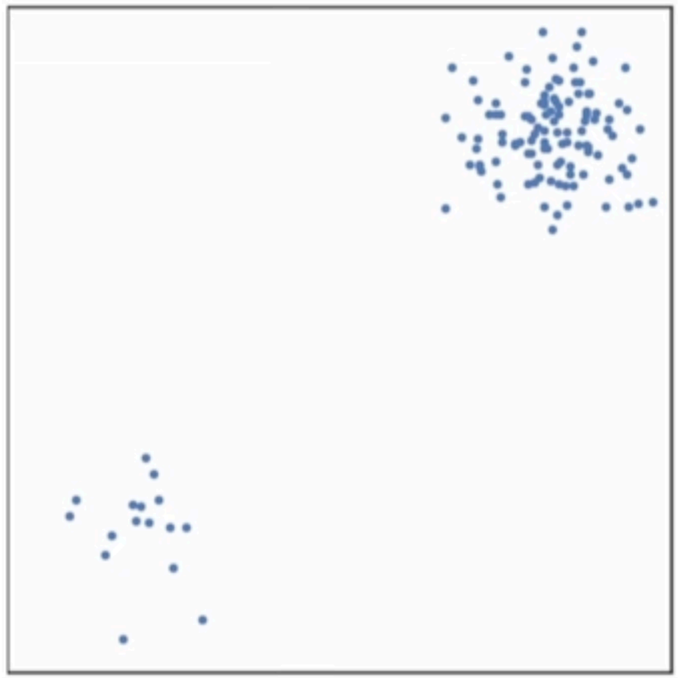
Y. Song et al., Generative Modeling by Estimating Gradients of the Data Distribution, NeurIPS2019

Training score-based models

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

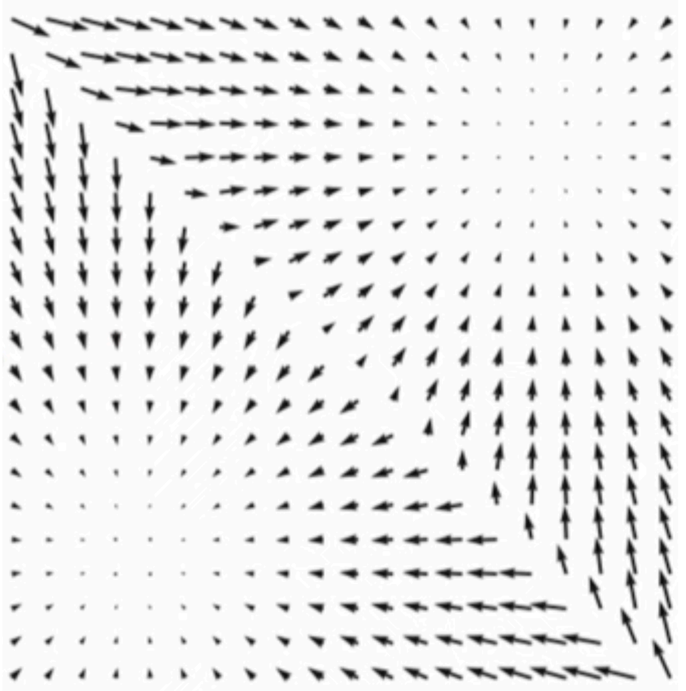


$D = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$



?

$s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{data}(\mathbf{x})$



Training score-based models

Objective

$$\min_{\theta} \underbrace{\mathbb{E}_{p_{data}(\mathbf{x})}\left[\frac{1}{2}\|\nabla_{\mathbf{x}}\log p_\theta(\mathbf{x}) - \nabla_{\mathbf{x}}\log p_{data}(\mathbf{x})\|_2^2\right]}_{\text{Fisher divergence } D_F(p_{data}\|p_\theta)}$$

If $\forall \mathbf{x}\ \nabla_{\mathbf{x}}\log p_\theta(\mathbf{x}) = \nabla_{\mathbf{x}}\log p_{data}(\mathbf{x})$ then $p_\theta(\mathbf{x}) = p_{data}(\mathbf{x})$

Training score-based models

Objective

$$\min_{\theta} \underbrace{\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\left[\frac{1}{2}\|\nabla_{\mathbf{x}}\log p_\theta(\mathbf{x}) - \nabla_{\mathbf{x}}\log p_{data}(\mathbf{x})\|_2^2\right]}_{\text{Fisher divergence } D_F(p_{data}\|p_\theta)} = \min_{\theta}\frac{1}{2}\mathop{\mathbb{E}}_{p(\mathbf{x})}\left[\|s_\theta(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}}\log p_{data}(\mathbf{x})}_{\text{Unknown}}\|_2^2\right]$$

If $\forall \mathbf{x}\ \nabla_{\mathbf{x}}\log p_\theta(\mathbf{x}) = \nabla_{\mathbf{x}}\log p_{data}(\mathbf{x})$ then $p_\theta(\mathbf{x}) = p_{data}(\mathbf{x})$

Training score-based models

Objective

$$\min_{\theta} \frac{1}{2} \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \left[\| s_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{data}(\mathbf{x}) \|_2^2 \right]$$

Unknown

Score matching

$$\min_{\theta} \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \left[\frac{1}{2} \| s_\theta(\mathbf{x}) \|_2^2 + tr(\nabla_{\mathbf{x}} s_\theta(\mathbf{x})) \right]$$

Jacobian

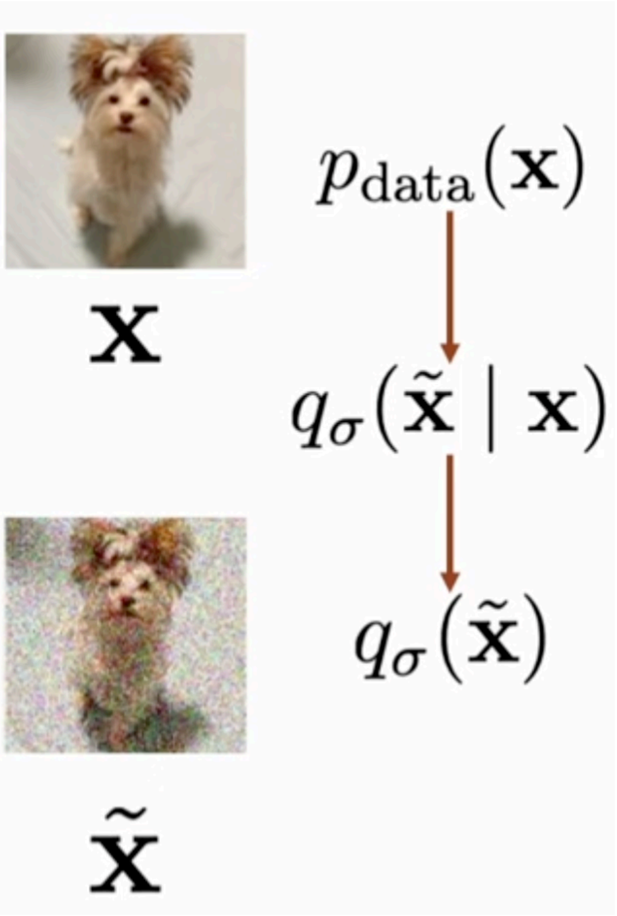
Too expensive to calculate in practice :(

Denoising score matching

Let's slightly perturb the data with Gaussian distribution

$$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2 I)$$

$$q_\sigma(\tilde{\mathbf{x}}) = \int p_{data}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) d\mathbf{x}$$



- Score estimation for $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$ is much easier
- For small noise $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) \approx \nabla_{\mathbf{x}} \log p_{data}(\mathbf{x})$

Denoising score matching

$$\frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 = \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 d\tilde{\mathbf{x}} =$$ Expand $\|\cdot\|_2^2$

Denoising score matching

$$\frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 = \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 d\tilde{\mathbf{x}} = \quad \text{Expand } \|\cdot\|_2^2$$

$$= \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\|\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 d\tilde{\mathbf{x}} + \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\|s_\theta(\tilde{\mathbf{x}})\|_2^2 d\tilde{\mathbf{x}} - \int q_\sigma(\tilde{\mathbf{x}})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}}) d\tilde{\mathbf{x}} =$$

$$= const + \frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}})\|_2^2 - \int q_\sigma(\tilde{\mathbf{x}})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}}) d\tilde{\mathbf{x}}$$

Denoising score matching

$$-\int q_\sigma(\tilde{\mathbf{x}})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})\,d\tilde{\mathbf{x}}$$

Denoising score matching

$$-\int q_\sigma(\tilde{\mathbf{x}})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})\,d\tilde{\mathbf{x}} = -\int q_\sigma(\tilde{\mathbf{x}})\frac{1}{q_\sigma(\tilde{\mathbf{x}})}\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})\,d\tilde{\mathbf{x}} =$$

Denoising score matching

$$-\int q_\sigma(\tilde{\mathbf{x}})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\mathbf{x})\,d\tilde{\mathbf{x}} = -\int q_\sigma(\tilde{\mathbf{x}})\frac{1}{q_\sigma(\tilde{\mathbf{x}})}\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\mathbf{x})\,d\tilde{\mathbf{x}} = -\int \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})\,d\tilde{\mathbf{x}} =$$

Denoising score matching

$$-\int q_\sigma(\tilde{\mathbf{x}})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int q_\sigma(\tilde{\mathbf{x}})\frac{1}{q_\sigma(\tilde{\mathbf{x}})}\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} =$$

$$= -\int \nabla_{\tilde{\mathbf{x}}}\left(\int p_{data}(\mathbf{x})q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} =$$

Denoising score matching

$$-\int q_\sigma(\tilde{\mathbf{x}})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int q_\sigma(\tilde{\mathbf{x}})\frac{1}{q_\sigma(\tilde{\mathbf{x}})}\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} =$$

$$= -\int \nabla_{\tilde{\mathbf{x}}}\left(\int p_{data}(\mathbf{x})q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int\left(\int p_{data}(\mathbf{x})\,\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} =$$

Denoising score matching

$$-\int q_\sigma(\tilde{\mathbf{x}})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int q_\sigma(\tilde{\mathbf{x}})\frac{1}{q_\sigma(\tilde{\mathbf{x}})}\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} =$$

$$= -\int \nabla_{\tilde{\mathbf{x}}}\left(\int p_{data}(\mathbf{x})q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int\left(\int p_{data}(\mathbf{x})\,\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} =$$

$$= -\int\left(\int p_{data}(\mathbf{x})q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} =$$

Log-derivative trick: $\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x}) = q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})$

Denoising score matching

$$-\int q_\sigma(\tilde{\mathbf{x}})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int q_\sigma(\tilde{\mathbf{x}})\frac{1}{q_\sigma(\tilde{\mathbf{x}})}\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} =$$

$$= -\int \nabla_{\tilde{\mathbf{x}}}\left(\int p_{data}(\mathbf{x})q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int\left(\int p_{data}(\mathbf{x})\,\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} =$$

$$= -\int\left(\int p(\mathbf{x})q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int\int p(\mathbf{x})q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})^\top s_\theta(\tilde{\mathbf{x}})d\mathbf{x}d\tilde{\mathbf{x}} =$$

Denoising score matching

$$-\int q_\sigma(\tilde{\mathbf{x}})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int q_\sigma(\tilde{\mathbf{x}})\frac{1}{q_\sigma(\tilde{\mathbf{x}})}\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} =$$

$$= -\int \nabla_{\tilde{\mathbf{x}}}\left(\int p_{data}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int\left(\int p_{data}(\mathbf{x})\,\nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} =$$

$$= -\int\left(\int p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})d\mathbf{x}\right)^\top s_\theta(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = -\int\int p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})^\top s_\theta(\tilde{\mathbf{x}})d\mathbf{x}d\tilde{\mathbf{x}} =$$

$$= -\mathop{\mathbb{E}}_{q(\tilde{\mathbf{x}}|\mathbf{x})}\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\left[\nabla_{\tilde{\mathbf{x}}}\log q(\tilde{\mathbf{x}}\,|\,\mathbf{x})^\top s_\theta(\tilde{\mathbf{x}})\right]$$

Denoising score matching

$$\frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 = \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 d\tilde{\mathbf{x}} = \quad \text{Expand } \|\cdot\|_2^2$$

$$= \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\|\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 d\tilde{\mathbf{x}} + \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\|s_\theta(\tilde{\mathbf{x}})\|_2^2 d\tilde{\mathbf{x}} - \int q_\sigma(\tilde{\mathbf{x}})\,\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top s_\theta(\tilde{\mathbf{x}}) d\tilde{\mathbf{x}} =$$

$$= const + \frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}})\|_2^2 - \mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\left[\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})^\top s_\theta(\tilde{\mathbf{x}})\right]$$

Denoising score matching

$$\frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 = const + \frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}})\|_2^2 - \mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\left[\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})^\top s_\theta(\tilde{\mathbf{x}})\right] =$$

Denoising score matching

$$\frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 = const + \frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}})\|_2^2 - \mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\left[\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})^\top s_\theta(\tilde{\mathbf{x}})\right] =$$

$$= const + \mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\|_2^2 - \frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\|\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\|_2^2 =$$

Denoising score matching

$$\frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 = const + \frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}})\|_2^2 - \mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\left[\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})^\top s_\theta(\tilde{\mathbf{x}})\right] =$$

$$= const + \mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\|_2^2 - \frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\|\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\|_2^2 =$$

$$= const + \boxed{\frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\|_2^2} + const$$

Denoising score matching

$$\frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}})}\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2 = \frac{1}{2}\mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\mathop{\mathbb{E}}_{p_{data}(\mathbf{x})}\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})\|_2^2 + const$$

$\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x})$ **is analytically calculated**

$q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}\,|\,\mathbf{x}, \sigma^2 I) \rightarrow \tilde{\mathbf{x}} = \mathbf{x} + \sigma \cdot \epsilon$, **where** $\epsilon \sim \mathcal{N}(0,I)$

$$\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x}) = -\frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2} = -\frac{\epsilon}{\sigma}$$

Denoising score matching

$$L_{DSM} = \mathop{\mathbb{E}}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2 = \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}(0,I)} \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \|s_\theta(\mathbf{x} + \sigma \cdot \epsilon) + \frac{\epsilon}{\sigma}\|_2^2$$

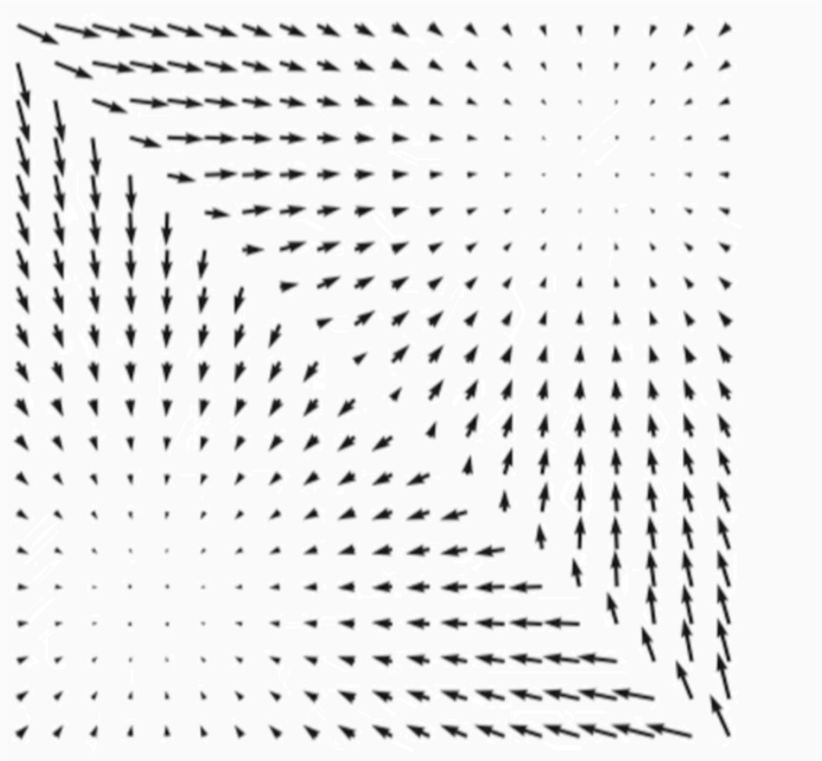
Training algorithm

1. Sample a batch $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \sim p_{data}$
2. Sample noise $\{\epsilon_1, \ldots, \epsilon_N\} \sim \mathcal{N}(0,I)$
3. Perturb the batch by $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \sigma \cdot \epsilon_i$
4. Update $\theta \leftarrow \theta - \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \|s_\theta(\tilde{\mathbf{x}}_i) + \frac{\epsilon_i}{\sigma}\|_2^2$

For good $\nabla_{\mathbf{x}} \log p_{data}(\mathbf{x})$ approximation

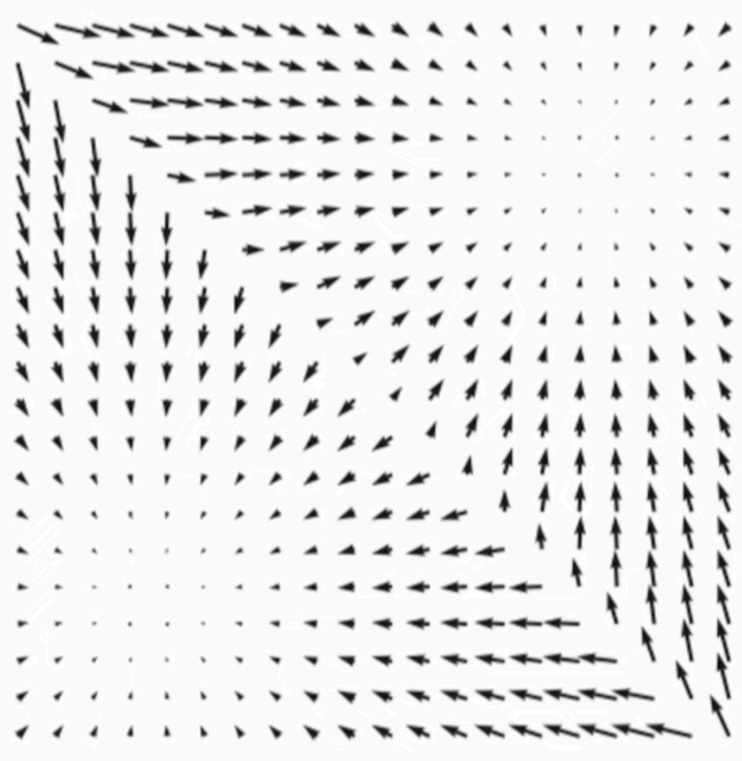
σ has to be very small, e.g., 0.001

Sampling with score function

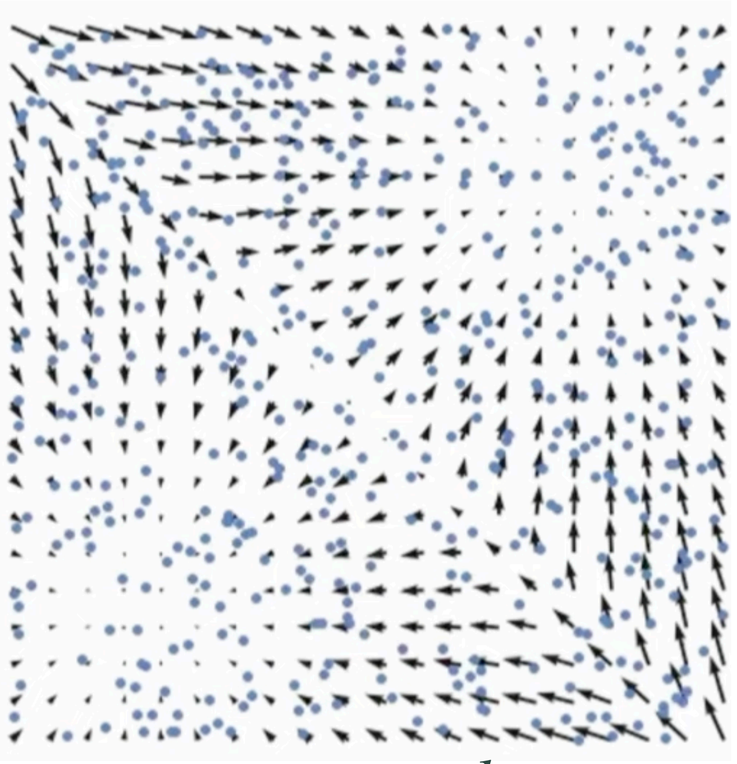


Scores $s_\theta(\mathbf{x})$

Sampling with score function

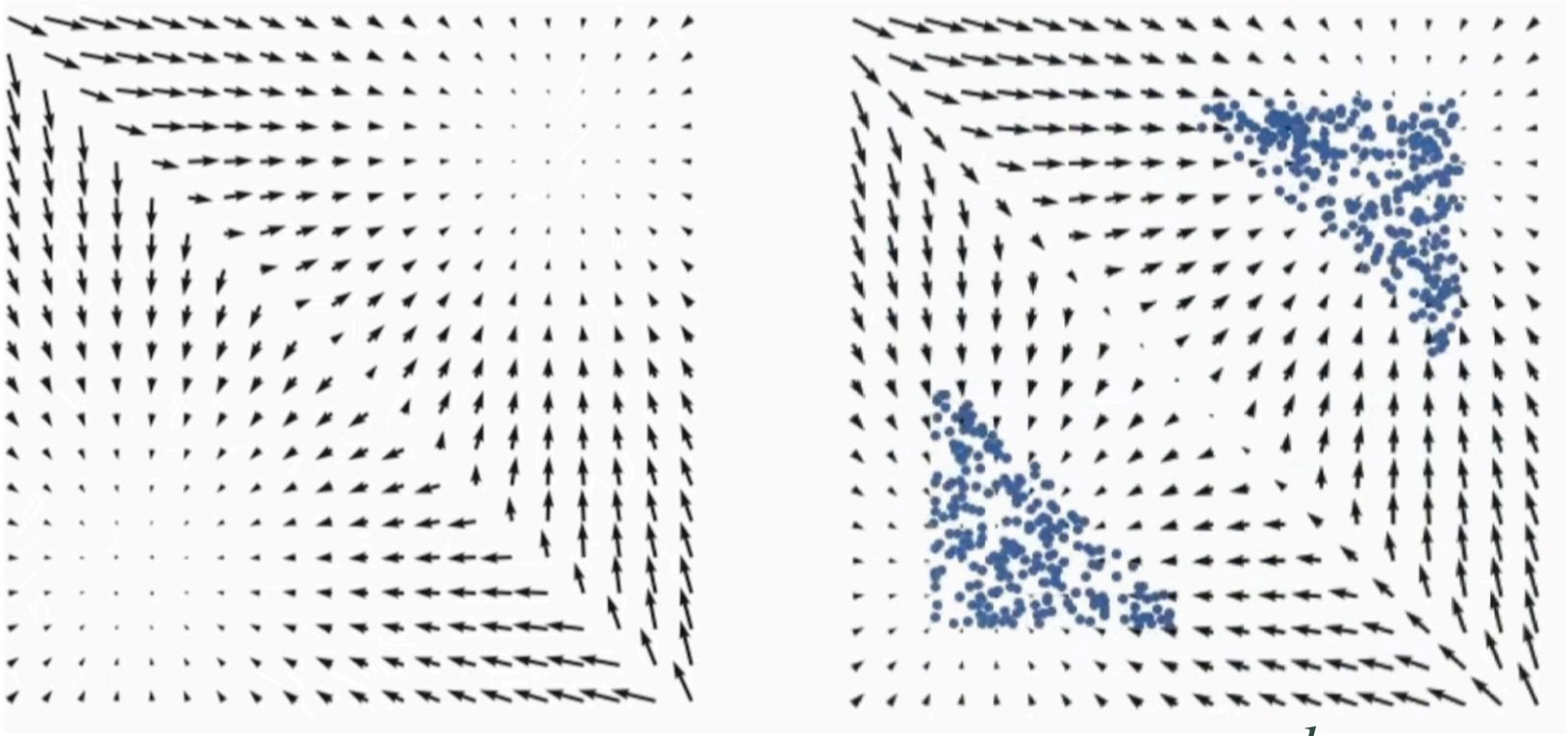


Scores $s_\theta(\mathbf{x})$



$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2} s_\theta(\mathbf{x})$$

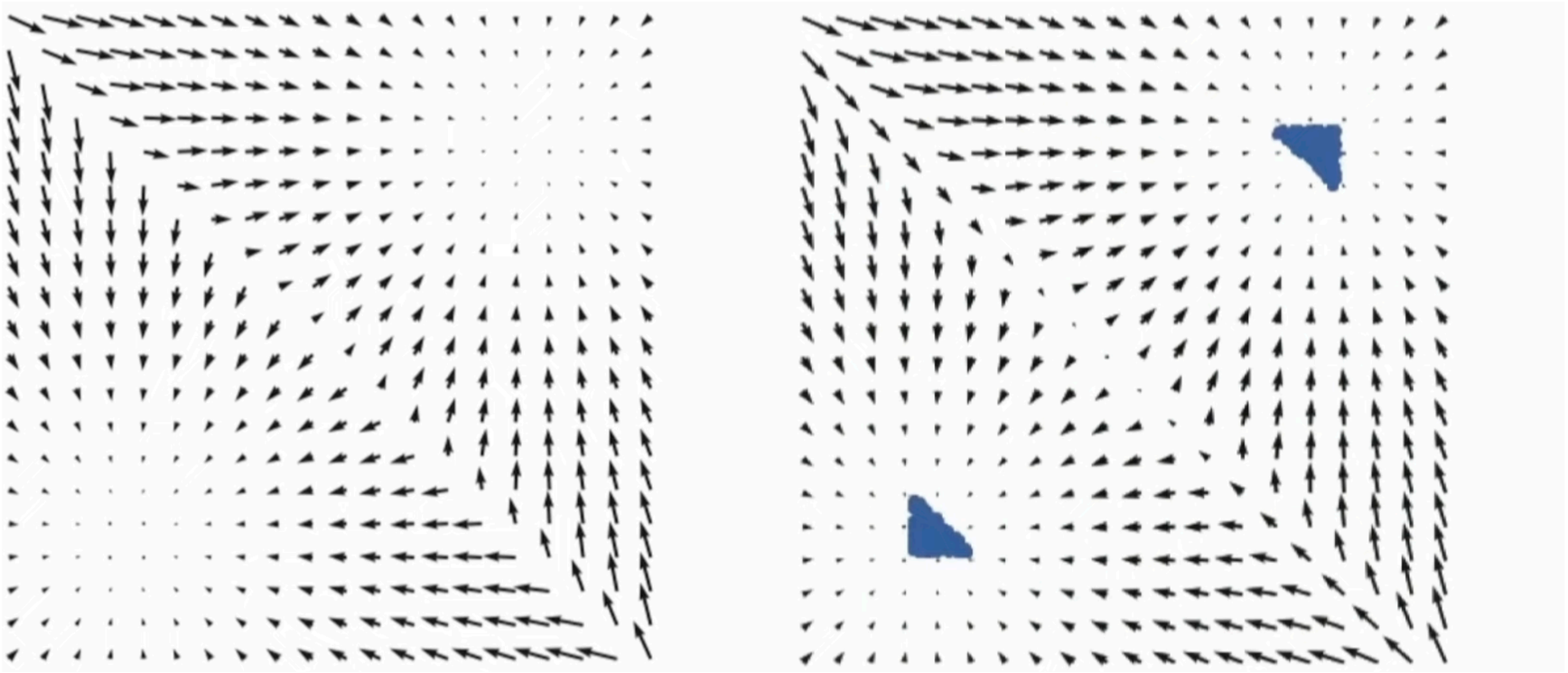
Sampling with score function



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

$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2} s_\theta(\mathbf{x})$$

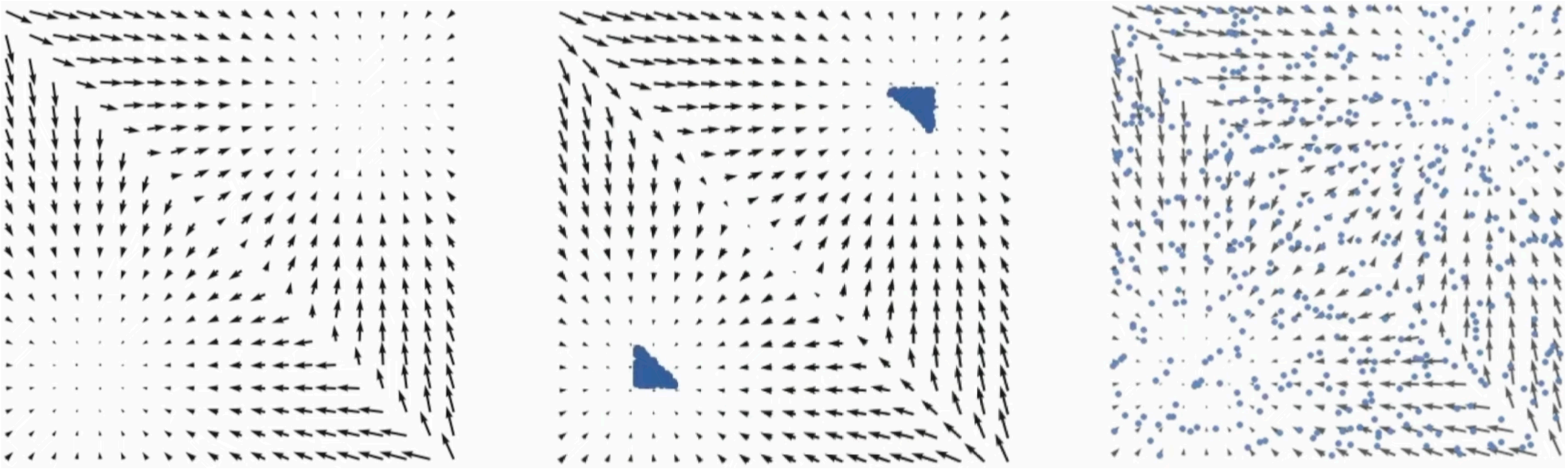
Sampling with score function



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

$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2} s_\theta(\mathbf{x})$$

Sampling with score function



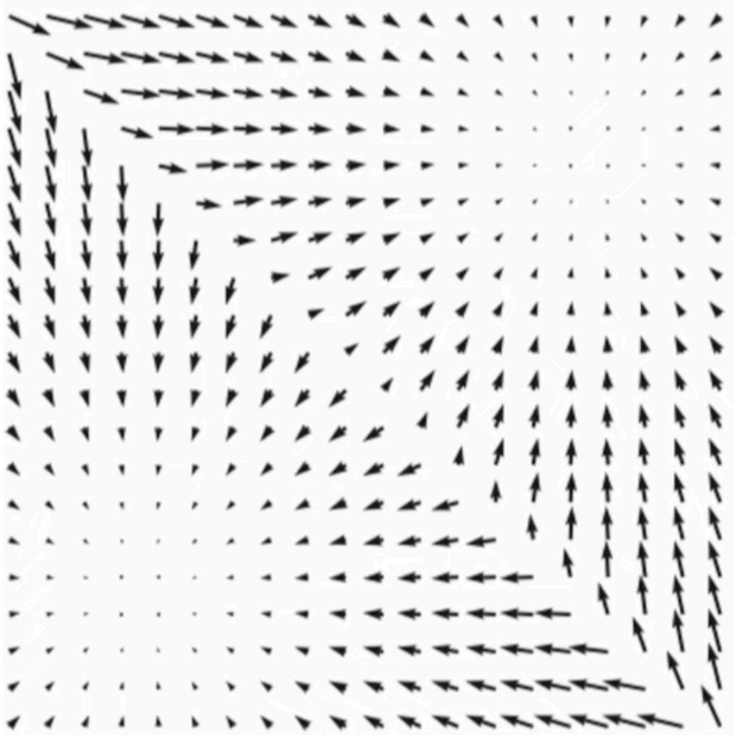
Scores $s_\theta(\mathbf{x})$

$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2} s_\theta(\mathbf{x})$$

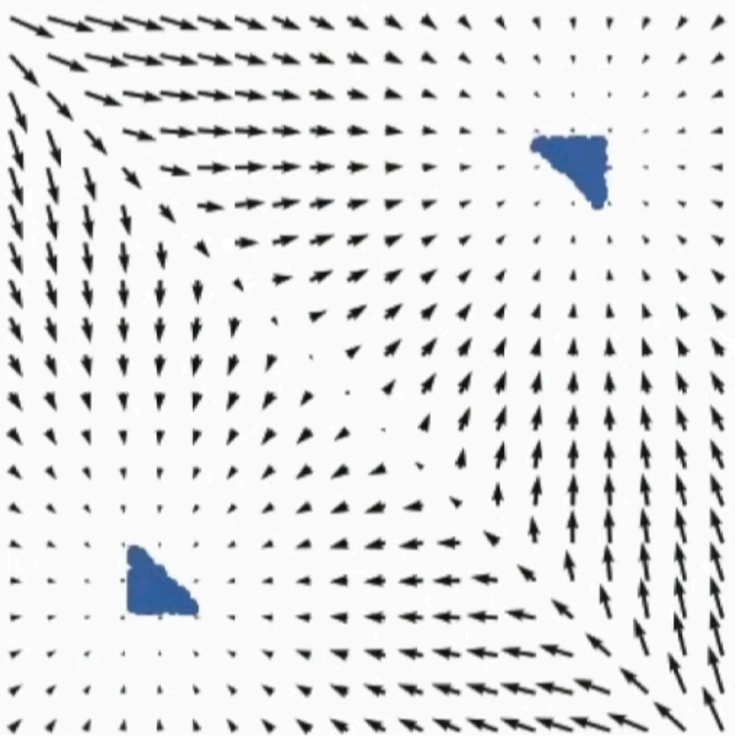
$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2} s_\theta(\mathbf{x}) + \sqrt{h}\epsilon^t$$

Langevin dynamics

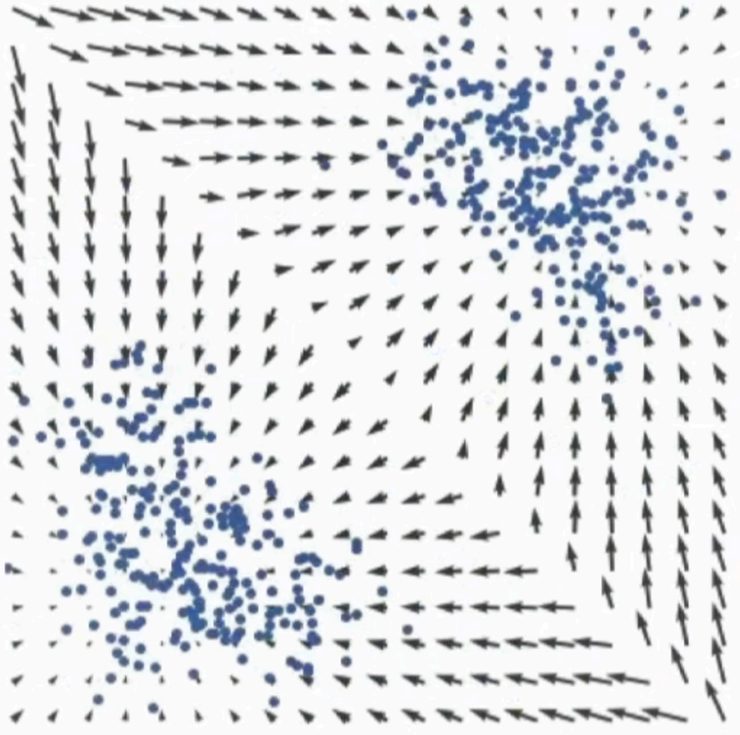
Sampling with score function



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



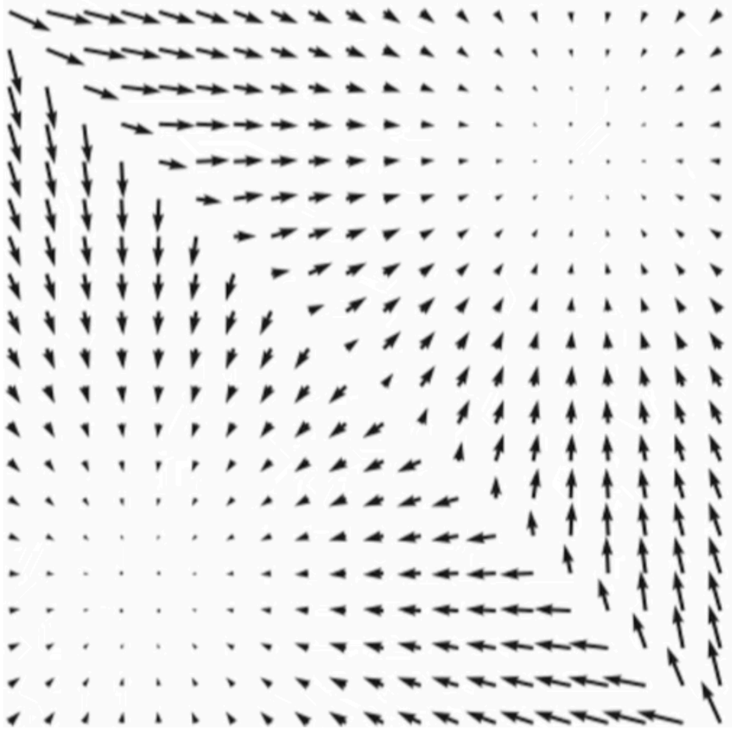
$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2} s_\theta(\mathbf{x})$$



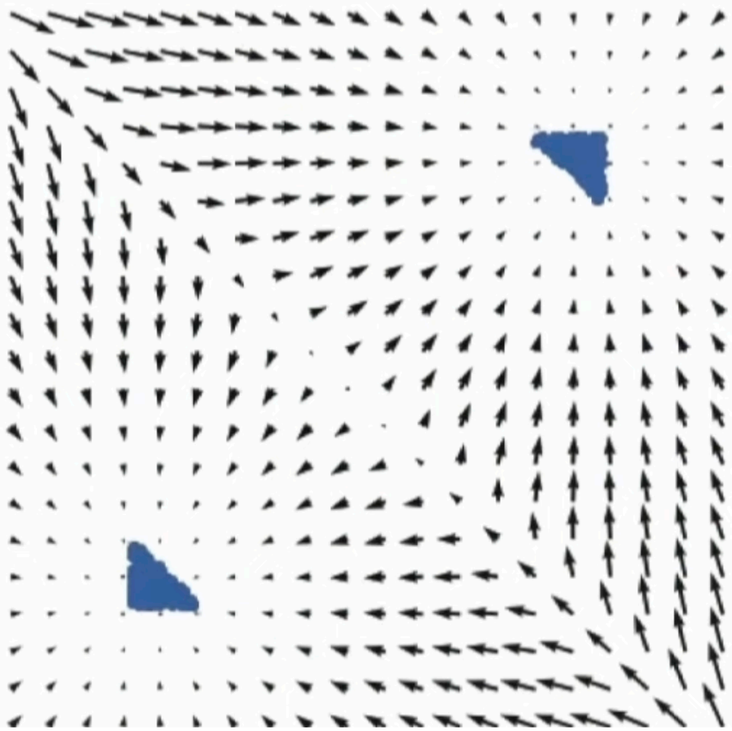
$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2} s_\theta(\mathbf{x}) + \sqrt{h}\epsilon^t$$

Langevin dynamics

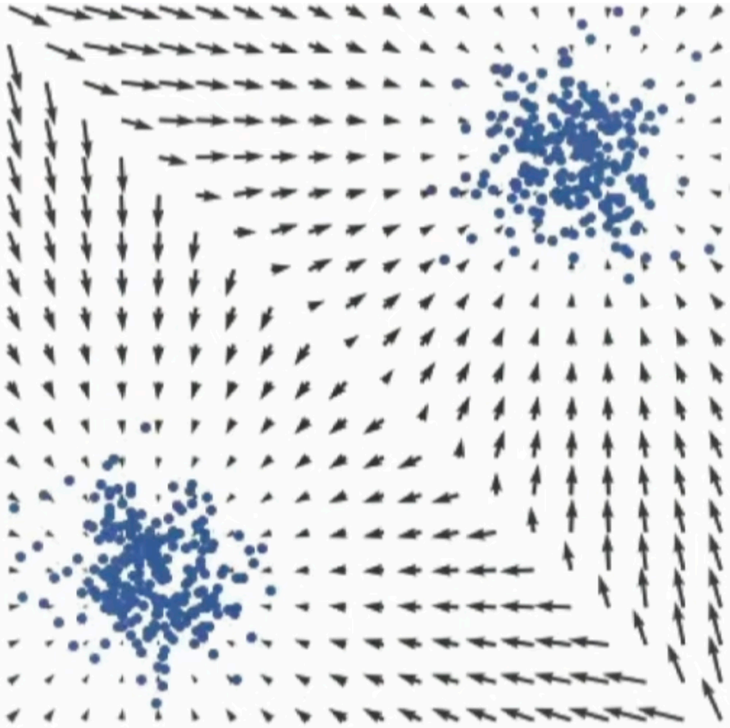
Sampling with score function



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



$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2} s_\theta(\mathbf{x})$$



$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2} s_\theta(\mathbf{x}) + \sqrt{h}\epsilon^t$$

Langevin dynamics

Langevin dynamics sampling

Langevin dynamics produces samples from $p(\mathbf{x})$ using $\nabla_{\mathbf{x}} \log p(\mathbf{x})$

Sampling algorithm

1. Sample initial point $\mathbf{x}^0 \sim \mathcal{N}(0, I)$
2. For $t \leftarrow 1, 2, \ldots, T$

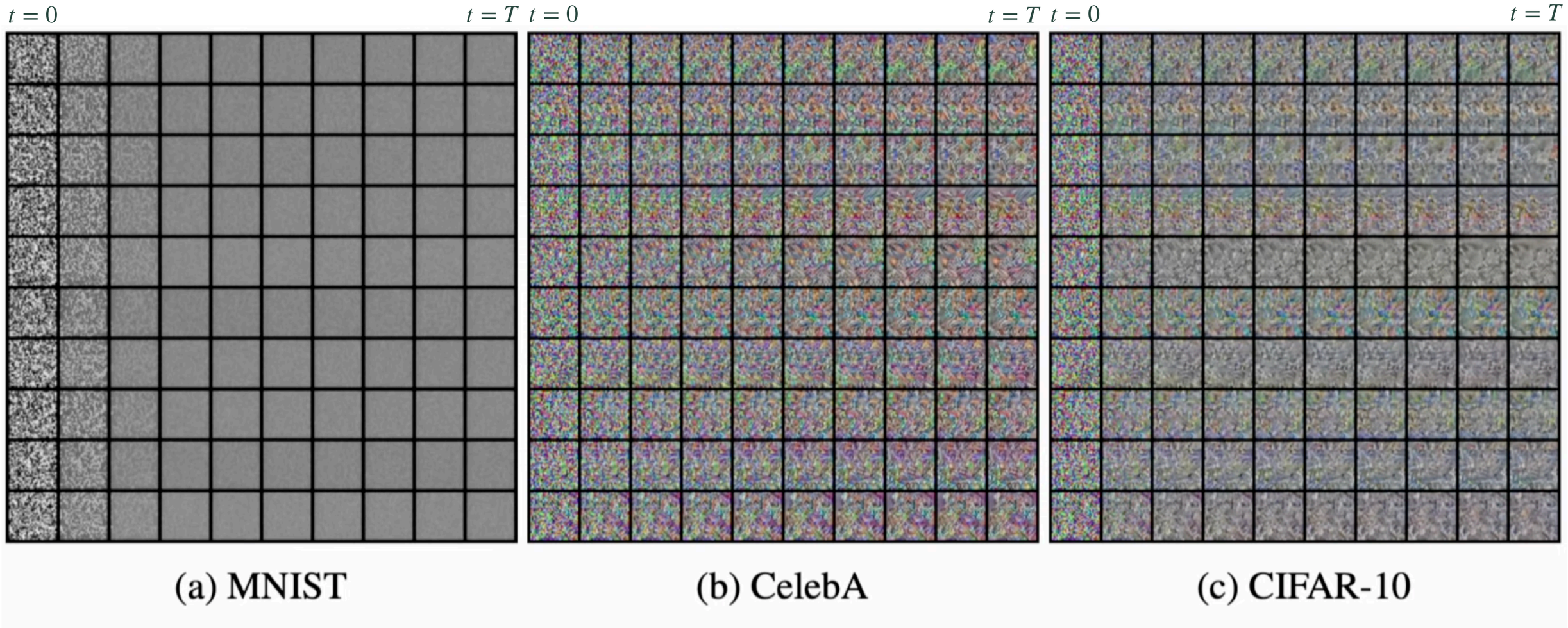
$$\epsilon^t \sim \mathcal{N}(0, I)$$

$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2} \nabla_{\mathbf{x}} \log p(\mathbf{x}^t) + \sqrt{h} \epsilon^t$$

For $h \to 0$ and $T \to \infty$: we are guaranteed to get $\mathbf{x}^T \sim p(\mathbf{x})$

Langevin dynamics sampling

Langevin dynamics using $s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$ does not work in practice :(



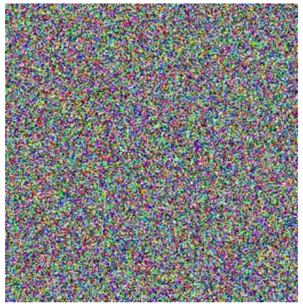
Langevin dynamics sampling

Initial points are from low-density regions

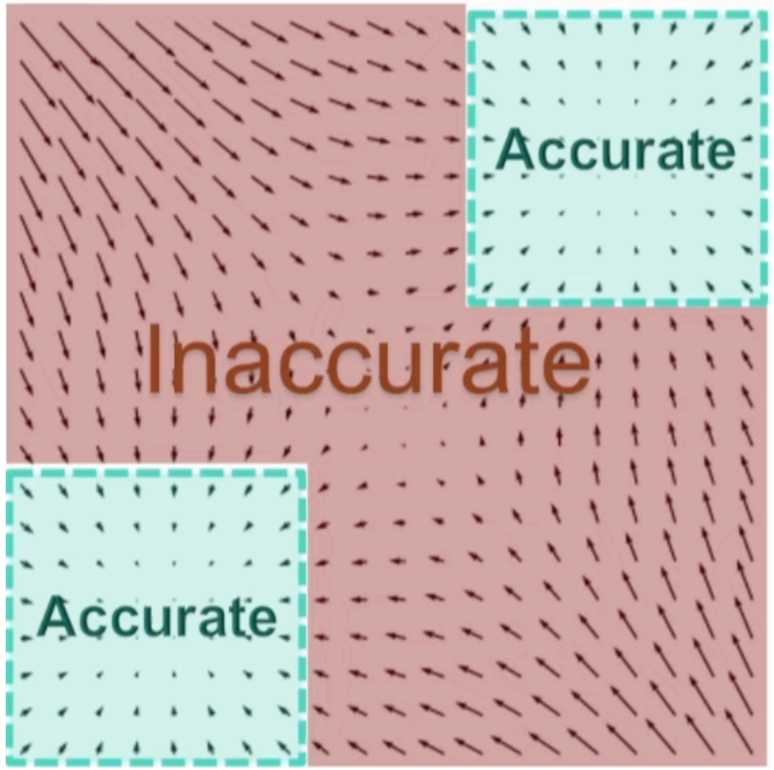
High density $p(\mathbf{x})$

Good $\nabla_{\mathbf{x}} \log p(\mathbf{x})$

$p(\mathbf{x}) \approx 0$



Undefined $\nabla_{\mathbf{x}} \log p(\mathbf{x})$



$s_\theta(\mathbf{x})$ does not observe such points during training

$\rightarrow$ walking in random directions

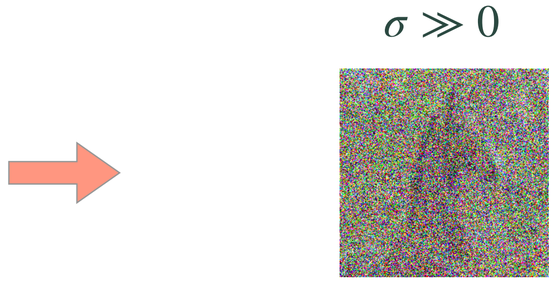
Song and Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. NeurIPS2019

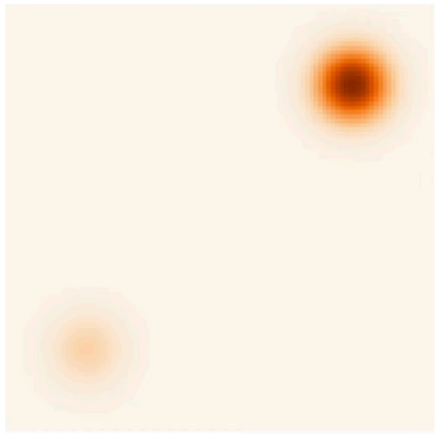
Langevin dynamics sampling

Let's perturb the data much stronger for denoising score matching

Perturbed distribution is much denser

But $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) \approx \nabla_{\mathbf{x}} \log p_{data}(\mathbf{x})$ for $\sigma \gg 0$





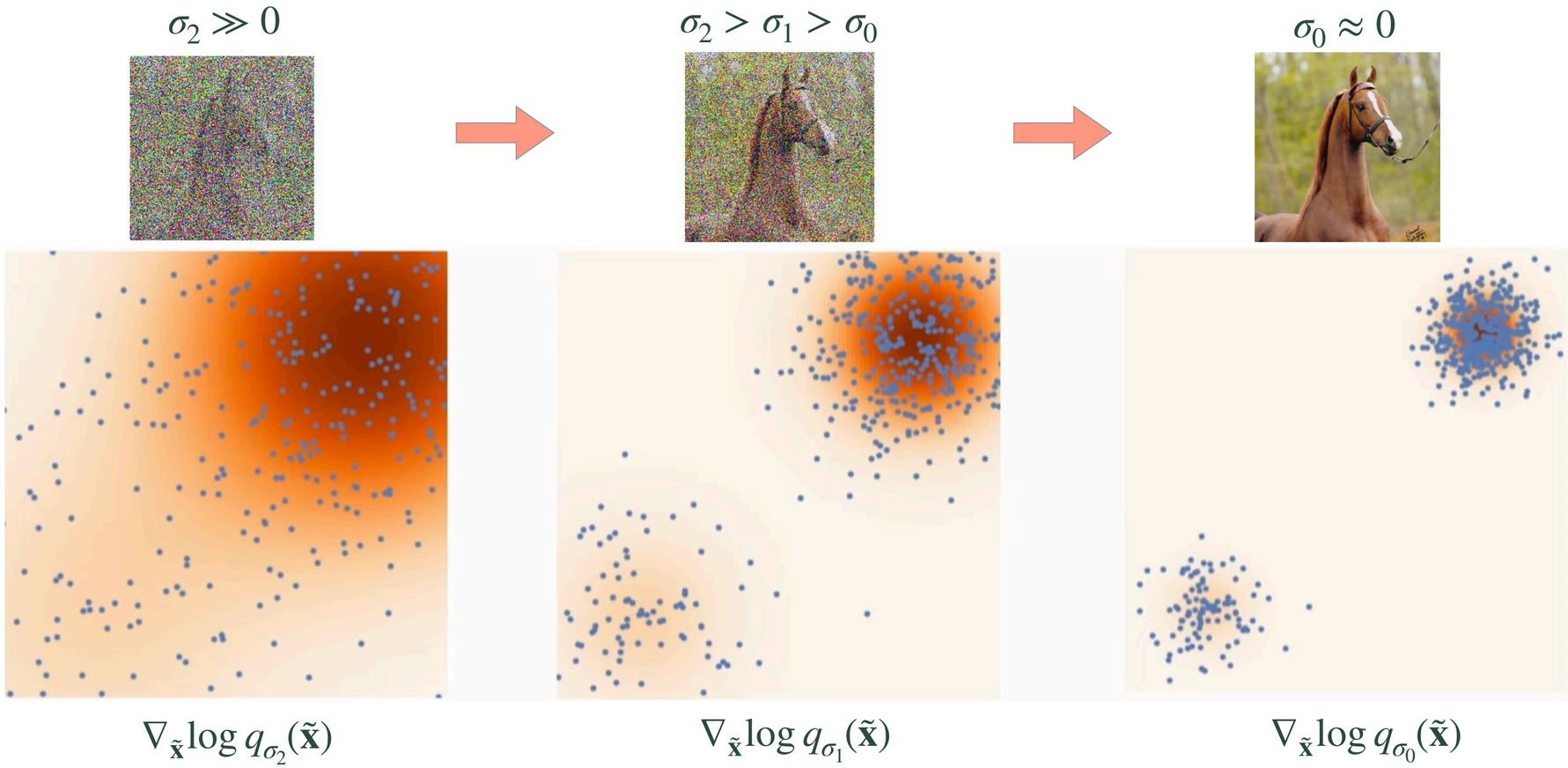
Sampling with multiple noise levels

Let's gradually transit from large σ_T to small σ_0

Key idea

$\tilde{\mathbf{x}} \sim q_{\sigma_t}(\tilde{\mathbf{x}})$ is a good initial point for $\nabla_{\tilde{\mathbf{x}}} \log q_{\sigma_{t-1}}(\tilde{\mathbf{x}})$

If $\Delta(\sigma_{t-1}, \sigma_t)$ is small



Annealed Langevin dynamics sampling

Sample initial point $\mathbf{x}_{\sigma_T}^0 \sim \mathcal{N}(0, \sigma_T I)$

For $\sigma_t \leftarrow \sigma_T, \ldots, \sigma_0$ // Anneal down the noise level

$\mathbf{x}_{\sigma_t}^0 \leftarrow \mathbf{x}_{\sigma_{t+1}}^M$

For $m \leftarrow 1, 2, \ldots, M$ // Langevin dynamics for $p_{\sigma_t}(\mathbf{x})$

$\epsilon^m \sim \mathcal{N}(0, I)$

$$\mathbf{x}_{\sigma_t}^{m+1} \leftarrow \mathbf{x}_{\sigma_t}^m + \frac{h}{2} \nabla_{\mathbf{x}} \log p_{\sigma_t}(\mathbf{x}_{\sigma_t}^m) + \sqrt{h} \epsilon^m$$

Usually $M = 1$ in practice

$\{\sigma_T, \ldots, \sigma_0\}$ schedule can be set at inference

Annealed Langevin dynamics sampling

Sample initial point $\mathbf{x}_{\sigma_T}^0 \sim \mathcal{N}(0, \sigma_T I)$

For $\sigma_t \leftarrow \sigma_T, \ldots, \sigma_0$ // Anneal down the noise level

$\mathbf{x}_{\sigma_t}^0 \leftarrow \mathbf{x}_{\sigma_{t+1}}^M$

For $m \leftarrow 1, 2, \ldots, M$ // Langevin dynamics for $p_{\sigma_t}(\mathbf{x})$

$\epsilon^m \sim \mathcal{N}(0, I)$

$$\mathbf{x}_{\sigma_t}^{m+1} \leftarrow \mathbf{x}_{\sigma_t}^m + \frac{h}{2} \nabla_{\mathbf{x}} \log p_{\sigma_t}(\mathbf{x}_{\sigma_t}^m) + \sqrt{h} \epsilon^m$$

Usually $M = 1$ in practice

$\{\sigma_T, \ldots, \sigma_0\}$ schedule can be set at inference

How to train $s_\theta(\tilde{\mathbf{x}})$ for multiple noise levels?

Final denoising score matching training

$$L_{DSM} = \mathop{\mathbb{E}}_{\sigma \sim U(\sigma_{min}, \sigma_{max})} \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}(0,I)} \mathop{\mathbb{E}}_{p_{data}(\mathbf{x})} \| s_\theta(\mathbf{x} + \sigma \cdot \epsilon, \sigma) + \frac{\epsilon}{\sigma} \|_2^2$$

Training algorithm

1. Sample a batch: $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \sim p_{data}$
2. Sample noise: $\{\epsilon_1, \ldots, \epsilon_N\} \sim \mathcal{N}(0,I)$
3. Sample noise levels: $\{\sigma_1, \ldots, \sigma_N\} \sim U(\sigma_{min}, \sigma_{max})$
4. Perturb the batch: $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \sigma_i \cdot \epsilon_i$
5. Update $\theta \leftarrow \theta - \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \| s_\theta(\tilde{\mathbf{x}}_i, \sigma_i) + \frac{\epsilon_i}{\sigma} \|_2^2$

$s_\theta(\tilde{\mathbf{x}}, \sigma)$ now approximates $\nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}})$ for multiple noise levels

Connection to DDPM

Simplify

$$L_{DSM} = \mathop{\mathbb{E}}_{\sigma,\epsilon,\mathbf{x}} \|s_\theta(\mathbf{x} + \sigma \cdot \epsilon, \sigma) + \frac{\epsilon}{\sigma}\|_2^2 = \mathop{\mathbb{E}}_{\sigma,\epsilon,\mathbf{x}} \cancel{\frac{1}{\sigma^2}} \|\sigma \cdot s_\theta(\mathbf{x} + \sigma \cdot \epsilon, \sigma) + \epsilon\|_2^2 \rightarrow \mathop{\mathbb{E}}_{\sigma,\epsilon,\mathbf{x}} \|\epsilon_\theta(\mathbf{x} + \sigma \cdot \epsilon, \sigma) - \epsilon\|_2^2$$

$$\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = -\frac{\epsilon}{\sigma} \rightarrow \text{reparametrize } s_\theta(\tilde{\mathbf{x}}, \sigma) \rightarrow s_\theta(\tilde{\mathbf{x}}, \sigma) = \boxed{-\frac{\epsilon_\theta(\tilde{\mathbf{x}}, \sigma)}{\sigma}}$$

Connection to DDPM

Simplify

$$L_{DSM} = \mathop{\mathbb{E}}_{\sigma,\epsilon,\mathbf{x}} \|s_\theta(\mathbf{x}+\sigma\cdot\epsilon,\sigma)+\frac{\epsilon}{\sigma}\|_2^2 = \mathop{\mathbb{E}}_{\sigma,\epsilon,\mathbf{x}} \cancel{\frac{1}{\sigma^2}} \|\sigma\cdot s_\theta(\mathbf{x}+\sigma\cdot\epsilon,\sigma)+\epsilon\|_2^2 \to \mathop{\mathbb{E}}_{\sigma,\epsilon,\mathbf{x}} \|\epsilon_\theta(\mathbf{x}+\sigma\cdot\epsilon,\sigma)-\epsilon\|_2^2$$

$$\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}\,|\,\mathbf{x}) = -\frac{\epsilon}{\sigma} \to \text{reparametrize } s_\theta(\tilde{\mathbf{x}},\sigma) \to s_\theta(\tilde{\mathbf{x}},\sigma) = \boxed{-\frac{\epsilon_\theta(\tilde{\mathbf{x}},\sigma)}{\sigma}}$$

DSM

$q_{\sigma_t}(\tilde{\mathbf{x}}\,|\,\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}\,|\,\mathbf{x},\sigma_t^2 I)$ | Variance exploding process: $q_{\sigma_T} = \mathcal{N}(\mathbf{x},\sigma_T^2 I), \sigma_T \gg 1$

DDPM

$q_{\sigma_t}(\tilde{\mathbf{x}}\,|\,\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}\,|\,\sqrt{1-\sigma_t^2}\cdot\mathbf{x},\sigma_t^2 I)$, where $\sigma_t = \sqrt{1-\bar{\alpha}_t}$ | Variance preserving process: $q_{\sigma_T} = \mathcal{N}(0,I)$

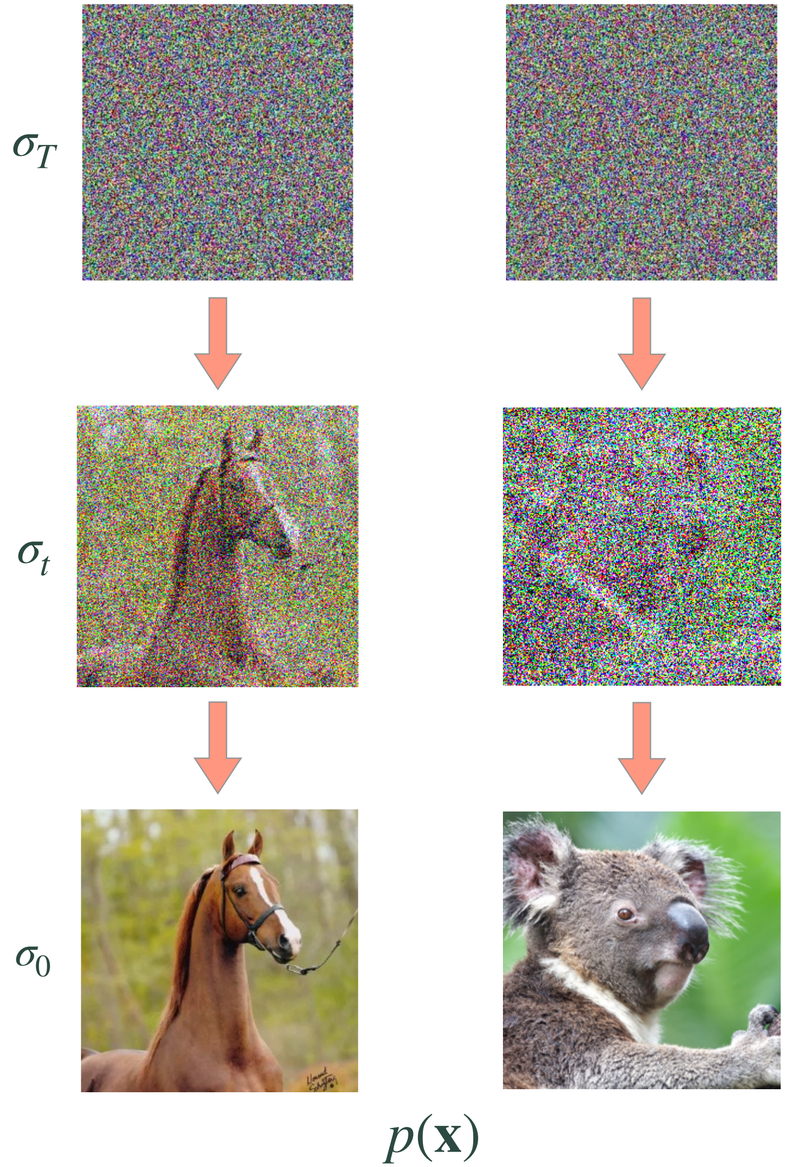
Unconditional generation

DSM produces samples from p_{data}

$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2}\nabla_{\mathbf{x}} \log p(\mathbf{x}^t) + \sqrt{h}\epsilon^t$$

p_{data} can contain many modes, e.g., object classes

Given a condition $\mathbf{y}$, how to sample from $p_{data}(\mathbf{x} \mid \mathbf{y})$ using the unconditional $s_\theta(\mathbf{x})$?



Classifier guidance

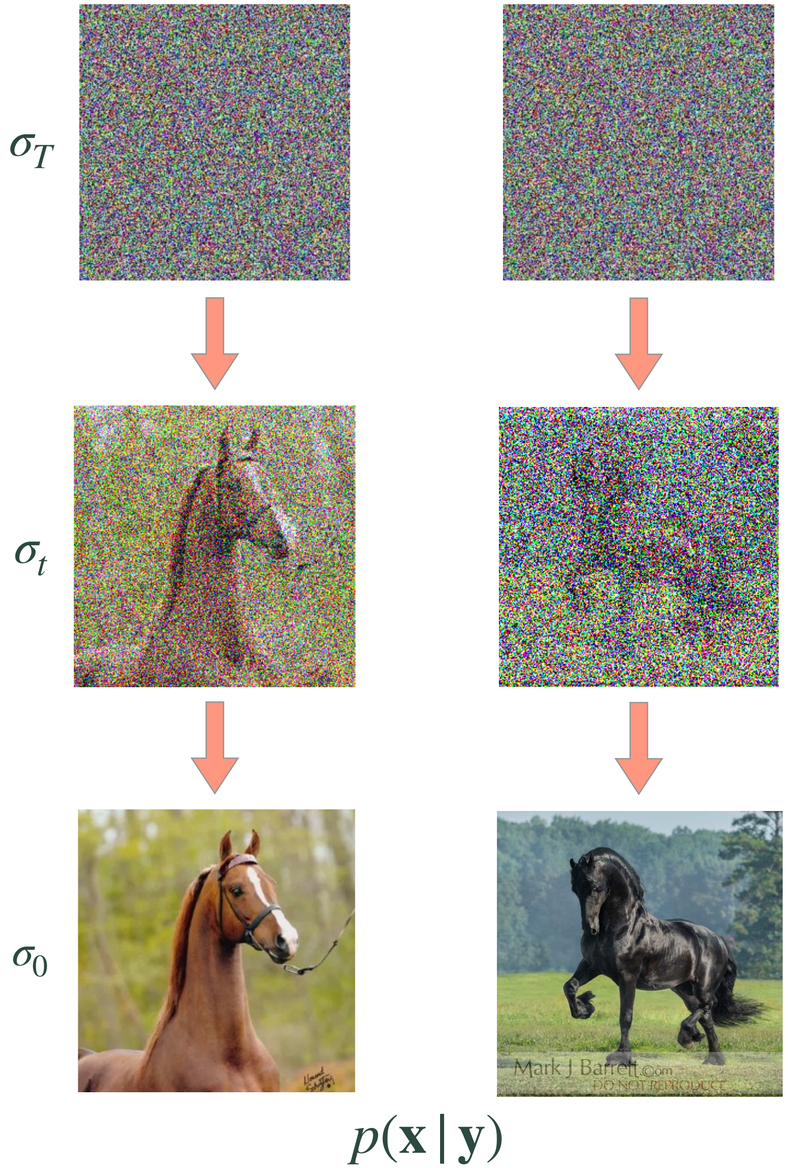
$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \frac{h}{2}\nabla_{\mathbf{x}}\log p_\gamma(\mathbf{x}^t \mid \mathbf{y}) + \sqrt{h}\epsilon^t$$

To enhance conditioning, consider a sharpened distribution $p_\gamma(\mathbf{x} \mid \mathbf{y})$

$p_\gamma(\mathbf{x} \mid \mathbf{y}) \propto p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})^\gamma$ | Bayes' rule

γ — guidance scale

$$\nabla_{\mathbf{x}}\log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}}\log p(\mathbf{x}) + \gamma\nabla_{\mathbf{x}}\log p(\mathbf{y} \mid \mathbf{x})$$
$$\approx s_\theta(\mathbf{x}) \qquad \approx ?$$



Classifier guidance

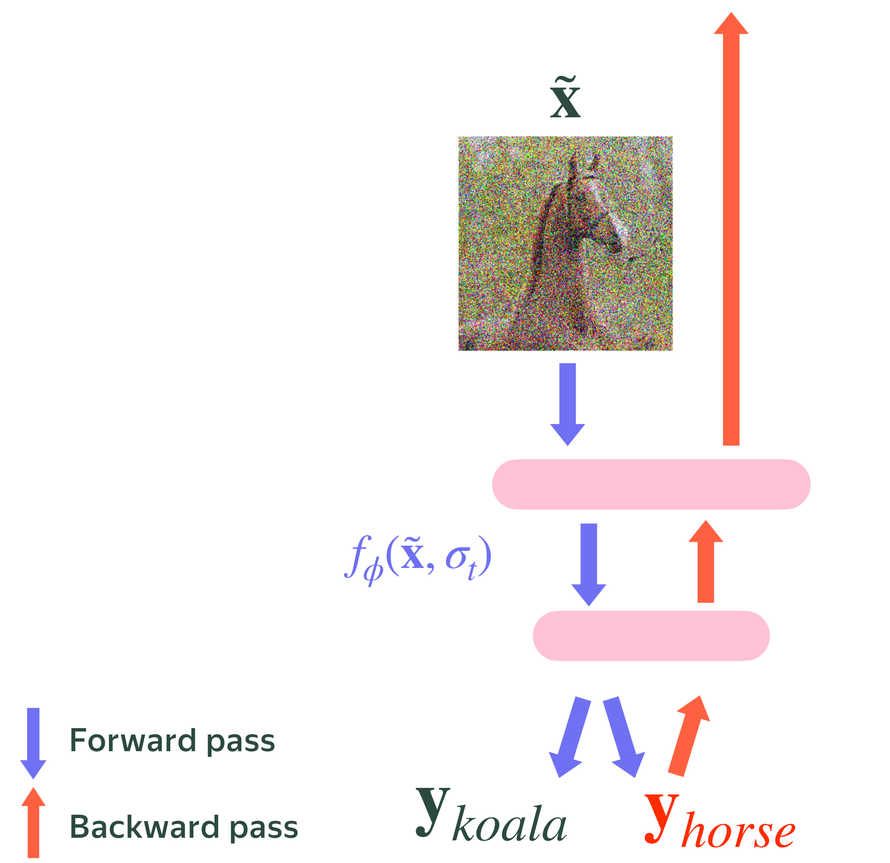
$p(\mathbf{y} \mid \mathbf{x})$ — can be modeled using a classifier $f_\phi(\mathbf{x}) \to \mathbf{y}$

How to get $\nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$?

1. Compute logits using $f_\phi(\mathbf{x})$ and select the $\mathbf{y}$-th one
2. Call autograd w.r.t. $\mathbf{x}$: $\nabla_{\mathbf{x}} f_\phi(\mathbf{x})$

How to deal with different noise levels?

- Train a new classifier on noisy images $f_\phi(\tilde{\mathbf{x}}, \sigma_t) \to \mathbf{y}$



DSM summary

Data noising is essential for training in practice

Difference noise levels are required for accurate sampling

Annealed Langevin dynamics for sampling over T noise levels

DDPM is a particular case of DSM with a different noising process

DSM enables a convenient conditioning mechanism