

# Deep Generative Models

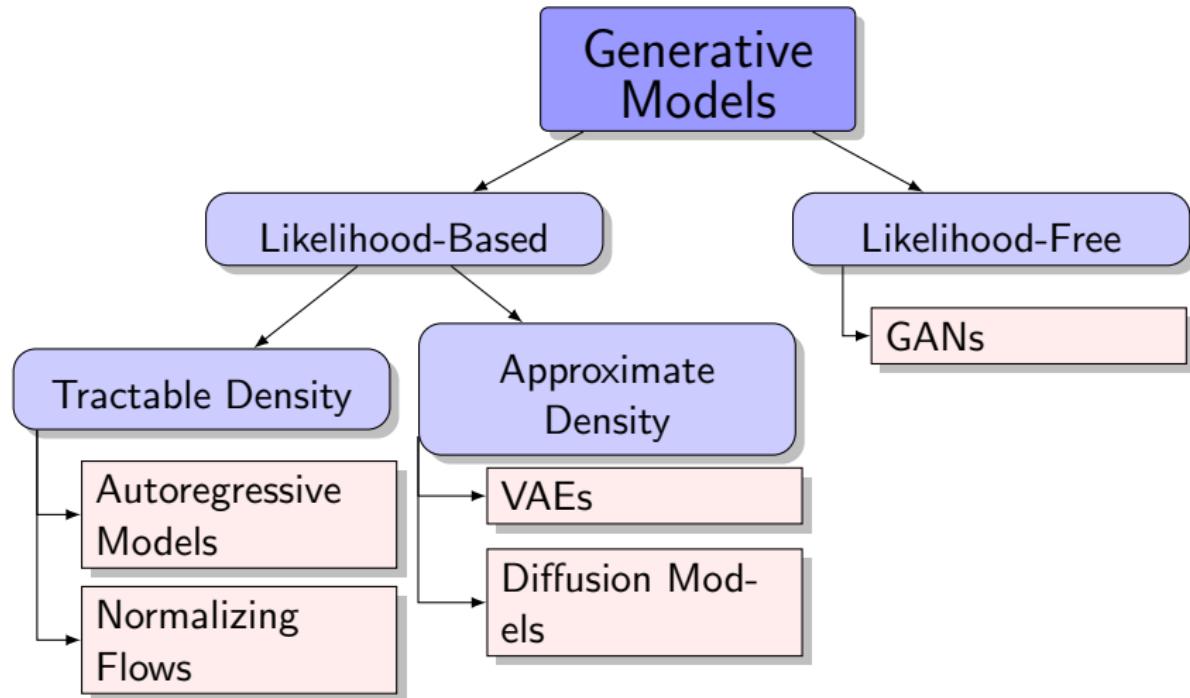
## Lecture 1

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Generative Models Zoo



# Outline

1. Generative Models Overview
2. Course Tricks
3. Problem Statement
4. Divergence Minimization Framework
5. Autoregressive Modeling

# Outline

1. Generative Models Overview
2. Course Tricks
3. Problem Statement
4. Divergence Minimization Framework
5. Autoregressive Modeling

## VAE – The First Scalable Approach for Image Generation



# DCGAN – The First Convolutional GAN for Image Generation



# StyleGAN – High-Quality Face Generation



Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

# Language Modeling at Scale

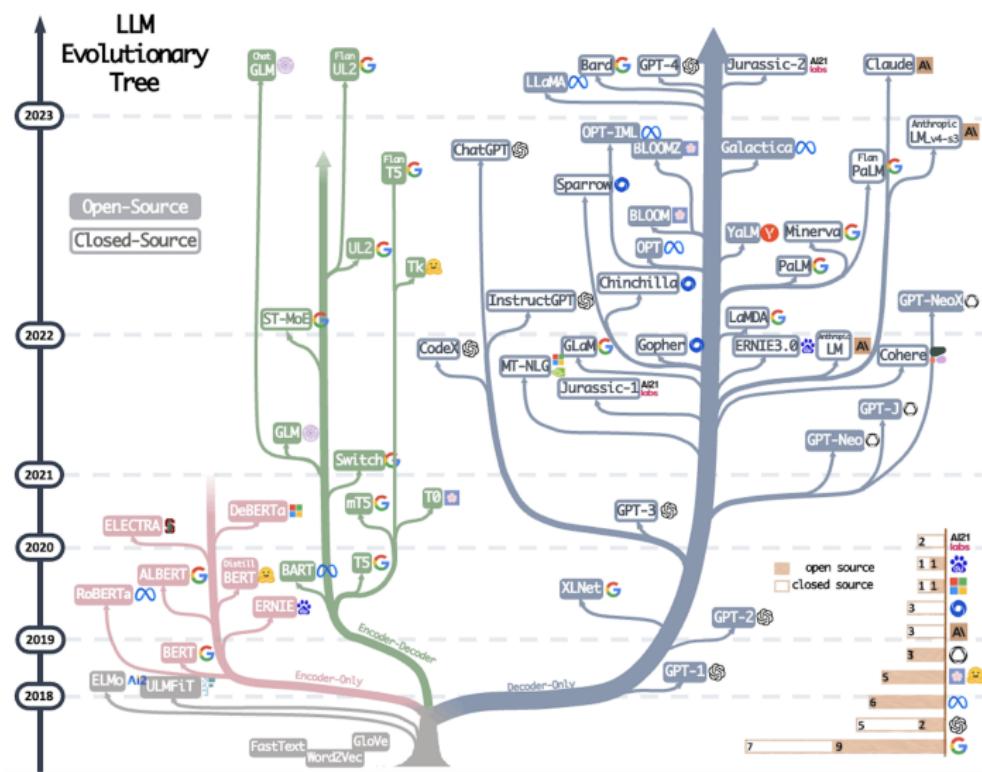
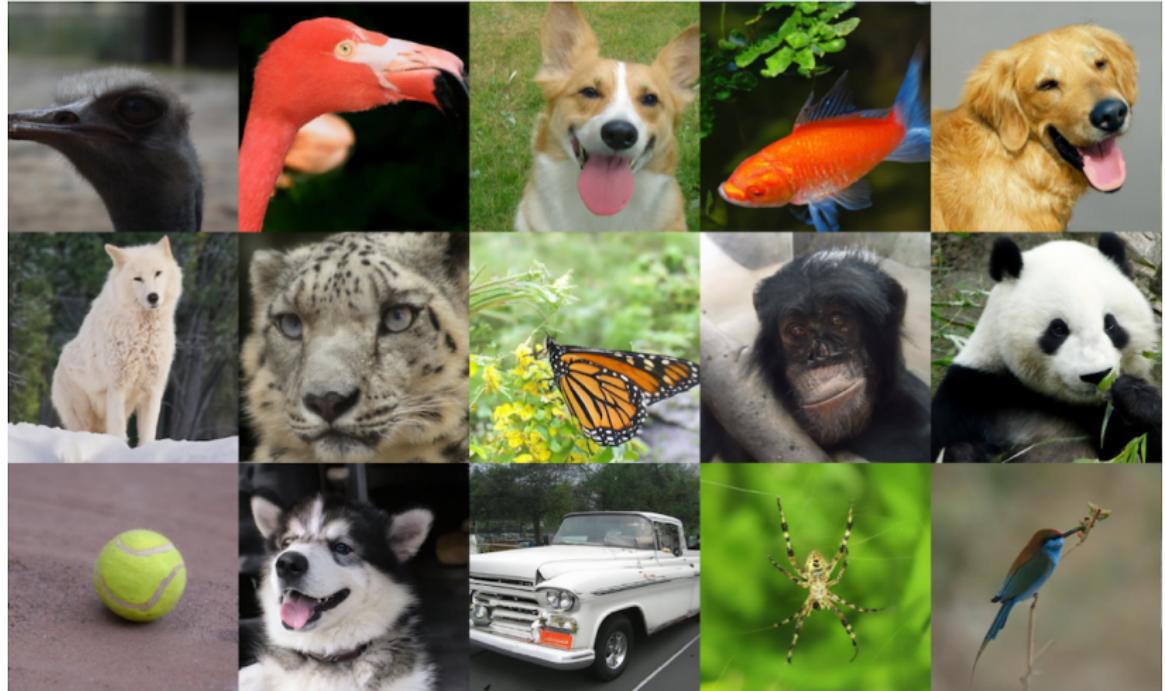


Image credit:

<https://blog.biocomm.ai/2023/05/14/open-source-proliferation-lm-evolutionary-tree/>

# Denoising Diffusion Probabilistic Model



# Midjourney – Impressive Text-to-Image Results



Image credit: <https://www.midjourney.com/explore>

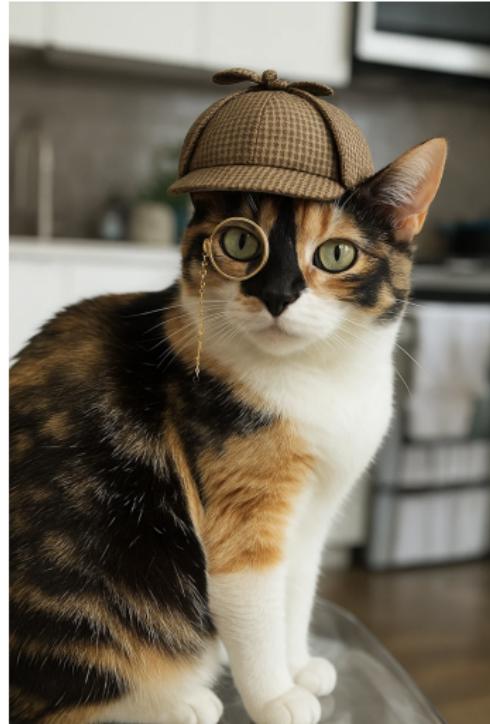
# Sora – Video Generation



*Image credit: <https://openai.com/index/sora>*

# GPT4o Image Editing

**Prompt:** Give this cat a detective hat and a monocle

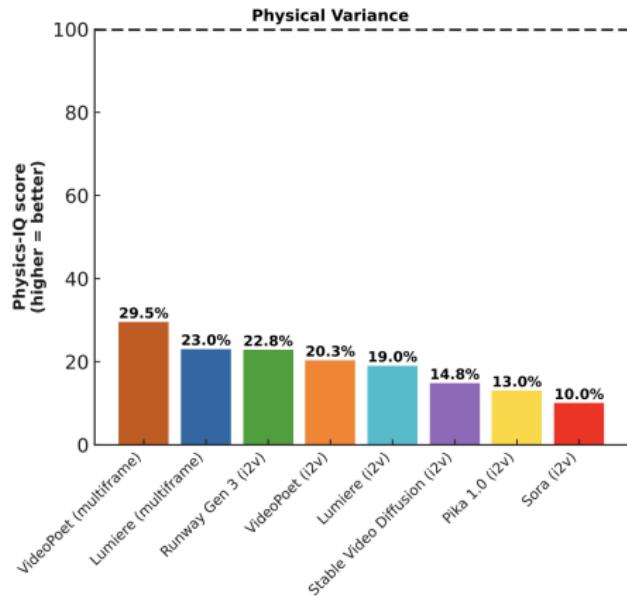


---

*Image credit: <https://openai.com/index/introducing-4o-image-generation/>*

# Open Problems in Generative Models

- ▶ Video generation
- ▶ 3D scene generation
- ▶ Understanding of physical processes
- ▶ Multimodal end-to-end models



# Outline

1. Generative Models Overview
2. Course Tricks
3. Problem Statement
4. Divergence Minimization Framework
5. Autoregressive Modeling

# Course Tricks I

## Log-Derivative Trick

Given a differentiable function  $p : \mathbb{R}^m \rightarrow \mathbb{R}$  (usually density function),

$$\nabla \log p(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \cdot \nabla p(\mathbf{x}).$$

## Jensen's Inequality

If  $\mathbf{x} \in \mathbb{R}^m$  is a continuous random variable with density  $p(\mathbf{x})$  and  $f : \mathbb{R}^m \rightarrow \mathbb{R}$  is convex, then

$$\mathbb{E}[f(\mathbf{x})] \geq f(\mathbb{E}[\mathbf{x}]).$$

## Monte Carlo Estimation

Let  $\mathbf{x} \in \mathbb{R}^m$  be a continuous random variable with density  $p(\mathbf{x})$ , and  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^d$  be any vector-valued function. Then,

$$\mathbb{E}_{p(\mathbf{x})}\mathbf{f}(\mathbf{x}) = \int p(\mathbf{x})\mathbf{f}(\mathbf{x})d\mathbf{x} \approx \frac{1}{n} \sum_{i=1}^n \mathbf{f}(\mathbf{x}_i), \quad \text{where } \mathbf{x}_i \sim p(\mathbf{x}).$$

## Course Tricks II

### Change of Variables Theorem (CoV)

Let  $\mathbf{x} \in \mathbb{R}^m$  be a random vector with density  $p(\mathbf{x})$ , and let  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  be a  $C^1$ -diffeomorphism ( $\mathbf{f}$  and  $\mathbf{f}^{-1}$  are continuously differentiable mappings). If  $\mathbf{z} = \mathbf{f}(\mathbf{x})$ , then

$$p(\mathbf{x}) = p(\mathbf{z}) |\det(\mathbf{J}_\mathbf{f})| = p(\mathbf{z}) \left| \det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(\mathbf{f}(\mathbf{x})) \left| \det \left( \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$
$$p(\mathbf{z}) = p(\mathbf{x}) |\det(\mathbf{J}_{\mathbf{f}^{-1}})| = p(\mathbf{x}) \left| \det \left( \frac{\partial \mathbf{x}}{\partial \mathbf{z}} \right) \right| = p(\mathbf{f}^{-1}(\mathbf{z})) \left| \det \left( \frac{\partial \mathbf{f}^{-1}(\mathbf{z})}{\partial \mathbf{z}} \right) \right|$$

### Proof (1D)

Assume  $f$  is monotonically increasing.

$$F_Y(y) = P(Y \leq y) = P(x \leq f^{-1}(y)) = F_X(f^{-1}(y))$$

$$p(y) = \frac{dF_Y(y)}{dy} = \frac{dF_X(f^{-1}(y))}{dy} = \frac{dF_X(x)}{dx} \frac{df^{-1}(y)}{dy} = p(x) \frac{df^{-1}(y)}{dy}$$

## Course Tricks III

### Law of the Unconscious Statistician (LOTUS)

Let  $\mathbf{x} \in \mathbb{R}^m$  be a continuous random variable with density  $p(\mathbf{x})$ , and let  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  be measurable. If  $\mathbf{y} = \mathbf{f}(\mathbf{x})$ , then

$$\mathbb{E}_{p(\mathbf{y})}\mathbf{g}(\mathbf{y}) = \int p(\mathbf{y})\mathbf{g}(\mathbf{y})d\mathbf{y} = \int p(\mathbf{x})\mathbf{g}(\mathbf{f}(\mathbf{x}))d\mathbf{x} = \mathbb{E}_{p(\mathbf{x})}\mathbf{g}(\mathbf{f}(\mathbf{x})).$$

### Dirac Delta Function

Any deterministic variable  $\mathbf{x}_0$  can be interpreted as a random variable with density  $p(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_0)$ .

$$\delta(\mathbf{x}) = \begin{cases} +\infty, & \mathbf{x} = \mathbf{x}_0 \\ 0, & \mathbf{x} \neq \mathbf{x}_0 \end{cases} \quad \int \delta(\mathbf{x})d\mathbf{x} = 1$$

$$\mathbb{E}_{p(\mathbf{x})}\mathbf{f}(\mathbf{x}) = \int \delta(\mathbf{x} - \mathbf{x}_0)\mathbf{f}(\mathbf{x})d\mathbf{x} = \mathbf{f}(\mathbf{x}_0)$$

# Outline

1. Generative Models Overview
2. Course Tricks
3. Problem Statement
4. Divergence Minimization Framework
5. Autoregressive Modeling

## Problem Statement

We're given **finite** number of i.i.d. samples  $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^m$  drawn from an **unknown** distribution  $p_{\text{data}}(\mathbf{x})$ .

## Objective

Our aim is to learn a distribution  $p_{\text{data}}(\mathbf{x})$  that allows us to:

- ▶ Generate new samples from  $p_{\text{data}}(\mathbf{x})$  (sample  $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ ) — **generation**.
- ▶ Evaluate  $p_{\text{data}}(\mathbf{x})$  on novel data (answering “How likely is an object  $\mathbf{x}$ ?”) — **density estimation**;

## Challenge

The data is high-dimensional and complex. For example, image datasets live in  $\mathbb{R}^{\text{width} \times \text{height} \times \text{channels}}$ . The curse of dimensionality makes accurately estimating  $p_{\text{data}}(\mathbf{x})$  infeasible.

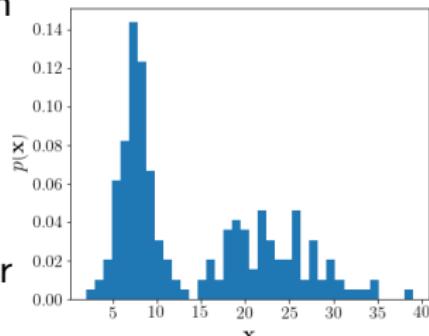
**Note:** here we use a strong assumption that our data is continuous (thus avoiding the domain of texts).

## Histogram as a Generative Model

Assume  $x \sim \text{Cat}(\pi)$ . The histogram model is fully characterized by

$$\hat{\pi}_k = \hat{\pi}(x = k) = \frac{\sum_{i=1}^n [x_i = k]}{n}.$$

**Curse of dimensionality:** The number of bins rises exponentially.



**MNIST example:**  $28 \times 28$  grayscale images, with each image  $\mathbf{x} = (x_1, \dots, x_{784})$ ,  $x_i \in \{0, 1\}$ :

$$p_{\text{data}}(\mathbf{x}) = p(x_1) \cdot p(x_2|x_1) \cdot \dots \cdot p(x_m|x_{m-1}, \dots, x_1).$$

A complete histogram would require  $2^{28 \times 28} - 1$  parameters for  $p_{\text{data}}(\mathbf{x})$ .

**Question:** How many parameters are required in these cases?

$$p_{\text{data}}(\mathbf{x}) = p(x_1) \cdot p(x_2) \cdot \dots \cdot p(x_m);$$

$$p_{\text{data}}(\mathbf{x}) = p(x_1) \cdot p(x_2|x_1) \cdot \dots \cdot p(x_m|x_{m-1}).$$

## Conditional Models

In practice, we're typically interested in learning conditional models (sampling from conditional distribution  $p_{\text{data}}(\mathbf{x}|\mathbf{y})$ ).

- ▶  $\mathbf{y} = \emptyset, \mathbf{x} = \text{image} \Rightarrow \text{unconditional image model}$
- ▶  $\mathbf{y} = \text{class label}, \mathbf{x} = \text{image} \Rightarrow \text{class-conditional image model}$
- ▶  $\mathbf{y} = \text{text prompt}, \mathbf{x} = \text{image} \Rightarrow \text{text-to-image model}$
- ▶  $\mathbf{y} = \text{image}, \mathbf{x} = \text{image} \Rightarrow \text{image-to-image model}$
- ▶  $\mathbf{y} = \text{image}, \mathbf{x} = \text{text} \Rightarrow \text{image-to-text (image captioning) model}$
- ▶  $\mathbf{y} = \text{English text}, \mathbf{x} = \text{Russian text} \Rightarrow \text{sequence-to-sequence model (machine translation) model}$
- ▶  $\mathbf{y} = \text{sound}, \mathbf{x} = \text{text} \Rightarrow \text{speech-to-text (automatic speech recognition) model}$
- ▶  $\mathbf{y} = \text{text}, \mathbf{x} = \text{sound} \Rightarrow \text{text-to-speech model}$

# Outline

1. Generative Models Overview
2. Course Tricks
3. Problem Statement
4. Divergence Minimization Framework
5. Autoregressive Modeling

## Divergences

- ▶ Let us fix a probabilistic model  $p_\theta(\mathbf{x})$  from a parametric family of distributions  $\{p(\mathbf{x}|\boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \Theta\}$ .
- ▶ Instead of searching among all possible distributions for the true  $p_{\text{data}}(\mathbf{x})$ , we seek a functional approximation  $p_\theta(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$ .

### What is a Divergence?

Let  $\mathcal{P}$  be the set of all probability distributions. A mapping  $D : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}$  is called a **divergence** if

- ▶  $D(\pi \| p) \geq 0$  for all  $\pi, p \in \mathcal{P}$
- ▶  $D(\pi \| p) = 0$  if and only if  $\pi \equiv p$

### Divergence Minimization Problem

$$\min_{\theta} D(p_{\text{data}} \| p_{\theta})$$

where  $p_{\text{data}}(\mathbf{x})$  is the true data distribution and  $p_\theta(\mathbf{x})$  is the model distribution.

# Forward KL vs Reverse KL (Kullback-Leibler Divergence)

## Forward KL

$$\text{KL}(p_{\text{data}} \| p_{\theta}) = \int p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\theta}(\mathbf{x})} d\mathbf{x} \rightarrow \min_{\theta}$$

## Reverse KL

$$\text{KL}(p_{\theta} \| p_{\text{data}}) = \int p_{\theta}(\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})} d\mathbf{x} \rightarrow \min_{\theta}$$

What's the practical distinction between these two objectives?

## Maximum Likelihood Estimation (MLE)

Let  $\{\mathbf{x}_i\}_{i=1}^n$  be i.i.d. observed samples.

$$\theta^* = \arg \max_{\theta} \prod_{i=1}^n p_{\theta}(\mathbf{x}_i) = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i).$$

## Forward KL vs Reverse KL: MLE as Forward KL

### Forward KL

$$\begin{aligned}\text{KL}(p_{\text{data}} \| p_{\theta}) &= \int p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\theta}(\mathbf{x})} d\mathbf{x} \\ &= \int p_{\text{data}}(\mathbf{x}) \log p_{\text{data}}(\mathbf{x}) d\mathbf{x} - \int p_{\text{data}}(\mathbf{x}) \log p_{\theta}(\mathbf{x}) d\mathbf{x} \\ &= -\mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\log p_{\theta}(\mathbf{x})] + \text{const} \\ &\approx -\frac{1}{n} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i) + \text{const} \rightarrow \min_{\theta}.\end{aligned}$$

Maximum likelihood estimation is thus equivalent to minimizing a Monte Carlo estimate of the forward KL divergence.

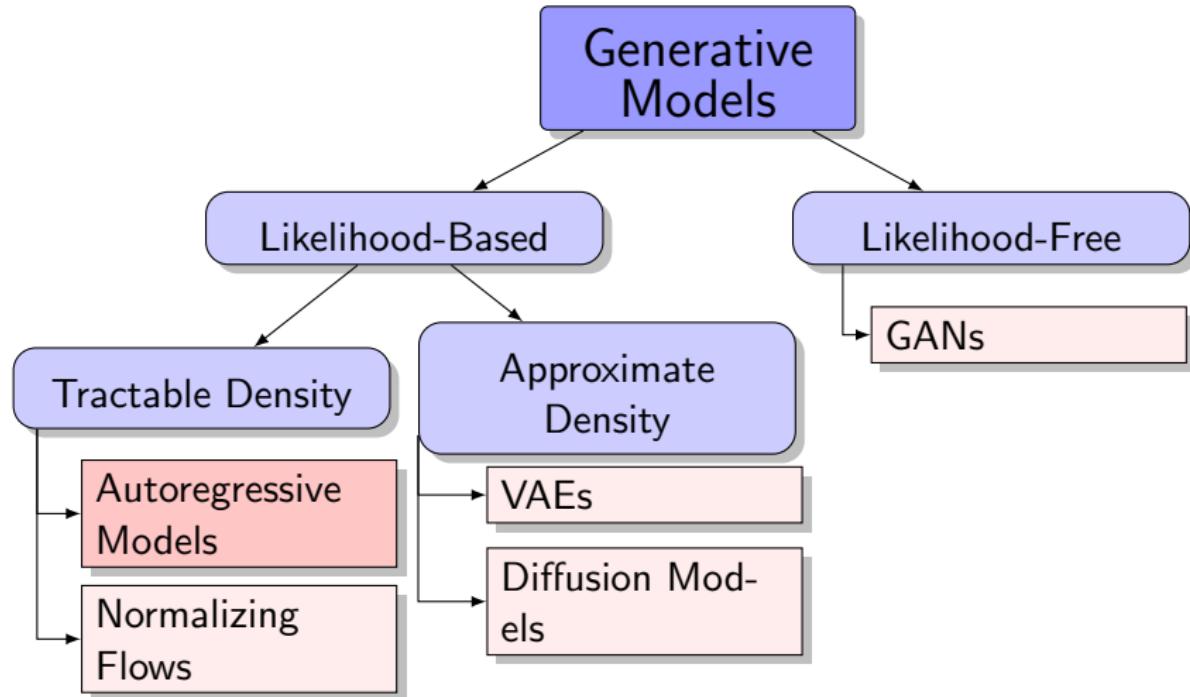
### Reverse KL

$$\begin{aligned}\text{KL}(p_{\theta} \| p_{\text{data}}) &= \int p_{\theta}(\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})} d\mathbf{x} \\ &= \mathbb{E}_{p_{\theta}(\mathbf{x})} [\log p_{\theta}(\mathbf{x}) - \log p_{\text{data}}(\mathbf{x})] \rightarrow \min_{\theta}\end{aligned}$$

# Outline

1. Generative Models Overview
2. Course Tricks
3. Problem Statement
4. Divergence Minimization Framework
5. Autoregressive Modeling

# Generative Models Zoo



# Autoregressive Modeling

## MLE Problem

$$\theta^* = \arg \max_{\theta} \prod_{i=1}^n p_{\theta}(\mathbf{x}_i) = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i)$$

- ▶ This maximization is typically solved via gradient-based optimization.
- ▶ Thus, efficient computation of both  $\log p_{\theta}(\mathbf{x})$  and its gradient  $\frac{\partial \log p_{\theta}(\mathbf{x})}{\partial \theta}$  is crucial.

## Likelihood as a Product of Conditionals

For  $\mathbf{x} = (x_1, \dots, x_m)$ ,  $\mathbf{x}_{1:j} = (x_1, \dots, x_j)$ ,

$$p_{\theta}(\mathbf{x}) = \prod_{j=1}^m p_{\theta}(x_j | \mathbf{x}_{1:j-1}); \quad \log p_{\theta}(\mathbf{x}) = \sum_{j=1}^m \log p_{\theta}(x_j | \mathbf{x}_{1:j-1})$$

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n \left[ \sum_{j=1}^m \log p_{\theta}(x_{ij} | \mathbf{x}_{i,1:j-1}) \right]$$

## Autoregressive Models

$$\log p_{\theta}(\mathbf{x}) = \sum_{j=1}^m \log p_{\theta}(x_j | \mathbf{x}_{1:j-1})$$

- ▶ Sampling is performed sequentially:
  - ▶ Sample  $\hat{x}_1 \sim p_{\theta}(x_1)$ ;
  - ▶ Sample  $\hat{x}_2 \sim p_{\theta}(x_2 | \hat{x}_1)$ ;
  - ▶ ...
  - ▶ Sample  $\hat{x}_m \sim p_{\theta}(x_m | \hat{\mathbf{x}}_{1:m-1})$ ;
  - ▶ The generated sample is  $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_m)$ .
- ▶ Each conditional  $p_{\theta}(x_j | \mathbf{x}_{1:j-1})$  can be modeled using a neural network.
- ▶ Modeling all conditionals separately isn't feasible. To address this, we share parameters across all conditionals.

# Autoregressive Models: MLP

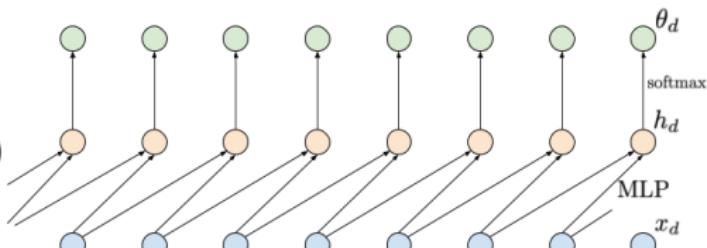
For large  $j$ , the conditional  $p_{\theta}(x_j | \mathbf{x}_{1:j-1})$  becomes intractable as the history  $\mathbf{x}_{1:j-1}$  grows variable-length.

## Markov Assumption

$$p_{\theta}(x_j | \mathbf{x}_{1:j-1}) = p_{\theta}(x_j | \mathbf{x}_{j-d:j-1}), \quad d \text{ is a fixed parameter.}$$

## Example

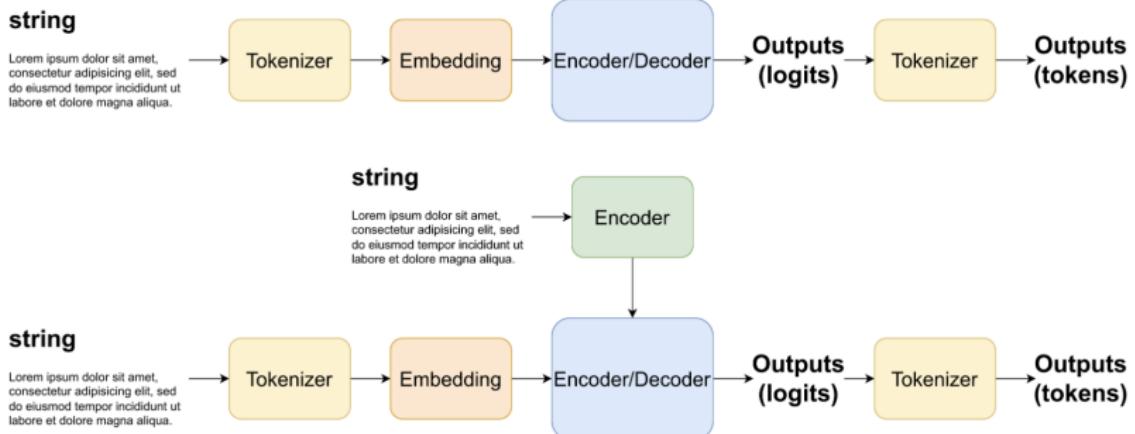
- ▶  $d = 2$
- ▶  $x_j \in \{0, 255\}$
- ▶  $\mathbf{h}_j = \text{MLP}_{\theta}(x_{j-1}, x_{j-2})$
- ▶  $\pi_j = \text{softmax}(\mathbf{h}_j)$
- ▶  $p_{\theta}(x_j | x_{j-1}, x_{j-2}) = \text{Cat}(\pi_j)$



Can we also model continuous-valued data, not just the discrete case?

# Autoregressive Models: LLM

$$p_{\theta}(x_j | \mathbf{x}_{1:j-1}) = p_{\theta}(x_j | \mathbf{x}_{j-d:j-1}), \quad d \text{ is the context window.}$$

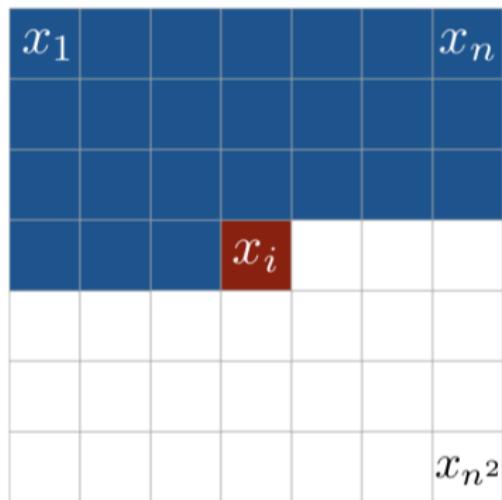


# Autoregressive Models for Images

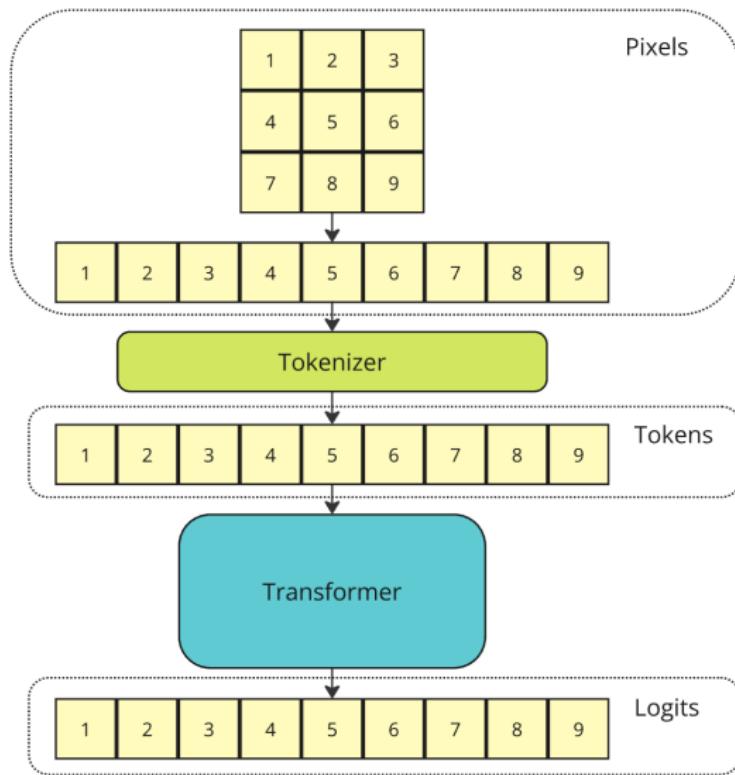
How do we model the distribution  $p_{\text{data}}(\mathbf{x})$  of natural images?

$$p_{\theta}(\mathbf{x}) = \prod_{j=1}^{\text{width} \times \text{height}} p_{\theta}(x_j | \mathbf{x}_{1:j-1})$$

- ▶ A pixel ordering must be selected; the raster scan is a standard choice.
- ▶ RGB channel dependencies can be modeled explicitly as well.



# Autoregressive Models: ImageGPT



## Summary

- ▶ Our target is to approximate the data distribution both for density estimation and for generation.
- ▶ The divergence minimization framework offers a principled way to learn distributions that match the data.
- ▶ Minimizing the forward KL divergence is equivalent to maximum likelihood estimation.
- ▶ Autoregressive models decompose the joint distribution as a product of conditionals.
- ▶ Autoregressive sampling is simple, but inherently sequential.
- ▶ Joint density evaluation multiplies all conditional probabilities  $p_{\theta}(x_j | \mathbf{x}_{1:j-1})$ .
- ▶ ImageGPT applies a transformer architecture to sequences of raster-ordered image pixels.

# Deep Generative Models

## Lecture 2

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

## 6. Normalizing Flows (NF)

## 7. NF Examples

Linear Normalizing Flows

Gaussian Autoregressive NF

Coupling Layer (RealNVP)

## 8. Latent Variable Models (LVM)

# Outline

## 6. Normalizing Flows (NF)

## 7. NF Examples

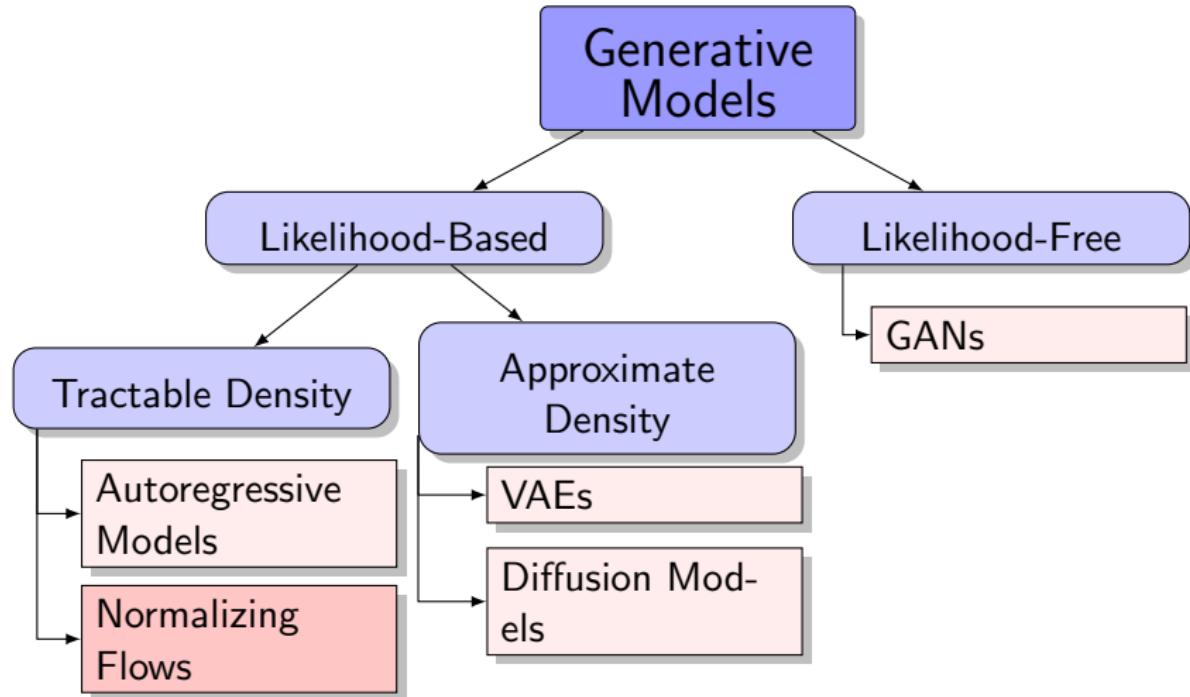
Linear Normalizing Flows

Gaussian Autoregressive NF

Coupling Layer (RealNVP)

## 8. Latent Variable Models (LVM)

# Generative Models Zoo



# Normalizing Flows: Prerequisites

## Jacobian Matrix

Let  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  be a differentiable function.

$$\mathbf{z} = \mathbf{f}(\mathbf{x}), \quad \mathbf{J} = \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_m}{\partial x_1} & \cdots & \frac{\partial z_m}{\partial x_m} \end{pmatrix} \in \mathbb{R}^{m \times m}$$

## Change of Variables Theorem (CoV)

Let  $\mathbf{x} \in \mathbb{R}^m$  be a random vector with density  $p(\mathbf{x})$ , and let  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  be a  $C^1$ -diffeomorphism ( $\mathbf{f}$  and  $\mathbf{f}^{-1}$  are continuously differentiable mappings). If  $\mathbf{z} = \mathbf{f}(\mathbf{x})$ , then

$$p(\mathbf{x}) = p(\mathbf{z}) |\det(\mathbf{J}_f)| = p(\mathbf{z}) \left| \det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(\mathbf{f}(\mathbf{x})) \left| \det \left( \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$
$$p(\mathbf{z}) = p(\mathbf{x}) |\det(\mathbf{J}_{f^{-1}})| = p(\mathbf{x}) \left| \det \left( \frac{\partial \mathbf{x}}{\partial \mathbf{z}} \right) \right| = p(\mathbf{f}^{-1}(\mathbf{z})) \left| \det \left( \frac{\partial \mathbf{f}^{-1}(\mathbf{z})}{\partial \mathbf{z}} \right) \right|$$

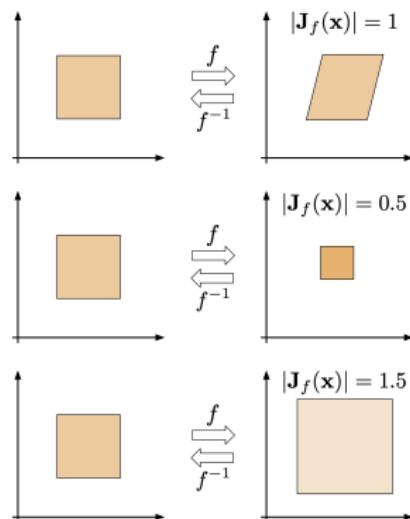
# Jacobian Determinant

## Inverse Function Theorem

If the function  $\mathbf{f}$  is invertible and its Jacobian is continuous and non-singular, then

$$\mathbf{J}_{\mathbf{f}^{-1}} = \mathbf{J}_{\mathbf{f}}^{-1}; \quad |\det(\mathbf{J}_{\mathbf{f}^{-1}})| = \frac{1}{|\det(\mathbf{J}_{\mathbf{f}})|}$$

- ▶  $\mathbf{x}$  and  $\mathbf{z}$  reside in the same space ( $\mathbb{R}^m$ ).
- ▶  $\mathbf{f}_\theta(\mathbf{x})$  is a parameterized transformation.
- ▶ The determinant of the Jacobian  $\mathbf{J} = \frac{\partial \mathbf{f}_\theta(\mathbf{x})}{\partial \mathbf{x}}$  quantifies how the volume is changed by the transformation.

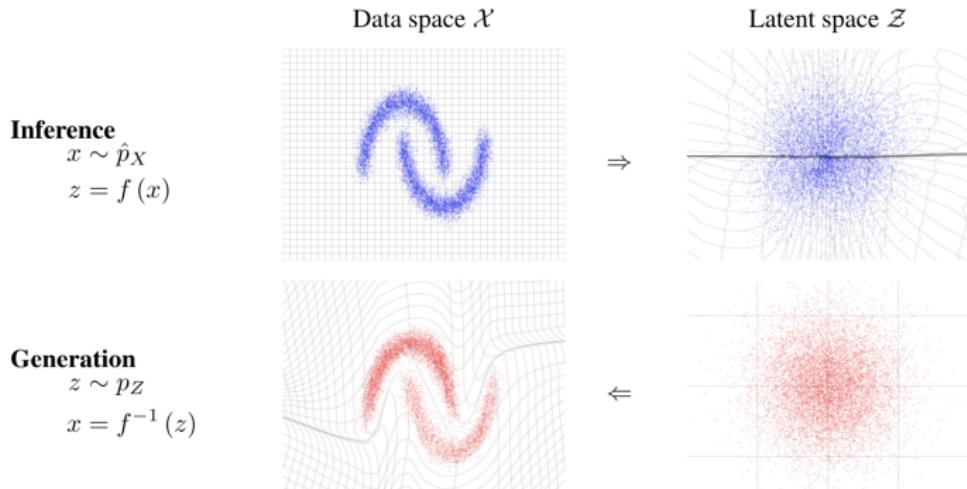


# Fitting Normalizing Flows

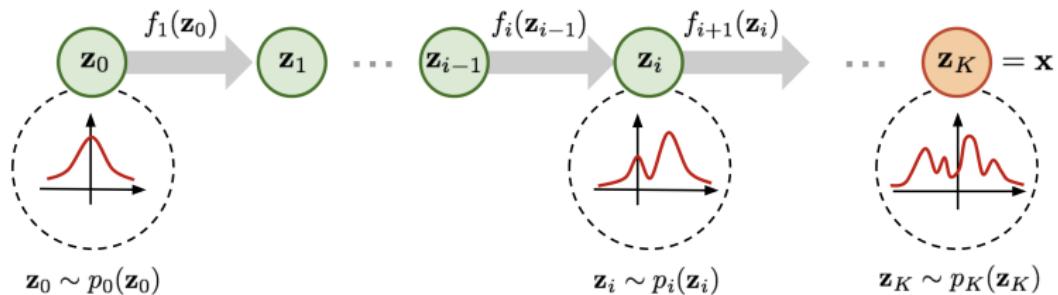
## MLE Problem

$$p_{\theta}(\mathbf{x}) = p(\mathbf{z}) \left| \det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(\mathbf{f}_{\theta}(\mathbf{x})) \left| \det \left( \frac{\partial \mathbf{f}_{\theta}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

$$\log p_{\theta}(\mathbf{x}) = \log p(\mathbf{f}_{\theta}(\mathbf{x})) + \log |\det(\mathbf{J}_f)| \rightarrow \max_{\theta}$$



# Composition of Normalizing Flows



## Theorem

If every  $\{\mathbf{f}_k\}_{k=1}^K$  satisfies the conditions of the change-of-variables theorem, then the composition  $\mathbf{f}(\mathbf{x}) = \mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x})$  also satisfies them.

$$\begin{aligned} p_{\theta}(\mathbf{x}) &= p(\mathbf{f}(\mathbf{x})) \left| \det \left( \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = p(\mathbf{f}(\mathbf{x})) \left| \det \left( \frac{\partial \mathbf{f}_K}{\partial \mathbf{f}_{K-1}} \cdots \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}} \right) \right| = \\ &= p(\mathbf{f}(\mathbf{x})) \prod_{k=1}^K \left| \det \left( \frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}} \right) \right| = p(\mathbf{f}(\mathbf{x})) \prod_{k=1}^K |\det(\mathbf{J}_{\mathbf{f}_k})| \end{aligned}$$

# Normalizing Flows (NF)

$$\log p_{\theta}(\mathbf{x}) = \log p(\mathbf{f}_{\theta}(\mathbf{x})) + \log |\det(\mathbf{J}_f)|$$

## Definition

A normalizing flow is a  $C^1$ -diffeomorphism that transforms data  $\mathbf{x}$  to noise  $\mathbf{z}$ .

- ▶ **Normalizing** refers to mapping samples from  $p_{\text{data}}(\mathbf{x})$  to a base distribution  $p(\mathbf{z})$ .
- ▶ **Flow** describes the sequence of transformations that maps samples from  $p(\mathbf{z})$  to the target, more complex distribution.

$$\mathbf{z} = \mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x}); \quad \mathbf{x} = \mathbf{f}_1^{-1} \circ \dots \circ \mathbf{f}_K^{-1}(\mathbf{z})$$

## Log-Likelihood

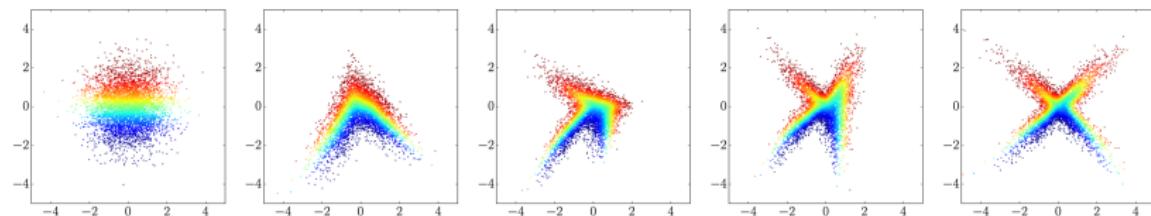
$$\log p_{\theta}(\mathbf{x}) = \log p(\mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x})) + \sum_{k=1}^K \log |\det(\mathbf{J}_{\mathbf{f}_k})|$$

where  $\mathbf{J}_{\mathbf{f}_k} = \frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}}$ .

**Note:** Here we consider only **continuous** random variables.

# Normalizing Flows

## Example: 4-Step NF



## NF Log-Likelihood

$$\log p_{\theta}(\mathbf{x}) = \log p(\mathbf{f}_{\theta}(\mathbf{x})) + \log |\det(\mathbf{J}_f)|$$

What's the computational complexity of evaluating this determinant?

## Requirements

- ▶ Efficient computation of the Jacobian  $\mathbf{J}_f = \frac{\partial \mathbf{f}_{\theta}(\mathbf{x})}{\partial \mathbf{x}}$
- ▶ Efficient inversion of the transformation  $\mathbf{f}_{\theta}(\mathbf{x})$

# Outline

## 6. Normalizing Flows (NF)

## 7. NF Examples

Linear Normalizing Flows

Gaussian Autoregressive NF

Coupling Layer (RealNVP)

## 8. Latent Variable Models (LVM)

# Outline

## 6. Normalizing Flows (NF)

## 7. NF Examples

Linear Normalizing Flows

Gaussian Autoregressive NF

Coupling Layer (RealNVP)

## 8. Latent Variable Models (LVM)

# Jacobian Structure

## Normalizing Flows Log-Likelihood

$$\log p_{\theta}(\mathbf{x}) = \log p(\mathbf{f}_{\theta}(\mathbf{x})) + \log \left| \det \left( \frac{\partial \mathbf{f}_{\theta}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

The principal computational challenge is evaluating the Jacobian determinant.

### What is $\det(\mathbf{J})$ in These Cases?

Consider a linear layer  $\mathbf{z} = \mathbf{W}\mathbf{x}$ ,  $\mathbf{W} \in \mathbb{R}^{m \times m}$ .

1.  $\mathbf{z}$  is a permutation of  $\mathbf{x}$ .
2.  $z_j$  depends only on  $x_j$ .

$$\log \left| \det \left( \frac{\partial \mathbf{f}_{\theta}(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = \log \left| \prod_{j=1}^m \frac{\partial f_{j,\theta}(x_j)}{\partial x_j} \right| = \sum_{j=1}^m \log \left| \frac{\partial f_{j,\theta}(x_j)}{\partial x_j} \right|$$

3.  $z_j$  depends only on  $\mathbf{x}_{1:j}$  (autoregressive dependency).

# Linear Normalizing Flows

$$\mathbf{z} = \mathbf{f}_{\theta}(\mathbf{x}) = \mathbf{W}\mathbf{x}, \quad \mathbf{W} \in \mathbb{R}^{m \times m}, \quad \theta = \mathbf{W}, \quad \mathbf{J}_f = \mathbf{W}^T$$

In general, matrix inversion has computational complexity  $O(m^3)$ .

## Invertibility

- ▶ Diagonal matrix:  $O(m)$ .
- ▶ Triangular matrix:  $O(m^2)$ .
- ▶ Directly parameterizing all invertible matrices in a continuous way is infeasible  
(there is not surjective function from  $\mathbb{R}^{m^2}$  to the set of all invertible matrices of size  $m \times m$ ).

# Linear Normalizing Flows

$$\mathbf{z} = \mathbf{f}_{\theta}(\mathbf{x}) = \mathbf{W}\mathbf{x}, \quad \mathbf{W} \in \mathbb{R}^{m \times m}, \quad \theta = \mathbf{W}, \quad \mathbf{J}_f = \mathbf{W}^T$$

## Matrix Decompositions

- ▶ **LU Decomposition:**

$$\mathbf{W} = \mathbf{P}\mathbf{L}\mathbf{U},$$

where  $\mathbf{P}$  is a permutation matrix,  $\mathbf{L}$  is lower triangular with positive diagonal, and  $\mathbf{U}$  is upper triangular with positive diagonal.

- ▶ **QR Decomposition:**

$$\mathbf{W} = \mathbf{Q}\mathbf{R},$$

where  $\mathbf{Q}$  is orthogonal, and  $\mathbf{R}$  is upper triangular with positive diagonal.

Decomposition is performed only at initialization; the decomposed matrices ( $\mathbf{P}, \mathbf{L}, \mathbf{U}$  or  $\mathbf{Q}, \mathbf{R}$ ) are optimized during training.

# Outline

## 6. Normalizing Flows (NF)

## 7. NF Examples

Linear Normalizing Flows

Gaussian Autoregressive NF

Coupling Layer (RealNVP)

## 8. Latent Variable Models (LVM)

# Gaussian Autoregressive Model

Consider the autoregressive model:

$$p_{\theta}(\mathbf{x}) = \prod_{j=1}^m p_{\theta}(x_j | \mathbf{x}_{1:j-1}), \quad p_{\theta}(x_j | \mathbf{x}_{1:j-1}) = \mathcal{N}(\mu_{j,\theta}(\mathbf{x}_{1:j-1}), \sigma_{j,\theta}^2(\mathbf{x}_{1:j-1}))$$

## Sampling

$$x_j = \sigma_{j,\theta}(\mathbf{x}_{1:j-1}) \cdot z_j + \mu_{j,\theta}(\mathbf{x}_{1:j-1}), \quad z_j \sim \mathcal{N}(0, 1)$$

## Inverse Transformation

$$z_j = \frac{x_j - \mu_{j,\theta}(\mathbf{x}_{1:j-1})}{\sigma_{j,\theta}(\mathbf{x}_{1:j-1})}$$

- ▶ This gives an  **$C^1$ -diffeomorphism** from  $p(\mathbf{z})$  to  $p_{\theta}(\mathbf{x})$  (assume that  $\sigma_j \neq 0$ ).
- ▶ This model is called an autoregressive (AR) NF with base distribution  $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ .
- ▶ The Jacobian matrix of this transformation is triangular.

# Gaussian Autoregressive NF

Forward Transformation:  $\mathbf{f}_\theta(\mathbf{x})$

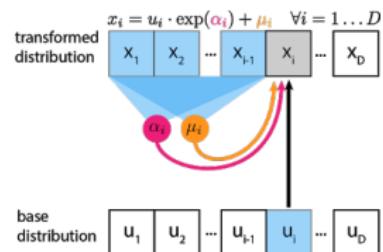
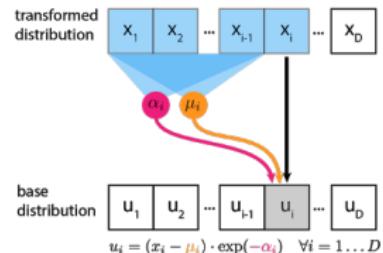
$$\mathbf{z} = \mathbf{f}_\theta(\mathbf{x})$$

$$z_j = \frac{x_j - \mu_{j,\theta}(\mathbf{x}_{1:j-1})}{\sigma_{j,\theta}(\mathbf{x}_{1:j-1})}$$

Inverse Transformation:  $\mathbf{f}_\theta^{-1}(\mathbf{z})$

$$\mathbf{x} = \mathbf{f}_\theta^{-1}(\mathbf{z})$$

$$x_j = \sigma_{j,\theta}(\mathbf{x}_{1:j-1}) \cdot z_j + \mu_{j,\theta}(\mathbf{x}_{1:j-1})$$



- ▶ Sampling must be done sequentially, but density estimation can be parallelized.
- ▶ The forward KL divergence is a natural objective for training.

# Outline

## 6. Normalizing Flows (NF)

## 7. NF Examples

Linear Normalizing Flows

Gaussian Autoregressive NF

Coupling Layer (RealNVP)

## 8. Latent Variable Models (LVM)

# RealNVP

Split  $\mathbf{x}$  and  $\mathbf{z}$  into two parts:

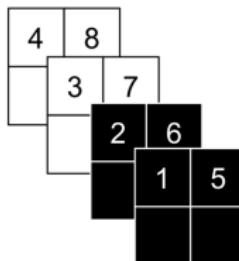
$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2] = [\mathbf{x}_{1:d}, \mathbf{x}_{d+1:m}]; \quad \mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2] = [\mathbf{z}_{1:d}, \mathbf{z}_{d+1:m}]$$

## Coupling Layer

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1 \\ \mathbf{x}_2 = \mathbf{z}_2 \odot \sigma_\theta(\mathbf{z}_1) + \mu_\theta(\mathbf{z}_1) \end{cases}$$

$$\begin{cases} \mathbf{z}_1 = \mathbf{x}_1 \\ \mathbf{z}_2 = (\mathbf{x}_2 - \mu_\theta(\mathbf{x}_1)) \odot \frac{1}{\sigma_\theta(\mathbf{x}_1)} \end{cases}$$

## Image Partitioning



- ▶ Checkerboard ordering corresponds to masking.
- ▶ Channelwise ordering relies on splitting.

# RealNVP

## Coupling Layer

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1 \\ \mathbf{x}_2 = \mathbf{z}_2 \odot \sigma_{\theta}(\mathbf{z}_1) + \mu_{\theta}(\mathbf{z}_1) \end{cases} \quad \begin{cases} \mathbf{z}_1 = \mathbf{x}_1 \\ \mathbf{z}_2 = (\mathbf{x}_2 - \mu_{\theta}(\mathbf{x}_1)) \odot \frac{1}{\sigma_{\theta}(\mathbf{x}_1)} \end{cases}$$

In both training and sampling, only a single forward pass is needed!  
Jacobian

$$\det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) = \det \begin{pmatrix} \mathbf{I}_d & \mathbf{0}_{d \times m-d} \\ \frac{\partial \mathbf{z}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{z}_2}{\partial \mathbf{x}_2} \end{pmatrix} = \prod_{j=1}^{m-d} \frac{1}{\sigma_{j,\theta}(\mathbf{x}_1)}$$

## Gaussian AR NF

$$\mathbf{x} = \mathbf{f}_{\theta}^{-1}(\mathbf{z}) \quad \Rightarrow \quad \mathbf{x}_j = \sigma_{j,\theta}(\mathbf{x}_{1:j-1}) \cdot \mathbf{z}_j + \mu_{j,\theta}(\mathbf{x}_{1:j-1})$$

$$\mathbf{z} = \mathbf{f}_{\theta}(\mathbf{x}) \quad \Rightarrow \quad \mathbf{z}_j = (\mathbf{x}_j - \mu_{j,\theta}(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\sigma_{j,\theta}(\mathbf{x}_{1:j-1})}.$$

How can the RealNVP layer be derived as a special instance of the  
Gaussian autoregressive NF?

# Outline

## 6. Normalizing Flows (NF)

## 7. NF Examples

Linear Normalizing Flows

Gaussian Autoregressive NF

Coupling Layer (RealNVP)

## 8. Latent Variable Models (LVM)

# Bayesian Framework

## Bayes' Theorem

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int p(x|\theta)p(\theta)d\theta}$$

- ▶  $x$ : observed variables;
- ▶  $\theta$ : unknown latent variables/parameters;
- ▶  $p_\theta(x) = p(x|\theta)$ : likelihood;
- ▶  $p(x) = \int p(x|\theta)p(\theta)d\theta$ : evidence;
- ▶  $p(\theta)$ : prior distribution;
- ▶  $p(\theta|x)$ : posterior distribution.

## Interpretation

- ▶ We begin with unknown variables  $\theta$  and a prior belief  $p(\theta)$ .
- ▶ Once data  $x$  is observed, the posterior  $p(\theta|x)$  incorporates both prior beliefs and evidence from the data.

## Bayesian Framework

Consider the case where the unobserved variables  $\theta$  are model parameters (i.e.,  $\theta$  are random variables).

- ▶  $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ : observed samples;
- ▶  $p(\theta)$ : prior distribution.

## Posterior Distribution

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} = \frac{p(\mathbf{X}|\theta)p(\theta)}{\int p(\mathbf{X}|\theta)p(\theta)d\theta}$$

If the evidence  $p(\mathbf{X})$  is intractable (due to high-dimensional integration), the posterior cannot be computed exactly.

## Maximum a Posteriori (MAP) Estimation

$$\theta^* = \arg \max_{\theta} p(\theta|\mathbf{X}) = \arg \max_{\theta} (\log p(\mathbf{X}|\theta) + \log p(\theta))$$

# Latent Variable Models (LVM)

## Maximum Likelihood Estimation (MLE) Problem

$$\boldsymbol{\theta}^* = \arg \max_{\theta} p_{\theta}(\mathbf{X}) = \arg \max_{\theta} \prod_{i=1}^n p_{\theta}(\mathbf{x}_i) = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i).$$

The distribution  $p_{\theta}(\mathbf{x})$  can be highly complex and often intractable (just like the true data distribution  $p_{\text{data}}(\mathbf{x})$ ).

## Extended Probabilistic Model

Introduce a latent variable  $\mathbf{z}$  for each observed sample  $\mathbf{x}$ :

$$p_{\theta}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z}); \quad \log p_{\theta}(\mathbf{x}, \mathbf{z}) = \log p_{\theta}(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}).$$

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

## Motivation

Both  $p_{\theta}(\mathbf{x}|\mathbf{z})$  and  $p(\mathbf{z})$  are usually much simpler than  $p_{\theta}(\mathbf{x})$ .

## Summary

- ▶ The CoV theorem provides a method for computing a random variable's density under an invertible transformation.
- ▶ Normalizing flows transform a simple base distribution into a complex one via a sequence of invertible mappings, each with efficient Jacobian determinants.
- ▶ Linear NFs capture invertible matrices by using matrix decompositions.
- ▶ Gaussian autoregressive NFs are AR models with triangular Jacobians.
- ▶ The RealNVP coupling layer provides an efficient normalizing flow (a special case of AR NF), supporting fast inference and sampling.
- ▶ The Bayesian framework generalizes nearly all standard machine learning methods.

# Deep Generative Models

## Lecture 3

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

9. Latent Variable Models (LVM) (continued)

10. Variational Evidence Lower Bound (ELBO)

11. Amortized Inference

12. ELBO Gradients, Reparametrization Trick

13. Variational Autoencoder (VAE)

# Outline

9. Latent Variable Models (LVM) (continued)

10. Variational Evidence Lower Bound (ELBO)

11. Amortized Inference

12. ELBO Gradients, Reparametrization Trick

13. Variational Autoencoder (VAE)

# Latent Variable Models (LVM)

## Maximum Likelihood Estimation (MLE) Problem

$$\boldsymbol{\theta}^* = \arg \max_{\theta} p_{\theta}(\mathbf{X}) = \arg \max_{\theta} \prod_{i=1}^n p_{\theta}(\mathbf{x}_i) = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i).$$

The distribution  $p_{\theta}(\mathbf{x})$  can be highly complex and often intractable (just like the true data distribution  $p_{\text{data}}(\mathbf{x})$ ).

## Extended Probabilistic Model

Introduce a latent variable  $\mathbf{z}$  for each observed sample  $\mathbf{x}$ :

$$p_{\theta}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z}); \quad \log p_{\theta}(\mathbf{x}, \mathbf{z}) = \log p_{\theta}(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}).$$

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

## Motivation

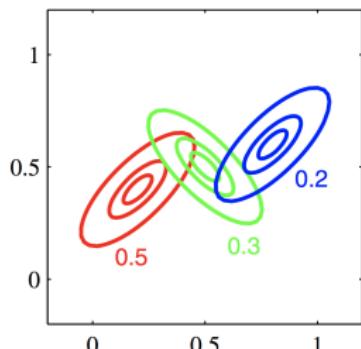
Both  $p_{\theta}(\mathbf{x}|\mathbf{z})$  and  $p(\mathbf{z})$  are usually much simpler than  $p_{\theta}(\mathbf{x})$ .

# Latent Variable Models (LVM)

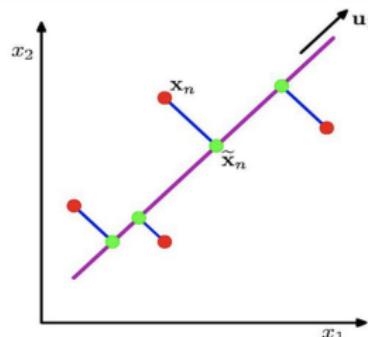
$$\log p_{\theta}(\mathbf{x}) = \log \int p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \rightarrow \max_{\theta}$$

## Examples

Mixture of Gaussians



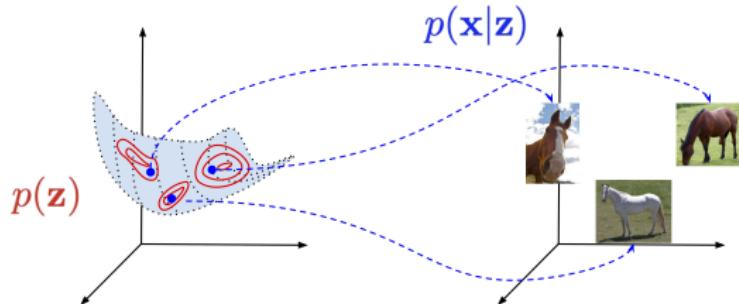
PCA Model



- ▶  $p_{\theta}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\Sigma}_{\mathbf{z}})$
- ▶  $p(\mathbf{z}) = \text{Cat}(\boldsymbol{\pi})$
- ▶  $p_{\theta}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2 \mathbf{I})$
- ▶  $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$

# MLE for LVM

$$\sum_{i=1}^n \log p_\theta(\mathbf{x}_i) = \sum_{i=1}^n \log \int p_\theta(\mathbf{x}_i|\mathbf{z}_i)p(\mathbf{z}_i)d\mathbf{z}_i \rightarrow \max_{\theta}.$$



## Naive Monte Carlo Estimation

$$\log p_\theta(\mathbf{x}) = \log \mathbb{E}_{p(\mathbf{z})} p_\theta(\mathbf{x}|\mathbf{z}) \geq \mathbb{E}_{p(\mathbf{z})} \log p_\theta(\mathbf{x}|\mathbf{z}) \approx \frac{1}{K} \sum_{k=1}^K \log p_\theta(\mathbf{x}|\mathbf{z}_k),$$

where  $\mathbf{z}_k \sim p(\mathbf{z})$ .

**Challenge:** As the dimensionality of  $\mathbf{z}$  increases, the number of samples needed to adequately cover the latent space grows exponentially.

# Outline

9. Latent Variable Models (LVM) (continued)

10. Variational Evidence Lower Bound (ELBO)

11. Amortized Inference

12. ELBO Gradients, Reparametrization Trick

13. Variational Autoencoder (VAE)

# ELBO Derivation I

## Inequality Derivation

$$\begin{aligned}\log p_{\theta}(\mathbf{x}) &= \log \int p_{\theta}(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \log \int \frac{q(\mathbf{z})}{q(\mathbf{z})} p_{\theta}(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \\ &= \log \mathbb{E}_q \left[ \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right] \geq \mathbb{E}_q \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} = \mathcal{L}_{q,\theta}(\mathbf{x})\end{aligned}$$

- ▶ Here,  $q(\mathbf{z})$  is any distribution such that  $\int q(\mathbf{z}) d\mathbf{z} = 1$ .
- ▶ We assume that  $\text{supp}(q(\mathbf{z})) = \text{supp}(p_{\theta}(\mathbf{z}|\mathbf{x})) = \mathbb{R}^d$ .

## Variational Evidence Lower Bound (ELBO)

$$\mathcal{L}_{q,\theta}(\mathbf{x}) = \mathbb{E}_q \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \leq \log p_{\theta}(\mathbf{x})$$

This inequality holds for any choice of  $q(\mathbf{z})$ .

## ELBO Derivation II

$$p_{\theta}(z|x) = \frac{p_{\theta}(x,z)}{p_{\theta}(x)}$$

### Equality Derivation

$$\begin{aligned}\mathcal{L}_{q,\theta}(x) &= \int q(z) \log \frac{p_{\theta}(x,z)}{q(z)} dz = \\ &= \int q(z) \log \frac{p_{\theta}(z|x)p_{\theta}(x)}{q(z)} dz = \\ &= \int q(z) \log p_{\theta}(x) dz + \int q(z) \log \frac{p_{\theta}(z|x)}{q(z)} dz = \\ &= \log p_{\theta}(x) - \text{KL}(q(z) \| p_{\theta}(z|x))\end{aligned}$$

### Variational Decomposition

$$\log p_{\theta}(x) = \mathcal{L}_{q,\theta}(x) + \text{KL}(q(z) \| p_{\theta}(z|x)) \geq \mathcal{L}_{q,\theta}(x).$$

Here,  $\text{KL}(q(z) \| p_{\theta}(z|x)) \geq 0$ .

## Variational Evidence Lower Bound (ELBO)

$$\begin{aligned}\mathcal{L}_{q,\theta}(\mathbf{x}) &= \int q(\mathbf{z}) \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} d\mathbf{z} \\ &= \int q(\mathbf{z}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} + \int q(\mathbf{z}) \log \frac{p(\mathbf{z})}{q(\mathbf{z})} d\mathbf{z} \\ &= \mathbb{E}_q \log p_{\theta}(\mathbf{x}|\mathbf{z}) - \text{KL}(q(\mathbf{z}) \| p(\mathbf{z}))\end{aligned}$$

## Log-Likelihood Decomposition

$$\begin{aligned}\log p_{\theta}(\mathbf{x}) &= \mathcal{L}_{q,\theta}(\mathbf{x}) + \text{KL}(q(\mathbf{z}) \| p_{\theta}(\mathbf{z}|\mathbf{x})) = \\ &= \mathbb{E}_q \log p_{\theta}(\mathbf{x}|\mathbf{z}) - \text{KL}(q(\mathbf{z}) \| p(\mathbf{z})) + \text{KL}(q(\mathbf{z}) \| p_{\theta}(\mathbf{z}|\mathbf{x})).\end{aligned}$$

- ▶ Instead of maximizing the likelihood, maximize the ELBO:

$$\max_{\theta} p_{\theta}(\mathbf{x}) \rightarrow \max_{q,\theta} \mathcal{L}_{q,\theta}(\mathbf{x})$$

- ▶ Maximizing the ELBO with respect to the **variational** distribution  $q$  is equivalent to minimizing the KL divergence:

$$\arg \max_q \mathcal{L}_{q,\theta}(\mathbf{x}) \equiv \arg \min_q \text{KL}(q(\mathbf{z}) \| p_{\theta}(\mathbf{z}|\mathbf{x})).$$

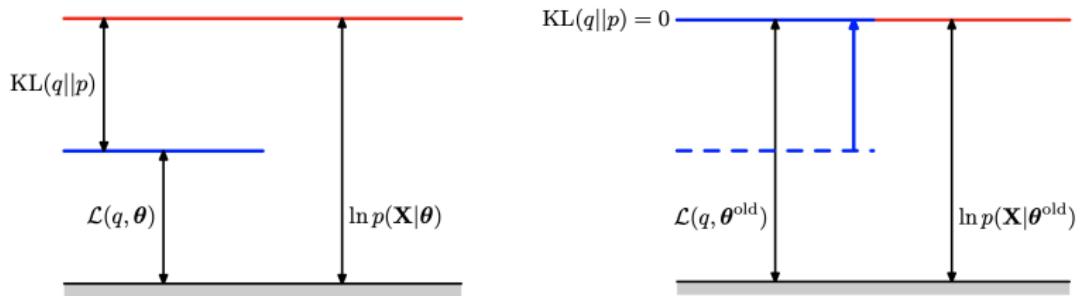
# Variational Posterior

$$\mathcal{L}_{q,\theta}(\mathbf{x}) = \mathbb{E}_q \log p_{\theta}(\mathbf{x}|\mathbf{z}) - \text{KL}(q(\mathbf{z})\|p(\mathbf{z})) \rightarrow \max_{q,\theta}.$$

What is the optimal distribution  $q^*(\mathbf{z})$  given fixed  $\theta^*$ ?

$$\begin{aligned} q^*(\mathbf{z}) &= \arg \max_q \mathcal{L}_{q,\theta^*}(\mathbf{x}) = \\ &= \arg \min_q \text{KL}(q(\mathbf{z})\|p_{\theta^*}(\mathbf{z}|\mathbf{x})) = p_{\theta^*}(\mathbf{z}|\mathbf{x}). \end{aligned}$$

Here we got the intuition about  $q(\mathbf{z})$  – it estimates the posterior  $p_{\theta^*}(\mathbf{z}|\mathbf{x})$ .



# Outline

9. Latent Variable Models (LVM) (continued)

10. Variational Evidence Lower Bound (ELBO)

11. Amortized Inference

12. ELBO Gradients, Reparametrization Trick

13. Variational Autoencoder (VAE)

# Parametric Variable Posterior

## Variational Posterior

$$q(\mathbf{z}) = \arg \max_q \mathcal{L}_{q,\theta^*}(\mathbf{x}) = \arg \min_q \text{KL}(q\|p) = p_{\theta^*}(\mathbf{z}|\mathbf{x}).$$

- ▶  $p_{\theta^*}(\mathbf{z}|\mathbf{x})$  may be **intractable**;
- ▶  $q(\mathbf{z})$  is individual for each data point  $\mathbf{x}$ .

## Amortized Variational Inference

We restrict the family of possible distributions  $q(\mathbf{z})$  to a parametric class  $q_\phi(\mathbf{z}|\mathbf{x})$ , **conditioned on data  $\mathbf{x}$**  and **parameterized by  $\phi$** .

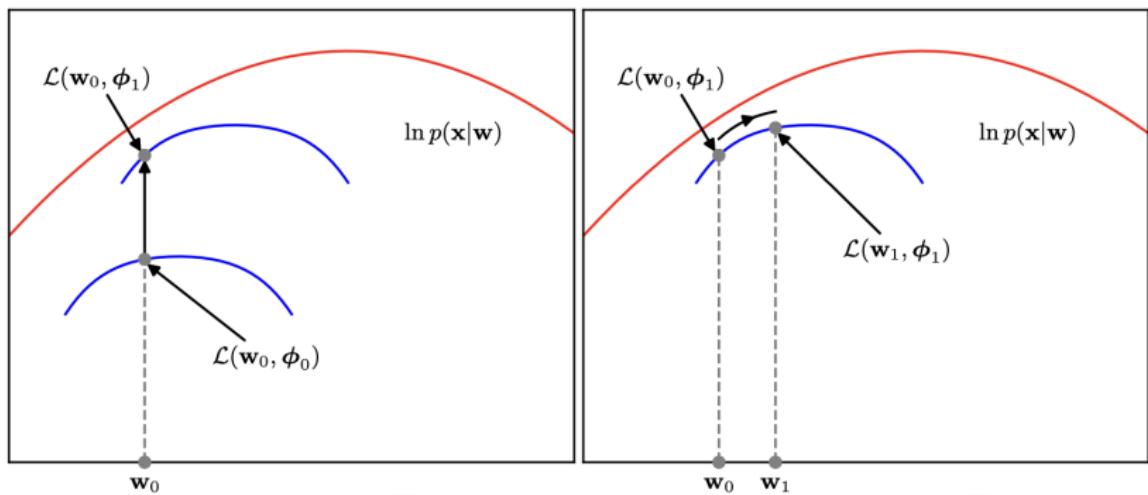
## Gradient Update

$$\begin{bmatrix} \phi_k \\ \theta_k \end{bmatrix} = \left[ \begin{array}{l} \phi_{k-1} + \eta \cdot \nabla_\phi \mathcal{L}_{\phi,\theta}(\mathbf{x}) \\ \theta_{k-1} + \eta \cdot \nabla_\theta \mathcal{L}_{\phi,\theta}(\mathbf{x}) \end{array} \right] \Big|_{(\phi_{k-1}, \theta_{k-1})}$$

# ELBO optimization

## Gradient Update

$$\begin{bmatrix} \phi_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} \phi_{k-1} + \eta \cdot \nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x}) \\ \theta_{k-1} + \eta \cdot \nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x}) \end{bmatrix} \Big|_{(\phi_{k-1}, \theta_{k-1})}$$



# ELBO optimization

## ELBO

$$\log p_{\theta}(\mathbf{x}) = \mathcal{L}_{\phi, \theta}(\mathbf{x}) + \text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p_{\theta}(\mathbf{z}|\mathbf{x})) \geq \mathcal{L}_{\phi, \theta}(\mathbf{x}).$$

$$\mathcal{L}_{q, \theta}(\mathbf{x}) = \mathbb{E}_q \log p_{\theta}(\mathbf{x}|\mathbf{z}) - \text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$$

## Gradient Update

$$\begin{bmatrix} \boldsymbol{\phi}_k \\ \boldsymbol{\theta}_k \end{bmatrix} = \left[ \begin{array}{l} \boldsymbol{\phi}_{k-1} + \eta \cdot \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\phi, \theta}(\mathbf{x}) \\ \boldsymbol{\theta}_{k-1} + \eta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\phi, \theta}(\mathbf{x}) \end{array} \right] \Big|_{(\boldsymbol{\phi}_{k-1}, \boldsymbol{\theta}_{k-1})}$$

- ▶  $\boldsymbol{\phi}$  denotes the parameters of the variational posterior  $q_{\phi}(\mathbf{z}|\mathbf{x})$ .
- ▶  $\boldsymbol{\theta}$  represents the parameters of the generative model  $p_{\theta}(\mathbf{x}|\mathbf{z})$ .

The remaining step is to obtain **unbiased** Monte Carlo estimates of the gradients:  $\nabla_{\boldsymbol{\phi}} \mathcal{L}_{\phi, \theta}(\mathbf{x})$  and  $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\phi, \theta}(\mathbf{x})$ .

# Outline

9. Latent Variable Models (LVM) (continued)

10. Variational Evidence Lower Bound (ELBO)

11. Amortized Inference

12. ELBO Gradients, Reparametrization Trick

13. Variational Autoencoder (VAE)

## ELBO Gradients: $\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

$$\mathcal{L}_{q, \theta}(\mathbf{x}) = \mathbb{E}_q \log p_{\theta}(\mathbf{x}|\mathbf{z}) - \text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$$

### Gradient $\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

$$\begin{aligned}\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \nabla_{\theta} \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} \\ &= \int q_{\phi}(\mathbf{z}|\mathbf{x}) \nabla_{\theta} \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} \\ &\approx \nabla_{\theta} \log p_{\theta}(\mathbf{x}|\mathbf{z}^*), \quad \mathbf{z}^* \sim q_{\phi}(\mathbf{z}|\mathbf{x}).\end{aligned}$$

### Naive Monte Carlo Estimation

$$\log p_{\theta}(\mathbf{x}) \geq \int \log p_{\theta}(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \approx \frac{1}{K} \sum_{k=1}^K \log p_{\theta}(\mathbf{x}|\mathbf{z}_k), \quad \mathbf{z}_k \sim p(\mathbf{z}).$$

The variational posterior  $q_{\phi}(\mathbf{z}|\mathbf{x})$  typically concentrates more probability mass in a much smaller region than the prior  $p(\mathbf{z})$ .

## ELBO Gradients: $\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

### Gradient $\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

Unlike the  $\theta$ -gradient, the density  $q_{\phi}(\mathbf{z}|\mathbf{x})$  now depends on  $\phi$ , so standard Monte Carlo estimation can't be applied:

$$\begin{aligned}\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \nabla_{\phi} \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} - \nabla_{\phi} \text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) \\ &\neq \int q_{\phi}(\mathbf{z}|\mathbf{x}) \nabla_{\phi} \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} - \nabla_{\phi} \text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))\end{aligned}$$

### Reparametrization Trick (LOTUS Trick)

Assume  $\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})$  is generated by a random variable  $\epsilon \sim p(\epsilon)$  via a deterministic mapping  $\mathbf{z} = \mathbf{g}_{\phi}(\mathbf{x}, \epsilon)$ . Then,

$$\mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} \mathbf{f}(\mathbf{z}) = \mathbb{E}_{\epsilon \sim p(\epsilon)} \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon))$$

**Note:** The LHS expectation is with respect to the parametric distribution  $q_{\phi}(\mathbf{z}|\mathbf{x})$ , while the RHS is for the non-parametric  $p(\epsilon)$ .

## ELBO Gradients: $\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

### Reparametrization Trick (LOTUS Trick)

$$\begin{aligned}\nabla_{\phi} \int q_{\phi}(\mathbf{z}|\mathbf{x}) \mathbf{f}(\mathbf{z}) d\mathbf{z} &= \nabla_{\phi} \int p(\epsilon) \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon)) d\epsilon = \\ &= \int p(\epsilon) \nabla_{\phi} \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon)) d\epsilon \approx \nabla_{\phi} \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon^*)),\end{aligned}$$

where  $\epsilon^* \sim p(\epsilon)$ .

### Variational Assumption

$$\begin{aligned}p(\epsilon) &= \mathcal{N}(0, \mathbf{I}); \quad \mathbf{z} = \mathbf{g}_{\phi}(\mathbf{x}, \epsilon) = \boldsymbol{\sigma}_{\phi}(\mathbf{x}) \odot \epsilon + \boldsymbol{\mu}_{\phi}(\mathbf{x}); \\ q_{\phi}(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\boldsymbol{\mu}_{\phi}(\mathbf{x}), \boldsymbol{\sigma}_{\phi}^2(\mathbf{x})).\end{aligned}$$

Here,  $\boldsymbol{\mu}_{\phi}(\cdot)$  and  $\boldsymbol{\sigma}_{\phi}(\cdot)$  are parameterized functions (outputs of a neural network).

Thus, we can write  $q_{\phi}(\mathbf{z}|\mathbf{x}) = \text{NN}_{e, \phi}(\mathbf{x})$ , the **encoder**.

## ELBO Gradient: $\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

$$\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x}) = \nabla_{\phi} \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} - \nabla_{\phi} \text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$$

### Reconstruction Term

$$\begin{aligned}\nabla_{\phi} \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} &= \int p(\epsilon) \nabla_{\phi} \log p_{\theta}(\mathbf{x}|\mathbf{g}_{\phi}(\mathbf{x}, \epsilon)) d\epsilon \approx \\ &\approx \nabla_{\phi} \log p_{\theta}(\mathbf{x} | \boldsymbol{\sigma}_{\phi}(\mathbf{x}) \odot \boldsymbol{\epsilon}^* + \boldsymbol{\mu}_{\phi}(\mathbf{x})), \quad \text{where } \boldsymbol{\epsilon}^* \sim \mathcal{N}(0, \mathbf{I})\end{aligned}$$

The generative distribution  $p_{\theta}(\mathbf{x}|\mathbf{z})$  can be implemented as a neural network.

We may write  $p_{\theta}(\mathbf{x}|\mathbf{z}) = \text{NN}_{d, \theta}(\mathbf{z})$ , called the **decoder**.

### KL Term

$p(\mathbf{z})$  is the prior over latents  $\mathbf{z}$ , typically  $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ .

$$\nabla_{\phi} \text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) = \nabla_{\phi} \text{KL}(\mathcal{N}(\boldsymbol{\mu}_{\phi}(\mathbf{x}), \boldsymbol{\sigma}_{\phi}^2(\mathbf{x})) \| \mathcal{N}(0, \mathbf{I}))$$

This expression admits a closed-form analytic solution.

# Outline

9. Latent Variable Models (LVM) (continued)

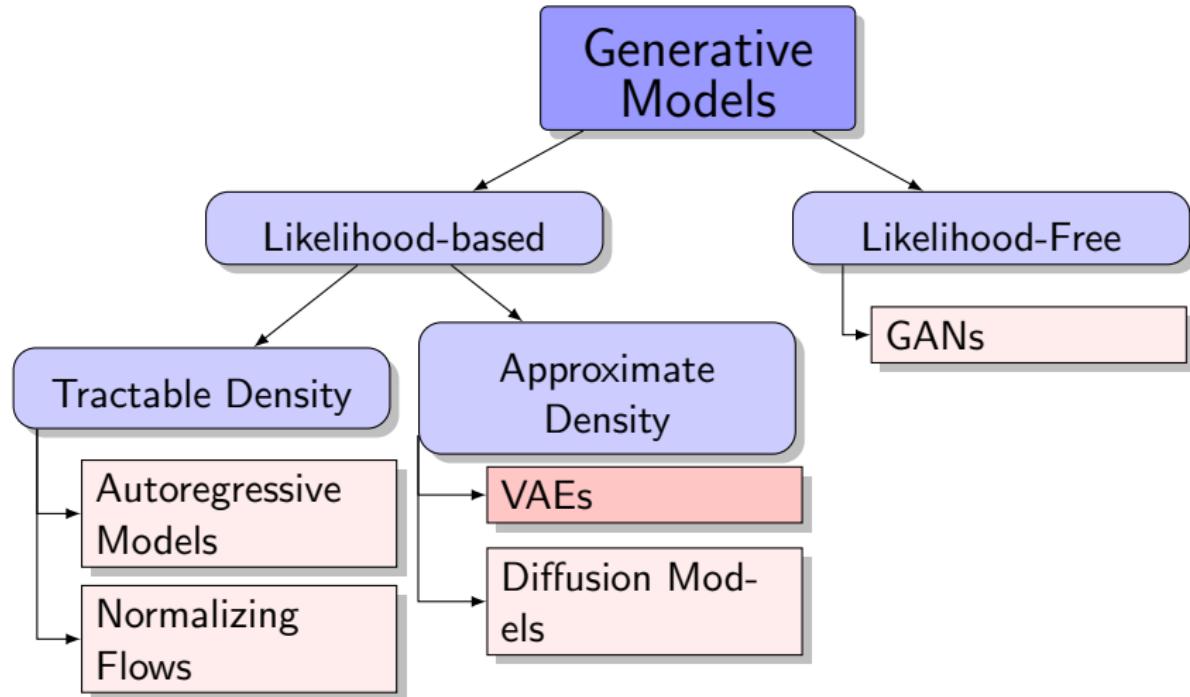
10. Variational Evidence Lower Bound (ELBO)

11. Amortized Inference

12. ELBO Gradients, Reparametrization Trick

13. Variational Autoencoder (VAE)

# Generative Models Zoo



# Variational Autoencoder (VAE)

## Training

- ▶ Pick a batch of samples  $\{\mathbf{x}_i\}_{i=1}^B$  (here we use Monte Carlo technique).
- ▶ Compute the objective for each sample (apply the reparametrization trick):

$$\epsilon^* \sim p(\epsilon); \quad \mathbf{z}^* = \mathbf{g}_\phi(\mathbf{x}, \epsilon^*);$$

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) \approx \log p_\theta(\mathbf{x}|\mathbf{z}^*) - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})).$$

- ▶ Update parameters via stochastic gradient steps with respect to  $\phi$  and  $\theta$ .

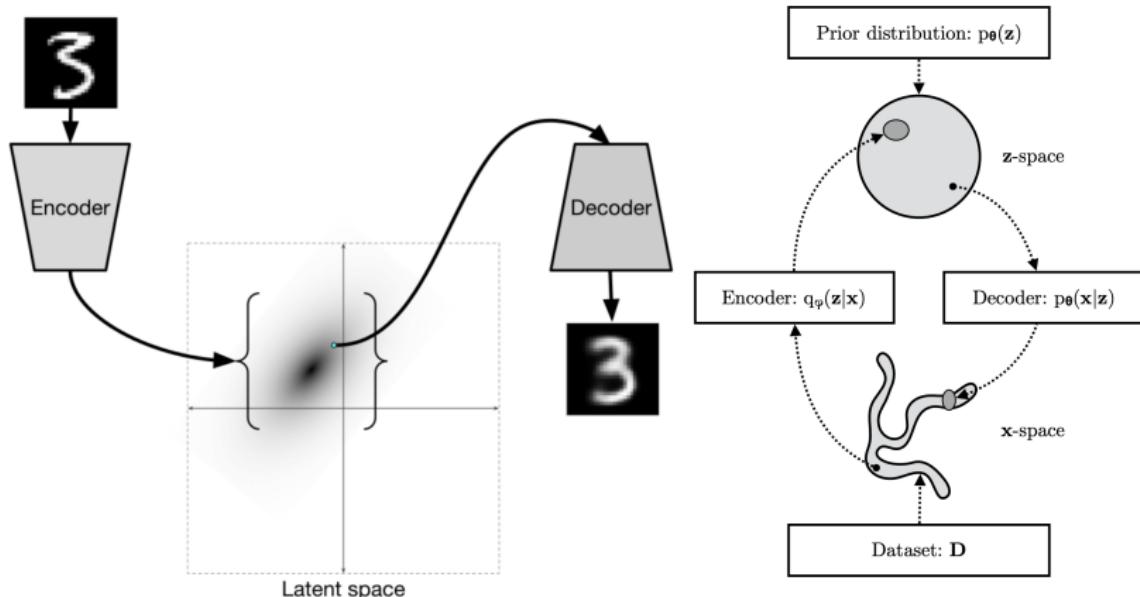
## Inference

- ▶ Sample  $\mathbf{z}^*$  from the prior  $p(\mathbf{z}) (\mathcal{N}(0, \mathbf{I}))$ ;
- ▶ Generate data from the decoder  $p_\theta(\mathbf{x}|\mathbf{z}^*)$ .

**Note:** The encoder  $q_\phi(\mathbf{z}|\mathbf{x})$  isn't needed during generation.

# Variational Autoencoder

$$\mathcal{L}_{q,\theta}(\mathbf{x}) = \mathbb{E}_q \log p_\theta(\mathbf{x}|\mathbf{z}) - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$



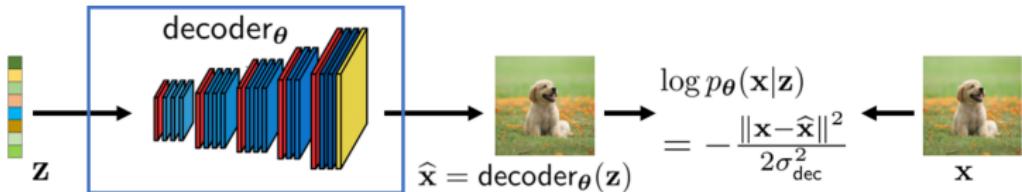
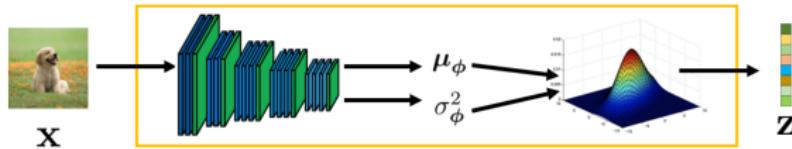
VAEs are widely used as a preliminary stage of projecting data onto low-dimensional space.

image credit: <http://ijdykeman.github.io/ml/2016/12/21/cvae.html>

Kingma D. P., Welling M., *An Introduction to Variational Autoencoders*, 2019

# Variational Autoencoder

- ▶ The encoder  $q_\phi(z|x) = \text{NN}_{e,\phi}(x)$  outputs  $\mu_\phi(x)$  and  $\sigma_\phi(x)$ .
- ▶ The decoder  $p_\theta(x|z) = \text{NN}_{d,\theta}(z)$  outputs parameters of the observed data distribution.



# VAE vs Normalizing Flows

|                   | VAE   | NF  |
|-------------------|---|---|
| <b>Objective</b>  | ELBO $\mathcal{L}$  | Forward KL/MLE  |
| <b>Encoder</b>    | stochastic<br>$\mathbf{z} \sim q_\phi(\mathbf{z} \mathbf{x})$   | deterministic<br>$\mathbf{z} = \mathbf{f}_\theta(\mathbf{x})$<br>$q_\theta(\mathbf{z} \mathbf{x}) = \delta(\mathbf{z} - \mathbf{f}_\theta(\mathbf{x}))$ |
| <b>Decoder</b>    | stochastic<br>$\mathbf{x} \sim p_\theta(\mathbf{x} \mathbf{z})$ | deterministic<br>$\mathbf{x} = \mathbf{g}_\theta(\mathbf{z})$<br>$p_\theta(\mathbf{x} \mathbf{z}) = \delta(\mathbf{x} - \mathbf{g}_\theta(\mathbf{z}))$ |
| <b>Parameters</b> | $\phi, \theta$  | $\theta \equiv \phi$  |

## Theorem

MLE for a normalizing flow is equivalent to maximizing the ELBO for a VAE where:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \delta(\mathbf{x} - \mathbf{f}_\theta^{-1}(\mathbf{z})) = \delta(\mathbf{x} - \mathbf{g}_\theta(\mathbf{z}));$$

$$q_\theta(\mathbf{z}|\mathbf{x}) = \delta(\mathbf{z} - \mathbf{f}_\theta(\mathbf{x})).$$

## Summary

- ▶ LVMs introduce latent representations for observed data, enabling more interpretable models.
- ▶ LVMs maximize the variational evidence lower bound (ELBO) to obtain maximum likelihood estimates for the parameters.
- ▶ Parametric posterior distribution  $q_\phi(\mathbf{z}|\mathbf{x})$  makes the method scalable.
- ▶ The reparametrization trick provides unbiased gradients with respect to the variational posterior  $q_\phi(\mathbf{z}|\mathbf{x})$ .
- ▶ The VAE model is a latent variable model parameterized by two neural networks: a stochastic encoder  $q_\phi(\mathbf{z}|\mathbf{x})$  and a stochastic decoder  $p_\theta(\mathbf{x}|\mathbf{z})$ .
- ▶ Nowadays, the main role of VAEs is to project data into low-dimensional latent space.

# Deep Generative Models

## Lecture 4

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

14. Discrete VAE Latent Representations

15. Vector Quantized VAE

16. ELBO Surgery

17. Learnable VAE Prior

# Outline

14. Discrete VAE Latent Representations

15. Vector Quantized VAE

16. ELBO Surgery

17. Learnable VAE Prior

# Discrete VAE Latents

## Motivation

- ▶ Previous VAE models have used **continuous** latent variables  $\mathbf{z}$ .
- ▶ For some modalities, **discrete** representations  $\mathbf{z}$  may be a more natural choice.
- ▶ Advanced autoregressive models (e.g., PixelCNN) are highly effective for distributions over discrete variables.
- ▶ Current transformer-like models process discrete tokens.

## ELBO

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \log p_{\theta}(\mathbf{x}|\mathbf{z}) - \text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) \rightarrow \max_{\phi, \theta} .$$

- ▶ Apply the reparametrization trick to obtain unbiased gradients.
- ▶ Use Gaussian distributions for  $q_{\phi}(\mathbf{z}|\mathbf{x})$  and  $p(\mathbf{z})$  to compute the KL analytically.

# Discrete VAE Latents

## Assumptions

- ▶ Let  $c \sim \text{Cat}(\boldsymbol{\pi})$ , where

$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_K), \quad \pi_k = P(c = k), \quad \sum_{k=1}^K \pi_k = 1.$$

- ▶ Suppose the VAE adopts a discrete latent variable  $c$  with prior  $p(c) = \text{Uniform}\{1, \dots, K\}$ .

## ELBO

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q_\phi(c|\mathbf{x})} \log p_\theta(\mathbf{x}|c) - \text{KL}(q_\phi(c|\mathbf{x}) \| p(c)) \rightarrow \max_{\phi, \theta} .$$

$$\text{KL}(q_\phi(c|\mathbf{x}) \| p(c)) = \sum_{k=1}^K q_\phi(k|\mathbf{x}) \log \frac{q_\phi(k|\mathbf{x})}{p(k)} =$$

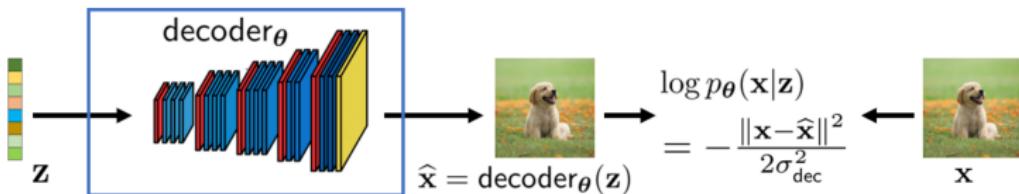
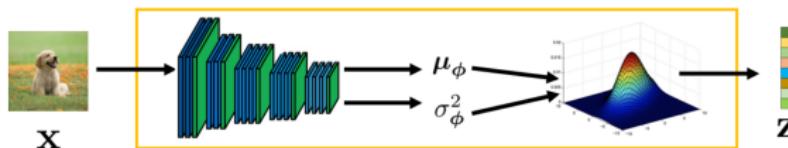
$$= \sum_{k=1}^K q_\phi(k|\mathbf{x}) \log q_\phi(k|\mathbf{x}) - \sum_{k=1}^K q_\phi(k|\mathbf{x}) \log p(k) =$$

$$= -H(q_\phi(c|\mathbf{x})) + \log K.$$

# Discrete VAE Latents

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q_{\phi}(c|\mathbf{x})} \log p_{\theta}(\mathbf{x}|c) + H(q_{\phi}(c|\mathbf{x})) - \log K \rightarrow \max_{\phi, \theta} .$$

- ▶ The encoder should output a discrete distribution  $q_{\phi}(c|\mathbf{x})$ .
- ▶ We need an analogue of the reparametrization trick for discrete  $q_{\phi}(c|\mathbf{x})$ .
- ▶ The decoder  $p_{\theta}(\mathbf{x}|c)$  must take a discrete random variable  $c$  as input.



# Outline

14. Discrete VAE Latent Representations

15. Vector Quantized VAE

16. ELBO Surgery

17. Learnable VAE Prior

# Vector Quantization

Define the codebook (dictionary) space  $\{\mathbf{e}_k\}_{k=1}^K$  with  $\mathbf{e}_k \in \mathbb{R}^L$  and  $K$  the number of codebook entries.

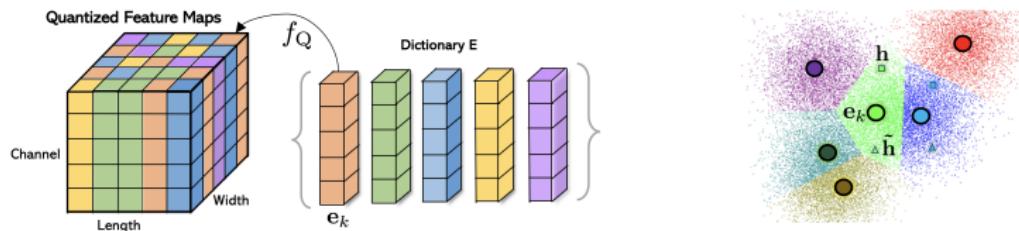
## Quantized Representation

A quantized vector  $\mathbf{z}_q \in \mathbb{R}^L$ , for any  $\mathbf{z} \in \mathbb{R}^L$ , is defined via nearest-neighbor lookup in the codebook:

$$\mathbf{z}_q = \mathbf{q}(\mathbf{z}) = \mathbf{e}_{k^*}, \quad \text{where } k^* = \arg \min_k \|\mathbf{z} - \mathbf{e}_k\|.$$

## Quantization Procedure

If the encoded tensor has spatial dimensions, quantization is independently applied to each of the  $W \times H$  locations.



## Vector Quantized VAE (VQ-VAE)

- ▶ The encoder outputs a continuous vector  $\mathbf{z}_e = \text{NN}_{e,\phi}(\mathbf{x}) \in \mathbb{R}^L$ .
- ▶ Quantization deterministically maps  $\mathbf{z}_e$  to its quantized codebook vector  $\mathbf{z}_q$ .
- ▶ The decoder is conditioned on codebook entries  $\mathbf{e}_c$ , i.e., via  $p_\theta(\mathbf{x}|\mathbf{e}_c)$  (instead of  $p_\theta(\mathbf{x}|c)$ ).

### Deterministic Variational Posterior

$$q_\phi(c = k^* | \mathbf{x}) = \begin{cases} 1, & \text{for } k^* = \arg \min_k \|\mathbf{z}_e - \mathbf{e}_k\|; \\ 0, & \text{otherwise.} \end{cases}$$

$$\text{KL}(q_\phi(c|\mathbf{x}) \| p(c)) = -\underbrace{\text{H}(q_\phi(c|\mathbf{x}))}_{=0} + \log K = \log K.$$

**Note:** The KL regularizer becomes constant and has no direct effect on the ELBO objective in this case.

# Vector Quantized VAE (VQ-VAE): Forward

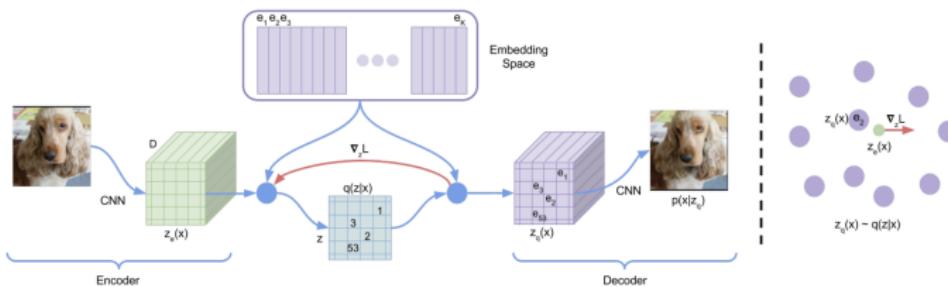
## Deterministic Variational Posterior

$$q_{\phi}(c = k^* | \mathbf{x}) = \begin{cases} 1, & \text{if } k^* = \arg \min_k \|\mathbf{z}_e - \mathbf{e}_k\|; \\ 0, & \text{otherwise.} \end{cases}$$

## ELBO

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q_{\phi}(c|\mathbf{x})} \log p_{\theta}(\mathbf{x}|\mathbf{e}_c) - \log K = \log p_{\theta}(\mathbf{x}|\mathbf{z}_q) - \log K,$$

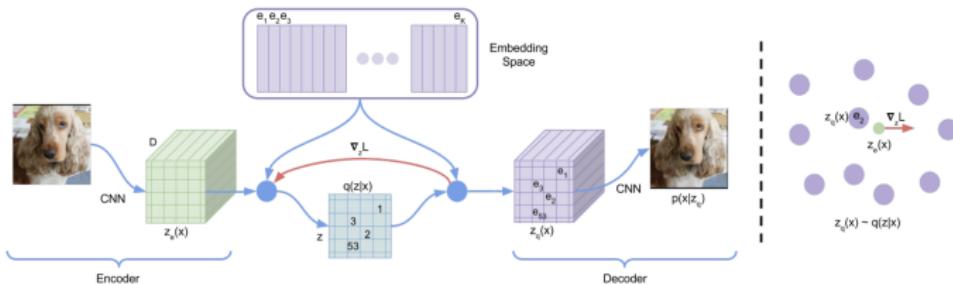
where  $\mathbf{z}_q = \mathbf{e}_{k^*}$ ,  $k^* = \arg \min_k \|\mathbf{z}_e - \mathbf{e}_k\|$ .



**Challenge:** The  $\arg \min$  operation is non-differentiable.

# Vector Quantized VAE (VQ-VAE): Backward ELBO

$$\mathcal{L}_{\phi, \theta}(x) = \log p_{\theta}(x|z_q) - \log K, \quad z_q = \mathbf{e}_{k^*}, \quad k^* = \arg \min_k \|\mathbf{z}_e - \mathbf{e}_k\|.$$



## Straight-Through Gradient Estimator

$$\begin{aligned} \frac{\partial \log p(\mathbf{x}|z_q, \theta)}{\partial \phi} &= \frac{\partial \log p_{\theta}(\mathbf{x}|z_q)}{\partial z_q} \cdot \frac{\partial z_q}{\partial \phi} = \\ &= \frac{\partial \log p_{\theta}(\mathbf{x}|z_q)}{\partial z_q} \cdot \frac{\partial z_q}{\partial z_e} \cdot \frac{\partial z_e}{\partial \phi} \approx \frac{\partial \log p_{\theta}(\mathbf{x}|z_q)}{\partial z_q} \cdot \frac{\partial z_e}{\partial \phi} \end{aligned}$$

# Vector Quantized VAE-2 (VQ-VAE-2)

Extension to the spatial domain:  $\mathbf{c} \in \{1, \dots, K\}^{W \times H}$

$$q_{\phi}(\mathbf{c}|\mathbf{x}) = \prod_{i=1}^W \prod_{j=1}^H q(c_{ij}|\mathbf{x}, \phi); \quad p(\mathbf{c}) = \prod_{i=1}^W \prod_{j=1}^H \text{Uniform}\{1, \dots, K\}.$$

## Sample Diversity



**VQ-VAE (Proposed)**

**BigGAN deep**

# Vector Quantized VAE (VQ-VAE): Final algorithm

## Training

- ▶ Vector-quantize (per spatial location if applicable):

$$k^* = \arg \min_k \|z_e - e_k\|_2, \quad z_q = e_{k^*}, \quad z_e = \text{NN}_{e,\phi}(x).$$

- ▶ Compute ELBO objective:

$$\mathcal{L}_{\phi,\theta}(x) = \log p_\theta(x|z_q) - \log K.$$

- ▶ Compute total loss with codebook and commitment losses (with stop-gradient  $\text{sg}[\cdot]$ ):

$$\mathcal{L} = -\mathcal{L}_{\phi,\theta}(x) + \|\text{sg}[z_e] - e_{k^*}\|_2^2 + \beta \|z_e - \text{sg}[e_{k^*}]\|_2^2$$

- ▶ Use straight-through gradient estimation for encoder.

## Sampling

- ▶ Sample  $c \sim p(c) = \text{Uniform}\{1, \dots, K\}$ .
- ▶ Generate  $x \sim p_\theta(x|e_c)$ .

# Outline

14. Discrete VAE Latent Representations

15. Vector Quantized VAE

16. ELBO Surgery

17. Learnable VAE Prior

# ELBO Surgery

$$\frac{1}{n} \sum_{i=1}^n \mathcal{L}_{\phi, \theta}(\mathbf{x}_i) = \frac{1}{n} \sum_{i=1}^n \left[ \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)} \log p_\theta(\mathbf{x}_i|\mathbf{z}) - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i) \| p(\mathbf{z})) \right].$$

## Theorem

$$\frac{1}{n} \sum_{i=1}^n \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i) \| p(\mathbf{z})) = \text{KL}(q_{\text{agg}, \phi}(\mathbf{z}) \| p(\mathbf{z})) + \mathbb{I}_q[\mathbf{x}, \mathbf{z}];$$

- ▶  $q_{\text{agg}, \phi}(\mathbf{z}) = \frac{1}{n} \sum_{i=1}^n q_\phi(\mathbf{z}|\mathbf{x}_i)$  denotes the **aggregated** variational posterior.
- ▶  $\mathbb{I}_q[\mathbf{x}, \mathbf{z}]$  is the mutual information between  $\mathbf{x}$  and  $\mathbf{z}$  under the data distribution  $p_{\text{data}}(\mathbf{x})$  and  $q_\phi(\mathbf{z}|\mathbf{x})$ .
- ▶ The first term encourages  $q_{\text{agg}, \phi}(\mathbf{z})$  to match the prior  $p(\mathbf{z})$ .
- ▶ The second term reduces the information about  $\mathbf{x}$  encoded in  $\mathbf{z}$ .

# ELBO Surgery

$$\frac{1}{n} \sum_{i=1}^n \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i) \| p(\mathbf{z})) = \text{KL}(q_{\text{agg},\phi}(\mathbf{z}) \| p(\mathbf{z})) + \mathbb{I}_q[\mathbf{x}, \mathbf{z}].$$

## Proof

$$\begin{aligned}\frac{1}{n} \sum_{i=1}^n \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i) \| p(\mathbf{z})) &= \frac{1}{n} \sum_{i=1}^n \int q_\phi(\mathbf{z}|\mathbf{x}_i) \log \frac{q_\phi(\mathbf{z}|\mathbf{x}_i)}{p(\mathbf{z})} d\mathbf{z} = \\ &= \frac{1}{n} \sum_{i=1}^n \int q_\phi(\mathbf{z}|\mathbf{x}_i) \log \frac{q_{\text{agg},\phi}(\mathbf{z}) q_\phi(\mathbf{z}|\mathbf{x}_i)}{p(\mathbf{z}) q_{\text{agg},\phi}(\mathbf{z})} d\mathbf{z} = \\ &= \int \frac{1}{n} \sum_{i=1}^n q_\phi(\mathbf{z}|\mathbf{x}_i) \log \frac{q_{\text{agg},\phi}(\mathbf{z})}{p(\mathbf{z})} d\mathbf{z} + \frac{1}{n} \sum_{i=1}^n \int q_\phi(\mathbf{z}|\mathbf{x}_i) \log \frac{q_\phi(\mathbf{z}|\mathbf{x}_i)}{q_{\text{agg},\phi}(\mathbf{z})} d\mathbf{z} = \\ &= \text{KL}(q_{\text{agg},\phi}(\mathbf{z}) \| p(\mathbf{z})) + \frac{1}{n} \sum_{i=1}^n \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i) \| q_{\text{agg},\phi}(\mathbf{z})) \\ \mathbb{I}_q[\mathbf{x}, \mathbf{z}] &= \frac{1}{n} \sum_{i=1}^n \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i) \| q_{\text{agg},\phi}(\mathbf{z})).\end{aligned}$$

# ELBO Surgery

## Revisiting the ELBO

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n \mathcal{L}_{\phi, \theta}(\mathbf{x}_i) &= \frac{1}{n} \sum_{i=1}^n [\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)} \log p_\theta(\mathbf{x}_i|\mathbf{z}) - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i)\|p(\mathbf{z}))] = \\ &= \underbrace{\frac{1}{n} \sum_{i=1}^n \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)} \log p_\theta(\mathbf{x}_i|\mathbf{z})}_{\text{Reconstruction Loss}} - \underbrace{\mathbb{I}_q[\mathbf{x}, \mathbf{z}]}_{\text{Mutual Information}} - \underbrace{\text{KL}(q_{\text{agg}, \phi}(\mathbf{z})\|p(\mathbf{z}))}_{\text{Marginal KL}} \end{aligned}$$

The prior distribution  $p(\mathbf{z})$  only appears in the last term.

## Optimal VAE Prior

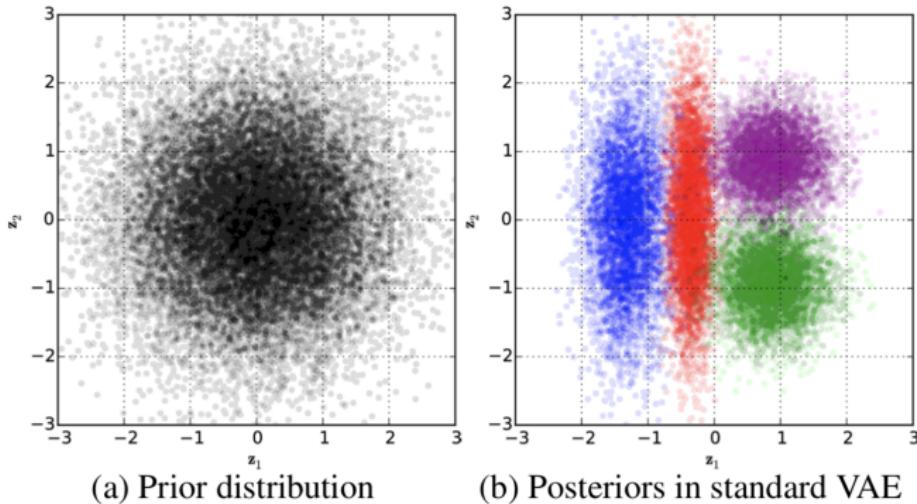
$$\text{KL}(q_{\text{agg}, \phi}(\mathbf{z})\|p(\mathbf{z})) = 0 \Leftrightarrow p(\mathbf{z}) = q_{\text{agg}, \phi}(\mathbf{z}) = \frac{1}{n} \sum_{i=1}^n q_\phi(\mathbf{z}|\mathbf{x}_i).$$

Hence, the optimal prior  $p(\mathbf{z})$  is the aggregated variational posterior  $q_{\text{agg}, \phi}(\mathbf{z})$ .

## Marginal KL

$$\text{KL}(q_{\text{agg}, \phi}(\mathbf{z}) \| p(\mathbf{z}))$$

- ▶  $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}_\phi^2(\mathbf{x}))$  is unimodal.
- ▶ It is generally believed that the **mismatch between  $p(\mathbf{z})$  and  $q_{\text{agg}, \phi}(\mathbf{z})$**  is the primary explanation for blurry VAE-generated images.



# Outline

14. Discrete VAE Latent Representations

15. Vector Quantized VAE

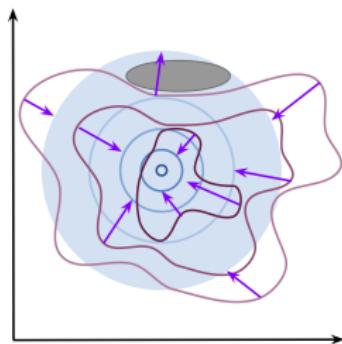
16. ELBO Surgery

17. Learnable VAE Prior

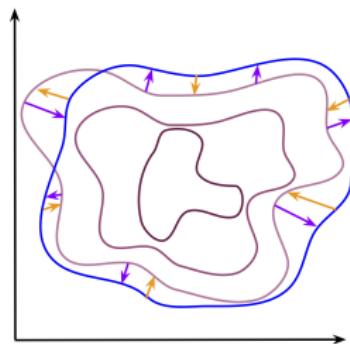
# Optimal VAE Prior

- ▶ Standard Gaussian  $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$  often leads to over-regularization.
- ▶  $p(\mathbf{z}) = q_{\text{agg}, \phi}(\mathbf{z}) = \frac{1}{n} \sum_{i=1}^n q_\phi(\mathbf{z} | \mathbf{x}_i)$  risks overfitting and incurs high computational cost.

Non-Learnable Prior  $p(\mathbf{z})$



Learnable Prior  $p_\lambda(\mathbf{z})$



$$\frac{1}{n} \sum_{i=1}^n \mathcal{L}_{\phi, \theta}(\mathbf{x}_i) = \text{RL} - \text{MI} - \text{KL}(q_{\text{agg}, \phi}(\mathbf{z}) \| p_\lambda(\mathbf{z}))$$

This is the forward KL divergence with respect to  $p_\lambda(\mathbf{z})$ .

image credit: [https://jmtomczak.github.io/blog/7/7\\_priors.html](https://jmtomczak.github.io/blog/7/7_priors.html)

# NF-Based VAE Prior

## NF Model in Latent Space

$$\log p_{\lambda}(\mathbf{z}) = \log p(\mathbf{z}^*) + \log \left| \det \left( \frac{d\mathbf{z}^*}{d\mathbf{z}} \right) \right| = \log p(\mathbf{f}_{\lambda}(\mathbf{z})) + \log |\det(\mathbf{J}_f)|$$

$$\mathbf{z} = \mathbf{g}_{\lambda}(\mathbf{z}^*) = \mathbf{f}_{\lambda}^{-1}(\mathbf{z}^*)$$

- ▶ RealNVP with coupling layers,
- ▶ Autoregressive normalizing flows (efficient  $\mathbf{f}_{\lambda}(\mathbf{z})$ , but  $\mathbf{g}_{\lambda}(\mathbf{z}^*)$  can be slow).

## ELBO with NF-Based VAE Prior

$$\begin{aligned}\mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \log p_{\theta}(\mathbf{x}|\mathbf{z}) - \text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) = \\ &= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p_{\theta}(\mathbf{x}|\mathbf{z}) + \log p_{\lambda}(\mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x})] = \\ &= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[ \log p_{\theta}(\mathbf{x}|\mathbf{z}) + \underbrace{\left( \log p(\mathbf{f}_{\lambda}(\mathbf{z})) + \log |\det(\mathbf{J}_f)| \right)}_{\text{NF-based prior}} - \log q_{\phi}(\mathbf{z}|\mathbf{x}) \right]\end{aligned}$$

## Summary

- ▶ Discrete VAE latents offer a natural class of latent variable models.
- ▶ Vector quantization provides a way to construct VAEs with discrete latents and deterministic variational posteriors.
- ▶ The straight-through gradient estimator allows gradients to pass as if quantization were an identity operation during backpropagation.
- ▶ ELBO surgery gives insights into the prior's influence in VAEs; the optimal prior is the aggregated variational posterior.
- ▶ The mismatch between  $p(\mathbf{z})$  and  $q_{\text{agg},\phi}(\mathbf{z})$  is widely regarded as the principal reason for VAE-generated image blurriness.
- ▶ Normalizing flow-based priors, including autoregressive flows, can be incorporated directly into VAEs.

# Deep Generative Models

## Lecture 5

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

18. Likelihood-Free Learning

19. Generative Adversarial Networks (GAN)

20. Wasserstein Distance

21. Wasserstein GAN

# Outline

18. Likelihood-Free Learning

19. Generative Adversarial Networks (GAN)

20. Wasserstein Distance

21. Wasserstein GAN

# Outline

18. Likelihood-Free Learning

19. Generative Adversarial Networks (GAN)

20. Wasserstein Distance

21. Wasserstein GAN

# Likelihood-Based Models

Poor Likelihood  
High-Quality Samples

$$p_1(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n \mathcal{N}(\mathbf{x} | \mathbf{x}_i, \epsilon \mathbf{I})$$

If  $\epsilon$  is very small, this model produces excellent, sharp samples but achieves poor likelihoods on test data.

High Likelihood  
Poor Samples

$$\begin{aligned} p_2(\mathbf{x}) &= 0.01p(\mathbf{x}) + 0.99p_{\text{noise}}(\mathbf{x}) \\ \log [0.01p(\mathbf{x}) + 0.99p_{\text{noise}}(\mathbf{x})] &\geq \\ \geq \log [0.01p(\mathbf{x})] &= \log p(\mathbf{x}) - \log 100 \end{aligned}$$

This model contains mostly noisy, irrelevant samples; for high dimensions,  $\log p(\mathbf{x})$  scales linearly with  $m$ .

- ▶ Likelihood isn't always a suitable metric for evaluating generative models.
- ▶ Sometimes, the likelihood function can't even be computed exactly.

# Likelihood-Free Learning

## Motivation

We're interested in approximating the true data distribution  $p_{\text{data}}(\mathbf{x})$ . Instead of searching over all distributions, let's learn a model  $p_{\theta}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$ .

Suppose we have two sets of samples:

- ▶  $\{\mathbf{x}_i\}_{i=1}^{n_1} \sim p_{\text{data}}(\mathbf{x})$  — real data;
- ▶  $\{\mathbf{x}_i\}_{i=1}^{n_2} \sim p_{\theta}(\mathbf{x})$  — generated (fake) data.

Define a discriminative model (classifier):

$$p(y = 1|\mathbf{x}) = P(\mathbf{x} \sim p_{\text{data}}(\mathbf{x})); \quad p(y = 0|\mathbf{x}) = P(\mathbf{x} \sim p_{\theta}(\mathbf{x}))$$

## Assumption

The generative model  $p_{\theta}(\mathbf{x})$  matches  $p_{\text{data}}(\mathbf{x})$  if a discriminative model  $p(y|\mathbf{x})$  can't distinguish between them — that is, if  $p(y = 1|\mathbf{x}) = 0.5$  for every  $\mathbf{x}$ .

# Generative Adversarial Networks (GAN)

- ▶ The more expressive the discriminator, the closer we get to the optimal  $p_\theta(\mathbf{x})$ .
- ▶ Standard classifiers are trained by minimizing cross-entropy loss  $-\mathbb{E}_{\hat{p}(\mathbf{x}, y)} \log p(y|\mathbf{x})$  with  
 $\hat{p}(\mathbf{x}, y) = \frac{1}{2}[y = 1]p_{\text{data}}(\mathbf{x}) + \frac{1}{2}[y = 0]p_\theta(\mathbf{x})$ .

## Cross-Entropy for Discriminator

$$\begin{aligned}& \min_{p(y|\mathbf{x})} \left[ -\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \log p(y=1|\mathbf{x}) - \mathbb{E}_{p_\theta(\mathbf{x})} \log p(y=0|\mathbf{x}) \right] \\& \max_{p(y|\mathbf{x})} \left[ \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \log p(y=1|\mathbf{x}) + \mathbb{E}_{p_\theta(\mathbf{x})} \log p(y=0|\mathbf{x}) \right]\end{aligned}$$

## Generative Model

Suppose  $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ , where  $p(\mathbf{z})$  is a base distribution, and  $p_\theta(\mathbf{x}|\mathbf{z}) = \delta(\mathbf{x} - \mathbf{G}_\theta(\mathbf{z}))$  is deterministic.

# Generative Adversarial Networks (GAN)

## Cross-Entropy for Discriminative Model

$$\max_{p(y|x)} [\mathbb{E}_{p_{\text{data}}(x)} \log p(y=1|x) + \mathbb{E}_{p_{\theta}(x)} \log p(y=0|x)]$$

- ▶ **Discriminator:** A classifier  $p_{\phi}(y=1|x) = D_{\phi}(x) \in [0, 1]$ , distinguishing real and generated samples. The discriminator aims to **maximize** cross-entropy.
- ▶ **Generator:** The generative model  $x = \mathbf{G}_{\theta}(z)$ ,  $z \sim p(z)$ , seeks to fool the discriminator. The generator aims to **minimize** cross-entropy.

## GAN Objective

$$\min_G \max_D [\mathbb{E}_{p_{\text{data}}(x)} \log D(x) + \mathbb{E}_{p_{\theta}(x)} \log(1 - D(x))]$$

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

# Outline

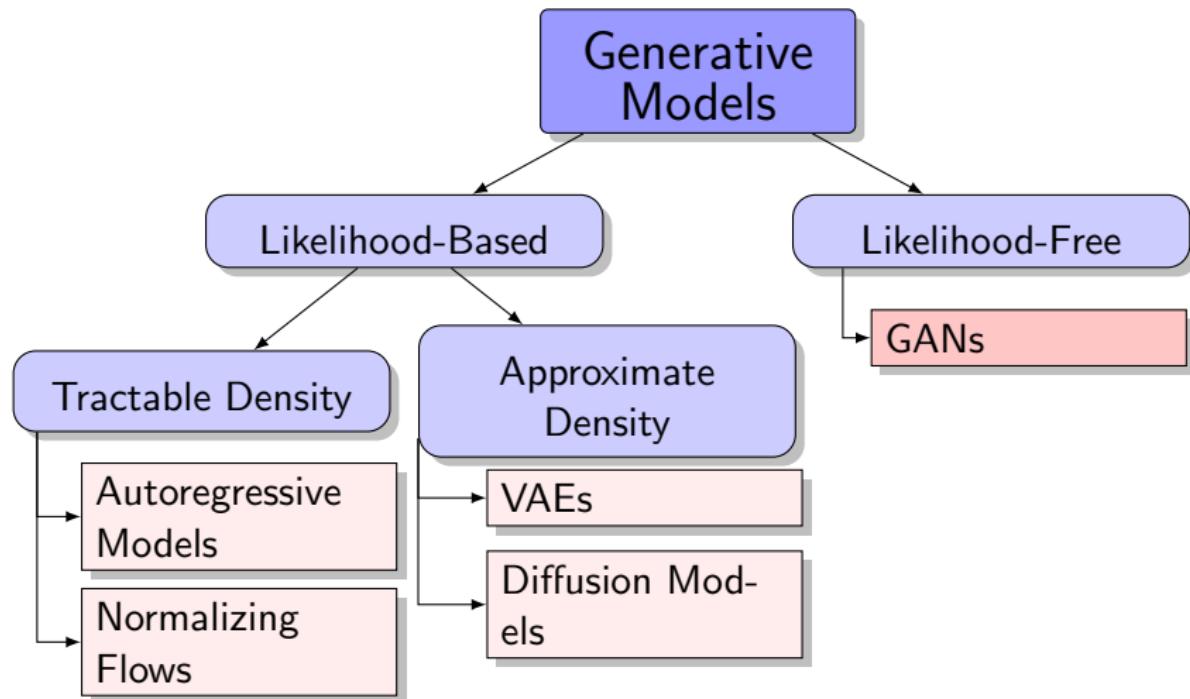
18. Likelihood-Free Learning

19. Generative Adversarial Networks (GAN)

20. Wasserstein Distance

21. Wasserstein GAN

# Generative Models Zoo



# GAN Optimality

## Theorem

The minimax game

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

achieves its global optimum when  $p_{\text{data}}(\mathbf{x}) = p_\theta(\mathbf{x})$ , and  $D^*(\mathbf{x}) = 0.5$ .

## Proof (Fixed $G$ )

$$\begin{aligned} V(G, D) &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p_\theta(\mathbf{x})} \log(1 - D(\mathbf{x})) \\ &= \int \underbrace{[p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) + p_\theta(\mathbf{x}) \log(1 - D(\mathbf{x}))]}_{y(D)} d\mathbf{x} \end{aligned}$$

$$\frac{dy(D)}{dD} = \frac{p_{\text{data}}(\mathbf{x})}{D(\mathbf{x})} - \frac{p_\theta(\mathbf{x})}{1 - D(\mathbf{x})} = 0 \quad \Rightarrow \quad D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_\theta(\mathbf{x})}$$

# GAN Optimality

Proof Continued (Fixed  $D = D^*$ )

$$\begin{aligned} V(G, D^*) &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \log \left( \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\theta}(\mathbf{x})} \right) + \mathbb{E}_{p_{\theta}(\mathbf{x})} \log \left( \frac{p_{\theta}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\theta}(\mathbf{x})} \right) \\ &= \text{KL} \left( p_{\text{data}}(\mathbf{x}) \parallel \frac{p_{\text{data}}(\mathbf{x}) + p_{\theta}(\mathbf{x})}{2} \right) + \text{KL} \left( p_{\theta}(\mathbf{x}) \parallel \frac{p_{\text{data}}(\mathbf{x}) + p_{\theta}(\mathbf{x})}{2} \right) - 2 \log 2 \\ &= 2 \text{JSD}(p_{\text{data}}(\mathbf{x}) \parallel p_{\theta}(\mathbf{x})) - 2 \log 2. \end{aligned}$$

Jensen-Shannon Divergence (Symmetric KL Divergence)

$$\text{JSD}(p_{\text{data}}(\mathbf{x}) \parallel p_{\theta}(\mathbf{x})) = \frac{1}{2} [\text{KL}(p_{\text{data}}(\mathbf{x}) \parallel \star) + \text{KL}(p_{\theta}(\mathbf{x}) \parallel \star)]$$

This can be regarded as a proper distance metric!

$$V(G^*, D^*) = -2 \log 2, \quad p_{\text{data}}(\mathbf{x}) = p_{\theta}(\mathbf{x}), \quad D^*(\mathbf{x}) = 0.5.$$

# GAN Optimality

## Theorem

The following minimax game

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

achieves its global optimum precisely when  $p_{\text{data}}(\mathbf{x}) = p_\theta(\mathbf{x})$ , and  $D^*(\mathbf{x}) = 0.5$ .

## Expectations

If the generator can express **any** function and the discriminator is **optimal** at every step, the generator **will converge** to the target distribution.

## Reality

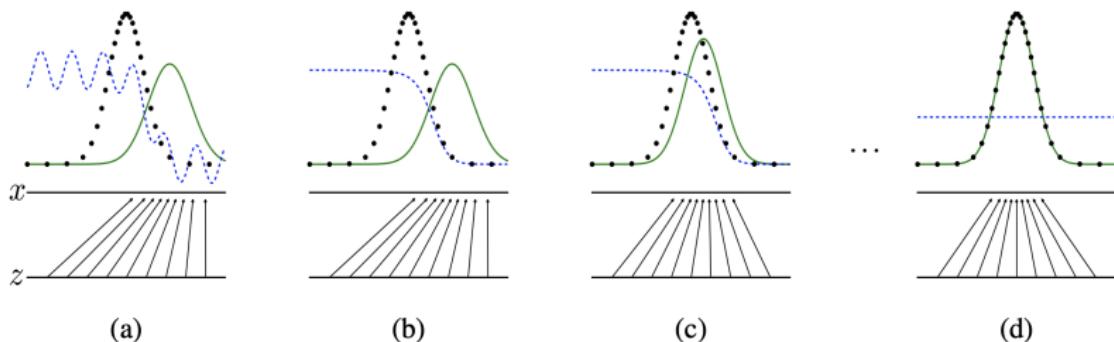
- ▶ Generator updates are performed in parameter space, and the discriminator is often imperfectly optimized.
- ▶ Generator and discriminator losses typically oscillate during GAN training.

# GAN Training

Assume both generator and discriminator are parametric models:  
 $D_\phi(\mathbf{x})$  and  $\mathbf{G}_\theta(\mathbf{z})$ .

## Objective

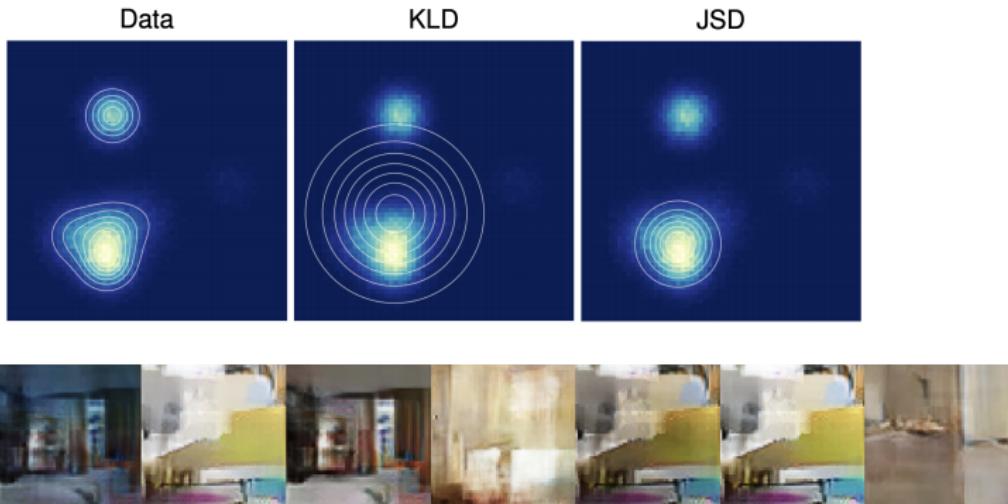
$$\min_{\theta} \max_{\phi} [\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \log D_\phi(\mathbf{x}) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D_\phi(\mathbf{G}_\theta(\mathbf{z})))]$$



- ▶  $\mathbf{z} \sim p(\mathbf{z})$  is a latent variable.
- ▶  $p_\theta(\mathbf{x}|\mathbf{z}) = \delta(\mathbf{x} - \mathbf{G}_\theta(\mathbf{z}))$  serves as a deterministic decoder (like normalizing flows).
- ▶ There is no encoder present.

## Mode Collapse

Mode collapse refers to the phenomenon where the generator in a GAN produces only one or a few different modes of the distribution.



Numerous methods have been proposed to tackle mode collapse: changing architectures, adding regularization terms, injecting noise.

Goodfellow I. J. et al. *Generative Adversarial Networks*, 2014

Metz L. et al. *Unrolled Generative Adversarial Networks*, 2016

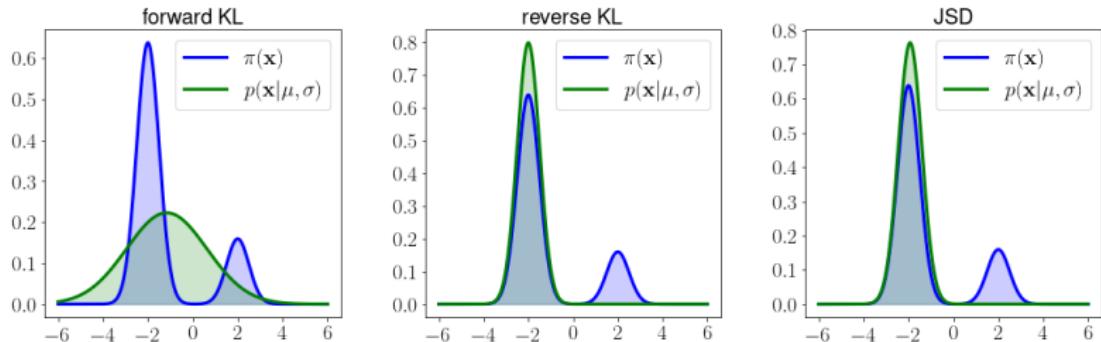
# Jensen-Shannon vs Kullback-Leibler Divergences

- ▶  $p_{\text{data}}(\mathbf{x})$  is a fixed mixture of two Gaussians.
- ▶  $p(\mathbf{x}|\mu, \sigma) = \mathcal{N}(\mu, \sigma^2)$ .

## Mode Covering vs. Mode Seeking

$$\text{KL}(\pi \| p) = \int \pi(\mathbf{x}) \log \frac{\pi(\mathbf{x})}{p(\mathbf{x})} d\mathbf{x}, \quad \text{KL}(p \| \pi) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{\pi(\mathbf{x})} d\mathbf{x}$$

$$\text{JSD}(\pi \| p) = \frac{1}{2} \left[ \text{KL}\left(\pi(\mathbf{x}) \| \frac{\pi(\mathbf{x}) + p(\mathbf{x})}{2}\right) + \text{KL}\left(p(\mathbf{x}) \| \frac{\pi(\mathbf{x}) + p(\mathbf{x})}{2}\right) \right]$$



# Outline

18. Likelihood-Free Learning

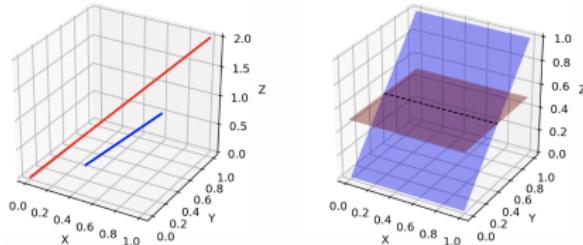
19. Generative Adversarial Networks (GAN)

20. Wasserstein Distance

21. Wasserstein GAN

# Theoretical Results

- ▶ The dimensionality of  $\mathbf{z}$  is less than that of  $\mathbf{x}$ , so  $p_\theta(\mathbf{x})$  with  $\mathbf{x} = \mathbf{G}_\theta(\mathbf{z})$  lives on a low-dimensional manifold.
- ▶ The true data distribution  $p_{\text{data}}(\mathbf{x})$  is also supported on a low-dimensional manifold.



- ▶ If  $p_{\text{data}}(\mathbf{x})$  and  $p_\theta(\mathbf{x})$  are disjoint, a smooth optimal discriminator can exist!
- ▶ For such low-dimensional, disjoint manifolds:

$$\text{KL}(p_{\text{data}} \parallel p_\theta) = \text{KL}(p_\theta \parallel p_{\text{data}}) = \infty, \quad \text{JSD}(p_{\text{data}} \parallel p_\theta) = \log 2$$

---

Weng L. From GAN to WGAN, 2019

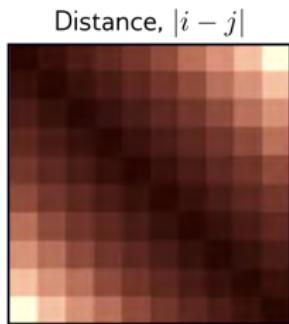
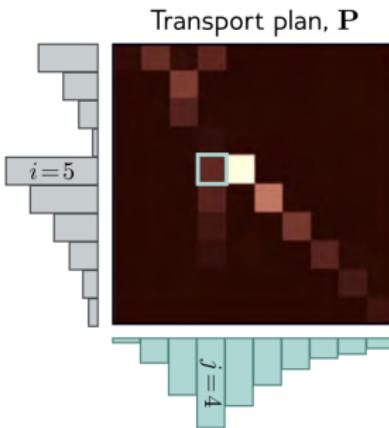
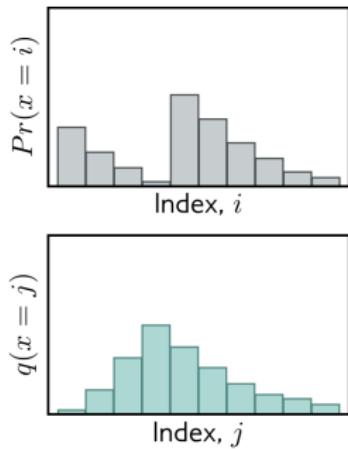
Arjovsky M., Bottou L. Towards Principled Methods for Training Generative Adversarial Networks, 2017

# Wasserstein Distance (Discrete)

Also known as the **Earth Mover's Distance**.

## Optimal Transport Formulation

The minimum cost of moving and transforming a pile of "dirt" shaped like one probability distribution to match another.



$$\begin{aligned} \text{Wasserstein distance} \\ = \sum \mathbf{P} \cdot |i - j| \end{aligned}$$

## Wasserstein Distance (Continuous)

$$W(\pi \| p) = \inf_{\gamma \in \Gamma(\pi, p)} \mathbb{E}_{(\mathbf{x}_1, \mathbf{x}_2) \sim \gamma} \|\mathbf{x}_1 - \mathbf{x}_2\| = \inf_{\gamma \in \Gamma(\pi, p)} \int \|\mathbf{x}_1 - \mathbf{x}_2\| \gamma(\mathbf{x}_1, \mathbf{x}_2) d\mathbf{x}_1 d\mathbf{x}_2$$

- ▶  $\gamma(\mathbf{x}_1, \mathbf{x}_2)$  is the transport plan: the amount of “dirt” assigned from  $\mathbf{x}_1$  to  $\mathbf{x}_2$ .

$$\int \gamma(\mathbf{x}_1, \mathbf{x}_2) d\mathbf{x} = p(\mathbf{x}_2); \quad \int \gamma(\mathbf{x}_1, \mathbf{x}_2) d\mathbf{x}_2 = \pi(\mathbf{x}_1).$$

- ▶  $\Gamma(\pi, p)$  denotes the set of all joint distributions  $\gamma(\mathbf{x}_1, \mathbf{x}_2)$  with marginals  $\pi$  and  $p$ .
- ▶  $\gamma(\mathbf{x}_1, \mathbf{x}_2)$  is the mass,  $\|\mathbf{x}_1 - \mathbf{x}_2\|$  is the distance.

## Wasserstein Metric

$$W_s(\pi, p) = \inf_{\gamma \in \Gamma(\pi, p)} \left( \mathbb{E}_{(\mathbf{x}_1, \mathbf{x}_2) \sim \gamma} \|\mathbf{x}_1 - \mathbf{x}_2\|^s \right)^{1/s}$$

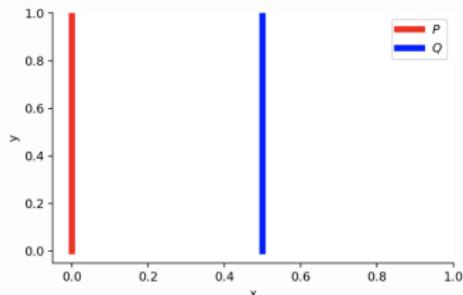
In our setting,  $W(\pi \| p) = W_1(\pi, p)$ , which is the transport cost formulation.

# Wasserstein Distance vs KL vs JSD

Consider two-dimensional distributions:

$$p_{\text{data}}(x, y) = (0, U[0, 1])$$

$$p_{\theta}(x, y) = (\theta, U[0, 1])$$



- $\theta = 0$ : Both distributions are identical.

$$\text{KL}(p_{\text{data}} \| p_{\theta}) = \text{KL}(p_{\theta} \| p_{\text{data}}) = \text{JSD}(p_{\theta} \| p_{\text{data}}) = W(p_{\text{data}} \| p_{\theta}) = 0$$

- $\theta \neq 0$ :

$$\text{KL}(p_{\text{data}} \| p_{\theta}) = \int_{U[0,1]} 1 \log \frac{1}{0} dy = \infty = \text{KL}(p_{\theta} \| p_{\text{data}})$$

$$\text{JSD}(p_{\text{data}} \| p_{\theta}) = \frac{1}{2} \left( \int_{U[0,1]} 1 \log \frac{1}{1/2} dy + \int_{U[0,1]} 1 \log \frac{1}{1/2} dy \right) = \log 2$$

$$W(p_{\text{data}} \| p_{\theta}) = |\theta|$$

# Wasserstein Distance vs KL vs JSD

## Theorem 1

Let  $\mathbf{G}_\theta(\mathbf{z})$  be (almost) any feedforward neural network, and  $p(\mathbf{z})$  a prior over  $\mathbf{z}$  such that  $\mathbb{E}_{p(\mathbf{z})}\|\mathbf{z}\| < \infty$ . Then  $W(p_{\text{data}} \| p_\theta)$  is continuous everywhere and differentiable almost everywhere.

## Theorem 2

Let  $\pi$  be a distribution on a compact space  $\mathcal{X}$  and let  $\{p_t\}_{t=1}^\infty$  be a sequence of distributions on  $\mathcal{X}$ .

$$\text{KL}(\pi \| p_t) \rightarrow 0 \quad (\text{or } \text{KL}(p_t \| \pi) \rightarrow 0) \tag{1}$$

$$\text{JSD}(\pi \| p_t) \rightarrow 0 \tag{2}$$

$$W(\pi \| p_t) \rightarrow 0 \tag{3}$$

In summary, as  $t \rightarrow \infty$ , (1)  $\Rightarrow$  (2), and (2)  $\Rightarrow$  (3).

# Outline

18. Likelihood-Free Learning

19. Generative Adversarial Networks (GAN)

20. Wasserstein Distance

21. Wasserstein GAN

# Wasserstein GAN

## Wasserstein Distance

$$W(\pi \| p) = \inf_{\gamma \in \Gamma(\pi, p)} \mathbb{E}_{(\mathbf{x}_1, \mathbf{x}_2) \sim \gamma} \|\mathbf{x}_1 - \mathbf{x}_2\| = \inf_{\gamma \in \Gamma(\pi, p)} \int \|\mathbf{x}_1 - \mathbf{x}_2\| \gamma(\mathbf{x}_1, \mathbf{x}_2) d\mathbf{x}_1 d\mathbf{x}_2$$

The infimum over all possible  $\gamma \in \Gamma(\pi, p)$  is computationally intractable.

## Theorem (Kantorovich-Rubinstein Duality)

$$W(\pi \| p) = \frac{1}{K} \max_{\|f\|_L \leq K} \left[ \mathbb{E}_{\pi(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{p(\mathbf{x})} f(\mathbf{x}) \right]$$

where  $f : \mathbb{R}^m \rightarrow \mathbb{R}$  is  $K$ -Lipschitz ( $\|f\|_L \leq K$ ):

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq K \|\mathbf{x}_1 - \mathbf{x}_2\|, \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}.$$

We can thus estimate  $W(\pi \| p)$  using only samples and a function  $f$ .

# Wasserstein GAN

## Theorem (Kantorovich-Rubinstein Duality)

$$W(p_{\text{data}} \| p_{\theta}) = \frac{1}{K} \max_{\|f\|_L \leq K} \left[ \mathbb{E}_{p_{\text{data}}(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{p_{\theta}(\mathbf{x})} f(\mathbf{x}) \right]$$

- ▶ We must ensure that  $f$  is  $K$ -Lipschitz continuous.
- ▶ Let  $f_{\phi}(\mathbf{x})$  be a feedforward neural network parameterized by  $\phi$ .
- ▶ If the weights  $\phi$  are restricted to a compact set  $\Phi$ , then  $f_{\phi}$  is  $K$ -Lipschitz.
- ▶ Clamp weights within the box  $\Phi = [-c, c]^d$  (e.g.  $c = 0.01$ ) after each update.

$$\begin{aligned} K \cdot W(p_{\text{data}} \| p_{\theta}) &= \max_{\|f\|_L \leq K} \left[ \mathbb{E}_{p_{\text{data}}(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{p_{\theta}(\mathbf{x})} f(\mathbf{x}) \right] \geq \\ &\geq \max_{\phi \in \Phi} \left[ \mathbb{E}_{p_{\text{data}}(\mathbf{x})} f_{\phi}(\mathbf{x}) - \mathbb{E}_{p_{\theta}(\mathbf{x})} f_{\phi}(\mathbf{x}) \right] \end{aligned}$$

# Wasserstein GAN

## Standard GAN Objective

$$\min_{\theta} \max_{\phi} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \log D_{\phi}(\mathbf{x}) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D_{\phi}(\mathbf{G}_{\theta}(\mathbf{z})))$$

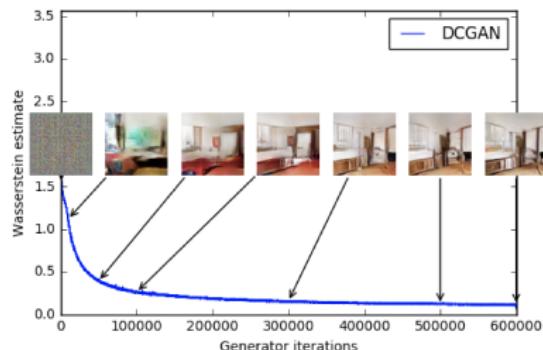
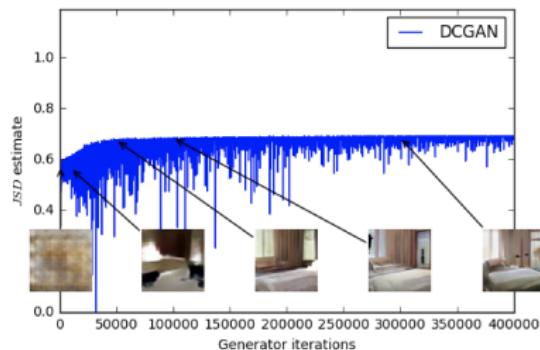
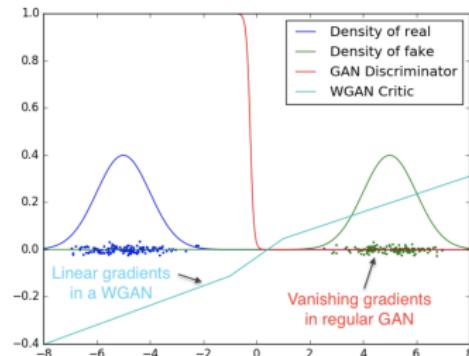
## WGAN Objective

$$\min_{\theta} W(p_{\text{data}} \| p_{\theta}) \approx \min_{\theta} \max_{\phi \in \Phi} \left[ \mathbb{E}_{p_{\text{data}}(\mathbf{x})} f_{\phi}(\mathbf{x}) - \mathbb{E}_{p(\mathbf{z})} f_{\phi}(\mathbf{G}_{\theta}(\mathbf{z})) \right]$$

- ▶ The discriminator  $D$  is replaced by function  $f$ : in WGAN, it is known as the **critic**, which is *not* a classifier.
- ▶ "*Weight clipping is a clearly terrible way to enforce a Lipschitz constraint.*"
  - ▶ If  $c$  is large, optimizing the critic is hard.
  - ▶ If  $c$  is small, gradients may vanish.

# Wasserstein GAN

- ▶ WGAN provides nonzero gradients even if distributions' supports are disjoint.
- ▶  $JSD(p_{\text{data}} \| p_{\theta})$  is poorly correlated with sample quality and remains near its maximum value  $\log 2 \approx 0.69$ .
- ▶  $W(p_{\text{data}} \| p_{\theta})$  is tightly correlated with quality.



## Summary

- ▶ Likelihood is not a reliable metric for generative model evaluation.
- ▶ Adversarial learning casts distribution matching as a minimax game.
- ▶ GANs, in theory, optimize the Jensen-Shannon divergence.
- ▶ KL and JS divergences fail as objectives when the model and data distributions are disjoint.
- ▶ The Earth Mover's (Wasserstein) distance provides a more meaningful loss for distribution matching.
- ▶ Kantorovich-Rubinstein duality allows us to compute the EM distance using only samples.
- ▶ Wasserstein GAN enforces the Lipschitz condition on the critic through weight clipping—although better alternatives exist.

# Deep Generative Models

## Lecture 6

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

## 22. Evaluation of Likelihood-Free Models

Frechet Inception Distance (FID)

Precision-Recall

CLIP Score

Human Evaluation

## 23. Langevin Dynamics

## 24. Score Matching

## 25. Denoising Score Matching

# Outline

## 22. Evaluation of Likelihood-Free Models

Frechet Inception Distance (FID)

Precision-Recall

CLIP Score

Human Evaluation

## 23. Langevin Dynamics

## 24. Score Matching

## 25. Denoising Score Matching

# Evaluation of Likelihood-Free Models

## Likelihood-Based Models

- ▶ **Train:** fit the model.
- ▶ **Validation:** tune hyperparameters.
- ▶ **Test:** assess generalization by reporting likelihood.

Not all models have tractable likelihoods  
(VAE: compare ELBO values; GAN: ???).

## Desirable Properties for Samples

- ▶ Sharpness



- ▶ Diversity



# Outline

## 22. Evaluation of Likelihood-Free Models

Frechet Inception Distance (FID)

Precision-Recall

CLIP Score

Human Evaluation

## 23. Langevin Dynamics

## 24. Score Matching

## 25. Denoising Score Matching

## Wasserstein Metric

$$W_s(\pi \| p) = \inf_{\gamma \in \Gamma(\pi, p)} (\mathbb{E}_{(\mathbf{x}_1, \mathbf{x}_2) \sim \gamma} \|\mathbf{x}_1 - \mathbf{x}_2\|^s)^{1/s}$$

## Wasserstein GAN (Optimal Transport)

$$W(\pi \| p) = \inf_{\gamma \in \Gamma(\pi, p)} \mathbb{E}_{(\mathbf{x}_1, \mathbf{x}_2) \sim \gamma} \|\mathbf{x}_1 - \mathbf{x}_2\| = \inf_{\gamma \in \Gamma(\pi, p)} \int \|\mathbf{x}_1 - \mathbf{x}_2\| \gamma(\mathbf{x}_1, \mathbf{x}_2) d\mathbf{x}_1 d\mathbf{x}_2$$

## Theorem

If  $\pi(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\pi, \boldsymbol{\Sigma}_\pi)$ ,  $p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ , then

$$W_2^2(\pi \| p) = \|\boldsymbol{\mu}_\pi - \boldsymbol{\mu}_p\|^2 + \text{tr} \left[ \boldsymbol{\Sigma}_\pi + \boldsymbol{\Sigma}_p - 2 \left( \boldsymbol{\Sigma}_\pi^{1/2} \boldsymbol{\Sigma}_p \boldsymbol{\Sigma}_\pi^{1/2} \right)^{1/2} \right]$$

## Frechet Inception Distance

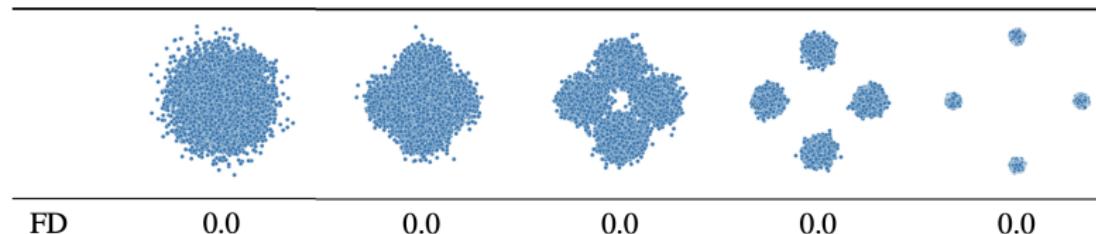
$$\text{FID}(p_{\text{data}}, p_\theta) = W_2^2(p_{\text{data}} \| p_\theta)$$

## Frechet Inception Distance (FID)

$$\text{FID}(p_{\text{data}}, p_{\theta}) = \|\boldsymbol{\mu}_{\text{data}} - \boldsymbol{\mu}_{\theta}\|^2 + \text{tr} \left[ \boldsymbol{\Sigma}_{\text{data}} + \boldsymbol{\Sigma}_{\theta} - 2 \left( \boldsymbol{\Sigma}_{\text{data}}^{1/2} \boldsymbol{\Sigma}_{\theta} \boldsymbol{\Sigma}_{\text{data}}^{1/2} \right)^{1/2} \right]$$

- ▶ FID is computed in the latent space  $\mathbf{z}$ .
- ▶ We use a pretrained image embedder to get latent representations  $\mathbf{z} = \mathbf{f}(\mathbf{x})$ .
- ▶  $\boldsymbol{\mu}_{\text{data}}$ ,  $\boldsymbol{\Sigma}_{\text{data}}$  and  $\boldsymbol{\mu}_{\theta}$ ,  $\boldsymbol{\Sigma}_{\theta}$  are statistics of latent representations for samples from  $p_{\text{data}}(\mathbf{x})$  and  $p_{\theta}(\mathbf{x})$ .

$$FID(p(\mathbf{x}), \mathcal{N}(0, \mathbf{I}))$$



## Frechet Inception Distance (FID)

$$\text{FID}(p_{\text{data}}, p_{\theta}) = \|\boldsymbol{\mu}_{\text{data}} - \boldsymbol{\mu}_{\theta}\|^2 + \text{tr} \left[ \boldsymbol{\Sigma}_{\text{data}} + \boldsymbol{\Sigma}_{\theta} - 2 \left( \boldsymbol{\Sigma}_{\text{data}}^{1/2} \boldsymbol{\Sigma}_{\theta} \boldsymbol{\Sigma}_{\text{data}}^{1/2} \right)^{1/2} \right]$$

### Drawbacks

- ▶ Depends on the pretrained classification network.
- ▶ Uses the normality assumption.
- ▶ May not correlate with human evaluation.

| Model                  | Model-A | Model-B |
|------------------------|---------|---------|
| FID                    | 21.40   | 18.42   |
| $\text{FID}_{\infty}$  | 20.16   | 17.19   |
| Human rater preference | 92.5%   | 6.9%    |

# Outline

## 22. Evaluation of Likelihood-Free Models

Frechet Inception Distance (FID)

Precision-Recall

CLIP Score

Human Evaluation

## 23. Langevin Dynamics

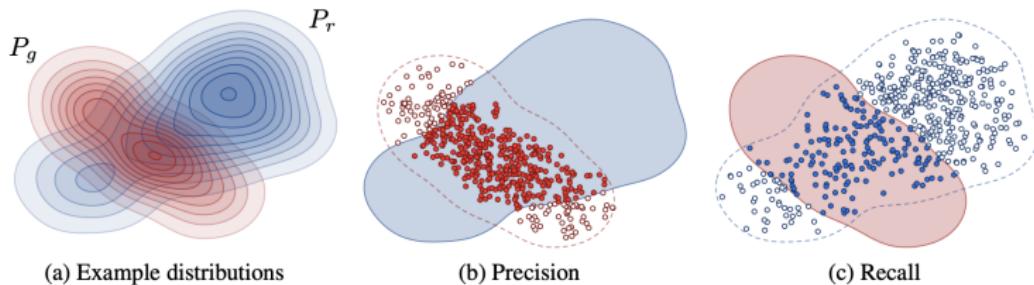
## 24. Score Matching

## 25. Denoising Score Matching

# Precision-Recall

## Desirable Properties for Samples

- ▶ **Sharpness:** generated samples should possess high visual quality.
- ▶ **Diversity:** their variation should match that in the training data.



- ▶ **Precision** denotes the fraction of generated images that look realistic.
- ▶ **Recall** measures how well the generator covers the training data manifold.

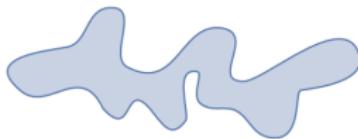
## Precision-Recall

- ▶  $\mathcal{S}_{\text{data}} = \{\mathbf{x}_i\}_{i=1}^n \sim p_{\text{data}}(\mathbf{x})$  – real samples;
- ▶  $\mathcal{S}_{\theta} = \{\mathbf{x}_i\}_{i=1}^n \sim p_{\theta}(\mathbf{x})$  – generated samples.

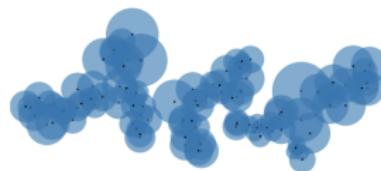
Define a binary function:

$$\mathbb{I}(\mathbf{x}, \mathcal{S}) = \begin{cases} 1, & \text{if } \exists \mathbf{x}' \in \mathcal{S} : \|\mathbf{x} - \mathbf{x}'\|_2 \leq \|\mathbf{x}' - \text{NN}_k(\mathbf{x}', \mathcal{S})\|_2; \\ 0, & \text{otherwise.} \end{cases}$$

$$\Pr(\mathcal{S}_{\text{data}}, \mathcal{S}_{\theta}) = \frac{1}{n} \sum_{\mathbf{x} \in \mathcal{S}_{\theta}} \mathbb{I}(\mathbf{x}, \mathcal{S}_{\text{data}}); \quad \text{Rec}(\mathcal{S}_{\text{data}}, \mathcal{S}_{\theta}) = \frac{1}{n} \sum_{\mathbf{x} \in \mathcal{S}_{\text{data}}} \mathbb{I}(\mathbf{x}, \mathcal{S}_{\theta}).$$



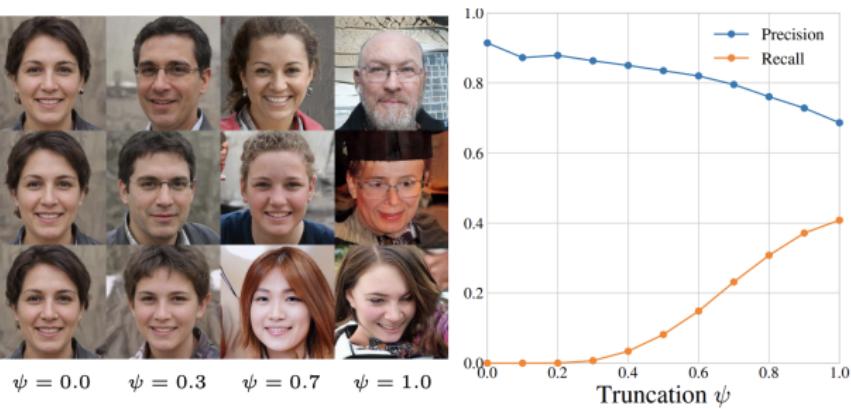
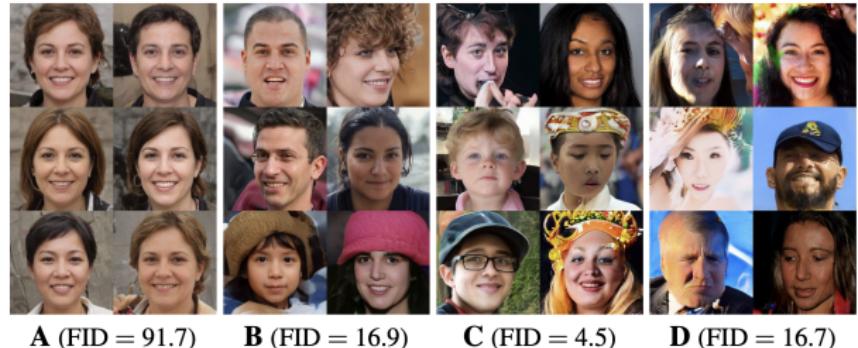
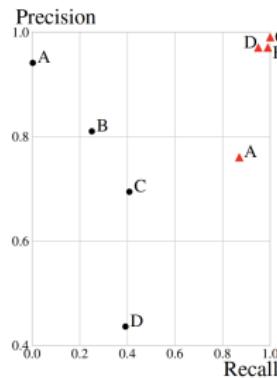
(a) True manifold



(b) Approx. manifold

Embed the samples using a pretrained network (as in FID).

# Precision-Recall



Kynkäanniemi T. et al. Improved precision and recall metric for assessing generative models, 2019

# Outline

## 22. Evaluation of Likelihood-Free Models

Frechet Inception Distance (FID)

Precision-Recall

**CLIP Score**

Human Evaluation

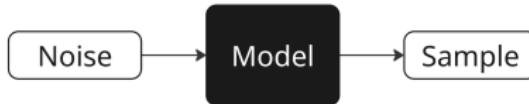
## 23. Langevin Dynamics

## 24. Score Matching

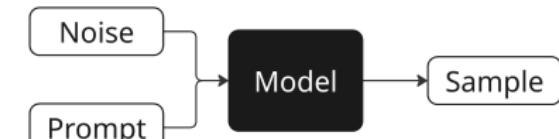
## 25. Denoising Score Matching

# CLIP Score

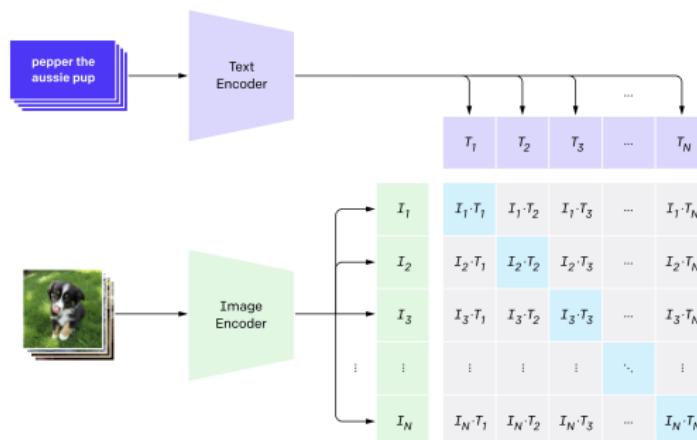
## Unconditional Model



## Conditional Model



We need a way to measure not only the quality of the generated image, but also how well it's aligned with the prompt.



# Outline

## 22. Evaluation of Likelihood-Free Models

Frechet Inception Distance (FID)

Precision-Recall

CLIP Score

Human Evaluation

## 23. Langevin Dynamics

## 24. Score Matching

## 25. Denoising Score Matching

# Human Evaluation

- ▶ No automated metric is perfect.
- ▶ The best way to evaluate generative models is by human assessment.
- ▶ It's important to assess various properties.

| Аспект        | Yandex ART 2.0 | Mj 6.1      | Mj 6        | Ideogram    | Recraft     | Google Imagen3 | Dall-E 3    | FLUX        | SBER Kandi3.1 |
|---------------|----------------|-------------|-------------|-------------|-------------|----------------|-------------|-------------|---------------|
| Релевантность | <b>0,59</b>    | <b>0,58</b> | <b>0,63</b> | <b>0,45</b> | <b>0,51</b> | <b>0,50</b>    | <b>0,50</b> | <b>0,54</b> | <b>0,75</b>   |
| Эстетика      | <b>0,49</b>    | <b>0,55</b> | <b>0,55</b> | <b>0,51</b> | <b>0,51</b> | <b>0,61</b>    | <b>0,61</b> | <b>0,54</b> | <b>0,59</b>   |
| Комплексность | <b>0,44</b>    | <b>0,73</b> | <b>0,70</b> | <b>0,68</b> | <b>0,76</b> | <b>0,75</b>    | <b>0,75</b> | <b>0,71</b> | <b>0,74</b>   |
| Дефектность   | <b>0,69</b>    | <b>0,57</b> | <b>0,68</b> | <b>0,55</b> | <b>0,59</b> | <b>0,63</b>    | <b>0,63</b> | <b>0,50</b> | <b>0,75</b>   |
| Предпочтение  | <b>0,66</b>    | <b>0,60</b> | <b>0,69</b> | <b>0,49</b> | <b>0,54</b> | <b>0,63</b>    | <b>0,63</b> | <b>0,51</b> | <b>0,84</b>   |

# Outline

## 22. Evaluation of Likelihood-Free Models

Frechet Inception Distance (FID)

Precision-Recall

CLIP Score

Human Evaluation

## 23. Langevin Dynamics

## 24. Score Matching

## 25. Denoising Score Matching

# Energy-Based Models

## Unnormalized Density

$$p_{\theta}(\mathbf{x}) = \frac{\hat{p}_{\theta}(\mathbf{x})}{Z_{\theta}}, \quad \text{where } Z_{\theta} = \int \hat{p}_{\theta}(\mathbf{x}) d\mathbf{x}$$

- ▶  $\hat{p}_{\theta}(\mathbf{x})$  can be any non-negative function.
- ▶ If we reparameterize as  $\hat{p}_{\theta}(\mathbf{x}) = \exp(-f_{\theta}(\mathbf{x}))$ , we eliminate the non-negativity constraint.

## Unnormalized Density

The gradient of the normalized log-density equals that of the unnormalized log-density:

$$\nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}) = \nabla_{\mathbf{x}} \log \hat{p}_{\theta}(\mathbf{x}) - \nabla_{\mathbf{x}} \log Z_{\theta} = \nabla_{\mathbf{x}} \log \hat{p}_{\theta}(\mathbf{x})$$

- ▶ Suppose we already have this density (normalized or not)  $p_{\theta}(\mathbf{x})$ .
- ▶ How can we sample from the model?

# Langevin Dynamics

## Theorem

Consider energy-based model  $p(\mathbf{x}) = \frac{\hat{p}(\mathbf{x})}{Z}$ ,  $\hat{p}(\mathbf{x}) = \exp(-f(\mathbf{x}))$ , with continuously differentiable  $f(\mathbf{x}) : \mathbb{R}^m \rightarrow \mathbb{R}$  that satisfies

- ▶  $L$ -smoothness:  $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$ ;
- ▶ Strong convexity:  $(\nabla f(\mathbf{x}) - \nabla f(\mathbf{y}))^\top (\mathbf{x} - \mathbf{y}) \geq m\|\mathbf{x} - \mathbf{y}\|^2$  for some  $m > 0$ .

Consider a Markov chain  $\mathbf{x}_{I+1} = \mathbf{x}_I + \frac{\eta}{2} \cdot \nabla_{\mathbf{x}_I} \log \hat{p}(\mathbf{x}_I) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}_I$ , where  $\boldsymbol{\epsilon}_I \sim \mathcal{N}(0, \mathbf{I})$ . Then, for any  $\eta < \frac{2}{L}$

- ▶ The Markov chain has a unique stationary distribution  $\pi_\eta$ .
- ▶  $W_2(\pi_\eta, p) \leq C\eta$ , and as  $\eta \rightarrow 0$  we have  $\pi_\eta \xrightarrow{d} p$ .

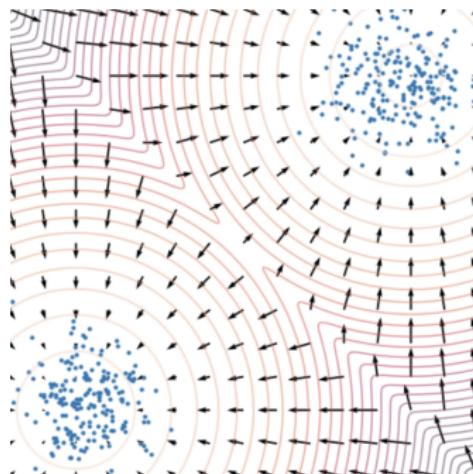
# Langevin Dynamics

## Theorem (Informal)

Let  $\mathbf{x}_0$  be a random vector. Under mild regularity conditions, samples from the following dynamics will eventually follow  $p_\theta(\mathbf{x})$  (for sufficiently small  $\eta$  and large  $I$ ):

$$\mathbf{x}_{I+1} = \mathbf{x}_I + \frac{\eta}{2} \cdot \nabla_{\mathbf{x}_I} \log p_\theta(\mathbf{x}_I) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}_I, \quad \boldsymbol{\epsilon}_I \sim \mathcal{N}(0, \mathbf{I}).$$

- ▶ What if  $\boldsymbol{\epsilon}_I = \mathbf{0}$ ?
- ▶ The density  $p_\theta(\mathbf{x})$  is the **stationary** distribution of the Markov chain.
- ▶ The gradient is taken with respect to  $\mathbf{x}$ , not  $\theta$ .
- ▶  $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$  defines a vector field.



# Outline

## 22. Evaluation of Likelihood-Free Models

Frechet Inception Distance (FID)

Precision-Recall

CLIP Score

Human Evaluation

## 23. Langevin Dynamics

## 24. Score Matching

## 25. Denoising Score Matching

# Score Matching

## Score Function

$$\mathbf{s}_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$$

## Langevin Dynamics

If we know the score function  $\mathbf{s}_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$ , we can generate samples from the model using Langevin dynamics:

$$\mathbf{x}_{I+1} = \mathbf{x}_I + \frac{\eta}{2} \cdot \nabla_{\mathbf{x}_I} \log p_\theta(\mathbf{x}_I) + \sqrt{\eta} \cdot \epsilon_I = \mathbf{x}_I + \frac{\eta}{2} \cdot \mathbf{s}_\theta(\mathbf{x}_I) + \sqrt{\eta} \cdot \epsilon_I.$$

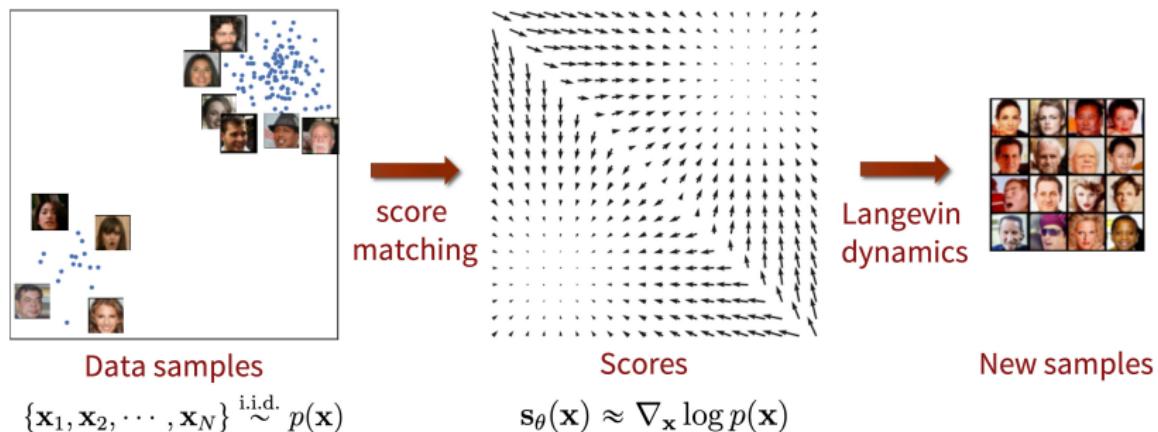
## Fisher Divergence

$$\begin{aligned} D_F(p_{\text{data}}, p_\theta) &= \frac{1}{2} \mathbb{E}_\pi \left\| \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \right\|_2^2 = \\ &= \frac{1}{2} \mathbb{E}_\pi \left\| \mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \right\|_2^2 \rightarrow \min_{\theta} \end{aligned}$$

# Score Matching

## Fisher Divergence

$$D_F(p_{\text{data}}, p_{\theta}) = \frac{1}{2} \mathbb{E}_{\pi} \| \mathbf{s}_{\theta}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \|_2^2 \rightarrow \min_{\theta}$$



**Problem:** We don't know  $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ .

# Outline

## 22. Evaluation of Likelihood-Free Models

Frechet Inception Distance (FID)

Precision-Recall

CLIP Score

Human Evaluation

## 23. Langevin Dynamics

## 24. Score Matching

## 25. Denoising Score Matching

# Denoising Score Matching

Let us perturb the original data  $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$  with Gaussian noise:

$$\mathbf{x}_\sigma = \mathbf{x} + \sigma \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \quad q(\mathbf{x}_\sigma | \mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma^2 \cdot \mathbf{I})$$

$$q(\mathbf{x}_\sigma) = \int q(\mathbf{x}_\sigma | \mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x}.$$

## Assumption

The solution to

$$\frac{1}{2} \mathbb{E}_{q(\mathbf{x}_\sigma)} \| \mathbf{s}_{\theta, \sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma) \|_2^2 \rightarrow \min_{\theta}$$

satisfies  $\mathbf{s}_{\theta, \sigma}(\mathbf{x}_\sigma) \approx \mathbf{s}_{\theta, 0}(\mathbf{x}_0) = \mathbf{s}_\theta(\mathbf{x})$  if  $\sigma$  is sufficiently small.

- ▶ The score function of the noised data nearly matches the score function of the original data.
- ▶ The score function  $\mathbf{s}_{\theta, \sigma}(\mathbf{x}_\sigma)$  is parameterized by  $\sigma$ .
- ▶ **Note:** We don't know  $q(\mathbf{x}_\sigma)$ , just as we don't know  $p_{\text{data}}(\mathbf{x})$ .

# Denoising Score Matching

## Theorem

Under mild regularity conditions, the following holds:

$$\begin{aligned}\mathbb{E}_{q(\mathbf{x}_\sigma)} \left\| \mathbf{s}_{\theta, \sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma) \right\|_2^2 &= \\ &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma | \mathbf{x})} \left\| \mathbf{s}_{\theta, \sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma | \mathbf{x}) \right\|_2^2 + \text{const}(\theta)\end{aligned}$$

## Gradient of the Noise Kernel

$$\mathbf{x}_\sigma = \mathbf{x} + \sigma \cdot \boldsymbol{\epsilon}, \quad q(\mathbf{x}_\sigma | \mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma^2 \cdot \mathbf{I})$$

$$\nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma | \mathbf{x}) = -\frac{\mathbf{x}_\sigma - \mathbf{x}}{\sigma^2} = -\frac{\boldsymbol{\epsilon}}{\sigma}$$

- ▶ The right-hand side doesn't require computing  $\nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma)$  or even  $\nabla_{\mathbf{x}_\sigma} \log p_{\text{data}}(\mathbf{x}_\sigma)$ .
- ▶  $\mathbf{s}_{\theta, \sigma}(\mathbf{x}_\sigma)$  is trained to **denoise** the noised samples  $\mathbf{x}_\sigma$ .

# Denoising Score Matching

Initial objective:

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|_2^2 \rightarrow \min_{\theta}$$

Noised objective:

$$\mathbb{E}_{q(\mathbf{x}_\sigma)} \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}} \log q(\mathbf{x}_\sigma)\|_2^2 \rightarrow \min_{\theta}$$

This is equivalent to a denoising task:

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})} \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x})\|_2^2 \rightarrow \min_{\theta}$$

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathcal{N}(0, \mathbf{I})} \left\| \mathbf{s}_{\theta,\sigma}(\mathbf{x} + \sigma \cdot \boldsymbol{\epsilon}) + \frac{\boldsymbol{\epsilon}}{\sigma} \right\|_2^2 \rightarrow \min_{\theta}$$

## Langevin Dynamics

$$\mathbf{x}_{I+1} = \mathbf{x}_I + \frac{\eta}{2} \cdot \mathbf{s}_{\theta,\sigma}(\mathbf{x}_I) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}_I, \quad \boldsymbol{\epsilon}_I \sim \mathcal{N}(0, \mathbf{I}).$$

## Summary

- ▶ Frechet Inception Distance is the most popular metric for evaluating implicit generative models.
- ▶ Precision-recall allows for choosing a model that balances sample quality and diversity.
- ▶ The CLIP score is widely used to measure text-to-image alignment.
- ▶ The gold standard for evaluating generated image quality is human assessment.
- ▶ Langevin dynamics enable sampling from generative models using gradients of the log-likelihood.
- ▶ Score matching proposes minimizing Fisher divergence to estimate the score function.
- ▶ Denoising score matching optimizes Fisher divergence on noisy data, making it estimable with samples.

# Deep Generative Models

## Lecture 7

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

## 26. Score Matching

- Denoising Score Matching (continued)
- Noise-Conditioned Score Network

## 27. Forward Gaussian Diffusion Process

## 28. Denoising Score Matching for Diffusion

## 29. Reverse Gaussian Diffusion Process

# Outline

## 26. Score Matching

- Denoising Score Matching (continued)
- Noise-Conditioned Score Network

## 27. Forward Gaussian Diffusion Process

## 28. Denoising Score Matching for Diffusion

## 29. Reverse Gaussian Diffusion Process

# Outline

## 26. Score Matching

Denoising Score Matching (continued)

Noise-Conditioned Score Network

## 27. Forward Gaussian Diffusion Process

## 28. Denoising Score Matching for Diffusion

## 29. Reverse Gaussian Diffusion Process

# Denoising Score Matching

## Theorem

$$\begin{aligned} \mathbb{E}_{q(\mathbf{x}_\sigma)} \underbrace{\|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma)\|_2^2}_{h(\mathbf{x}_\sigma)} &= \\ &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})} \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x})\|_2^2 + \text{const}(\theta) \end{aligned}$$

## Proof

$$\begin{aligned} \mathbb{E}_{q(\mathbf{x}_\sigma)} h(\mathbf{x}_\sigma) &= \int q(\mathbf{x}_\sigma) h(\mathbf{x}_\sigma) d\mathbf{x}_\sigma = \\ &= \int \left( \int q(\mathbf{x}_\sigma|\mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x} \right) h(\mathbf{x}_\sigma) d\mathbf{x}_\sigma = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})} h(\mathbf{x}_\sigma) \\ \mathbb{E}_{q(\mathbf{x}_\sigma)} \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma)\|_2^2 &= \\ &= \mathbb{E}_{q(\mathbf{x}_\sigma)} \left[ \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma)\|^2 + \underbrace{\|\nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma)\|_2^2}_{\text{const}(\theta)} - 2\mathbf{s}_{\theta,\sigma}^\top(\mathbf{x}_\sigma) \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma) \right] \end{aligned}$$

# Denoising Score Matching

## Theorem

$$\begin{aligned} \mathbb{E}_{q(\mathbf{x}_\sigma)} \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma)\|_2^2 &= \\ = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})} \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x})\|_2^2 + \text{const}(\theta) \end{aligned}$$

## Proof (Continued)

$$\begin{aligned} \mathbb{E}_{q(\mathbf{x}_\sigma)} [\mathbf{s}_{\theta,\sigma}^\top(\mathbf{x}_\sigma) \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma)] &= \int q(\mathbf{x}_\sigma) \left[ \mathbf{s}_{\theta,\sigma}^\top(\mathbf{x}_\sigma) \frac{\nabla_{\mathbf{x}_\sigma} q(\mathbf{x}_\sigma)}{q(\mathbf{x}_\sigma)} \right] d\mathbf{x}_\sigma = \\ &= \int \left[ \mathbf{s}_{\theta,\sigma}^\top(\mathbf{x}_\sigma) \nabla_{\mathbf{x}_\sigma} \left( \int q(\mathbf{x}_\sigma|\mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x} \right) \right] d\mathbf{x}_\sigma = \\ &= \int \int p_{\text{data}}(\mathbf{x}) [\mathbf{s}_{\theta,\sigma}^\top(\mathbf{x}_\sigma) \nabla_{\mathbf{x}_\sigma} q(\mathbf{x}_\sigma|\mathbf{x})] d\mathbf{x}_\sigma d\mathbf{x} = \\ &= \int \int p_{\text{data}}(\mathbf{x}) q(\mathbf{x}_\sigma|\mathbf{x}) [\mathbf{s}_{\theta,\sigma}^\top(\mathbf{x}_\sigma) \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x})] d\mathbf{x}_\sigma d\mathbf{x} = \\ &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})} [\mathbf{s}_{\theta,\sigma}^\top(\mathbf{x}_\sigma) \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x})] \end{aligned}$$

# Denoising Score Matching

## Theorem

$$\begin{aligned} \mathbb{E}_{q(\mathbf{x}_\sigma)} \underbrace{\|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma)\|_2^2}_{h(\mathbf{x}_\sigma)} &= \\ = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})} \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x})\|_2^2 + \text{const}(\theta) \end{aligned}$$

## Proof (Continued)

$$\begin{aligned} \mathbb{E}_{q(\mathbf{x}_\sigma)} h(\mathbf{x}_\sigma) &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})} h(\mathbf{x}_\sigma) \\ \mathbb{E}_{q(\mathbf{x}_\sigma)} [\mathbf{s}_{\theta,\sigma}^\top(\mathbf{x}_\sigma) \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma)] &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})} [\mathbf{s}_{\theta,\sigma}^\top(\mathbf{x}_\sigma) \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x})] \\ \mathbb{E}_{q(\mathbf{x}_\sigma)} \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma)\|_2^2 &= \\ = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})} [\|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma)\|^2 - 2\mathbf{s}_{\theta,\sigma}^\top(\mathbf{x}_\sigma) \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x})] + \text{const}(\theta) &= \\ = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})} \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x})\|_2^2 + \text{const}(\theta) \end{aligned}$$

# Denoising Score Matching

Original objective:

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|_2^2 \rightarrow \min_{\theta}$$

Noisy objective:

$$\mathbb{E}_{q(\mathbf{x}_\sigma)} \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}} \log q(\mathbf{x}_\sigma)\|_2^2 \rightarrow \min_{\theta}$$

This is equivalent to a denoising task:

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})} \|\mathbf{s}_{\theta,\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x})\|_2^2 \rightarrow \min_{\theta}$$

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathcal{N}(0, \mathbf{I})} \left\| \mathbf{s}_{\theta,\sigma}(\mathbf{x} + \sigma \boldsymbol{\epsilon}) + \frac{\boldsymbol{\epsilon}}{\sigma} \right\|_2^2 \rightarrow \min_{\theta}$$

## Langevin Dynamics

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \frac{\eta}{2} \cdot \mathbf{s}_{\theta,\sigma}(\mathbf{x}_l) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}_l, \quad \boldsymbol{\epsilon}_l \sim \mathcal{N}(0, \mathbf{I})$$

# Outline

## 26. Score Matching

Denoising Score Matching (continued)

Noise-Conditioned Score Network

## 27. Forward Gaussian Diffusion Process

## 28. Denoising Score Matching for Diffusion

## 29. Reverse Gaussian Diffusion Process

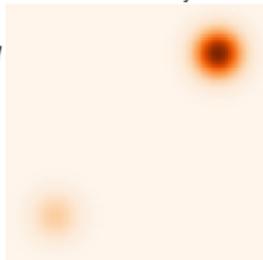
# Denoising Score Matching

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathcal{N}(0, \mathbf{I})} \left\| \mathbf{s}_{\theta, \sigma}(\mathbf{x} + \sigma \epsilon) + \frac{\epsilon}{\sigma} \right\|_2^2 \rightarrow \min_{\theta}$$

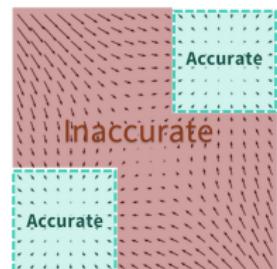
$$\mathbf{x}_{I+1} = \mathbf{x}_I + \frac{\eta}{2} \cdot \mathbf{s}_{\theta, \sigma}(\mathbf{x}_I) + \sqrt{\eta} \cdot \epsilon_I$$

- ▶ For **small**  $\sigma$ ,  $\mathbf{s}_{\theta, \sigma}(\mathbf{x})$  becomes inaccurate and Langevin dynamics fails to traverse modes
- ▶ For **large**  $\sigma$ , robustness in low-density regions is achieved, but the model learns a distribution that is overly corrupted

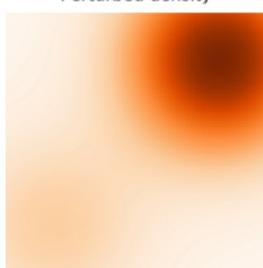
Data density



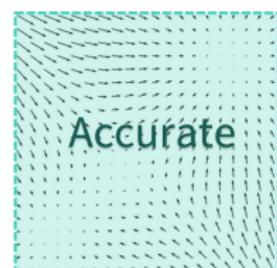
Estimated scores



Perturbed density



Estimated scores



# Noise-Conditioned Score Network (NCSN)

- ▶ Specify a sequence of noise levels:  $\sigma_1 < \sigma_2 < \dots < \sigma_T$
- ▶ Perturb each data point with different noise levels:  
 $\mathbf{x}_t = \mathbf{x} + \sigma_t \epsilon$ , so  $\mathbf{x}_t \sim q(\mathbf{x}_t)$
- ▶ Choose  $\sigma_1, \sigma_T$  such that:

$$q(\mathbf{x}_1) \approx p_{\text{data}}(\mathbf{x}), \quad q(\mathbf{x}_T) \approx \mathcal{N}(0, \sigma_T^2 \mathbf{I})$$

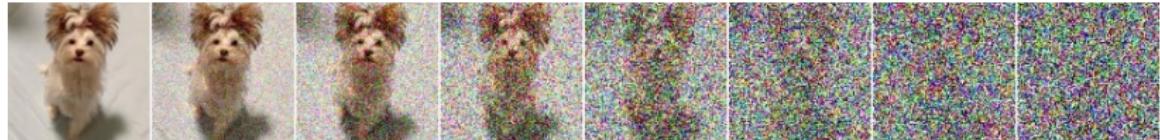
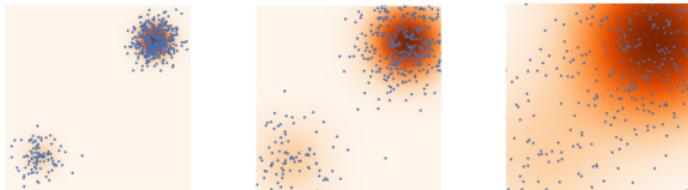
$$\sigma_1$$

$$<$$

$$\sigma_2$$

$$<$$

$$\sigma_3$$



# Noise-Conditioned Score Network (NCSN)

Train the denoising score function  $s_{\theta, \sigma_t}(\mathbf{x}_t)$  for each noise level  $\sigma_t$  using a unified weighted objective:

$$\sum_{t=1}^T \sigma_t^2 \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x})} \|s_{\theta, \sigma_t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x})\|_2^2 \rightarrow \min_{\theta}$$

Here,  $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}) = -\frac{\mathbf{x}_t - \mathbf{x}}{\sigma_t^2} = -\frac{\epsilon}{\sigma_t}$

## Training

1. Sample  $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$
2. Sample  $t \sim U\{1, T\}$  and  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
3. Construct noisy image  $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \epsilon$
4. Evaluate loss  $\mathcal{L} = \sigma_t^2 \left\| s_{\theta, \sigma_t}(\mathbf{x}_t) + \frac{\epsilon}{\sigma_t} \right\|^2$

How do we sample from such a model?

# Noise-Conditioned Score Network (NCSN)

## Sampling (Annealed Langevin Dynamics)

- ▶ Sample initial point  $\mathbf{x}_0 \sim \mathcal{N}(0, \sigma_T^2 \mathbf{I}) \approx q(\mathbf{x}_T)$
- ▶ At each noise level, apply  $L$  steps of Langevin dynamics:

$$\mathbf{x}_l = \mathbf{x}_{l-1} + \frac{\eta_t}{2} \mathbf{s}_{\theta, \sigma_t}(\mathbf{x}_{l-1}) + \sqrt{\eta_t} \boldsymbol{\epsilon}_l,$$

- ▶ Update  $\mathbf{x}_0 := \mathbf{x}_L$  and reduce to the next lower  $\sigma_t$



# Outline

## 26. Score Matching

- Denoising Score Matching (continued)
- Noise-Conditioned Score Network

## 27. Forward Gaussian Diffusion Process

## 28. Denoising Score Matching for Diffusion

## 29. Reverse Gaussian Diffusion Process

# Forward Gaussian Diffusion Process

Let  $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ ,  $\beta_t \ll 1$ . Define a Markov chain:

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

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

## Langevin Dynamics

$$\mathbf{x}_{I+1} = \mathbf{x}_I + \frac{\eta}{2} \cdot \nabla_{\mathbf{x}_I} \log p_{\theta}(\mathbf{x}_I) + \sqrt{\eta} \boldsymbol{\epsilon}_I, \quad \boldsymbol{\epsilon}_I \sim \mathcal{N}(0, \mathbf{I})$$

$$\begin{aligned} \mathbf{x}_t &= \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}_t \approx \left(1 - \frac{\beta_t}{2}\right) \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}_t = \\ &= \mathbf{x}_{t-1} + \frac{\beta_t}{2} (-\mathbf{x}_{t-1}) + \sqrt{\beta_t} \boldsymbol{\epsilon}_t \end{aligned}$$

- ▶  $\beta_t = \eta$
- ▶  $\nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1} | \theta) = -\mathbf{x}_{t-1} = \nabla_{\mathbf{x}_{t-1}} \log \mathcal{N}(0, \mathbf{I})$

# Forward Gaussian Diffusion Process

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

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

## Statement 1

Let  $\alpha_t = 1 - \beta_t$  and  $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s = \prod_{s=1}^t (1 - \beta_s)$ . Then

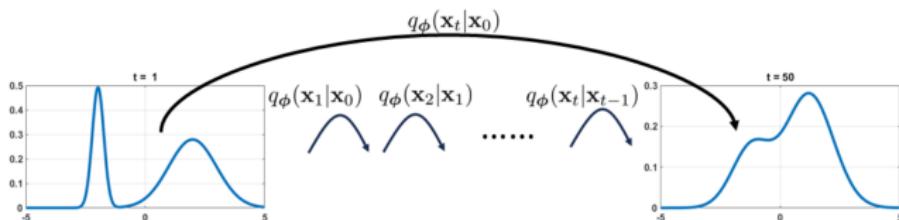
$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$$

Thus, samples at any timestep  $t$  can be generated directly from  $\mathbf{x}_0$

$$\begin{aligned}\mathbf{x}_t &= \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_t = \\ &= \sqrt{\alpha_t} (\sqrt{\alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_{t-1}} \boldsymbol{\epsilon}_{t-1}) + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_t = \\ &= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + (\sqrt{\alpha_t (1 - \alpha_{t-1})} \boldsymbol{\epsilon}_{t-1} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_t) = \\ &= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \boldsymbol{\epsilon}'_t = \\ &= \dots = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})\end{aligned}$$

# Forward Gaussian Diffusion Process

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N} \left( \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I} \right); \quad q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N} \left( \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I} \right)$$



## Statement 2

Applying the Markov chain to any distribution  $p_{\text{data}}(\mathbf{x})$  yields  $\mathbf{x}_\infty \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$ , the **stationary** (limiting) distribution:

$$p_\infty(\mathbf{x}) = \int q(\mathbf{x} | \mathbf{x}') p_\infty(\mathbf{x}') d\mathbf{x}'$$

$$p_\infty(\mathbf{x}) = \int q(\mathbf{x}_\infty | \mathbf{x}_0) p_{\text{data}}(\mathbf{x}_0) d\mathbf{x}_0 \approx \mathcal{N}(0, \mathbf{I}) \int p_{\text{data}}(\mathbf{x}_0) d\mathbf{x}_0 = \mathcal{N}(0, \mathbf{I})$$

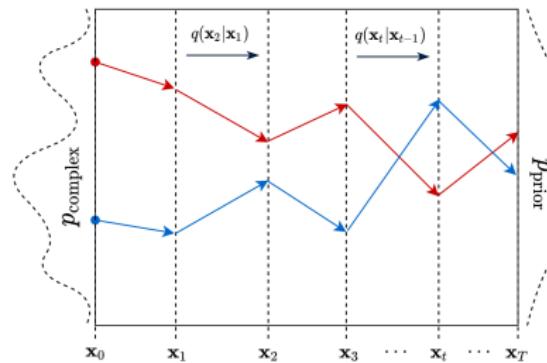
---

Chan S. Tutorial on Diffusion Models for Imaging and Vision, 2024

Sohl-Dickstein J. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, 2015

# Forward Gaussian Diffusion Process

**Diffusion** describes the migration of particles from regions of high density to those of low density.



1.  $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$
2.  $\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), t \geq 1$
3. After  $T \gg 1$  steps:  $\mathbf{x}_T \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$

If this process can be reversed, we can sample from  $p_{\text{data}}(\mathbf{x})$  by starting from noise  $p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$ .

Our goal now becomes inverting this diffusion.

# Outline

## 26. Score Matching

- Denoising Score Matching (continued)
- Noise-Conditioned Score Network

## 27. Forward Gaussian Diffusion Process

## 28. Denoising Score Matching for Diffusion

## 29. Reverse Gaussian Diffusion Process

# Denoising Score Matching

## NCSN

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0, \sigma_t^2 \mathbf{I}), \quad q(\mathbf{x}_1) \approx p_{\text{data}}(\mathbf{x}), \quad q(\mathbf{x}_T) \approx \mathcal{N}(0, \sigma_T^2 \mathbf{I})$$

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}) = -\frac{\mathbf{x}_t - \mathbf{x}}{\sigma_t^2}$$

## Gaussian Diffusion

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}), \quad q(\mathbf{x}_1) \approx p_{\text{data}}(\mathbf{x}), \quad q(\mathbf{x}_T) \approx \mathcal{N}(0, \mathbf{I})$$

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0) = -\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0}{1 - \bar{\alpha}_t}$$

## Theorem (Denoising Score Matching)

$$\begin{aligned} \mathbb{E}_{q(\mathbf{x}_t)} \|\mathbf{s}_{\theta,t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)\|_2^2 &= \\ &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x})} \|\mathbf{s}_{\theta,t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x})\|_2^2 + \text{const}(\theta) \end{aligned}$$

**Note:** This enables applying the NCSN approach with annealed Langevin dynamics to diffusion-based denoising models.

# Outline

## 26. Score Matching

Denoising Score Matching (continued)

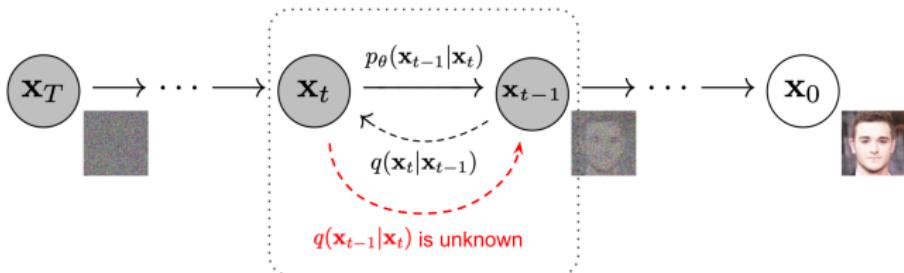
Noise-Conditioned Score Network

## 27. Forward Gaussian Diffusion Process

## 28. Denoising Score Matching for Diffusion

## 29. Reverse Gaussian Diffusion Process

# Reverse Gaussian Diffusion Process



Forward Process

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

Reverse Process

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

$q(\mathbf{x}_{t-1})$  and  $q(\mathbf{x}_t)$  are intractable:

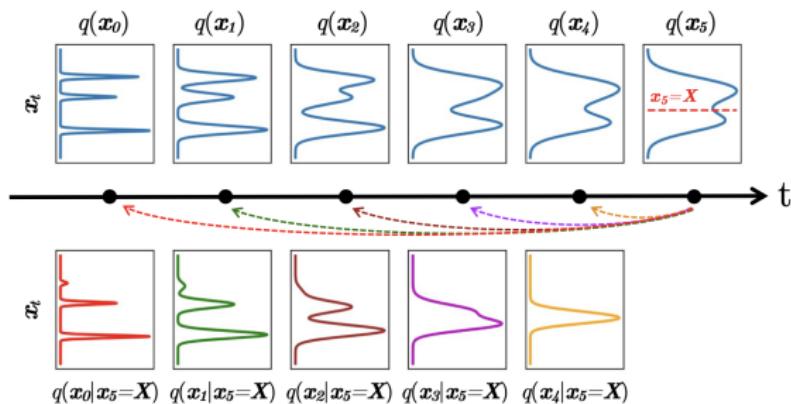
$$q(\mathbf{x}_t) = \int q(\mathbf{x}_t | \mathbf{x}_0) p_{\text{data}}(\mathbf{x}_0) d\mathbf{x}_0$$

# Reverse Gaussian Diffusion Process

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1})}{q(\mathbf{x}_t)}$$

Theorem (Feller, 1949)

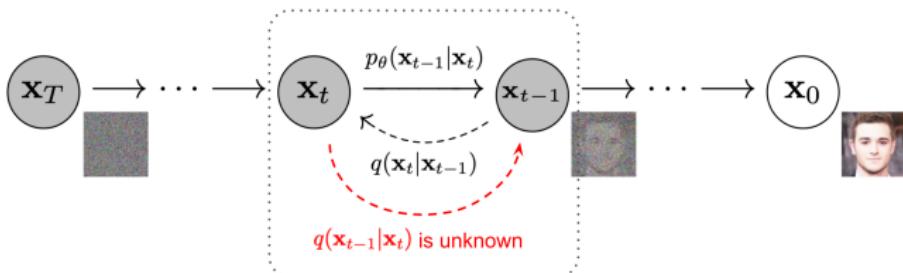
If  $\beta_t$  is sufficiently small,  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  is Gaussian (thus, diffusion requires  $T \approx 1000$  steps for convergence)



Feller W. On the theory of stochastic processes, with particular reference to applications, 1949

Xiao Z., Kreis K., Vahdat A. Tackling the generative learning trilemma with denoising diffusion GANs, 2021

# Reverse Gaussian Diffusion Process (Ancestral Sampling)



Define the reverse process as:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) \approx p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mu_{\theta,t}(\mathbf{x}_t), \sigma_{\theta,t}^2(\mathbf{x}_t))$$

Feller's theorem justifies this Gaussian assumption.

## Forward Process

1.  $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$
2.  $\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}$
3.  $\mathbf{x}_T \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$

## Reverse Process

1.  $\mathbf{x}_T \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$
2.  $\mathbf{x}_{t-1} = \sigma_{\theta,t}(\mathbf{x}_t) \boldsymbol{\epsilon} + \mu_{\theta,t}(\mathbf{x}_t)$
3.  $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$

**Note:** The forward process is non-learnable, i.e., it does not involve trainable parameters

## Summary

- ▶ Denoising score matching minimizes the Fisher divergence on corrupted samples, making the divergence estimable via sampling
- ▶ The noise-conditioned score network leverages a range of noise levels and annealed Langevin dynamics to learn the score function and enable sampling
- ▶ The Gaussian diffusion process is a Markov chain that incrementally corrupts data with carefully structured Gaussian noise
- ▶ Denoising score matching, together with Langevin dynamics, can be applied to the Gaussian diffusion process
- ▶ The reverse process reconstructs data from noise samples, although its precise form is intractable

# Deep Generative Models

## Lecture 8

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

30. Conditioned Reverse Distribution
31. Gaussian Diffusion Model as VAE
32. ELBO Derivation
33. Reparametrization
34. Denoising Diffusion Probabilistic Model (DDPM)

# Outline

30. Conditioned Reverse Distribution

31. Gaussian Diffusion Model as VAE

32. ELBO Derivation

33. Reparametrization

34. Denoising Diffusion Probabilistic Model (DDPM)

# Conditioned Reverse Distribution

## Reverse Kernel (**Intractable**)

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})\cancel{q(\mathbf{x}_{t-1})}}{q(\mathbf{x}_t)}$$

## Conditioned Reverse Kernel (**Tractable**)

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\ &= \frac{\mathcal{N}(\sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \cdot \mathcal{N}(\sqrt{\bar{\alpha}_{t-1}} \cdot \mathbf{x}_0, (1-\bar{\alpha}_{t-1}) \cdot \mathbf{I})}{\mathcal{N}(\sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0, (1-\bar{\alpha}_t) \cdot \mathbf{I})} \\ &= \mathcal{N}(\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\boldsymbol{\beta}}_t \cdot \mathbf{I}) \end{aligned}$$

Here,

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \cdot \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t} \cdot \mathbf{x}_0; \\ \tilde{\boldsymbol{\beta}}_t &= \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} = \text{const.} \end{aligned}$$

## Distribution Summary

**Forward process** maps any distribution  $p_{\text{data}}(\mathbf{x})$  to  $\mathcal{N}(0, \mathbf{I})$  by injection of noise:

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

**Reverse process** refers to an intractable distribution that can be approximated by a normal distribution (with unknown parameters) for small  $\beta_t$ :

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t) = \frac{q(\mathbf{x}_t | \mathbf{x}_{t-1}) q(\mathbf{x}_{t-1})}{q(\mathbf{x}_t)} \approx \mathcal{N}(\boldsymbol{\mu}_{\theta,t}(\mathbf{x}_t), \boldsymbol{\sigma}_{\theta,t}^2(\mathbf{x}_t))$$

**Conditioned reverse process** is a normal distribution with known parameters, describing how to denoise a noisy image  $\mathbf{x}_t$  when we know the clean image  $\mathbf{x}_0$ .

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

# Outline

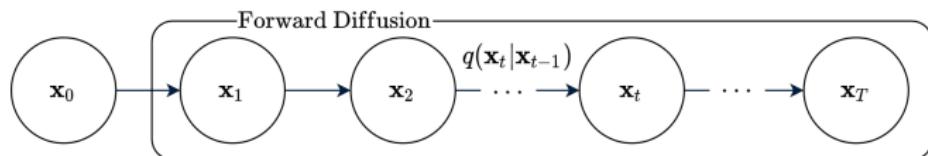
30. Conditioned Reverse Distribution
31. Gaussian Diffusion Model as VAE
32. ELBO Derivation
33. Reparametrization
34. Denoising Diffusion Probabilistic Model (DDPM)

# Gaussian Diffusion Model as VAE

Let's treat  $\mathbf{z} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$  as a latent variable (**note:** each  $\mathbf{x}_t$  has the same dimension), and  $\mathbf{x} = \mathbf{x}_0$  as the observed variable.

## Latent Variable Model

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



## Forward Diffusion

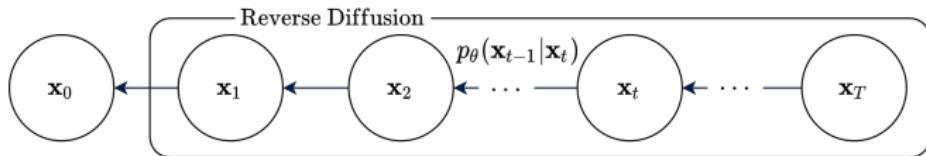
- ▶ Variational posterior distribution (encoder)

$$q(\mathbf{z}|\mathbf{x}) = q(\mathbf{x}_1, \dots, \mathbf{x}_T | \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}).$$

- ▶ **Note:** there are no learnable parameters.

# Gaussian Diffusion Model as VAE

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



## Reverse Diffusion

- ▶ Generative distribution (decoder)

$$p_{\theta}(\mathbf{x}|\mathbf{z}) = p_{\theta}(\mathbf{x}_0|\mathbf{x}_1).$$

- ▶ Prior distribution

$$p_{\theta}(\mathbf{z}) = p_{\theta}(\mathbf{x}_1, \dots, \mathbf{x}_T) = \prod_{t=2}^T p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) \cdot p(\mathbf{x}_T).$$

**Note:** This differs from the vanilla VAE due to the complex decoder  $p_{\theta}(\mathbf{x}|\mathbf{z})$  and the standard normal prior  $p(\mathbf{z})$ .

# Outline

30. Conditioned Reverse Distribution
31. Gaussian Diffusion Model as VAE
32. ELBO Derivation
33. Reparametrization
34. Denoising Diffusion Probabilistic Model (DDPM)

# ELBO for Gaussian Diffusion Model

## Standard ELBO

$$\log p_{\theta}(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} = \mathcal{L}_{\phi, \theta}(\mathbf{x}) \rightarrow \max_{q, \theta}$$

## Derivation

$$\begin{aligned}\mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p_{\theta}(\mathbf{x}_0, \mathbf{x}_{1:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \\ &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T) \prod_{t=1}^T p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}\end{aligned}$$

- ▶ Let's try to decompose the ELBO into individual KL divergence terms.
- ▶ We need to replace  $q(\mathbf{x}_t|\mathbf{x}_{t-1})$  with  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  in the denominator.
- ▶ Let's condition on  $\mathbf{x}_0$  to make the reverse  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  tractable.

# ELBO for Gaussian Diffusion Model

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) q(\mathbf{x}_t | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_0)}$$

Derivation (continued)

$$\begin{aligned}\mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \log \frac{p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{\prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1})} \\&= \mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \log \frac{p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{\prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0)} \\&= \mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \log \frac{p(\mathbf{x}_T) p_\theta(\mathbf{x}_0 | \mathbf{x}_1) \prod_{t=2}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_1 | \mathbf{x}_0) \prod_{t=2}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0)} \\&= \mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \log \frac{p(\mathbf{x}_T) p_\theta(\mathbf{x}_0 | \mathbf{x}_1) \prod_{t=2}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_1 | \mathbf{x}_0) \prod_{t=2}^T \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) q(\mathbf{x}_t | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_0)}} \\&= \mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \log \frac{p(\mathbf{x}_T) p_\theta(\mathbf{x}_0 | \mathbf{x}_1) \prod_{t=2}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_T | \mathbf{x}_0) \prod_{t=2}^T q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}\end{aligned}$$

# ELBO for Gaussian Diffusion Model

## Derivation (continued)

$$\begin{aligned}\mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T) p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) \prod_{t=2}^T p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_T|\mathbf{x}_0) \prod_{t=2}^T q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} = \\ &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) + \log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} + \sum_{t=2}^T \log \left( \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \right) \right] = \\ &= \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) + \mathbb{E}_{q(\mathbf{x}_T|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} + \\ &\quad + \sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{x}_0)} \log \left( \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \right) = \\ &= \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) - \text{KL}(q(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T)) - \\ &\quad - \sum_{t=2}^T \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \text{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\mathcal{L}_t}\end{aligned}$$

# ELBO for Gaussian Diffusion Model

$$\begin{aligned}\mathcal{L}_{\phi, \theta}(\mathbf{x}) = & \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) - \text{KL}(q(\mathbf{x}_T|\mathbf{x}_0)\|p(\mathbf{x}_T)) - \\ & - \sum_{t=2}^T \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \text{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\|p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\mathcal{L}_t}\end{aligned}$$

- ▶ First term is the decoder distribution

$$\log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) = \log \mathcal{N}(\mathbf{x}_0|\boldsymbol{\mu}_{\theta,t}(\mathbf{x}_1), \boldsymbol{\sigma}_{\theta,t}^2(\mathbf{x}_1)),$$

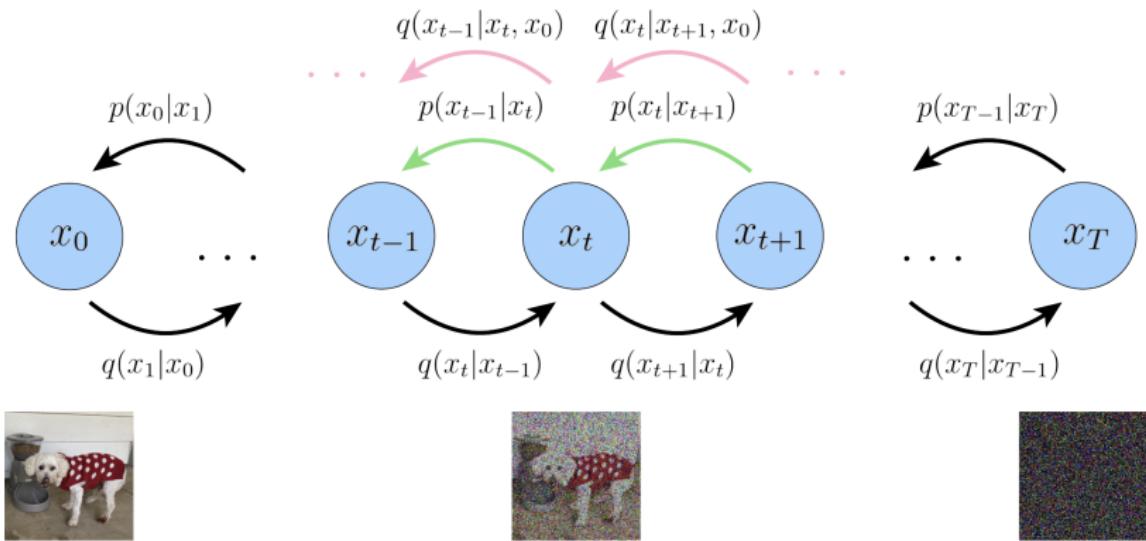
with  $\mathbf{x}_1 \sim q(\mathbf{x}_1|\mathbf{x}_0)$ .

- ▶ Second term is constant:

- ▶  $p(\mathbf{x}_T) = \mathcal{N}(0, \mathbf{I})$ ;
- ▶  $q(\mathbf{x}_T|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_T} \cdot \mathbf{x}_0, (1 - \bar{\alpha}_T) \cdot \mathbf{I})$ .

- ▶ Third term is the main contributor to the ELBO.

# ELBO for Gaussian Diffusion Model



$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \text{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))$$

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

$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1} | \boldsymbol{\mu}_{\theta,t}(\mathbf{x}_t), \boldsymbol{\sigma}_{\theta,t}^2(\mathbf{x}_t))$$

## ELBO for Gaussian Diffusion Model

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \text{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))$$

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

$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1} | \mu_{\theta,t}(\mathbf{x}_t), \sigma_{\theta,t}^2(\mathbf{x}_t))$$

Let's assume that

$$\sigma_{\theta,t}^2(\mathbf{x}_t) = \tilde{\beta}_t \mathbf{I} \quad \Rightarrow \quad p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1} | \mu_{\theta,t}(\mathbf{x}_t), \tilde{\beta}_t \mathbf{I}).$$

Theoretically, the optimal  $\sigma_{\theta,t}^2(\mathbf{x}_t)$  lies in  $[\tilde{\beta}_t, \beta_t]$ :

- ▶  $\beta_t$  is optimal for  $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$ ;
- ▶  $\tilde{\beta}_t$  is optimal for  $\mathbf{x}_0 \sim \delta(\mathbf{x}_0 - \mathbf{x}^*)$ .

$$\begin{aligned}\mathcal{L}_t &= \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \text{KL}\left(\mathcal{N}(\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}) \| \mathcal{N}(\mu_{\theta,t}(\mathbf{x}_t), \tilde{\beta}_t \mathbf{I})\right) \\ &= \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[ \frac{1}{2\tilde{\beta}_t} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_{\theta,t}(\mathbf{x}_t)\|^2 \right]\end{aligned}$$

# ELBO for Gaussian Diffusion Model

## Training

1. Obtain a sample  $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$ .
2. Generate a noisy image  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \boldsymbol{\epsilon}$ , with  $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ .
3. Compute the ELBO

$$\begin{aligned}\mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{x}_1 | \mathbf{x}_0)} \log p_{\theta}(\mathbf{x}_0 | \mathbf{x}_1) - \text{KL}(q(\mathbf{x}_T | \mathbf{x}_0) \| p(\mathbf{x}_T)) - \\ &\quad - \underbrace{\sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \left[ \frac{1}{2\tilde{\beta}_t} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_{\theta, t}(\mathbf{x}_t)\|^2 \right]}_{\mathcal{L}_t}\end{aligned}$$

## Sampling

1. Sample  $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ .
2. Denoise:  $\mathbf{x}_{t-1} = \mu_{\theta, t}(\mathbf{x}_t) + \sqrt{\tilde{\beta}_t} \cdot \boldsymbol{\epsilon}$ ,  $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ .

# Outline

30. Conditioned Reverse Distribution
31. Gaussian Diffusion Model as VAE
32. ELBO Derivation
33. Reparametrization
34. Denoising Diffusion Probabilistic Model (DDPM)

# Reparametrization of DDPM

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

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

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

- ▶ There is a linear relationship between  $\epsilon$ ,  $\mathbf{x}_t$ , and  $\mathbf{x}_0$ .
- ▶ Let's try to rewrite this mean using only  $\mathbf{x}_t$  and  $\epsilon$ .

$$\begin{aligned}\tilde{\mu}_t(\mathbf{x}_t, \epsilon) &= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \cdot \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)}{1 - \bar{\alpha}_t} \cdot \left( \frac{\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon}{\sqrt{\bar{\alpha}_t}} \right) \\ &= \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}} \cdot \epsilon\end{aligned}$$

# Reparametrization of DDPM

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

## Reparametrization

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}} \cdot \epsilon$$

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

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

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

At every step of the reverse process, we attempt to predict the noise  $\epsilon$  that was used in the forward diffusion process!

# Reparametrization of DDPM

$$\begin{aligned}\mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) - \text{KL}(q(\mathbf{x}_T|\mathbf{x}_0)\|p(\mathbf{x}_T)) - \\ &\quad - \sum_{t=2}^{\top} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \text{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\|p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\mathcal{L}_t} \\ \mathcal{L}_t &= \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \mathbf{I})} \left[ \frac{(1 - \alpha_t)^2}{2\tilde{\beta}_t \alpha_t (1 - \bar{\alpha}_t)} \left\| \epsilon - \epsilon_{\theta, t}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon) \right\|^2 \right]\end{aligned}$$

Let's drop the scaling coefficient.

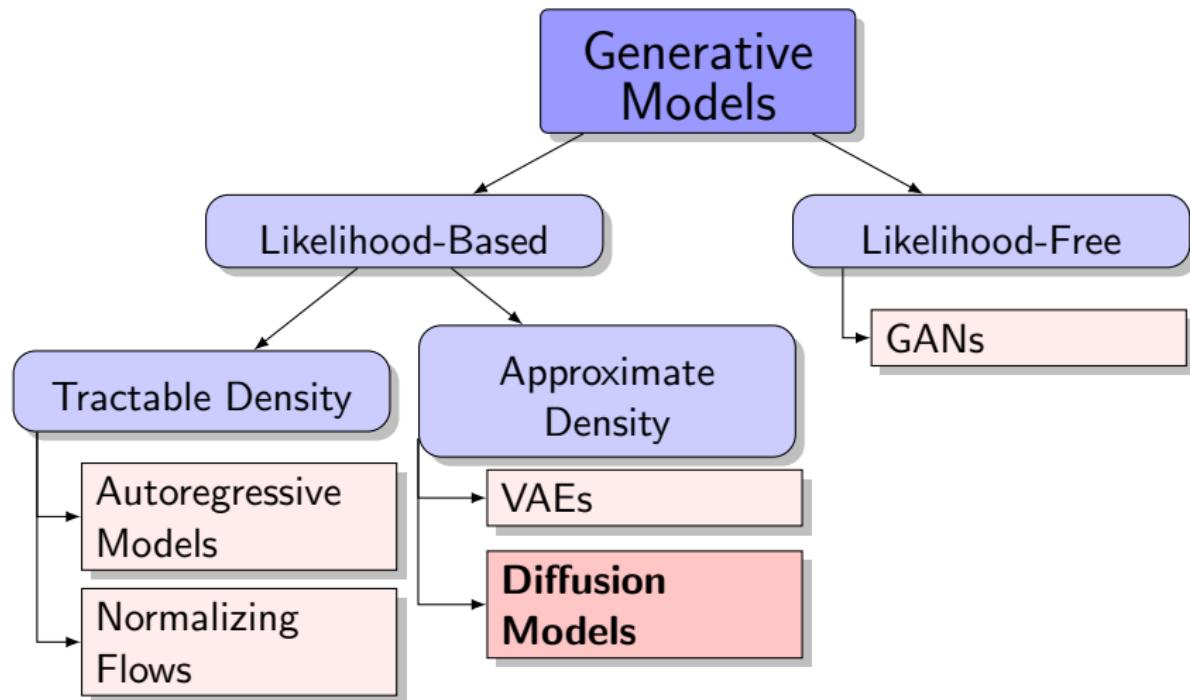
## Simplified Objective

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

# Outline

30. Conditioned Reverse Distribution
31. Gaussian Diffusion Model as VAE
32. ELBO Derivation
33. Reparametrization
34. Denoising Diffusion Probabilistic Model (DDPM)

# Generative Models Zoo



# Denoising Diffusion Probabilistic Model (DDPM)

## DDPM is a VAE Model

- ▶ The encoder is a fixed Gaussian Markov chain  $q(\mathbf{x}_1, \dots, \mathbf{x}_T | \mathbf{x}_0)$ .
- ▶ The latent variable is hierarchical (at each step, its dimension equals the input's).
- ▶ The decoder is a simple Gaussian model  $p_\theta(\mathbf{x}_0 | \mathbf{x}_1)$ .
- ▶ The prior distribution is given by a parametric Gaussian Markov chain  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$ .

### Forward Process

1.  $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ ;
2.  $\mathbf{x}_t = \sqrt{1 - \beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}$ ;
3.  $\mathbf{x}_T \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$ .

### Reverse Process

1.  $\mathbf{x}_T \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$ ;
2.  $\mathbf{x}_{t-1} = \sigma_{\theta, t}(\mathbf{x}_t) \cdot \boldsymbol{\epsilon} + \mu_{\theta, t}(\mathbf{x}_t)$ ;
3.  $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ ;

# Denoising Diffusion Probabilistic Model (DDPM)

## Training

1. Obtain a sample  $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$ .
2. Sample time index  $t \sim U\{1, T\}$  and noise  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ .
3. Generate noisy image  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon$ .
4. Compute the loss  $\mathcal{L}_{\text{simple}} = \|\epsilon - \epsilon_{\theta,t}(\mathbf{x}_t)\|^2$ .

## Sampling (Ancestral Sampling)

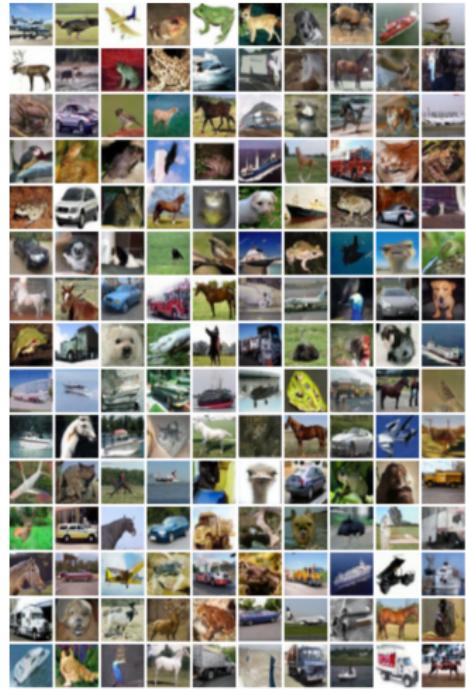
1. Sample  $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ .
2. Compute the mean of  $p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\boldsymbol{\mu}_{\theta,t}(\mathbf{x}_t), \tilde{\beta}_t \cdot \mathbf{I})$ :

$$\boldsymbol{\mu}_{\theta,t}(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}} \cdot \epsilon_{\theta,t}(\mathbf{x}_t)$$

3. Denoise:  $\mathbf{x}_{t-1} = \boldsymbol{\mu}_{\theta,t}(\mathbf{x}_t) + \sqrt{\tilde{\beta}_t} \cdot \epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I})$ .

# Denoising Diffusion Probabilistic Model (DDPM)

## Samples



## Summary

- ▶ DDPM approximates the reverse process using normality assumptions.
- ▶ DDPM can be interpreted as a VAE with a hierarchy of latent variables.
- ▶ The ELBO for DDPM may be formulated as a sum over many KL divergence terms.
- ▶ At each step, DDPM predicts the noise that was injected in the forward process.
- ▶ DDPM is a VAE model that tries to invert the forward diffusion process via variational inference.
- ▶ DDPMs are quite slow, since the model must be applied  $T$  times for sampling.

# Deep Generative Models

## Lecture 9

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

## Recap of Lecture 7

Let us perturb the original data with Gaussian noise  
 $q(\mathbf{x}_\sigma | \mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma^2 \cdot \mathbf{I})$ .

$$q(\mathbf{x}_\sigma) = \int q(\mathbf{x}_\sigma | \mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x}.$$

Then the solution of

$$\frac{1}{2} \mathbb{E}_{q(\mathbf{x}_\sigma)} \| \mathbf{s}_{\theta, \sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma) \|_2^2 \rightarrow \min_{\theta}$$

satisfies  $\mathbf{s}_{\theta, \sigma}(\mathbf{x}_\sigma) \approx \mathbf{s}_{\theta, 0}(\mathbf{x}_0) = \mathbf{s}_\theta(\mathbf{x})$  if  $\sigma$  is sufficiently small.

**Theorem (Denoising Score Matching)**

$$\begin{aligned} & \mathbb{E}_{q(\mathbf{x}_\sigma)} \| \mathbf{s}_{\theta, \sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma) \|_2^2 = \\ &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_\sigma | \mathbf{x})} \| \mathbf{s}_{\theta, \sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma | \mathbf{x}) \|_2^2 + \text{const}(\theta) \end{aligned}$$

Here,  $\nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma | \mathbf{x}) = -\frac{\mathbf{x}_\sigma - \mathbf{x}}{\sigma^2} = -\frac{\epsilon}{\sigma}$ .  $\mathbf{s}_{\theta, \sigma}(\mathbf{x}_\sigma)$  attempts to **denoise** a corrupted sample.

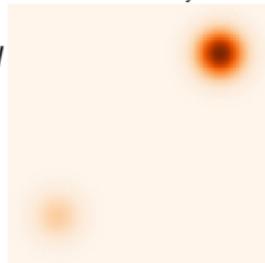
## Recap of Lecture 7

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathcal{N}(0, \mathbf{I})} \left\| \mathbf{s}_{\theta, \sigma}(\mathbf{x} + \sigma \boldsymbol{\epsilon}) + \frac{\boldsymbol{\epsilon}}{\sigma} \right\|_2^2 \rightarrow \min_{\theta}$$

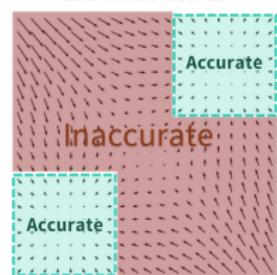
$$\mathbf{x}_{I+1} = \mathbf{x}_I + \frac{\eta}{2} \cdot \mathbf{s}_{\theta, \sigma}(\mathbf{x}_I) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}_I$$

- ▶ For **small**  $\sigma$ ,  $\mathbf{s}_{\theta, \sigma}(\mathbf{x})$  becomes inaccurate and Langevin dynamics fails to traverse modes
- ▶ For **large**  $\sigma$ , robustness in low-density regions is achieved, but the model learns a distribution that is overly corrupted

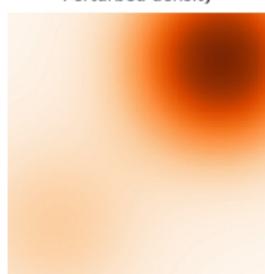
Data density



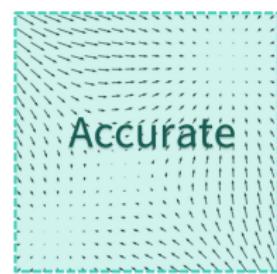
Estimated scores



Perturbed density



Estimated scores



# Recap of Lecture 7

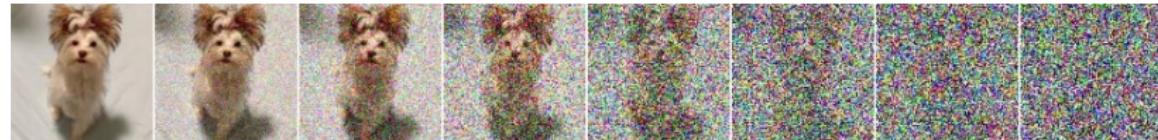
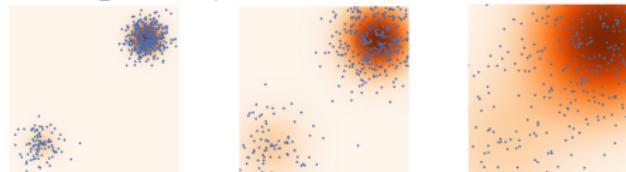
## Noise-Conditioned Score Network

- ▶ Define a sequence of noise levels:  $\sigma_1 < \sigma_2 < \dots < \sigma_T$ .
- ▶ Train a denoised score function  $s_{\theta, \sigma_t}(\mathbf{x}_t)$  for each noise level:

$$\sum_{t=1}^T \sigma_t^2 \cdot \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x})} \| s_{\theta, \sigma_t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}) \|_2^2 \rightarrow \min_{\theta}$$

- ▶ Sample using **annealed** Langevin dynamics (for  $t = 1, \dots, T$ ).

$$\sigma_1 < \sigma_2 < \sigma_3$$



## Recap of Lecture 7

### NCSN Training

1. Obtain a sample  $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$ .
2. Sample noise level  $t \sim U\{1, T\}$  and noise  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ .
3. Construct noisy image  $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \cdot \epsilon$ .
4. Compute the loss  $\mathcal{L} = \sigma_t^2 \cdot \|\mathbf{s}_{\theta, \sigma_t}(\mathbf{x}_t) + \frac{\epsilon}{\sigma_t}\|^2$ .

### NCSN Sampling (Annealed Langevin Dynamics)

- ▶ Sample  $\mathbf{x}_0 \sim \mathcal{N}(0, \sigma_T^2 \cdot \mathbf{I}) \approx q(\mathbf{x}_T)$ .
- ▶ Apply  $L$  steps of Langevin dynamics:

$$\mathbf{x}_l = \mathbf{x}_{l-1} + \frac{\eta_t}{2} \cdot \mathbf{s}_{\theta, \sigma_t}(\mathbf{x}_{l-1}) + \sqrt{\eta_t} \cdot \epsilon_l.$$

- ▶ Update  $\mathbf{x}_0 := \mathbf{x}_L$  and proceed to the next  $\sigma_t$ .

# Recap of Lecture 7

## Forward Gaussian Diffusion Process

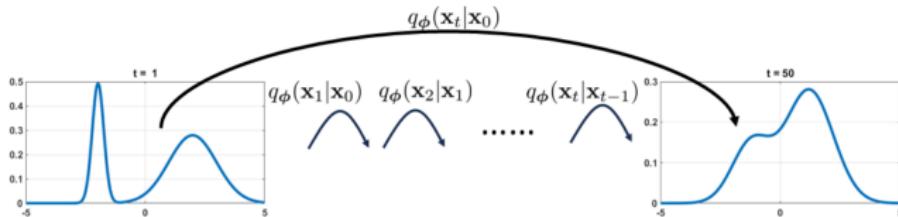
Let  $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ ,  $\beta_t \ll 1$ ,  $\alpha_t = 1 - \beta_t$  and  $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$ .

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

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

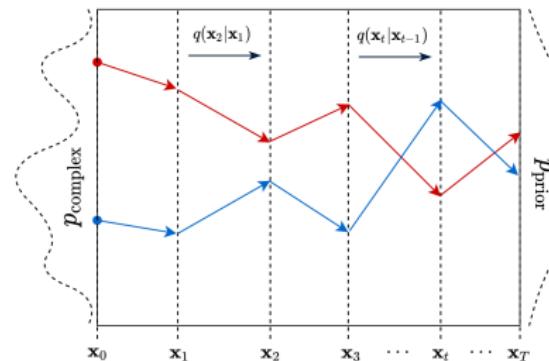
$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1 - \beta_t} \cdot \mathbf{x}_{t-1}, \beta_t \cdot \mathbf{I});$$

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



## Recap of Lecture 7

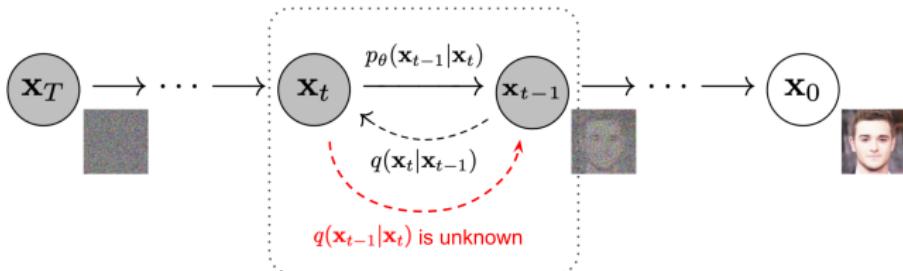
**Diffusion** describes the process where particles migrate from regions of high density to regions of low density.



1.  $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ ;
2.  $\mathbf{x}_t = \sqrt{1 - \beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}$ , with  $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ ,  $t \geq 1$ ;
3.  $\mathbf{x}_T \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$ , for  $T \gg 1$ .

If we can invert this process, we would have a way to sample  $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$  using noise samples, i.e.  $p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$ . Hence, our objective becomes to reverse this process.

# Recap of Lecture 7



## Reverse Process (Ancestral Sampling)

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1})}{q(\mathbf{x}_t)} \approx p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mu_{\theta,t}(\mathbf{x}_t), \sigma_{\theta,t}^2(\mathbf{x}_t))$$

The Feller theorem guarantees this approximation is valid.

## Forward Process

1.  $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x});$
2.  $\mathbf{x}_t = \sqrt{1 - \beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \epsilon;$
3.  $\mathbf{x}_T \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I}).$

## Reverse Process

1.  $\mathbf{x}_T \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I});$
2.  $\mathbf{x}_{t-1} = \sigma_{\theta,t}(\mathbf{x}_t) \cdot \epsilon + \mu_{\theta,t}(\mathbf{x}_t);$
3.  $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x});$

# Outline

35. DDPM as a Score-Based Generative Model

36. Guidance

Classifier Guidance

Classifier-Free Guidance

37. Continuous-Time Normalizing Flows

# Outline

35. DDPM as a Score-Based Generative Model

36. Guidance

Classifier Guidance

Classifier-Free Guidance

37. Continuous-Time Normalizing Flows

# Denoising Diffusion as a Score-Based Generative Model

## DDPM Objective

$$\begin{aligned}\mathcal{L}_t &= \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \mathbf{I})} \left[ C_{1,t} \cdot \|\epsilon_{\theta,t}(\mathbf{x}_t) - \epsilon\|_2^2 \right] \\ &= \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \mathbf{I})} \left[ C_{2,t} \cdot \left\| \frac{\epsilon_{\theta,t}(\mathbf{x}_t)}{\sqrt{1 - \bar{\alpha}_t}} - \frac{\epsilon}{\sqrt{1 - \bar{\alpha}_t}} \right\|_2^2 \right]\end{aligned}$$

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

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0) = -\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0}{1 - \bar{\alpha}_t} = -\frac{\epsilon}{\sqrt{1 - \bar{\alpha}_t}}.$$

We can reparameterize the model as:

$$\mathbf{s}_{\theta,t}(\mathbf{x}_t) = \frac{\epsilon_{\theta,t}(\mathbf{x}_t)}{\sqrt{1 - \bar{\alpha}_t}} = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t).$$

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \left[ C_{2,t} \cdot \left\| \mathbf{s}_{\theta,t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0) \right\|_2^2 \right]$$

# DDPM vs NCSN: Objectives

## DDPM Objective

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}_0)} \mathbb{E}_{t \sim U\{1, T\}} \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \left[ C_{2,t} \left\| \mathbf{s}_{\theta,t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0) \right\|_2^2 \right]$$

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

In practice, this coefficient is often omitted.

## NCSN Objective

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}_0)} \mathbb{E}_{t \sim U\{1, T\}} \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \left\| \mathbf{s}_{\theta, \sigma_t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0) \right\|_2^2$$

$$\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \cdot \boldsymbol{\epsilon}$$

**Maximizing the ELBO leads to the same objective as denoising score matching!**

# DDPM vs NCSN: Sampling

## DDPM Sampling (Ancestral Sampling)

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

$$\begin{aligned}\mathbf{x}_{t-1} &= \boldsymbol{\mu}_{\theta,t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon} \\&= \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}} \cdot \boldsymbol{\epsilon}_{\theta,t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon} \\&= \frac{1}{\sqrt{1 - \beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1 - \beta_t}} \cdot \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon}\end{aligned}$$

## NCSN Sampling (Annealed Langevin Dynamics)

- ▶ Sample  $\mathbf{x}_T^0 \sim \mathcal{N}(0, \sigma_T^2 \mathbf{I}) \approx q(\mathbf{x}_T)$ .
- ▶ Perform  $L$  steps of Langevin dynamics:

$$\mathbf{x}_t^l = \mathbf{x}_t^{l-1} + \frac{\eta_t}{2} \cdot \mathbf{s}_{\theta,\sigma_t}(\mathbf{x}_t^{l-1}) + \sqrt{\eta_t} \cdot \boldsymbol{\epsilon}_t^l.$$

- ▶ Set  $\mathbf{x}_{t-1}^0 = \mathbf{x}_t^L$  and move to the next  $\sigma_t$ .

# DDPM vs NCSN: Summary

## Summary

- ▶ Different Markov chains:
  - ▶ DDPM:  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \boldsymbol{\epsilon};$
  - ▶ NCSN:  $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \cdot \boldsymbol{\epsilon}.$
  - ▶ One can generalize to  $q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\alpha_t \cdot \mathbf{x}_0, \sigma_t^2 \mathbf{I}).$
- ▶ The objectives coincide: ELBO  $\equiv$  score-matching.
- ▶ The sampling procedures differ:
  - ▶ Ancestral sampling in DDPM;
  - ▶ Annealed Langevin dynamics for NCSN;
  - ▶ Hybrid approaches that combine both updates are possible.

---

Kingma D. et al. *Variational Diffusion Models*, 2021

Song Y. et al. *Score-Based Generative Modeling through Stochastic Differential Equations*, 2020

# Outline

35. DDPM as a Score-Based Generative Model

36. Guidance

Classifier Guidance

Classifier-Free Guidance

37. Continuous-Time Normalizing Flows

## Guidance

- ▶ Up to now, we have focused on **unconditional** generative models  $p_\theta(\mathbf{x})$ .
- ▶ In practice, most generative models are **conditional** (in diffusion era it is called guided):  $p_\theta(\mathbf{x}|\mathbf{y})$ .
- ▶ Here,  $\mathbf{y}$  might denote a class label or **text** (as in text-to-image tasks).



Кот ныряет в бассейн, как ребенок на обложке альбома Nevermind, реалистично



рука человека с пятью пальцами, ни четырьмя, ни шестью, а с 5 (пять) пальцами

## Conditional Models

In practice, we're typically interested in learning conditional models (sampling from conditional distribution  $p_{\text{data}}(\mathbf{x}|\mathbf{y})$ ).

- ▶  $\mathbf{y} = \emptyset, \mathbf{x} = \text{image} \Rightarrow \text{unconditional image model}$
- ▶  $\mathbf{y} = \text{class label}, \mathbf{x} = \text{image} \Rightarrow \text{class-conditional image model}$
- ▶  $\mathbf{y} = \text{text prompt}, \mathbf{x} = \text{image} \Rightarrow \text{text-to-image model}$
- ▶  $\mathbf{y} = \text{image}, \mathbf{x} = \text{image} \Rightarrow \text{image-to-image model}$
- ▶  $\mathbf{y} = \text{image}, \mathbf{x} = \text{text} \Rightarrow \text{image-to-text (image captioning) model}$
- ▶  $\mathbf{y} = \text{English text}, \mathbf{x} = \text{Russian text} \Rightarrow \text{sequence-to-sequence model (machine translation) model}$
- ▶  $\mathbf{y} = \text{sound}, \mathbf{x} = \text{text} \Rightarrow \text{speech-to-text (automatic speech recognition) model}$
- ▶  $\mathbf{y} = \text{text}, \mathbf{x} = \text{sound} \Rightarrow \text{text-to-speech model}$

# Label Guidance

**Label:** Ostrich (10th ImageNet class)



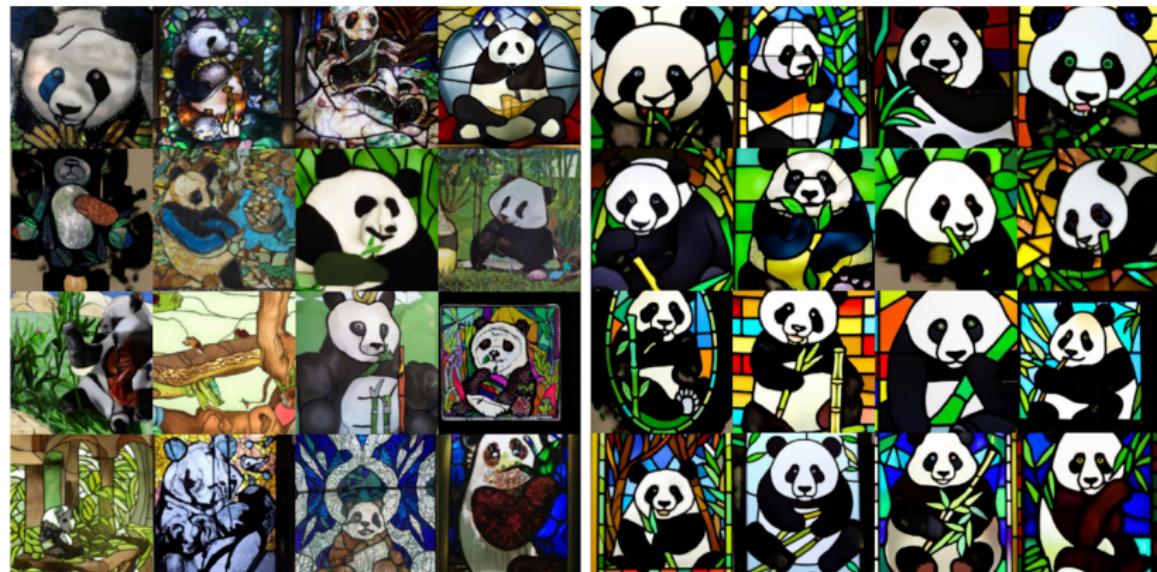
VQ-VAE (Proposed)

BigGAN deep

# Text Guidance

**Prompt:** a stained glass window of a panda eating bamboo

Left:  $\gamma = 1$ , Right:  $\gamma = 3$ .



# Guidance in Generative Models

## How to make guided model?

Instead of sampling from  $p_\theta(\mathbf{x})$ , we sample from  $p_\theta(\mathbf{x}|\mathbf{y})$ .

Given **supervised** data  $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ , we can treat  $\mathbf{y}$  as an additional model input:

- ▶  $p_\theta(x_j|\mathbf{x}_{1:j-1}, \mathbf{y})$  for AR models;
- ▶ Encoder  $q_\phi(z|\mathbf{x}, \mathbf{y})$  and decoder  $p_\theta(\mathbf{x}|z, \mathbf{y})$  for VAEs;
- ▶  $G_\theta(z, \mathbf{y})$  for NFs and GANs;
- ▶  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$  for DDPMs.

## Challenge

- ▶ Empirically, images sampled with this procedure do not fit well enough to the desired label  $\mathbf{y}$ .
- ▶ Being able to control the strength of guidance is especially valuable.

# Outline

35. DDPM as a Score-Based Generative Model

36. Guidance

Classifier Guidance

Classifier-Free Guidance

37. Continuous-Time Normalizing Flows

# Classifier Guidance

## DDPM Sampling

1. Sample  $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ .
2. Generate the denoised image (unconditional generation):

$$\begin{aligned}\mathbf{x}_{t-1} &= \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \sigma_t \cdot \epsilon \\ &= \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \sigma_t \cdot \epsilon\end{aligned}$$

## Guided Generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t | \mathbf{y}) + \sigma_t \cdot \epsilon$$

What is the link between  $\nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t)$  and  $\nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t | \mathbf{y})$ ?

# Classifier Guidance: Guided Score Function

## Guided Generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1 - \beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1 - \beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t | \mathbf{y}) + \sigma_t \cdot \epsilon$$

## Guided Generation

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t | \mathbf{y}) &= \nabla_{\mathbf{x}_t} \log \left( \frac{p_{\theta}(\mathbf{x}_t) p(\mathbf{y} | \mathbf{x}_t)}{p(\mathbf{y})} \right) \\ &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y} | \mathbf{x}_t) \\ &= \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y} | \mathbf{x}_t)\end{aligned}$$

## Guided Score Function

$$\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t | \mathbf{y}).$$

$$\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y} | \mathbf{x}_t)$$

# Classifier Guidance: Guidance Scale

## Guided Score Function

$$\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

- ▶ Let us assume  $\mathbf{y}$  is a class label.
- ▶  $p(\mathbf{y}|\mathbf{x}_t)$  is a classifier for noisy inputs.
- ▶  $p(\mathbf{y}|\mathbf{x}_t)$  is responsible for model guidance.

## Guidance Scale

It is a natural idea to scale up the contribution of the guidance

$$\mathbf{s}_{\theta,t}^{\gamma}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

- ▶ The **guidance scale**  $\gamma$  adjusts the strength of classifier guidance.
- ▶  $\mathbf{s}_{\theta,t}^{\gamma}(\mathbf{x}_t, \mathbf{y})$  is not the true guided score function  $\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y})$ .

# Classifier Guidance: Distribution Sharpening

## Scaled Guided Score Function

$$\mathbf{s}_{\theta,t}^{\gamma}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

## Scaled Conditional Distribution

$$\begin{aligned}\nabla_{\mathbf{x}_t}^{\gamma} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) \\ &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)^{\gamma} \\ &= \nabla_{\mathbf{x}_t} \log \left( \frac{p_{\theta}(\mathbf{x}_t)p(\mathbf{y}|\mathbf{x}_t)^{\gamma}}{Z} \right)\end{aligned}$$

**Note:** Increasing  $\gamma$  sharpens  $p(\mathbf{y}|\mathbf{x}_t)$ , making it more contrast

$$\hat{p}(\mathbf{y}|\mathbf{x}_t) \propto p(\mathbf{y}|\mathbf{x}_t)^{\gamma}.$$

# Classifier Guidance: Overview

## Training

- ▶ Train the DDPM as before.
- ▶ Train an additional classifier  $p(\mathbf{y}|\mathbf{x}_t)$  on noisy data (note that it is dependent on time  $t$ ).

## Guided DDPM Sampling

1. Sample  $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ .
2. Generate the denoised image (using scaled guided score function):

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1 - \beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1 - \beta_t}} \cdot \mathbf{s}_{\theta, t}^{\gamma}(\mathbf{x}_t, \mathbf{y}) + \sigma_t \cdot \boldsymbol{\epsilon}$$

# Outline

35. DDPM as a Score-Based Generative Model

36. Guidance

Classifier Guidance

Classifier-Free Guidance

37. Continuous-Time Normalizing Flows

# Classifier-Free Guidance

- ▶ The previous approach relies on training an additional classifier  $p(\mathbf{y}|\mathbf{x}_t)$  for noisy images.
- ▶ We now introduce a method to sidestep this requirement.

$$\nabla_{\mathbf{x}_t}^{\gamma} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

## Bayes theorem

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log \left( \frac{p_{\theta}(\mathbf{x}_t|\mathbf{y})p(\mathbf{y})}{p_{\theta}(\mathbf{x}_t)} \right) \\ &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t)\end{aligned}$$

## Scaled Guided Score Function

$$\begin{aligned}\nabla_{\mathbf{x}_t}^{\gamma} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) = \\ &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot (\nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t)) = \\ &= (1 - \gamma) \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y})\end{aligned}$$

# Classifier-Free Guidance: Formulation

## Scaled Guided Score Function

$$\nabla_{\mathbf{x}_t}^{\gamma} \log p_{\theta}(\mathbf{x}_t | \mathbf{y}) = (1 - \gamma) \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t | \mathbf{y})$$

$$\mathbf{s}_{\theta,t}^{\gamma}(\mathbf{x}_t, \mathbf{y}) = (1 - \gamma) \cdot \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \gamma \cdot \mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y})$$

## Naive training approach

- ▶ Train an unguided score function model  $\mathbf{s}_{\theta,t}(\mathbf{x}_t)$ .
- ▶ Train a guided score function model  $\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y})$ .
- ▶ Use their convex combination at inference.

## Guided Sampling

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1 - \beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1 - \beta_t}} \cdot \mathbf{s}_{\theta,t}^{\gamma}(\mathbf{x}_t, \mathbf{y}) + \sigma_t \cdot \epsilon$$

## How to avoid training two separate score function models?

# Classifier-Free Guidance

$$\mathbf{s}_{\theta,t}^{\gamma}(\mathbf{x}_t, \mathbf{y}) = (1 - \gamma) \cdot \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \gamma \cdot \mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y})$$

## CFG algorithm

- ▶ Introduce "the absence of conditioning" label  $\mathbf{y} = \emptyset$ .
- ▶ Use it to get unguided score function  $\mathbf{s}_{\theta,t}(\mathbf{x}_t) = \mathbf{s}_{\theta,t}(\mathbf{x}_t, \emptyset)$ .
- ▶ Train a single model  $\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y})$  using **supervised** data, but artificially drop the labels  $\mathbf{y}$  with some fixed probability (simulating the case of  $\mathbf{y} = \emptyset$ ).
- ▶ Apply the model twice during inference to get  $\mathbf{s}_{\theta,t}(\mathbf{x}_t, \emptyset)$  and  $\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y})$ .

## Guided Sampling

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1 - \beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1 - \beta_t}} \cdot \mathbf{s}_{\theta,t}^{\gamma}(\mathbf{x}_t, \mathbf{y}) + \sigma_t \cdot \boldsymbol{\epsilon}$$

# Outline

35. DDPM as a Score-Based Generative Model

36. Guidance

Classifier Guidance

Classifier-Free Guidance

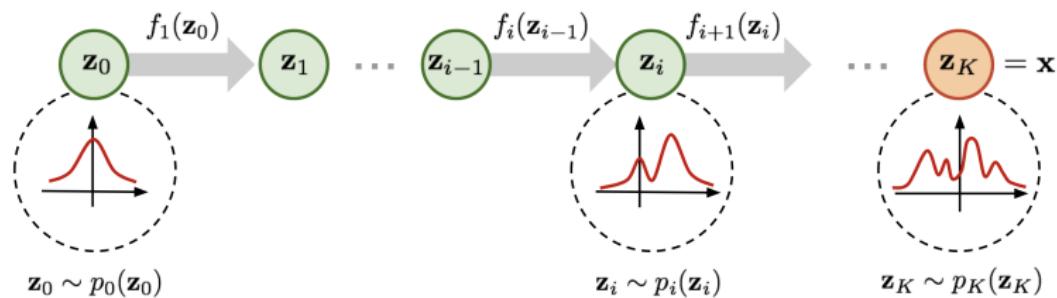
37. Continuous-Time Normalizing Flows

# Discrete-Time Normalizing Flows

## Change of Variable Theorem (CoV)

Let  $\mathbf{x}$  be a random variable with density  $p(\mathbf{x})$ , and let  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  be a differentiable and **invertible** transformation. If  $\mathbf{z} = \mathbf{f}(\mathbf{x})$ ,  $\mathbf{x} = \mathbf{f}^{-1}(\mathbf{z}) = \mathbf{g}(\mathbf{z})$ , then

$$p(\mathbf{x}) = p(\mathbf{z}) |\det(\mathbf{J}_\mathbf{f})| = p(\mathbf{z}) \left| \det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(\mathbf{f}(\mathbf{x})) \left| \det \left( \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$



$$\log p_\theta(\mathbf{x}) = \log p(\mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x})) + \sum_{k=1}^K \log \left| \det \left( \frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}} \right) \right|.$$

# Towards Continuous-Time Normalizing Flows

- ▶ Up to this point, we have considered discrete-time normalizing flows:

$$\mathbf{x}_{t+1} = \mathbf{f}_\theta(\mathbf{x}_t, t); \quad \log p(\mathbf{x}_{t+1}) = \log p(\mathbf{x}_t) - \log \left| \det \frac{\partial \mathbf{f}_\theta(\mathbf{x}_t)}{\partial \mathbf{x}_t} \right|.$$

- ▶ Let us now move to the general case of continuous time, using a mapping  $\mathbf{x}(t) : \mathbb{R} \rightarrow \mathbb{R}^m$  to describe continuous dynamics.

## Continuous-Time Dynamics

Consider an Ordinary Differential Equation (ODE):

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_\theta(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0.$$

$$\mathbf{x}(t_1) = \int_{t_0}^{t_1} \mathbf{f}_\theta(\mathbf{x}(t), t) dt + \mathbf{x}_0$$

Here,  $\mathbf{f}_\theta : \mathbb{R}^m \times [t_0, t_1] \rightarrow \mathbb{R}^m$  is a **vector field**.

# Ordinary Differential Equations (ODEs)

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_\theta(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0.$$

$$\mathbf{x}(t_1) = \int_{t_0}^{t_1} \mathbf{f}_\theta(\mathbf{x}(t), t) dt + \mathbf{x}_0$$

## Flow

Let call **the flow**  $\psi : \mathbb{R}^m \times [t_0, t_1] \rightarrow \mathbb{R}^m$  the solution of ODE:

$$\frac{d\psi_t(\mathbf{x}_0)}{dt} = \mathbf{f}_\theta(\psi_t(\mathbf{x}_0), t); \quad \text{with initial condition } \psi_0(\mathbf{x}_0) = \mathbf{x}_0.$$

## Numerical Solution of ODEs

$$\psi_t(\mathbf{x}_0) = \int_{t_0}^{t_1} \mathbf{f}_\theta(\mathbf{x}(t), t) dt + \mathbf{x}_0 \approx \texttt{ODESolve}_f(\mathbf{x}_0, \theta, t_0, t_1).$$

Here, we require the numerical routine  $\texttt{ODESolve}_f(\mathbf{x}_0, \theta, t_0, t_1)$ .

# Numerical Solution of ODEs

$$\psi_t(\mathbf{x}_0) = \int_{t_0}^{t_1} \mathbf{f}_{\theta}(\mathbf{x}(t), t) dt + \mathbf{x}_0 \approx \text{ODESolve}_f(\mathbf{x}_0, \theta, t_0, t_1).$$

`ODESolvef(x0, θ, t0, t1)` consists of sequence of iterative update steps.

## Euler Update Step

$$\frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h} = \mathbf{f}_{\theta}(\mathbf{x}(t), t)$$

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h \cdot \mathbf{f}_{\theta}(\mathbf{x}(t), t)$$

## Heun's Update Step

$$\mathbf{x}'(t+h) = \mathbf{x}(t) + h \cdot \mathbf{f}_{\theta}(\mathbf{x}(t), t)$$

$$\mathbf{x}(t+h) = \mathbf{x}(t) + \frac{h}{2} \cdot (\mathbf{f}_{\theta}(\mathbf{x}(t), t) + \mathbf{f}_{\theta}(\mathbf{x}'(t+h), t+h))$$

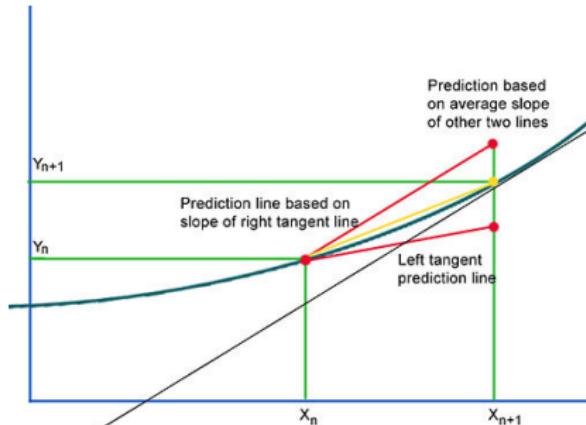


Image credit: [https://en.wikipedia.org/wiki/Heun's\\_method](https://en.wikipedia.org/wiki/Heun's_method)

# Continuous-Time Normalizing Flows: Neural ODE

## Neural ODE

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_\theta(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0$$

## Euler ODESolve

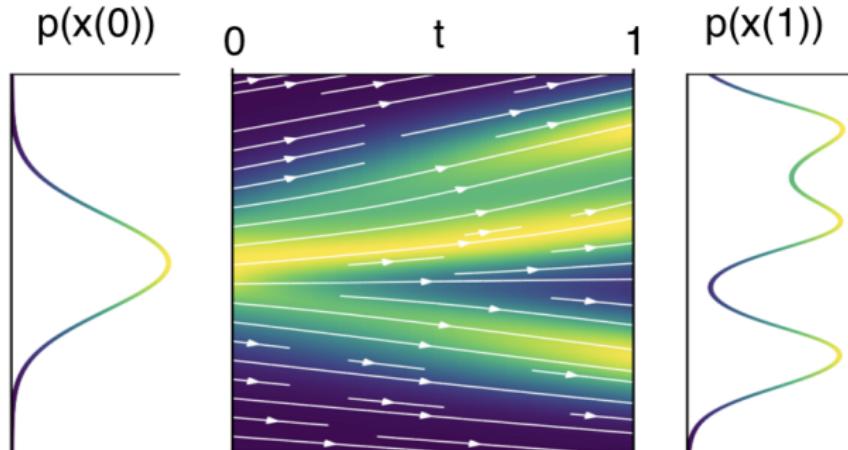
$$\mathbf{x}(t + h) = \mathbf{x}(t) + h \cdot \mathbf{f}_\theta(\mathbf{x}(t), t)$$

- ▶ Consider  $[t_0, t_1] = [0, 1]$  for simplicity.
- ▶ If  $\mathbf{x}(0)$  is a random variable with density  $p_0(\mathbf{x})$ ,
- ▶ Then, for any  $t$ ,  $\mathbf{x}(t)$  is a random variable with density  $p_t(\mathbf{x})$ .

# Continuous-Time Normalizing Flows: Intuition

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_\theta(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0$$

- ▶  $p_t(\mathbf{x}) = p(\mathbf{x}, t)$  describes the **probability path** interpolating between  $p_0(\mathbf{x})$  and  $p_1(\mathbf{x})$ .
- ▶ What is the difference between  $p_t(\mathbf{x}(t))$  and  $p_t(\mathbf{x})$ ?



# Continuous-Time Normalizing Flows: Reversibility

## Theorem (Picard)

If  $\mathbf{f}$  is continuously differentiable with a bounded derivative in  $\mathbf{x}$  and continuous in  $t$ , then the ODE has a **unique solution** given by a flow  $\psi_t$ .

This guarantees the ODE is **uniquely reversible**.

$$\psi_1(\mathbf{x}_0) = \mathbf{x}_0 + \int_0^1 \mathbf{f}_\theta(\psi_t(\mathbf{x}_0), t) dt$$

$$\mathbf{x}(1) = \mathbf{x}(0) + \int_0^1 \mathbf{f}_\theta(\mathbf{x}(t), t) dt$$

$$\mathbf{x}(0) = \mathbf{x}(1) + \int_1^0 \mathbf{f}_\theta(\mathbf{x}(t), t) dt$$

**Note:** Unlike discrete-time flows,  $\mathbf{f}$  need not be invertible (uniqueness ensures bijection). How can we compute  $p_t(\mathbf{x})$  at arbitrary  $t$ ?

## Summary

- ▶ DDPM and NCSN are intimately connected at the objective level.
- ▶ Classifier guidance provides a technique to turn an unconditional model into a conditional one by training an auxiliary classifier on noisy data.
- ▶ Classifier-free guidance removes the need for such a classifier, yielding a practical recipe now widely used.
- ▶ Continuous-time normalizing flows leverage neural ODEs to define continuous-time trajectories  $\mathbf{x}(t)$ , relaxing many constraints of discrete-time flows.
- ▶ If  $\mathbf{x}_0$  is a random variable, this yields a **probability path**  $p_t(\mathbf{x})$  as time evolves.

# Deep Generative Models

## Lecture 10

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

38. Continuity Equation for NF Log-Likelihood

39. SDE Basics

40. Probability Flow ODE

41. Reverse SDE

# Outline

38. Continuity Equation for NF Log-Likelihood

39. SDE Basics

40. Probability Flow ODE

41. Reverse SDE

# Continuous-Time NF

## Theorem (Continuity Equation)

If  $\mathbf{f}$  is uniformly Lipschitz continuous in  $\mathbf{x}$  and continuous in  $t$ , then

$$\frac{d \log p_t(\mathbf{x}(t))}{dt} = -\text{tr} \left( \frac{\partial \mathbf{f}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} \right)$$

This result states: given  $\mathbf{x}_0 = \mathbf{x}(0)$ , the solution to the continuity equation gives the density  $p_1(\mathbf{x}(1))$ .

## Solution of the Continuity Equation

$$\log p_1(\mathbf{x}(1)) = \log p_0(\mathbf{x}(0)) - \int_0^1 \text{tr} \left( \frac{\partial \mathbf{f}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} \right) dt.$$

- ▶ This provides the density along the trajectory (not the total probability path).
- ▶ However, the latter term is difficult to estimate efficiently.

# Outline

38. Continuity Equation for NF Log-Likelihood

39. SDE Basics

40. Probability Flow ODE

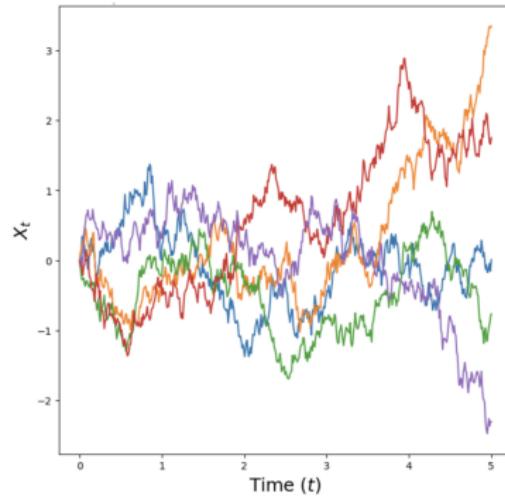
41. Reverse SDE

# Stochastic Differential Equation (SDE)

## Wiener Process

$\mathbf{w}(t)$  is the standard Wiener process (Brownian motion), defined by:

1.  $\mathbf{w}(0) = 0$  (almost surely);
2.  $\mathbf{w}(t)$  has independent increments;
3.  $\mathbf{w}(t)$  trajectories are continuous;
4.  $\mathbf{w}(t) - \mathbf{w}(s) \sim \mathcal{N}(0, (t-s)\mathbf{I})$  for  $t > s$ ;



$d\mathbf{w} = \mathbf{w}(t + dt) - \mathbf{w}(t) = \mathcal{N}(0, \mathbf{I} \cdot dt) = \epsilon \cdot \sqrt{dt}$ , where  
 $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ .

# Stochastic Differential Equation (SDE)

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t) \Rightarrow d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt$$

Let's define a stochastic process  $\mathbf{x}(t)$  with initial condition  $\mathbf{x}(0) \sim p_0(\mathbf{x}) = p_{\text{data}}(\mathbf{x})$ :

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

- ▶  $\mathbf{f}(\mathbf{x}, t) : \mathbb{R}^m \times [0, 1] \rightarrow \mathbb{R}^m$  is the **drift** term (vector field).
- ▶  $\mathbf{g}(t) : \mathbb{R} \rightarrow \mathbb{R}$  is the **diffusion** term (if  $\mathbf{g}(t) = 0$ , we recover the standard ODE).
- ▶  $\mathbf{w}(t)$  is the standard Wiener process ( $d\mathbf{w} = \epsilon \cdot \sqrt{dt}$ ).
- ▶ We do not have the flow  $\psi_t(\mathbf{x}_0)$  notion anymore, since trajectories are stochastic.

# Stochastic Differential Equation (SDE)

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

## Theorem

If  $\mathbf{f}$  is continuously differentiable with a bounded derivative in  $\mathbf{x}$  and continuous in  $t$  and  $g(t)$  is continuous then the SDE has the solution given by unique process  $\mathbf{x}(t)$ .

- ▶ Unlike ODEs, the initial condition  $\mathbf{x}(0)$  doesn't uniquely determine the trajectory.
- ▶ There are two sources of randomness:
  - ▶ the initial distribution  $p_0(\mathbf{x})$ ;
  - ▶ the Wiener process  $\mathbf{w}(t)$ .

# Stochastic Differential Equation (SDE)

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

Discretizing the SDE (Euler Update Step) – SDESolve

$$\mathbf{x}(t + dt) = \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t) \cdot dt + g(t) \cdot \epsilon \cdot \sqrt{dt}$$

If  $dt = 1$ , then

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t, t) + g(t) \cdot \epsilon$$

- ▶ At any time  $t$ , the process has density  $p_t(\mathbf{x}) = p(\mathbf{x}, t)$ .
- ▶  $p : \mathbb{R}^m \times [0, 1] \rightarrow \mathbb{R}_+$  specifies a **probability path** from  $p_0(\mathbf{x})$  to  $p_1(\mathbf{x})$ .
- ▶ How can we obtain the probability path  $p_t(\mathbf{x})$  for  $\mathbf{x}(t)$ ?

# Stochastic Differential Equation (SDE)

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}, \quad d\mathbf{w} = \boldsymbol{\epsilon} \cdot \sqrt{dt}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}).$$

Theorem (Kolmogorov-Fokker-Planck)

The evolution of  $p_t(\mathbf{x})$  is governed by

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = -\text{div}(\mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x})) + \frac{1}{2}g^2(t)\Delta_{\mathbf{x}}p_t(\mathbf{x})$$

Here,

$$\text{div}(\mathbf{v}) = \sum_{i=1}^m \frac{\partial v_i(\mathbf{x})}{\partial x_i} = \text{tr}\left(\frac{\partial \mathbf{v}(\mathbf{x})}{\partial \mathbf{x}}\right)$$

$$\Delta_{\mathbf{x}}p_t(\mathbf{x}) = \sum_{i=1}^m \frac{\partial^2 p_t(\mathbf{x})}{\partial x_i^2} = \text{tr}\left(\frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2}\right)$$

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = \text{tr}\left(-\frac{\partial}{\partial \mathbf{x}}[\mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2}g^2(t)\frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2}\right)$$

# Stochastic Differential Equation (SDE)

## Theorem (Kolmogorov-Fokker-Planck)

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = \text{tr} \left( -\frac{\partial}{\partial \mathbf{x}} [\mathbf{f}(\mathbf{x}, t) p_t(\mathbf{x})] + \frac{1}{2} g^2(t) \frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2} \right)$$

- ▶ The KFP theorem is necessary and sufficient condition (it is uniquely defines  $p_t(\mathbf{x})$ ).
- ▶ This generalizes the continuity equation for continuous-time NF:

$$\frac{d \log p_t(\mathbf{x}(t))}{dt} = -\text{tr} \left( \frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial \mathbf{x}} \right).$$

### Special Case: constant density $p_t(\mathbf{x})$

Let find the SDE for which  $p_t(\mathbf{x}) = \text{const}$  (i.e., if  $\mathbf{x}(0) \sim p_0(\mathbf{x})$ , then  $\mathbf{x}(t) \sim p_0(\mathbf{x})$ ).

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = 0 \Leftrightarrow \text{tr} \left( -\frac{\partial}{\partial \mathbf{x}} [\mathbf{f}(\mathbf{x}, t) p_t(\mathbf{x})] + \frac{1}{2} g^2(t) \frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2} \right) = 0$$

## Langevin SDE (Special Case)

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = 0 \Leftrightarrow \text{tr} \left( -\frac{\partial}{\partial \mathbf{x}} [\mathbf{f}(\mathbf{x}, t) p_t(\mathbf{x})] + \frac{1}{2} g^2(t) \frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2} \right) = 0$$

$$\frac{\partial}{\partial \mathbf{x}} [\mathbf{f}(\mathbf{x}, t) p_t(\mathbf{x})] = \frac{1}{2} g^2(t) \frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2}$$

$$\mathbf{f}(\mathbf{x}, t) p_t(\mathbf{x}) = \frac{1}{2} g^2(t) \frac{\partial p_t(\mathbf{x})}{\partial \mathbf{x}}$$

$$\mathbf{f}(\mathbf{x}, t) = \frac{1}{2} g^2(t) \frac{1}{p_t(\mathbf{x})} \frac{\partial p_t(\mathbf{x})}{\partial \mathbf{x}} = \frac{1}{2} g^2(t) \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x})$$

Let  $g(t) = 1$ , then  $\mathbf{f}(\mathbf{x}, t) = \frac{1}{2} \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x})$ .

$$d\mathbf{x} = \frac{1}{2} \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) dt + \mathbf{1} \cdot d\mathbf{w}$$

## Langevin SDE (Special Case)

Let find the SDE for which  $p_t(\mathbf{x}) = \text{const}$  (i.e., if  $\mathbf{x}(0) \sim p_0(\mathbf{x})$ , then  $\mathbf{x}(t) \sim p_0(\mathbf{x})$ ).

$$d\mathbf{x} = \frac{1}{2} \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) dt + \mathbf{1} \cdot d\mathbf{w}$$

## Discretized Langevin SDE

$$\mathbf{x}_{t+1} - \mathbf{x}_t = \frac{\eta}{2} \cdot \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}, \quad \eta \approx dt.$$

## Langevin Dynamic

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{\eta}{2} \cdot \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}, \quad \eta \approx dt.$$

We (partially) explained, why Langevin dynamics is working.

# Outline

38. Continuity Equation for NF Log-Likelihood

39. SDE Basics

40. Probability Flow ODE

41. Reverse SDE

# Probability Flow ODE

## ODE and Continuity Equation

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt$$

$$\frac{d \log p_t(\mathbf{x}(t))}{dt} = -\text{tr} \left( \frac{\partial \mathbf{f}_\theta(\mathbf{x}, t)}{\partial \mathbf{x}} \right) \Leftrightarrow \frac{\partial p_t(\mathbf{x})}{\partial t} = -\text{div}(\mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x}))$$

The only source of randomness is the initial distribution  $p_0(\mathbf{x})$ .

## SDE and KFP Equation

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

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = -\text{div}(\mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x})) + \frac{1}{2}g^2(t)\Delta_{\mathbf{x}}p_t(\mathbf{x})$$

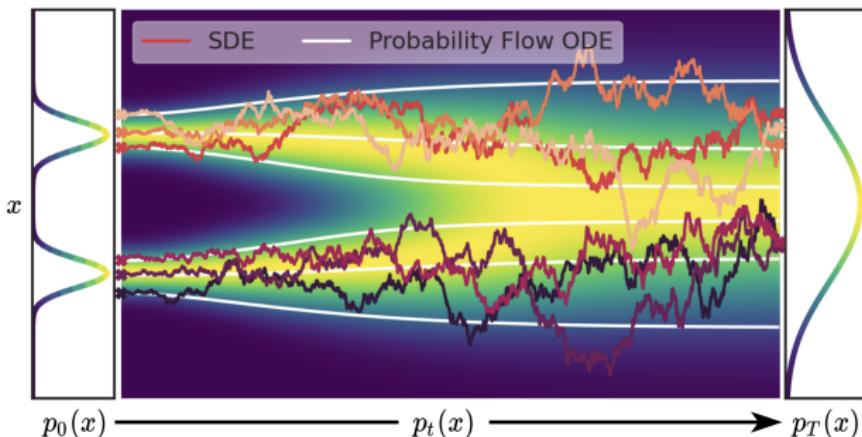
Now there are two sources of randomness: the initial distribution  $p_0(\mathbf{x})$  and the Wiener process  $\mathbf{w}(t)$ .

# Probability Flow ODE

## Theorem

Suppose the SDE  $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$  induces the probability path  $p_t(\mathbf{x})$ . Then, there exists an ODE with the same probability path  $p_t(\mathbf{x})$ , given by

$$d\mathbf{x} = \left( \mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt$$



# Probability Flow ODE

## Theorem

Suppose the SDE  $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$  induces the probability path  $p_t(\mathbf{x})$ . Then, there exists an ODE with the same probability path  $p_t(\mathbf{x})$ , given by

$$d\mathbf{x} = \left( \mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt$$

## Proof

$$\begin{aligned}\frac{\partial p_t(\mathbf{x})}{\partial t} &= \text{tr} \left( -\frac{\partial}{\partial \mathbf{x}} [\mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2}g^2(t)\frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2} \right) = \\ &= \text{tr} \left( -\frac{\partial}{\partial \mathbf{x}} \left[ \mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x}) - \frac{1}{2}g^2(t)\frac{\partial p_t(\mathbf{x})}{\partial \mathbf{x}} \right] \right) = \\ &= \text{tr} \left( -\frac{\partial}{\partial \mathbf{x}} \left[ \mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x}) - \frac{1}{2}g^2(t)p_t(\mathbf{x})\frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right] \right) = \\ &= \text{tr} \left( -\frac{\partial}{\partial \mathbf{x}} \left[ \left( \mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) p_t(\mathbf{x}) \right] \right)\end{aligned}$$

# Probability Flow ODE

## Proof (Continued)

$$\begin{aligned}\frac{\partial p_t(\mathbf{x})}{\partial t} &= \text{tr} \left( -\frac{\partial}{\partial \mathbf{x}} \left[ \left( \mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g^2(t) \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) p_t(\mathbf{x}) \right] \right) = \\ &= \text{tr} \left( -\frac{\partial}{\partial \mathbf{x}} \left[ \tilde{\mathbf{f}}(\mathbf{x}, t) p_t(\mathbf{x}) \right] \right) = -\text{div} \left( \tilde{\mathbf{f}}(\mathbf{x}, t) p_t(\mathbf{x}) \right)\end{aligned}$$

$$\tilde{\mathbf{f}}(\mathbf{x}, t) = \mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g^2(t) \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}); \quad \tilde{g}(t) = 0$$

$$d\mathbf{x} = \tilde{\mathbf{f}}(\mathbf{x}, t) dt + 0 \cdot d\mathbf{w} = \left( \mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g^2(t) \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt$$

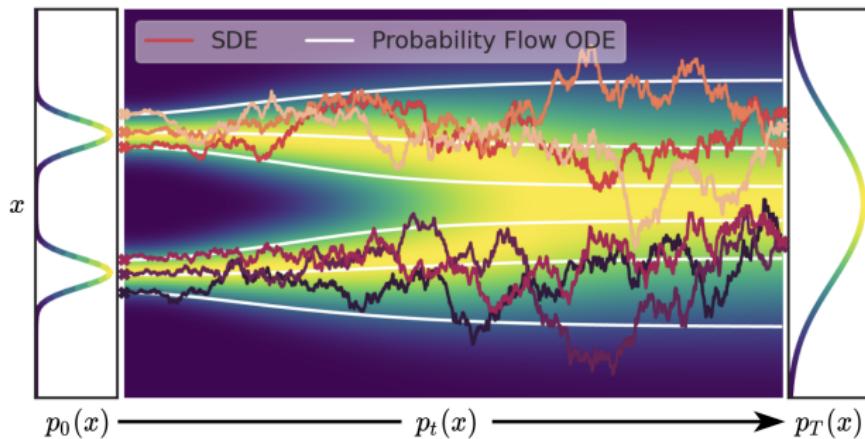
$$\frac{\partial p_t(\mathbf{x})}{\partial t} = -\text{div}(\tilde{\mathbf{f}}(\mathbf{x}, t) p_t(\mathbf{x})) + \frac{1}{2} \tilde{g}^2(t) \Delta_{\mathbf{x}} p_t(\mathbf{x})$$

# Probability Flow ODE

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w} \quad - \text{SDE}$$

$$d\mathbf{x} = \left( \mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt \quad - \text{Probability Flow ODE}$$

- ▶ The term  $\mathbf{s}(\mathbf{x}, t) = \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x})$  is the score function in continuous time.
- ▶ The ODE produces more stable trajectories.



# Outline

38. Continuity Equation for NF Log-Likelihood

39. SDE Basics

40. Probability Flow ODE

41. Reverse SDE

## Reverse SDE

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt, \quad \mathbf{x}(t + dt) = \mathbf{x}(t) + \mathbf{f}(\mathbf{x}, t)dt$$

Here  $dt$  can be  $> 0$  or  $< 0$ .

## Reverse ODE

Let  $\tau = 1 - t$  ( $d\tau = -dt$ ).

$$d\mathbf{x} = -\mathbf{f}(\mathbf{x}, 1 - \tau)d\tau$$

- ▶ How do we reverse the SDE  $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$ ?
- ▶ The Wiener process introduces randomness that must be reversed.

## Theorem

There exists a reverse SDE for  $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$ , given by:

$$d\mathbf{x} = \left( \mathbf{f}(\mathbf{x}, t) - g^2(t) \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt + g(t)d\mathbf{w}$$

where  $dt < 0$ .

# Reverse SDE

## Theorem

There exists a reverse SDE for  $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$ , given by:

$$d\mathbf{x} = \left( \mathbf{f}(\mathbf{x}, t) - g^2(t) \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt + g(t)d\mathbf{w}$$

where  $dt < 0$ .

**Note:** Again, the score function appears:  $\mathbf{s}(\mathbf{x}, t) = \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x})$ .

## Proof Sketch

- ▶ Convert the initial SDE to a probability flow ODE.
- ▶ Reverse the probability flow ODE.
- ▶ Convert the reversed probability flow ODE back to an SDE.

# Reverse SDE

## Proof

- ▶ Convert the initial SDE to a probability flow ODE:

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

$$d\mathbf{x} = \left( \mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt$$

- ▶ Reverse the probability flow ODE:

$$d\mathbf{x} = \left( \mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt$$

$$d\mathbf{x} = \left( -\mathbf{f}(\mathbf{x}, 1 - \tau) + \frac{1}{2}g^2(1 - \tau)\frac{\partial}{\partial \mathbf{x}} \log p_{1-\tau}(\mathbf{x}) \right) d\tau$$

- ▶ Convert the reversed probability flow ODE back to an SDE:

$$d\mathbf{x} = \left( -\mathbf{f}(\mathbf{x}, 1 - \tau) + \frac{1}{2}g^2(1 - \tau)\frac{\partial}{\partial \mathbf{x}} \log p_{1-\tau}(\mathbf{x}) \right) d\tau$$

$$d\mathbf{x} = \left( -\mathbf{f}(\mathbf{x}, 1 - \tau) + g^2(1 - \tau)\frac{\partial}{\partial \mathbf{x}} \log p_{1-\tau}(\mathbf{x}) \right) d\tau + g(1 - \tau)d\mathbf{w}$$

# Reverse SDE

## Theorem

There exists a reverse SDE for  $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$ , given by:

$$d\mathbf{x} = \left( \mathbf{f}(\mathbf{x}, t) - g^2(t) \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt + g(t)d\mathbf{w}$$

where  $dt < 0$ .

## Proof (Continued)

$$d\mathbf{x} = \left( -\mathbf{f}(\mathbf{x}, 1 - \tau) + g^2(1 - \tau) \frac{\partial}{\partial \mathbf{x}} \log p_{1-\tau}(\mathbf{x}) \right) d\tau + g(1 - \tau)d\mathbf{w}$$

$$d\mathbf{x} = \left( \mathbf{f}(\mathbf{x}, t) - g^2(t) \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt + g(t)d\mathbf{w}$$

Here  $d\tau > 0$  and  $dt < 0$ .

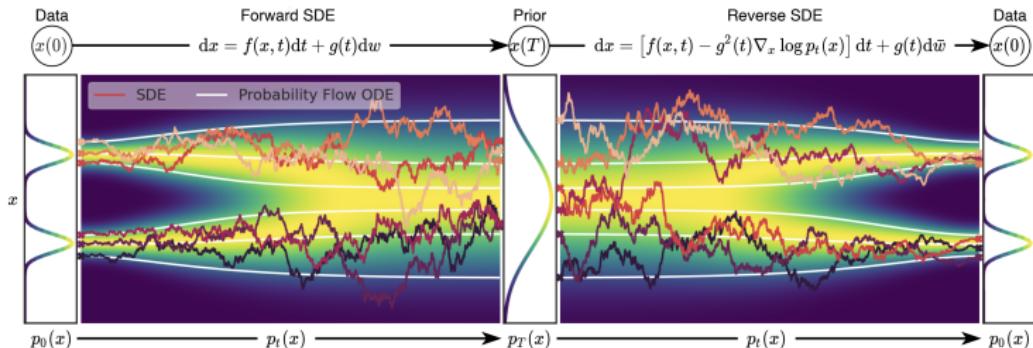
# Reverse SDE

$$dx = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w} \quad - \text{SDE}$$

$$dx = \left( \mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt \quad - \text{Probability Flow ODE}$$

$$dx = \left( \mathbf{f}(\mathbf{x}, t) - g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt + g(t)d\bar{w} \quad - \text{Reverse SDE}$$

- ▶ This framework allows us to transform one distribution into another via an SDE with a prescribed probability path  $p_t(\mathbf{x})$ .
- ▶ We can invert this process using the score function.



Song Y., et al. Score-Based Generative Modeling through Stochastic Differential Equations, 2020

## Summary

- ▶ The continuity equation allows us to compute  $\log p(\mathbf{x}, t)$  at any time  $t$ .
- ▶ An SDE defines a stochastic process with drift and diffusion terms; ODEs are a special case of SDEs.
- ▶ The KFP equation describes the probability dynamics of an SDE.
- ▶ The Langevin SDE preserves a constant probability path.
- ▶ Every SDE admits a corresponding probability flow ODE following the same probability path.
- ▶ SDEs can be reversed using the score function.

# Deep Generative Models

## Lecture 11

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

42. Diffusion and Score Matching SDEs

43. Score-Based Generative Models Through SDEs

44. Flow Matching

45. Conditional Flow Matching

# Outline

42. Diffusion and Score Matching SDEs

43. Score-Based Generative Models Through SDEs

44. Flow Matching

45. Conditional Flow Matching

# Score Matching SDE

## Denoising Score Matching

$$\mathbf{x}_t = \mathbf{x} + \sigma_t \cdot \epsilon_t, \quad q(\mathbf{x}_t | \mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma_t^2 \cdot \mathbf{I})$$

$$\mathbf{x}_{t-1} = \mathbf{x} + \sigma_{t-1} \cdot \epsilon_{t-1}, \quad q(\mathbf{x}_{t-1} | \mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma_{t-1}^2 \cdot \mathbf{I})$$

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \sqrt{\sigma_t^2 - \sigma_{t-1}^2} \cdot \epsilon, \quad q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_{t-1}, (\sigma_t^2 - \sigma_{t-1}^2) \cdot \mathbf{I})$$

Let's transform this Markov chain into the continuous stochastic process  $\mathbf{x}(t)$  by letting  $T \rightarrow \infty$ :

$$\begin{aligned}\mathbf{x}(t) &= \mathbf{x}(t - dt) + \sqrt{\sigma^2(t) - \sigma^2(t - dt)} \cdot \epsilon \\ &= \mathbf{x}(t - dt) + \sqrt{\frac{\sigma^2(t) - \sigma^2(t - dt)}{dt}} dt \cdot \epsilon \\ &= \mathbf{x}(t - dt) + \sqrt{\frac{d[\sigma^2(t)]}{dt}} \cdot d\mathbf{w}\end{aligned}$$

## Score Matching SDE

$$\mathbf{x}(t) = \mathbf{x}(t - dt) + \sqrt{\frac{d[\sigma^2(t)]}{dt}} \cdot d\mathbf{w}$$

## Variance Exploding SDE

$$d\mathbf{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}} \cdot d\mathbf{w}$$

$\sigma(t)$  is a monotonically increasing function.

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}, \quad \mathbf{f}(\mathbf{x}, t) = 0, \quad g(t) = \sqrt{\frac{d[\sigma^2(t)]}{dt}}$$

$$d\mathbf{x} = \left( -\frac{1}{2} \frac{d[\sigma^2(t)]}{dt} \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt \quad (\text{probability flow ODE})$$

$$d\mathbf{x} = \left( -\frac{d[\sigma^2(t)]}{dt} \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt + \sqrt{\frac{d[\sigma^2(t)]}{dt}} d\mathbf{w} \quad (\text{reverse SDE})$$

# Diffusion SDE

## Denoising Diffusion

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

Let's turn this Markov chain into a continuous stochastic process by letting  $T \rightarrow \infty$  and setting  $\beta_t = \beta\left(\frac{t}{T}\right) \cdot \frac{1}{T}$  (where  $dt = \frac{1}{T}$ ):

$$\begin{aligned}\mathbf{x}(t) &= \sqrt{1 - \beta(t)dt} \cdot \mathbf{x}(t - dt) + \sqrt{\beta(t)dt} \cdot \boldsymbol{\epsilon} \approx \\ &\approx \left(1 - \frac{1}{2}\beta(t)dt\right) \cdot \mathbf{x}(t - dt) + \sqrt{\beta(t)dt} \cdot \boldsymbol{\epsilon} = \\ &= \mathbf{x}(t - dt) - \frac{1}{2}\beta(t)\mathbf{x}(t - dt)dt + \sqrt{\beta(t)} \cdot d\mathbf{w}\end{aligned}$$

## Variance Preserving SDE

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

# Diffusion SDE

## Variance Preserving SDE

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

$$\mathbf{f}(\mathbf{x}, t) = -\frac{1}{2}\beta(t)\mathbf{x}(t), \quad g(t) = \sqrt{\beta(t)}$$

Variance is preserved as long as  $\mathbf{x}(0)$  has unit variance.

$$d\mathbf{x} = \left( -\frac{1}{2}\beta(t)\mathbf{x}(t) - \frac{1}{2}\beta(t)\frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt \quad (\text{probability flow ODE})$$

$$d\mathbf{x} = \left( -\frac{1}{2}\beta(t)\mathbf{x}(t) - \beta(t)\frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right) dt + \sqrt{\beta(t)}d\mathbf{w} \quad (\text{reverse SDE})$$

## Diffusion SDE

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

### Variance Exploding SDE (NCSN)

$$d\mathbf{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}} \cdot d\mathbf{w}$$

### Variance Preserving SDE (DDPM)

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

## Efficient Solvers

- ▶ Converting SDEs to PF-ODEs yields more efficient inference.
- ▶ We can apply any ODESolve procedure to reduce the number of inference steps.
- ▶ In practice, this reduces the number of steps from 100–1000 to 20–50.

# Outline

42. Diffusion and Score Matching SDEs

43. Score-Based Generative Models Through SDEs

44. Flow Matching

45. Conditional Flow Matching

# Score-Based Generative Models Through SDEs

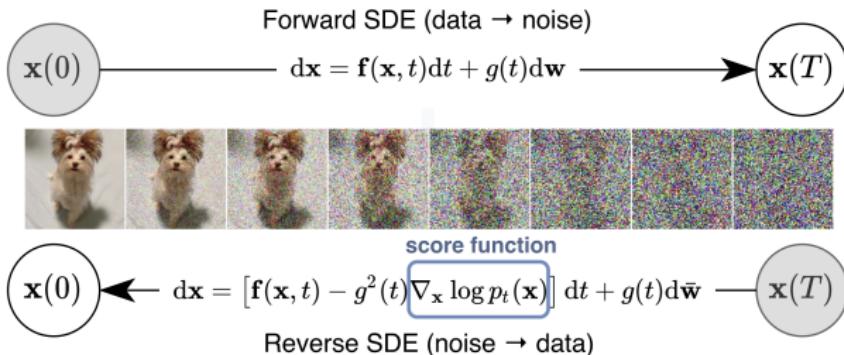
## Discrete-Time Objective

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}_0)} \mathbb{E}_{t \sim U\{1, T\}} \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \| \mathbf{s}_{\theta, t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0) \|_2^2$$

Is it possible to train score-based diffusion models in continuous time?

## Continuous-Time Objective

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}(0))} \mathbb{E}_{t \sim U[0, 1]} \mathbb{E}_{q(\mathbf{x}(t) | \mathbf{x}(0))} \| \mathbf{s}_{\theta}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log q(\mathbf{x}(t) | \mathbf{x}(0)) \|_2^2$$



# Score-Based Generative Models Through SDEs

## Continuous-Time Objective

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}(0))} \mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{q(\mathbf{x}(t)|\mathbf{x}(0))} \| \mathbf{s}_{\theta}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log q(\mathbf{x}(t)|\mathbf{x}(0)) \|_2^2$$

$$q(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}\left(\boldsymbol{\mu}(t, \mathbf{x}(0)), \boldsymbol{\sigma}^2(t, \mathbf{x}(0)) \cdot \mathbf{I}\right)$$

$$\nabla_{\mathbf{x}(t)} \log q(\mathbf{x}(t)|\mathbf{x}(0)) = -\frac{1}{\sigma} \odot (\mathbf{x}(t) - \boldsymbol{\mu})$$

**Note:** Normality holds for  $\mathbf{f}(\mathbf{x}, t)$  affine in  $\mathbf{x}$ .

$$d\mathbf{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}} \cdot d\mathbf{w} \quad (\text{Variance Exploding SDE})$$

$$d\mathbf{x} = -\frac{1}{2} \beta(t) \mathbf{x}(t) dt + \sqrt{\beta(t)} \cdot d\mathbf{w} \quad (\text{Variance Preserving SDE})$$

Is it possible to explicitly derive  $\boldsymbol{\mu}(t, \mathbf{x}(0))$  and  $\boldsymbol{\Sigma}(t, \mathbf{x}(0))$  for VE-SDE and VP-SDE?

# Score-Based Generative Models Through SDEs

$$q(\mathbf{x}(t) | \mathbf{x}(0)) = \mathcal{N}\left(\boldsymbol{\mu}(t, \mathbf{x}(0)), \boldsymbol{\Sigma}(t, \mathbf{x}(0))\right)$$

## Theorem

The moments of the SDE  $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$  satisfy:

$$\frac{d\boldsymbol{\mu}(t, \mathbf{x}(0))}{dt} = \mathbb{E}[\mathbf{f}(\mathbf{x}(t), t) | \mathbf{x}(0)]$$

$$\frac{d\boldsymbol{\Sigma}(t, \mathbf{x}(0))}{dt} = \mathbb{E} \left[ \mathbf{f} \cdot (\mathbf{x}(t) - \boldsymbol{\mu})^\top + (\mathbf{x}(t) - \boldsymbol{\mu}) \cdot \mathbf{f}^\top | \mathbf{x}(0) \right] + g^2(t) \cdot \mathbf{I}$$

## Proof

$$\begin{aligned}\mathbb{E}[d\mathbf{x} | \mathbf{x}(0)] &= \mathbb{E}[\mathbf{f}(\mathbf{x}, t)dt | \mathbf{x}(0)] + \mathbb{E}[g(t)d\mathbf{w} | \mathbf{x}(0)] \\ &= \mathbb{E}[\mathbf{f}(\mathbf{x}, t) | \mathbf{x}(0)] dt + g(t)\mathbb{E}[d\mathbf{w} | \mathbf{x}(0)] \\ &= \mathbb{E}[\mathbf{f}(\mathbf{x}, t) | \mathbf{x}(0)] dt\end{aligned}$$

# Score-Based Generative Models Through SDEs

## Theorem

$$\frac{d\mu(t, \mathbf{x}(0))}{dt} = \mathbb{E} [\mathbf{f}(\mathbf{x}(t), t) | \mathbf{x}(0)]$$

## Proof (Continued)

$$\mathbb{E} [d\mathbf{x} | \mathbf{x}(0)] = \mathbb{E} [\mathbf{f}(\mathbf{x}, t) | \mathbf{x}(0)] dt$$

$$\frac{d\mathbb{E} [\mathbf{x}(t) | \mathbf{x}(0)]}{dt} = \frac{d\mu(t, \mathbf{x}(0))}{dt} = \mathbb{E} [\mathbf{f}(\mathbf{x}, t) | \mathbf{x}(0)]$$

## Examples

**NCSN:**  $\mathbf{f}(\mathbf{x}, t) = 0 \Rightarrow \mu = \mathbf{x}(0)$

**DDPM:**  $\mathbf{f}(\mathbf{x}, t) = -\frac{1}{2}\beta(t)\mathbf{x}(t) \Rightarrow \frac{d\mu}{dt} = -\frac{1}{2}\beta(t)\mu$

$$\mu = \mathbf{x}(0) \exp \left( -\frac{1}{2} \int_0^t \beta(s) ds \right)$$

# Score-Based Generative Models Through SDEs

## Training

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}(0))} \mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{q(\mathbf{x}(t)|\mathbf{x}(0))} \| \mathbf{s}_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log q(\mathbf{x}(t)|\mathbf{x}(0)) \|_2^2$$

$$q(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}\left(\boldsymbol{\mu}(t, \mathbf{x}(0)), \boldsymbol{\Sigma}(t, \mathbf{x}(0))\right)$$

## NCSN

$$q(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}\left(\mathbf{x}(0), [\sigma^2(t) - \sigma^2(0)] \cdot \mathbf{I}\right)$$

## DDPM

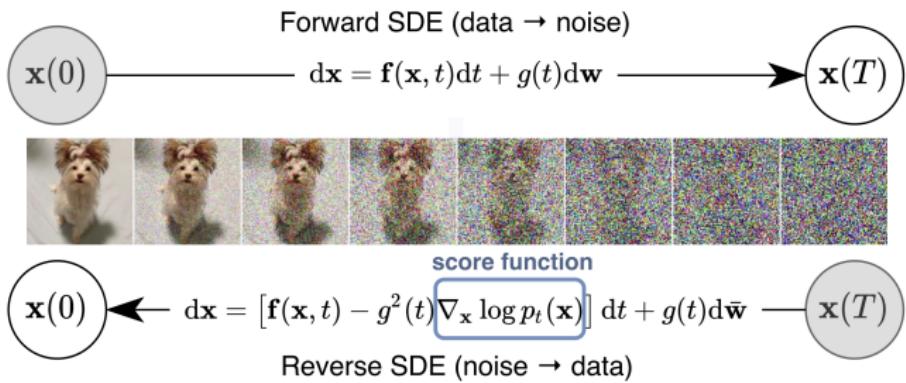
$$q(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}\left(\mathbf{x}(0)e^{-\frac{1}{2} \int_0^t \beta(s) ds}, \left(1 - e^{-\int_0^t \beta(s) ds}\right) \cdot \mathbf{I}\right)$$

Here we omit the derivations of the variance.

# Score-Based Generative Models Through SDEs

## Sampling

Solve the reverse SDE using numerical solvers (`SDESolve`).



- ▶ Discretizing the reverse SDE provides ancestral sampling.
- ▶ Discretizing the probability flow ODE yields deterministic sampling.

# Outline

42. Diffusion and Score Matching SDEs

43. Score-Based Generative Models Through SDEs

44. Flow Matching

45. Conditional Flow Matching

# Continuous-Time Normalizing Flows

Let's return to ODE dynamics  $\mathbf{x}_t = \mathbf{x}(t)$  in the interval  $t \in [0, 1]$ :

- ▶  $\mathbf{x}_0 \sim p_0(\mathbf{x}) = p(\mathbf{x})$ ,  $\mathbf{x}_1 \sim p_1(\mathbf{x}) = p_{\text{data}}(\mathbf{x})$ ;
- ▶  $p(\mathbf{x})$  is a base distribution (e.g.,  $\mathcal{N}(0, \mathbf{I})$ ), and  $p_{\text{data}}(\mathbf{x})$  is the true data distribution.

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{f}(\mathbf{x}_t, t), \quad \text{with initial condition } \mathbf{x}(0) = \mathbf{x}_0.$$

## KFP Theorem (Continuity Equation)

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = -\text{div}(\mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x})) \Leftrightarrow \frac{d \log p_t(\mathbf{x}(t))}{dt} = -\text{tr}\left(\frac{\partial \mathbf{f}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)}\right)$$

- ▶ It's hard to solve the continuity equation directly due to the trace term.
- ▶ There's a method (the adjoint method) that solves this equation directly, but it's unstable and unscalable.

# Continuous-Time Normalizing Flows

## KFP Theorem (Continuity Equation)

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = -\text{div}(\mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x})) \Leftrightarrow \frac{d \log p_t(\mathbf{x}(t))}{dt} = -\text{tr}\left(\frac{\partial \mathbf{f}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)}\right)$$

- ▶ Knowing the vector field  $\mathbf{f}(\mathbf{x}, t)$ , the KFP (or continuity) equation allows us to compute the density  $p_t(\mathbf{x})$ .
- ▶ Flow matching provides an alternative approach to Neural ODEs.

## Flow Matching

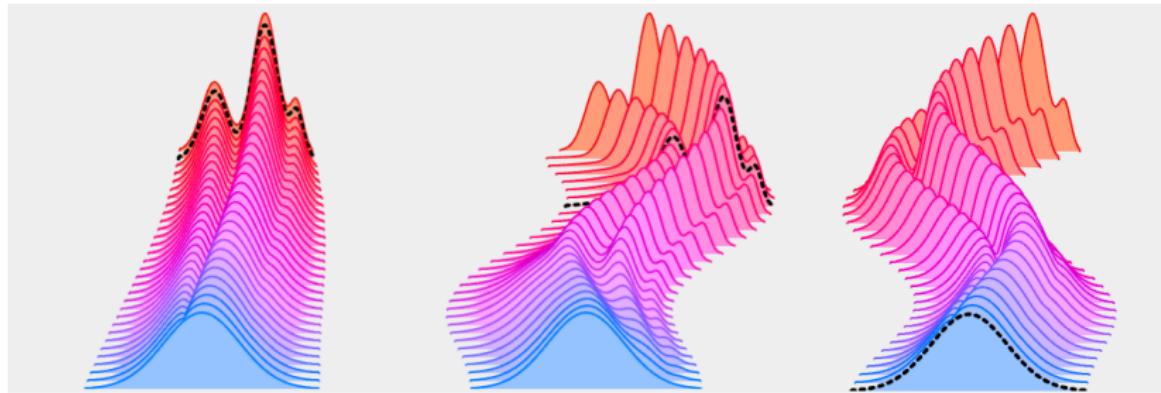
$$\mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x})} \|\mathbf{f}(\mathbf{x}, t) - \mathbf{f}_\theta(\mathbf{x}, t)\|^2 \rightarrow \min_{\theta}$$

- ▶ Approximate the true vector field  $\mathbf{f}(\mathbf{x}, t)$  using  $\mathbf{f}_\theta(\mathbf{x}, t)$ .
- ▶ Use  $\mathbf{f}_\theta(\mathbf{x}, t)$  for deterministic sampling from the ODE.

# Flow Matching

$$\mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x})} \|\mathbf{f}(\mathbf{x}, t) - \mathbf{f}_\theta(\mathbf{x}, t)\|^2 \rightarrow \min_{\theta}$$

- ▶ There are infinitely many possible  $\mathbf{f}(\mathbf{x}, t)$  between  $p_{\text{data}}(\mathbf{x})$  and  $p(\mathbf{x})$ .
- ▶ The true vector field  $\mathbf{f}(\mathbf{x}, t)$  is **unknown**.
- ▶ We need to select the "best"  $\mathbf{f}(\mathbf{x}, t)$  and make the objective tractable.



# Outline

42. Diffusion and Score Matching SDEs

43. Score-Based Generative Models Through SDEs

44. Flow Matching

45. Conditional Flow Matching

# Flow Matching

## Latent Variable Model

Let's introduce the latent variable  $\mathbf{z}$ :

$$p_t(\mathbf{x}) = \int p_t(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

Here,  $p_t(\mathbf{x}|\mathbf{z})$  is a **conditional probability path**. The conditional probability path  $p_t(\mathbf{x}|\mathbf{z})$  satisfies the KFP theorem:

$$\frac{\partial p_t(\mathbf{x}|\mathbf{z})}{\partial t} = -\text{div}(\mathbf{f}(\mathbf{x}, \mathbf{z}, t)p_t(\mathbf{x}|\mathbf{z})),$$

where  $\mathbf{f}(\mathbf{x}, \mathbf{z}, t)$  is a **conditional vector field**:

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{f}(\mathbf{x}_t, t) \quad \Rightarrow \quad \frac{d\mathbf{x}_t}{dt} = \mathbf{f}(\mathbf{x}_t, \mathbf{z}, t)$$

What's the relationship between  $\mathbf{f}(\mathbf{x}, t)$  and  $\mathbf{f}(\mathbf{x}, \mathbf{z}, t)$ ?

## Flow Matching

$$\frac{\partial p_t(\mathbf{x}|\mathbf{z})}{\partial t} = -\text{div}(\mathbf{f}(\mathbf{x}, \mathbf{z}, t) p_t(\mathbf{x}|\mathbf{z})),$$

### Theorem

The following vector field generates the probability path  $p_t(\mathbf{x})$ :

$$\mathbf{f}(\mathbf{x}, t) = \mathbb{E}_{p_t(\mathbf{z}|\mathbf{x})} \mathbf{f}(\mathbf{x}, \mathbf{z}, t) = \int \mathbf{f}(\mathbf{x}, \mathbf{z}, t) \frac{p_t(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{p_t(\mathbf{x})} d\mathbf{z}$$

### Proof

$$\begin{aligned}\frac{\partial p_t(\mathbf{x})}{\partial t} &= \frac{\partial}{\partial t} \int p_t(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} = \int \left( \frac{\partial p_t(\mathbf{x}|\mathbf{z})}{\partial t} \right) p(\mathbf{z}) d\mathbf{z} = \\ &= \int (-\text{div}(\mathbf{f}(\mathbf{x}, \mathbf{z}, t) p_t(\mathbf{x}|\mathbf{z}))) p(\mathbf{z}) d\mathbf{z} = \\ &= -\text{div} \left( \int \mathbf{f}(\mathbf{x}, \mathbf{z}, t) p_t(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \right) = -\text{div}(\mathbf{f}(\mathbf{x}, t) p_t(\mathbf{x}))\end{aligned}$$

# Flow Matching

## Flow Matching (FM)

$$\mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x})} \|\mathbf{f}(\mathbf{x}, t) - \mathbf{f}_\theta(\mathbf{x}, t)\|^2 \rightarrow \min_{\theta}$$

## Conditional Flow Matching (CFM)

$$\mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x}|\mathbf{z})} \|\mathbf{f}(\mathbf{x}, \mathbf{z}, t) - \mathbf{f}_\theta(\mathbf{x}, t)\|^2 \rightarrow \min_{\theta}$$

### Theorem

If  $\text{supp}(p_t(\mathbf{x})) = \mathbb{R}^m$ , then the optimal value of the FM objective equals the optimal value of the CFM objective.

### Proof

This can be proved in a similar way as in the denoising score matching theorem.

## Summary

- ▶ Score matching (NCSN) and diffusion models (DDPM) are discretizations of SDEs (variance exploding and variance preserving).
- ▶ It's possible to train continuous-in-time score-based generative models using forward and reverse SDEs.
- ▶ Discretizing the reverse SDE yields ancestral sampling of the DDPM.
- ▶ Flow matching suggests fitting the vector field directly.
- ▶ Conditional flow matching introduces the latent variable  $z$ , reformulating the initial task in terms of conditional dynamics.

# Deep Generative Models

## Lecture 12

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

46. Conditional Flow Matching

47. Conical Gaussian Paths

48. Linear Interpolation

# Outline

46. Conditional Flow Matching

47. Conical Gaussian Paths

48. Linear Interpolation

# Conditional Flow Matching

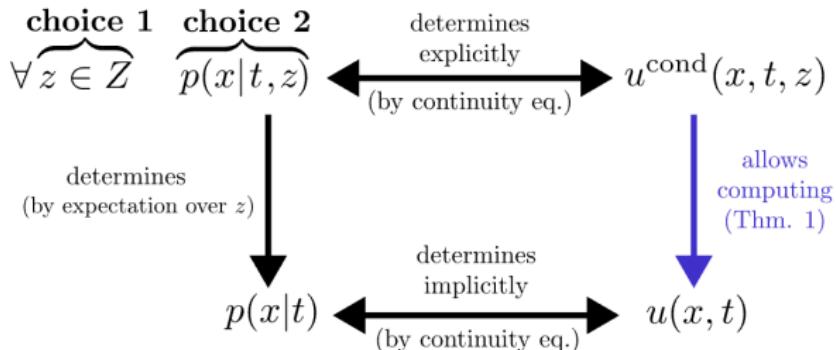
## Theorem

$$\begin{aligned} \arg \min_{\theta} \mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{x \sim p_t(x)} \|f(x, t) - f_{\theta}(x, t)\|^2 = \\ = \arg \min_{\theta} \mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{z \sim p(z)} \mathbb{E}_{x \sim p_t(x|z)} \|f(x, z, t) - f_{\theta}(x, t)\|^2 \end{aligned}$$

## Proof

$$\begin{aligned} \mathbb{E}_{x \sim p_t(x)} \|f(x, t) - f_{\theta}(x, t)\|^2 = \\ = \mathbb{E}_{z \sim p(z)} \mathbb{E}_{x \sim p_t(x|z)} [\|f_{\theta}(x, t)\|^2 - 2f_{\theta}^{\top}(x, t)f(x, t)] + \text{const}(\theta) \\ \mathbb{E}_{p_t(x)} [f_{\theta}^{\top}(x, t)f(x, t)] = \int p_t(x) \left[ f_{\theta}^{\top}(x, t) \left( \int p_t(z|x)f(x, z, t)dz \right) \right] dx = \\ = \int \int p_t(x)p_t(z|x) [f_{\theta}^{\top}(x, t)f(x, z, t)] dz dx = \\ = \mathbb{E}_{p(z)} \mathbb{E}_{p_t(x|z)} [f_{\theta}^{\top}(x, t)f(x, z, t)] \end{aligned}$$

# Conditional Flow Matching



- ▶ We don't want to directly model  $p_t(\mathbf{x})$ , since it's complex.
- ▶ We've shown it's possible to solve the CFM task instead of the FM task.
- ▶ Let's choose a convenient conditioning latent variable  $\mathbf{z}$ .
- ▶ We'll parametrize  $p_t(\mathbf{x}|\mathbf{z})$  instead of  $p_t(\mathbf{x})$ . It should satisfy the following constraints:

$$p(\mathbf{x}) = \mathcal{N}(0, \mathbf{I}) = \mathbb{E}_{p(\mathbf{z})} p_0(\mathbf{x}|\mathbf{z}); \quad p_{\text{data}}(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z})} p_1(\mathbf{x}|\mathbf{z}).$$

# Conditional Flow Matching

$$\mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x}|\mathbf{z})} \|\mathbf{f}(\mathbf{x}, \mathbf{z}, t) - \mathbf{f}_\theta(\mathbf{x}, t)\|^2 \rightarrow \min_{\theta}$$

$$p(\mathbf{x}) = \mathcal{N}(0, \mathbf{I}) = \mathbb{E}_{p(\mathbf{z})} p_0(\mathbf{x}|\mathbf{z}); \quad p_{\text{data}}(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z})} p_1(\mathbf{x}|\mathbf{z}).$$

## Training

1. Sample a time  $t \sim U[0, 1]$  and  $\mathbf{z} \sim p(\mathbf{z})$ .
2. Draw  $\mathbf{x}_t \sim p_t(\mathbf{x}|\mathbf{z})$ .
3. Compute the loss  $\mathcal{L} = \|\mathbf{f}(\mathbf{x}_t, \mathbf{z}, t) - \mathbf{f}_\theta(\mathbf{x}_t, t)\|^2$ .

## Sampling

1. Sample  $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$ .
2. Solve the ODE to obtain  $\mathbf{x}_1$ :

$$\mathbf{x}_1 = \text{ODESolve}_f(\mathbf{x}_0, \theta, t_0 = 0, t_1 = 1).$$

# Conditional Flow Matching

$$\mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{z \sim p(z)} \mathbb{E}_{x \sim p_t(x|z)} \|f(x, z, t) - f_\theta(x, t)\|^2 \rightarrow \min_{\theta}$$

## Open Questions

- Q1 How should we choose the conditioning latent variable  $z$ ?
- Q2 How can we define  $p_t(x|z)$  to enforce the constraints?

## Gaussian Conditional Probability Path [Q2]

$$p_t(x|z) = \mathcal{N}(\mu_t(z), \sigma_t^2(z))$$

- ▶ There are infinitely many vector fields that generate a particular probability path.
- ▶ Let's consider the following dynamics:

$$x_t = \mu_t(z) + \sigma_t(z) \odot \epsilon, \quad \text{with fixed } \epsilon \sim \mathcal{N}(0, I)$$

# Flow Matching

## Gaussian Conditional Probability Path

$$p_t(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_t(\mathbf{z}), \boldsymbol{\sigma}_t^2(\mathbf{z})) ; \quad \mathbf{x}_t = \boldsymbol{\mu}_t(\mathbf{z}) + \boldsymbol{\sigma}_t(\mathbf{z}) \odot \boldsymbol{\epsilon}$$

Is it possible to derive the expression for  $\mathbf{f}(\mathbf{x}_t, \mathbf{z}, t)$ ?

### Statement

$$\mathbf{f}(\mathbf{x}_t, \mathbf{z}, t) = \boldsymbol{\mu}'_t(\mathbf{z}) + \frac{\boldsymbol{\sigma}'_t(\mathbf{z})}{\boldsymbol{\sigma}_t(\mathbf{z})} \odot (\mathbf{x}_t - \boldsymbol{\mu}_t(\mathbf{z}))$$

### Proof

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{f}(\mathbf{x}_t, \mathbf{z}, t); \quad \boldsymbol{\epsilon} = \frac{1}{\boldsymbol{\sigma}_t(\mathbf{z})} \odot (\mathbf{x}_t - \boldsymbol{\mu}_t(\mathbf{z}))$$

$$\frac{d\mathbf{x}_t}{dt} = \boldsymbol{\mu}'_t(\mathbf{z}) + \boldsymbol{\sigma}'_t(\mathbf{z}) \odot \boldsymbol{\epsilon} = \boldsymbol{\mu}'_t(\mathbf{z}) + \frac{\boldsymbol{\sigma}'_t(\mathbf{z})}{\boldsymbol{\sigma}_t(\mathbf{z})} \odot (\mathbf{x}_t - \boldsymbol{\mu}_t(\mathbf{z}))$$

# Outline

46. Conditional Flow Matching

47. Conical Gaussian Paths

48. Linear Interpolation

# Endpoint Conditioning

## Conditional Flow Matching

$$\mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x}|\mathbf{z})} \|\mathbf{f}(\mathbf{x}, \mathbf{z}, t) - \mathbf{f}_\theta(\mathbf{x}, t)\|^2 \rightarrow \min_{\theta}$$

Let's define our latent variable  $\mathbf{z}$ .

## Conditioning Latent Variable [Q1]

Let us choose  $\mathbf{z} = \mathbf{x}_1$ . Then  $p(\mathbf{z}) = p_1(\mathbf{x}_1)$ .

$$p_t(\mathbf{x}) = \int p_t(\mathbf{x}|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1$$

We need to ensure the boundary constraints:

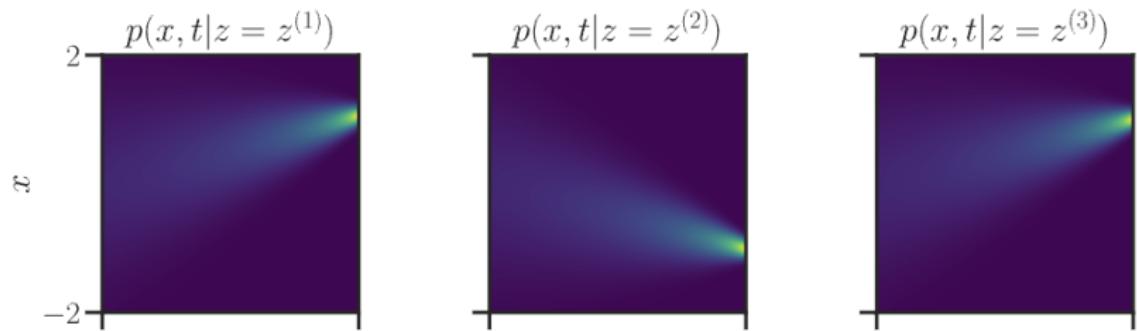
$$\begin{cases} p(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z})} p_0(\mathbf{x}|\mathbf{z}); (= \mathcal{N}(0, \mathbf{I})) \\ p_{\text{data}}(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z})} p_1(\mathbf{x}|\mathbf{z}) \end{cases} \Rightarrow \begin{cases} p_0(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(0, \mathbf{I}); \\ p_1(\mathbf{x}|\mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_1) \end{cases}$$

## Conical Gaussian Paths

$$p_0(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(0, \mathbf{I}); \quad p_1(\mathbf{x}|\mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_1)$$

## Gaussian Conditional Probability Path

$$p_t(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(\boldsymbol{\mu}_t(\mathbf{x}_1), \boldsymbol{\sigma}_t^2(\mathbf{x}_1)); \quad \mathbf{x}_t = \boldsymbol{\mu}_t(\mathbf{x}_1) + \boldsymbol{\sigma}_t(\mathbf{x}_1) \odot \epsilon$$



Let's consider straight conditional paths:

$$\begin{cases} \boldsymbol{\mu}_t(\mathbf{x}_1) = t\mathbf{x}_1 \\ \boldsymbol{\sigma}_t(\mathbf{x}_1) = 1 - t \end{cases} \Rightarrow \begin{cases} p_t(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(t\mathbf{x}_1, (1-t)^2 \cdot \mathbf{I}) \\ \mathbf{x}_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0 \end{cases}$$

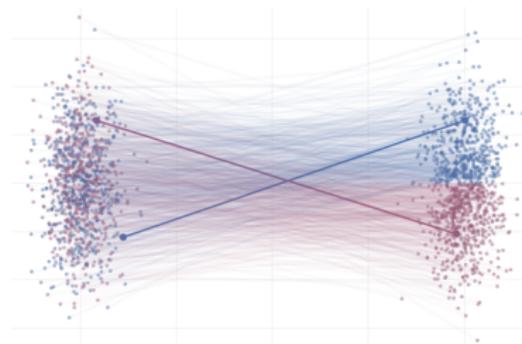
# Conical Gaussian Paths

$$p_t(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(t\mathbf{x}_1, (1-t)^2 \mathbf{I}); \quad \mathbf{x}_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0$$

## Conditional Vector Field

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{f}(\mathbf{x}_t, \mathbf{x}_1, t) = \boldsymbol{\mu}'_t(\mathbf{x}_1) + \frac{\boldsymbol{\sigma}'_t(\mathbf{x}_1)}{\boldsymbol{\sigma}_t(\mathbf{x}_1)} \odot (\mathbf{x}_t - \boldsymbol{\mu}_t(\mathbf{x}_1))$$

$$\begin{aligned}\mathbf{f}(\mathbf{x}_t, \mathbf{x}_1, t) &= \mathbf{x}_1 - \frac{1}{1-t} \cdot (\mathbf{x}_t - t\mathbf{x}_1) = \frac{\mathbf{x}_1 - \mathbf{x}_t}{1-t} = \\ &= \frac{\mathbf{x}_1 - t\mathbf{x}_1 - (1-t)\mathbf{x}_0}{1-t} = \mathbf{x}_1 - \mathbf{x}_0\end{aligned}$$



The conditional vector field  $\mathbf{f}(\mathbf{x}_t, \mathbf{x}_1, t)$  defines straight lines between  $p_{\text{data}}(\mathbf{x})$  and  $\mathcal{N}(0, \mathbf{I})$ .

# Conical Gaussian Paths

$$\begin{aligned} & \mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{z \sim p(z)} \mathbb{E}_{x \sim p_t(x|z)} \|f(x, z, t) - f_\theta(x, t)\|^2 = \\ &= \mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{x_1 \sim p_{\text{data}}(x)} \mathbb{E}_{x \sim p_t(x|x_1)} \left\| \frac{x_1 - x}{1-t} - f_\theta(x, t) \right\|^2 = \\ &= \mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{x_1 \sim p_{\text{data}}(x)} \mathbb{E}_{x_0 \sim \mathcal{N}(0, I)} \| (x_1 - x_0) - f_\theta(tx_1 + (1-t)x_0, t) \|^2 \end{aligned}$$

- ▶ We fit straight lines between the noise distribution  $p(x)$  and the data distribution  $p_{\text{data}}(x)$ .
- ▶ The **marginal** path  $p_t(x)$  does not give straight lines.



# Conditional Flow Matching

$$\mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{\mathbf{x}_1 \sim p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})} \|(\mathbf{x}_1 - \mathbf{x}_0) - \mathbf{f}_{\theta}(\mathbf{x}_t, t)\|^2 \rightarrow \min_{\theta}$$

## Training

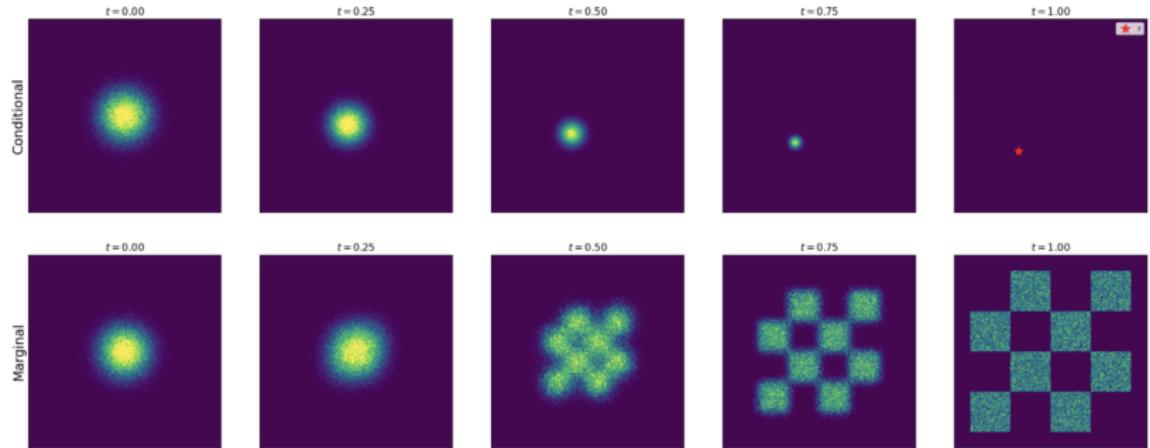
1. Sample  $\mathbf{x}_1 \sim p_{\text{data}}(\mathbf{x})$ .
2. Sample time  $t \sim U[0, 1]$  and  $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$ .
3. Obtain the noisy image  $\mathbf{x}_t = t\mathbf{x}_1 + (1 - t)\mathbf{x}_0$ .
4. Compute the loss  $\mathcal{L} = \|(\mathbf{x}_1 - \mathbf{x}_0) - \mathbf{f}_{\theta}(\mathbf{x}_t, t)\|^2$ .

## Sampling

1. Sample  $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$ .
2. Solve the ODE to obtain  $\mathbf{x}_1$ :

$$\mathbf{x}_1 = \text{ODESolve}_f(\mathbf{x}_0, \theta, t_0 = 0, t_1 = 1)$$

# Conical Gaussian Paths



- ▶ Conical gaussian paths give us the way to construct generative model.
- ▶ Now we extend it to image-to-image formulation (mapping between two distinct  $p_{\text{data}_1}(\mathbf{x})$  and  $p_{\text{data}_2}(\mathbf{x})$ ).

# Outline

46. Conditional Flow Matching

47. Conical Gaussian Paths

48. Linear Interpolation

# Pair Conditioning

## Conditional Flow Matching

$$\mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x}|\mathbf{z})} \|\mathbf{f}(\mathbf{x}, \mathbf{z}, t) - \mathbf{f}_\theta(\mathbf{x}, t)\|^2 \rightarrow \min_{\theta}$$

## Conditioning Latent Variable [Q1]

Let us choose  $\mathbf{z} = (\mathbf{x}_0, \mathbf{x}_1)$ . Then  $p(\mathbf{z}) = p(\mathbf{x}_0, \mathbf{x}_1) = p_0(\mathbf{x}_0)p_1(\mathbf{x}_1)$ .

$$p_t(\mathbf{x}) = \int p_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) p_0(\mathbf{x}_0) p_1(\mathbf{x}_1) d\mathbf{x}_0 d\mathbf{x}_1$$

We must enforce boundary constraints:

$$\begin{cases} p_0(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z})} p_0(\mathbf{x}|\mathbf{z}); \\ p_1(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z})} p_1(\mathbf{x}|\mathbf{z}) \end{cases} \Rightarrow \begin{cases} p_0(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_0) \\ p_1(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_1) \end{cases}$$

## Linear Interpolation

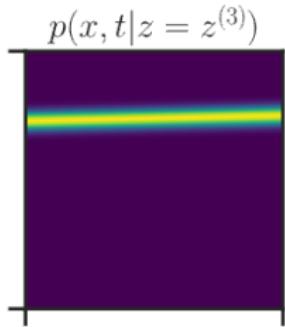
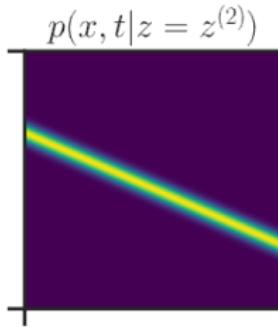
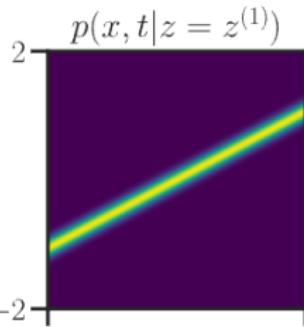
$$p_0(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_0); \quad p_1(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_1)$$

## Gaussian Conditional Probability Path

$$p_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \mathcal{N}(\boldsymbol{\mu}_t(\mathbf{x}_0, \mathbf{x}_1), \boldsymbol{\sigma}_t^2(\mathbf{x}_0, \mathbf{x}_1)); \quad \mathbf{x}_t = \boldsymbol{\mu}_t(\mathbf{x}_0, \mathbf{x}_1) + \boldsymbol{\sigma}_t(\mathbf{x}_0, \mathbf{x}_1) \odot \boldsymbol{\epsilon}$$

Let's consider straight conditional paths:

$$\begin{cases} \boldsymbol{\mu}_t(\mathbf{x}_1) = t\mathbf{x}_1 + (1-t)\mathbf{x}_0 \\ \boldsymbol{\sigma}_t(\mathbf{x}_1) = \boldsymbol{\epsilon} \end{cases} \Rightarrow \begin{cases} p_0(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_0) \\ p_1(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_1) \end{cases}$$



# Flow Matching: Conical Paths vs. Linear Interpolation

$$z = x_1$$

$$p_t(x|x_1) = \mathcal{N}(tx_1, (1-t)^2 I)$$

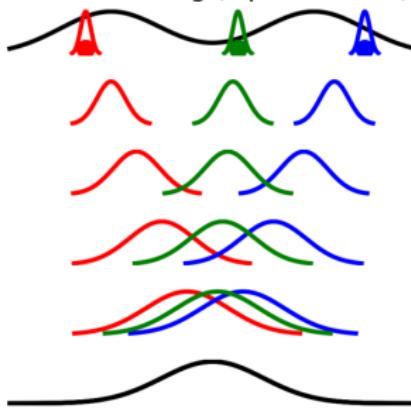
$$x_t = tx_1 + (1-t)x_0$$

$$z = (x_0, x_1)$$

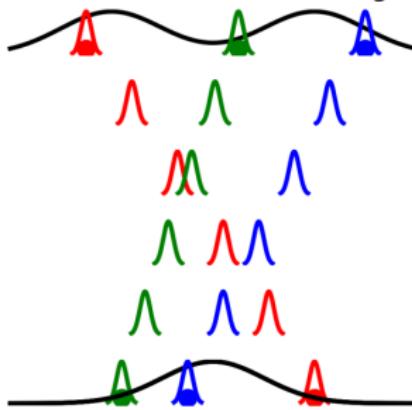
$$p_t(x|x_0, x_1) = \mathcal{N}(tx_1 + (1-t)x_0, \epsilon^2 I)$$

$$x_t = tx_1 + (1-t)x_0$$

Flow Matching (Lipman et al.)



Conditional Flow Matching



## Linear Interpolation

$$p_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \mathcal{N}\left(t\mathbf{x}_1 + (1-t)\mathbf{x}_0, \epsilon^2 \mathbf{I}\right); \quad \mathbf{x}_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0$$

## Conditional Vector Field

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{f}(\mathbf{x}_t, \mathbf{x}_0, \mathbf{x}_1, t) = \boldsymbol{\mu}'_t(\mathbf{x}_0, \mathbf{x}_1) + \frac{\boldsymbol{\sigma}'_t(\mathbf{x}_0, \mathbf{x}_1)}{\boldsymbol{\sigma}_t(\mathbf{x}_0, \mathbf{x}_1)} \odot (\mathbf{x}_t - \boldsymbol{\mu}_t(\mathbf{x}_0, \mathbf{x}_1))$$
$$\mathbf{f}(\mathbf{x}_t, \mathbf{x}_0, \mathbf{x}_1, t) = \mathbf{x}_1 - \mathbf{x}_0$$

## Conditional Flow Matching

$$\mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x}|\mathbf{z})} \|\mathbf{f}(\mathbf{x}, \mathbf{z}, t) - \mathbf{f}_\theta(\mathbf{x}, t)\|^2 =$$
$$\mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{(\mathbf{x}_0, \mathbf{x}_1) \sim p(\mathbf{x}_0, \mathbf{x}_1)} \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1)} \|(\mathbf{x}_1 - \mathbf{x}_0) - \mathbf{f}_\theta(\mathbf{x}, t)\|^2$$

- ▶ This yields the same procedure as for conical paths!
- ▶ Now, we do not require that  $p_0(\mathbf{x})$  is necessarily  $\mathcal{N}(0, \mathbf{I})$ .

## Conditional Flow Matching

- ▶ This conditioning allows us to transport any distribution  $p_0(\mathbf{x})$  to any distribution  $p_1(\mathbf{x})$ .
- ▶ It's possible to apply this approach to paired tasks, e.g., style transfer.

### Training Procedure

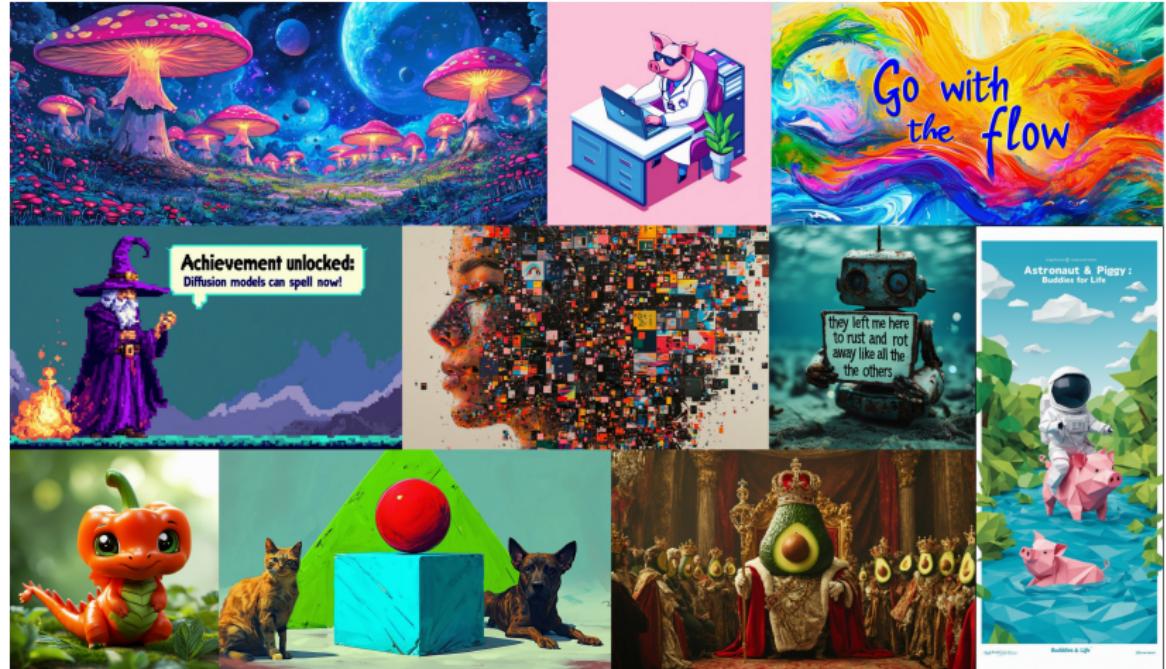
1. Sample  $(\mathbf{x}_0, \mathbf{x}_1) \sim p(\mathbf{x}_0, \mathbf{x}_1)$ .
2. Sample time  $t \sim U[0, 1]$ .
3. Compute the noisy image  $\mathbf{x}_t = t\mathbf{x}_1 + (1 - t)\mathbf{x}_0$ .
4. Compute the loss  $\mathcal{L} = \|(\mathbf{x}_1 - \mathbf{x}_0) - \mathbf{f}_{\theta}(\mathbf{x}_t, t)\|^2$ .

### Sampling

1. Sample  $\mathbf{x}_0 \sim p_0(\mathbf{x})$ .
2. Solve the ODE to obtain  $\mathbf{x}_1$ :

$$\mathbf{x}_1 = \text{ODESolve}_f(\mathbf{x}_0, \theta, t_0 = 0, t_1 = 1)$$

# Stable Diffusion 3: Scalable Flow Matching



## Summary

- ▶ Conditional flow matching makes the FM objective tractable.
- ▶ Conical Gaussian paths use endpoint conditioning  $\mathbf{z} = \mathbf{x}_1$  and serve as an effective FM technique.
- ▶ Linear Interpolation paths use pair conditioning  $\mathbf{z} = (\mathbf{x}_0, \mathbf{x}_1)$  and yields the same procedure, but is more general (suitable for paired tasks).

# Deep Generative Models

## Lecture 13

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

49. Link between Flow Matching and Score-Based Models

50. Discrete Diffusion Models

Forward Discrete Process

Reverse Discrete Diffusion

# Outline

49. Link between Flow Matching and Score-Based Models

50. Discrete Diffusion Models

Forward Discrete Process

Reverse Discrete Diffusion

# Score-Based Generative Models through SDEs

## Training

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}(0))} \mathbb{E}_{t \sim U[0,1]} \mathbb{E}_{q(\mathbf{x}(t)|\mathbf{x}(0))} \| \mathbf{s}_{\theta}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log q(\mathbf{x}(t)|\mathbf{x}(0)) \|_2^2$$

## Variance Exploding SDE (NCSN)

$$q(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}(\mathbf{x}(0), [\sigma^2(t) - \sigma^2(0)] \cdot \mathbf{I}), \quad \sigma(0) = 0$$

## Variance Preserving SDE (DDPM)

$$q(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}(\mathbf{x}(0)\alpha(t), (1 - \alpha(t)^2) \cdot \mathbf{I}); \quad \alpha(t) = e^{-\frac{1}{2} \int_0^t \beta(s) ds}$$

Flow matching uses reverse time direction:

$$p_t(\mathbf{x}|\mathbf{x}_1) = q_{1-t}(\mathbf{x}|\mathbf{x}_0 = \mathbf{x}_1)$$

# Score-Based Generative Models through SDEs

$$p_t(\mathbf{x}|\mathbf{x}_1) = q_{1-t}(\mathbf{x}|\mathbf{x}_0 = \mathbf{x}_1)$$

**VE (NCSN):**  $p_t(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1, \sigma_{1-t}^2 \cdot \mathbf{I})$

**VP (DDPM):**  $p_t(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(\alpha_{1-t}\mathbf{x}_1, (1 - \alpha_{1-t}^2) \cdot \mathbf{I})$

## Flow Matching Probability Path

$$p_t(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(t\mathbf{x}_1, (1 - t)^2 \mathbf{I}); \quad \mathbf{f}(\mathbf{x}_t, \mathbf{x}_1, t) = \frac{\mathbf{x}_1 - \mathbf{x}_t}{1 - t}$$

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{f}(\mathbf{x}_t, \mathbf{x}_1, t) = \boldsymbol{\mu}'_t(\mathbf{x}_1) + \frac{\boldsymbol{\sigma}'_t(\mathbf{x}_1)}{\boldsymbol{\sigma}_t(\mathbf{x}_1)} \odot (\mathbf{x}_t - \boldsymbol{\mu}_t(\mathbf{x}_1))$$

Let's derive the conditional vector fields for VE (NCSN) and VP (DDPM).

## Flow Matching vs. Score-Based SDE Models

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{f}(\mathbf{x}_t, \mathbf{x}_1, t) = \boldsymbol{\mu}'_t(\mathbf{x}_1) + \frac{\sigma'_t(\mathbf{x}_1)}{\sigma_t(\mathbf{x}_1)} \odot (\mathbf{x}_t - \boldsymbol{\mu}_t(\mathbf{x}_1))$$

### Variance Exploding SDE Probability Path

$$p_t(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1, \sigma_{1-t}^2 \mathbf{I}) \quad \Rightarrow \quad \mathbf{f}(\mathbf{x}_t, \mathbf{x}_1, t) = -\frac{\sigma'_{1-t}}{\sigma_{1-t}}(\mathbf{x}_t - \mathbf{x}_1)$$

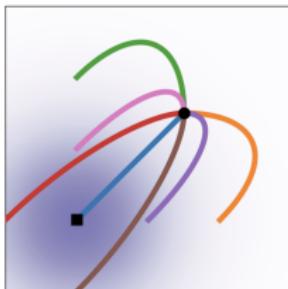
### Variance Preserving SDE Probability Path

$$p_t(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(\alpha_{1-t}\mathbf{x}_1, (1 - \alpha_{1-t}^2)\mathbf{I}) \Rightarrow \mathbf{f}(\mathbf{x}_t, \mathbf{x}_1, t) = \frac{\alpha'_{1-t}}{1 - \alpha_{1-t}^2} \cdot (\alpha_{1-t}\mathbf{x}_t - \mathbf{x}_1)$$

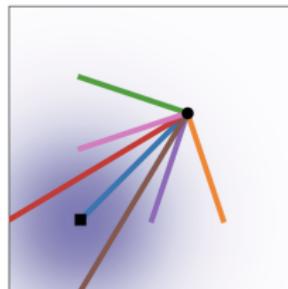
Thus, VE/VP SDE models correspond to particular choices of the Gaussian probability path within the flow matching framework.

# Flow Matching vs. Score-Based SDE Models

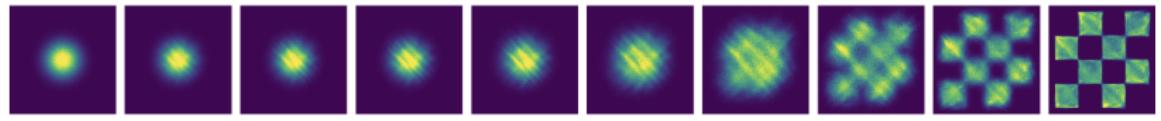
## Trajectories



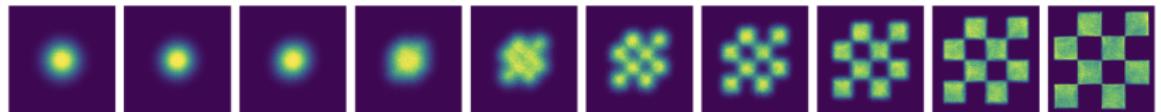
Diffusion



OT



Score matching w/ Diffusion



Flow Matching w/ OT

# Outline

49. Link between Flow Matching and Score-Based Models

50. Discrete Diffusion Models

Forward Discrete Process

Reverse Discrete Diffusion

# Discrete or Continuous Diffusion Models?

**Reminder:** Diffusion models define a forward corruption process and a reverse denoising process. Previously, we studied diffusion models with continuous states  $\mathbf{x}(t) \in \mathbb{R}^m$ .

## Continuous state space

- ▶ **Discrete time**  $t \in \{0, 1, \dots, T\} \Rightarrow \text{DDPM / NCSN.}$
- ▶ **Continuous time**  $t \in [0, 1] \Rightarrow \text{Score-based SDE models.}$

Now we turn to diffusion over discrete-value states  
 $\mathbf{x}(t) \in \{1, \dots, K\}^m$ .

## Discrete state space

- ▶ **Discrete time**  $t \in \{0, 1, \dots, T\}.$
- ▶ **Continuous time**  $t \in [0, 1].$

Let's discuss why we need discrete diffusion models.

# Why Discrete Diffusion Models?

While autoregressive (AR) models dominate discrete-data domains (e.g., text or sequences), they have fundamental limitations.

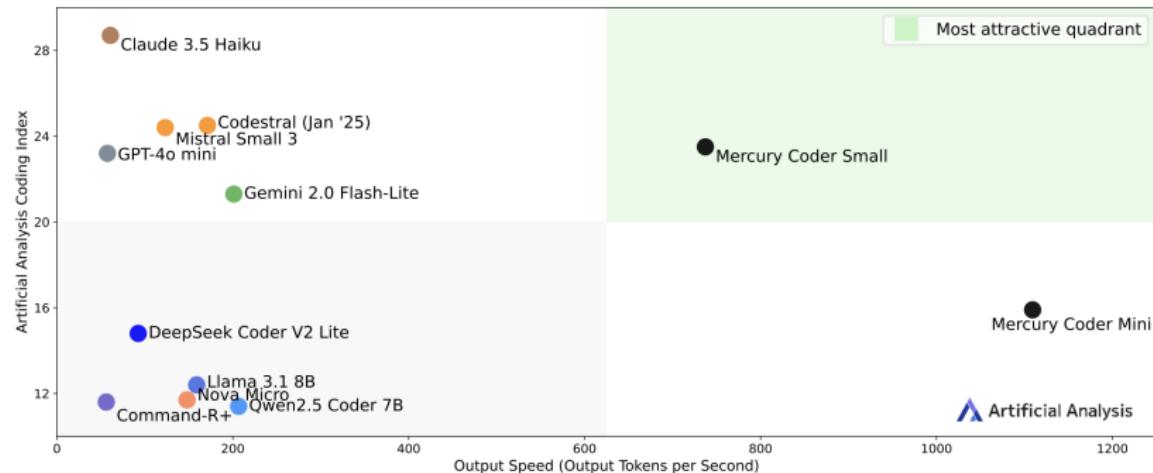
## Key advantages of discrete diffusion

- ▶ **Parallel generation:** diffusion enables sampling all tokens simultaneously, unlike AR's strictly left-to-right process.
- ▶ **Flexible infilling:** diffusion can mask arbitrary parts of a sequence and reconstruct them, rather than generating only from prefix to suffix.
- ▶ **Robustness:** diffusion avoids the "exposure bias" caused by teacher forcing in AR training.
- ▶ **Unified framework:** diffusion generalizes naturally to discrete domains that do not suit continuous Gaussian noise.

# 2025 – Big Bang of Discrete Diffusion Models

## Coding Index vs. Output Speed: Smaller models

Artificial Analysis Coding Index (represents the average of LiveCodeBench & SciCode);  
Output Speed: Output Tokens per Second; 1,000 Input Tokens; Coding focused workload



# Outline

49. Link between Flow Matching and Score-Based Models

50. Discrete Diffusion Models

Forward Discrete Process

Reverse Discrete Diffusion

## Forward Discrete Process

### Continuous Diffusion Markov Chain

In continuous diffusion, the forward Markov chain is defined by progressively corrupting data with Gaussian noise:

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

### Discrete Diffusion Markov Chain

For discrete data, we instead define a Markov chain over categorical states:

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \text{Cat}(\mathbf{Q}_t \mathbf{x}_{t-1}),$$

- ▶ Each  $\mathbf{x}_t \in \{0, 1\}^K$  is a **one-hot vector** encoding the categorical state (it is just one token).
- ▶ What is the transition matrix  $\mathbf{Q}_t$ ?

## Forward Process over Time

### Transition Matrix

$\mathbf{Q}_t \in [0, 1]^{K \times K}$  is a **transition matrix** where each column gives transition probabilities from one state to all others, and columns sum to 1:

$$[\mathbf{Q}_t]_{ij} = q(x_t = i | x_{t-1} = j), \quad \sum_{i=1}^K [\mathbf{Q}_t]_{ij} = 1.$$

- ▶ The forward diffusion gradually destroys information through repeated random transitions.
- ▶ Applying the transition  $t$  times yields the marginal distribution:

$$q(\mathbf{x}_t | \mathbf{x}_0) = \text{Cat}(\mathbf{Q}_{1:t} \mathbf{x}_0), \quad \mathbf{Q}_{1:t} = \mathbf{Q}_t \mathbf{Q}_{t-1} \cdots \mathbf{Q}_1.$$

- ▶ As  $t \rightarrow T$ , the process drives the data toward a stationary distribution.
- ▶ We design the transition matrices  $\mathbf{Q}_t$  to achieve this behavior.

## Transition Matrix

- ▶ The choice of  $\mathbf{Q}_t$  determines how information is erased and what the stationary distribution becomes.
- ▶  $\mathbf{Q}_t$  and  $\mathbf{Q}_{1:t}$  should be easy to compute for each  $t$ .

### Common choices

- ▶ **Uniform diffusion**

$$\mathbf{Q}_t = (1 - \beta_t)\mathbf{I} + \beta_t \mathbf{U}, \quad \mathbf{U}_{ij} = \frac{1}{K}.$$

Each token is replaced by a uniformly random symbol with probability  $\beta_t$ . The stationary distribution is uniform noise.

- ▶ **Absorbing diffusion**

$$\mathbf{Q}_t = (1 - \beta_t)\mathbf{I} + \beta_t \mathbf{e}_m \mathbf{1}^\top.$$

Tokens are gradually replaced by a special mask  $m$ ; the stationary distribution is fully masked.

## Transition Matrix

$$q(\mathbf{x}_t | \mathbf{x}_0) = \text{Cat}(\mathbf{Q}_{1:t} \mathbf{x}_0), \quad \mathbf{Q}_{1:t} = \mathbf{Q}_t \mathbf{Q}_{t-1} \cdots \mathbf{Q}_1.$$

### Uniform Diffusion

$$\mathbf{Q}_t = (1 - \beta_t) \mathbf{I} + \beta_t \mathbf{U}, \quad \mathbf{U}_{ij} = \frac{1}{K}.$$

$$\mathbf{Q}_{1:t} = \bar{\alpha}_t \mathbf{I} + (1 - \bar{\alpha}_t) \mathbf{U}, \quad \bar{\alpha}_t = \prod_{s=1}^t (1 - \beta_s).$$

- ▶ Each token retains its original value with prob.  $\bar{\alpha}_t$ .
- ▶ It becomes uniformly random with prob.  $(1 - \bar{\alpha}_t)$ .
- ▶ As  $t \rightarrow T$ , the process converges to the stationary uniform distribution.

# Transition Matrix

## Absorbing Diffusion

$$\mathbf{Q}_t = (1 - \beta_t) \mathbf{I} + \beta_t \mathbf{e}_m \mathbf{1}^\top,$$

$$\mathbf{Q}_{1:t} = \bar{\alpha}_t \mathbf{I} + (1 - \bar{\alpha}_t) \mathbf{e}_m \mathbf{1}^\top, \quad \bar{\alpha}_t = \prod_{s=1}^t (1 - \beta_s).$$

- ▶ Each token retains its original value with prob.  $\bar{\alpha}_t$ .
- ▶ It becomes  $\mathbf{e}_m$  with prob.  $(1 - \bar{\alpha}_t)$ .
- ▶ As  $t \rightarrow T$ , all tokens converge to the mask state:  
 $q(\mathbf{x}_T) \approx \text{Cat}(\mathbf{e}_m)$ .
- ▶ This makes the process analogous to **masked language modeling**.

# Uniform vs. Absorbing Transition Matrix

| Aspect                  | Uniform Diffusion   | Absorbing Diffusion  |
|-------------------------|---|--|
| $\mathbf{Q}_t$          | $(1 - \beta_t)\mathbf{I} + \beta_t\mathbf{U}$               | $(1 - \beta_t)\mathbf{I} + \beta_t\mathbf{e}_m\mathbf{1}^\top$               |
| $\mathbf{Q}_{1:t}$      | $\bar{\alpha}_t\mathbf{I} + (1 - \bar{\alpha}_t)\mathbf{U}$ | $\bar{\alpha}_t\mathbf{I} + (1 - \bar{\alpha}_t)\mathbf{e}_m\mathbf{1}^\top$ |
| $\mathbf{Q}_{1:\infty}$ | $\mathbf{U}$  | $\text{Cat}(\mathbf{e}_m)$   |
| Interpretation          | Random replacement  | Gradual masking of tokens  |
| Application             | Image diffusion   | Text diffusion $\approx$ Masked LM   |

## Observation

Both schemes gradually destroy information, but differ in their stationary limit. Absorbing diffusion bridges diffusion and masked-language-model objectives.

# Outline

49. Link between Flow Matching and Score-Based Models

50. Discrete Diffusion Models

Forward Discrete Process

Reverse Discrete Diffusion

# Posterior of the Forward Process

## ELBO

$$\begin{aligned}\mathcal{L}_{\phi, \theta}(\mathbf{x}) = & \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) - \text{KL}(q(\mathbf{x}_T|\mathbf{x}_0)\|p(\mathbf{x}_T)) - \\ & - \sum_{t=2}^T \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \text{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\|p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\mathcal{L}_t}\end{aligned}$$

- ▶ Conditioned reverse distribution  $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  played crucial role in the continuous-state diffusion model.
- ▶ It shows the probability of a previous state given the noisy state  $\mathbf{x}_t$  and the original clean data  $\mathbf{x}_0$ .

## Discrete conditioned reverse distribution

$$\begin{aligned}q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\ &= \frac{\text{Cat}(\mathbf{Q}_t) \cdot \text{Cat}(\mathbf{Q}_{1:t-1})}{\text{Cat}(\mathbf{Q}_{1:t})}.\end{aligned}$$

# Posterior of the Forward Process

Discrete conditioned reverse distribution

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \text{Cat}\left(\frac{\mathbf{Q}_t \mathbf{x}_t \odot \mathbf{Q}_{1:t-1} \mathbf{x}_0}{\mathbf{x}_t^\top \mathbf{Q}_{1:t} \mathbf{x}_0}\right).$$

Recall the ELBO term

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \text{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)),$$

- ▶ Both  $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$  and  $q(\mathbf{x}_t | \mathbf{x}_0)$  are known analytically from the forward process.
- ▶ The reverse process  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$  is a learned categorical distribution:

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \text{Cat}(\pi_\theta(\mathbf{x}_t, t)),$$

where  $\pi_\theta$  is a neural network.

# Discrete-time ELBO for Discrete Diffusion

## ELBO term

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \text{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)).$$

## Categorical KL

$$\text{KL}(\text{Cat}(\mathbf{q}) \parallel \text{Cat}(\mathbf{p})) = \sum_{k=1}^K q_k \log \frac{q_k}{p_k} = H(\mathbf{q}, \mathbf{p}) - H(\mathbf{q}),$$

- ▶  $H(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0))$  is a constant w.r.t.  $\theta$ .
- ▶  $H(\mathbf{q}, \mathbf{p}) = -\sum_k q_k \log p_k$  is a **cross-entropy loss**.

Therefore, minimizing  $\mathcal{L}_t$  w.r.t.  $\theta$  is equivalent to minimizing

$$\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} H(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0), p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)).$$

## Summary

- ▶ Diffusion and score-based models are special cases of the flow matching approach, but use curved trajectories.
- ▶ Diffusion approach has several key advantages over autoregressive approach.
- ▶ Forward discrete diffusion process defines Markov chain with discrete states.
- ▶ There are several ways to make it tractable (uniform / absorbing transitions).
- ▶ Reverse discrete diffusion process uses the variational approach to invert forward process.
- ▶ Discrete-state ELBO for discrete diffusion is a cross-entropy loss.

# Deep Generative Models

## Lecture 14

Roman Isachenko

Moscow Institute of Physics and Technology  
Yandex School of Data Analysis

2025, Autumn

# Outline

## 51. Discrete Diffusion

From Token to Sequence

Absorbing Diffusion

Continuous-Time Formulation

## 52. Course Overview

# Outline

## 51. Discrete Diffusion

From Token to Sequence

Absorbing Diffusion

Continuous-Time Formulation

## 52. Course Overview

# Outline

## 51. Discrete Diffusion

From Token to Sequence

Absorbing Diffusion

Continuous-Time Formulation

## 52. Course Overview

# From Token to Sequence

## One-hot sequence representation

$$\mathbf{x}_t \in \{0, 1\}^K \Leftrightarrow \mathbf{X}_t \in \{0, 1\}^{K \times m}$$

Here  $\mathbf{X}_t$  is a one-hot representation of a sequence of tokens.

## Independent Token-wise Forward Process

$$q(\mathbf{X}_t | \mathbf{X}_{t-1}) = \prod_{i=1}^m q(\mathbf{x}_t^i | \mathbf{x}_{t-1}^i) = \text{Cat}(\mathbf{Q}_t \mathbf{X}_{t-1})$$

- ▶ Each position  $i$  evolves according to its own Markov chain.
- ▶ Often the same transition matrix  $\mathbf{Q}_t$  is shared across  $i$ .

## Continuous Diffusion Analogy

- ▶ In Gaussian DDPMs with diagonal covariance, noise is independent per pixel.
- ▶ Structure is not in the noise; it is learned by the reverse model.

## From Token to Sequence

$$q(\mathbf{X}_t | \mathbf{X}_{t-1}) = \prod_{i=1}^m q(\mathbf{x}_t^i | \mathbf{x}_{t-1}^i) = \text{Cat}(\mathbf{Q}_t \mathbf{X}_{t-1})$$
$$q(\mathbf{X}_t | \mathbf{X}_{t-1}) = \text{Cat}(\mathbf{Q}_{1:t} \mathbf{X}_0)$$

## Conditioned Reverse Distribution

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \text{Cat} \left( \frac{\mathbf{Q}_t \mathbf{x}_t \odot \mathbf{Q}_{1:t-1} \mathbf{x}_0}{\mathbf{x}_t^\top \mathbf{Q}_{1:t} \mathbf{x}_0} \right).$$
$$q(\mathbf{X}_{t-1} | \mathbf{X}_t, \mathbf{X}_0) = \prod_{i=1}^m q(\mathbf{x}_{t-1}^i | \mathbf{x}_t^i, \mathbf{x}_0^i).$$

- ▶ All distributions defined by the forward process are factorized.
- ▶ Dependence appears in the learned reverse model.

## Reverse Model for Sequence

$$p_{\theta}(\mathbf{X}_{t-1}|\mathbf{X}_t) = \prod_{i=1}^m p_{\theta}(\mathbf{x}_{t-1}^i|\mathbf{X}_{\textcolor{violet}{t}}).$$

- ▶ The output factorizes (parallel prediction across positions).
- ▶ Each factor conditions on the entire noisy sequence  $\mathbf{X}_{\textcolor{violet}{t}}$ .
- ▶ This is exactly the **masked language modeling** pattern.

Objective:  $\mathcal{L}_t$  term

$$\begin{aligned} \text{KL}(q(\mathbf{X}_{t-1}|\mathbf{X}_t, \mathbf{X}_0) \| p_{\theta}(\mathbf{X}_{t-1}|\mathbf{X}_t)) \\ = \sum_{i=1}^m \text{KL}(q(\mathbf{x}_{t-1}^i|\mathbf{x}_t^i, \mathbf{x}_0^i) \| p_{\theta}(\mathbf{x}_{t-1}^i|\mathbf{x}_t)). \end{aligned}$$

Final objective: masked LM

$$\mathcal{L} = \sum_{t=1}^T \sum_{i=1}^m \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[ -\log p_{\theta}(\mathbf{x}_0^i|\mathbf{X}_t) \right].$$

# Outline

## 51. Discrete Diffusion

From Token to Sequence

Absorbing Diffusion

Continuous-Time Formulation

## 52. Course Overview

## Absorbing Diffusion: Forward Process

Let's restrict to the case of absorbing transition matrix.

$$\mathbf{Q}_t = (1 - \beta_t) \mathbf{I} + \beta_t \mathbf{e}_m \mathbf{1}^\top, \quad \bar{\alpha}_t = \prod_{s=1}^t (1 - \beta_s).$$
$$\mathbf{Q}_{1:t} = \bar{\alpha}_t \mathbf{I} + (1 - \bar{\alpha}_t) \mathbf{e}_m \mathbf{1}^\top.$$

Each position is either still clean or already masked:

$$q(\mathbf{x}_t | \mathbf{x}_0) = \bar{\alpha}_t [\mathbf{x}_t = \mathbf{x}_0] + (1 - \bar{\alpha}_t) [\mathbf{x}_t = \mathbf{e}_m]$$

$$\mathbf{Q}_t = \begin{pmatrix} 1 - \beta_t & 0 & \textcolor{violet}{0} \\ 0 & 1 - \beta_t & \textcolor{violet}{0} \\ \beta_t & \beta_t & \textcolor{violet}{1} \end{pmatrix} \Rightarrow \text{the masked state is absorbing.}$$

What happens in the conditioned reverse process  $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ ?

# Absorbing Diffusion

## Conditioned reverse distribution

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \begin{cases} [\mathbf{x}_{t-1} = \mathbf{x}_t], & \text{if } \mathbf{x}_t \neq \mathbf{e}_m, \\ \rho_t [\mathbf{x}_{t-1} = \mathbf{x}_0] + (1 - \rho_t) [\mathbf{x}_{t-1} = \mathbf{e}_m], & \text{if } \mathbf{x}_t = \mathbf{e}_m, \end{cases}$$

where

$$\rho_t = \frac{\beta_t \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}.$$

- ▶ If  $\mathbf{x}_t \neq \mathbf{e}_m$ , then the token must be unchanged:  $\mathbf{x}_{t-1} = \mathbf{x}_t$ .
- ▶ Observing an unmasked token at time  $t$  fixes the entire history:  $\mathbf{x}_{t-1} = \mathbf{x}_t = \dots = \mathbf{x}_0$ .
- ▶ If  $\mathbf{x}_t = \mathbf{e}_m$ , the previous token may be either clean or masked.
- ▶ With probability  $\rho_t$ , masking occurred exactly at step  $t$  (so  $\mathbf{x}_{t-1} = \mathbf{x}_0$ ).
- ▶ With probability  $(1 - \rho_t)$ , the token was already masked earlier (so  $\mathbf{x}_{t-1} = \mathbf{e}_m$ ).

# Absorbing Diffusion

## Sequence Distribution

$$q(\mathbf{X}_{t-1} | \mathbf{X}_t, \mathbf{X}_0) = \prod_{i=1}^m q(\mathbf{x}_{t-1}^i | \mathbf{x}_t^i, \mathbf{x}_0^i).$$

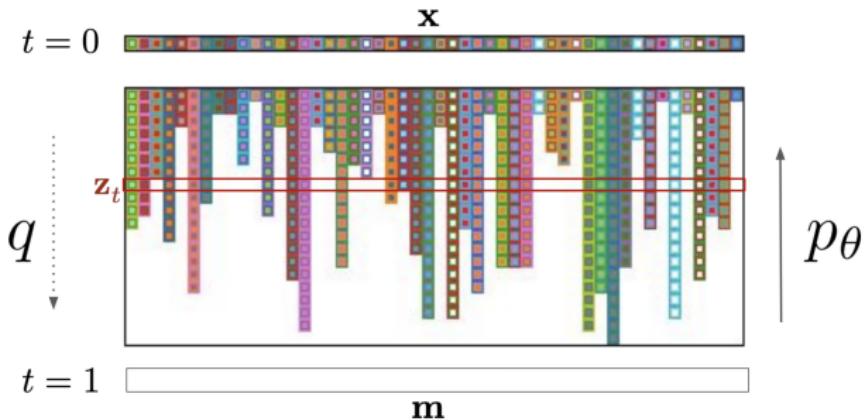
Each position  $i$  has two possible cases:

$$\mathbf{x}_t^i \neq \mathbf{e}_m \Rightarrow \mathbf{x}_{t-1}^i = \mathbf{x}_t^i, \quad \mathbf{x}_t^i = \mathbf{e}_m \Rightarrow \mathbf{x}_{t-1}^i \in \{\mathbf{x}_0^i, \mathbf{e}_m\}.$$

## Interpretation

- ▶ The forward process produces **random partial observations** of the clean sequence.
- ▶ If a token is visible at time  $t$ , the reverse distribution is deterministic.
- ▶ Unmasked tokens yield a deterministic posterior and therefore contribute only a constant to the ELBO. Therefore, only masked tokens contribute to the training loss.

# Absorbing Diffusion



$$p_\theta(\mathbf{X}_{t-1}|\mathbf{X}_t) = \prod_{i=1}^m p_\theta(\mathbf{x}_{t-1}^i|\mathbf{X}_t).$$

Objective: sequence-level  $\mathcal{L}_t$

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{X}_t|\mathbf{X}_0)} \sum_{i=1}^m \rho_t[\mathbf{x}_t^i = \mathbf{e}_m] [-\log p_\theta(\mathbf{x}_0^i|\mathbf{X}_t)] + \text{const}$$

# Outline

## 51. Discrete Diffusion

From Token to Sequence

Absorbing Diffusion

Continuous-Time Formulation

## 52. Course Overview

## From Discrete Time to Mask Rate

In absorbing diffusion, the forward process is

$$q(\mathbf{x}_t | \mathbf{x}_0) = \bar{\alpha}_t [\mathbf{x}_t = \mathbf{x}_0] + (1 - \bar{\alpha}_t) [\mathbf{x}_t = \mathbf{e}_m].$$

- ▶ The distribution depends on  $t$  only through the scalar

$$\lambda_t = 1 - \bar{\alpha}_t \in [0, 1].$$

- ▶ We can therefore reparameterize the corruption level by

$$t \quad \Rightarrow \quad \lambda \in [0, 1].$$

- ▶ We directly define a family of corrupted distributions indexed by a continuous mask rate  $\lambda$ :

$$q(\mathbf{x}_\lambda | \mathbf{x}_0) = (1 - \lambda) [\mathbf{x}_\lambda = \mathbf{x}_0] + \lambda [\mathbf{x}_\lambda = \mathbf{e}_m].$$

## Discrete ELBO Revisited

Recall the per-step ELBO term for absorbing diffusion:

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \sum_{i=1}^m \rho_t[\mathbf{x}_t^i = \mathbf{e}_m] [-\log p_\theta(\mathbf{x}_0^i | \mathbf{X}_t)] + \text{const.}$$

Replacing the discrete index  $t$  with the continuous mask rate  $\lambda$ , the training objective becomes

$$\mathcal{L} = \int_0^1 w(\lambda) \mathbb{E}_{q_\lambda(\mathbf{x}_\lambda | \mathbf{x}_0)} \sum_{i=1}^m [\mathbf{x}_\lambda^i = \mathbf{e}_m] [-\log p_\theta(\mathbf{x}_0^i | \mathbf{X}_\lambda)] d\lambda.$$

### Interpretation

Training corresponds to optimizing a **continuous mixture of masked language modeling objectives** with different mask rates.

# Algorithm: Masked Diffusion Language Model (MDLM)

## Training

1. Sample  $\mathbf{X}_0 \sim p_{\text{data}}(\mathbf{X})$  and  $\lambda \sim w(\lambda)$ .
2. Corrupt the sequence by independent masking:

$$\mathbf{x}_\lambda^i = \begin{cases} \mathbf{e}_m, & \text{with prob. } \lambda, \\ \mathbf{x}_0^i, & \text{with prob. } 1 - \lambda, \end{cases} \quad i = 1, \dots, m.$$

3. Predict token distributions in parallel:

$$p_\theta(\mathbf{X}_0 | \mathbf{X}_\lambda) = \prod_{i=1}^m p_\theta(\mathbf{x}_0^i | \mathbf{X}_\lambda).$$

4. Compute the masked-CE loss:

$$\mathcal{L}(\theta) = \sum_{i=1}^m [\mathbf{x}_\lambda^i = \mathbf{e}_m] \left[ -\log p_\theta(\mathbf{x}_0^i | \mathbf{X}_\lambda) \right].$$

# Algorithm: Masked Diffusion Language Model (MDLM)

## Sampling

1. Initialize  $\mathbf{X} \leftarrow \mathbf{e}_m \mathbf{1}^\top$  (fully masked).
2. For a decreasing schedule  $\lambda_1 > \lambda_2 > \dots > \lambda_L$ :
  - 2.1 Predict  $p_\theta(\mathbf{x}^i | \mathbf{X})$  for all positions in parallel.
  - 2.2 Unmask a subset of positions to reach the next mask rate  $\lambda_{\ell+1}$  (e.g., sample tokens for newly-unmasked positions).
3. Return the final sequence  $\mathbf{X}$  (fully unmasked).

# Outline

## 51. Discrete Diffusion

From Token to Sequence

Absorbing Diffusion

Continuous-Time Formulation

## 52. Course Overview

# Course Overview: Problem Statement

## Goal

Learn a generative model  $p_{\theta}(\mathbf{x})$  that matches the data distribution  $p_{\text{data}}(\mathbf{x})$ .

## Three similar lenses

- ▶ **Divergence minimization:**

$$\min_{\theta} D(p_{\text{data}} || p_{\theta}) \quad (\text{KL, JS, Wasserstein, etc.})$$

- ▶ **Likelihood-based modeling:** maximize  $\log p_{\theta}(\mathbf{x})$  (NF, VAE, diffusion-as-VAE).
- ▶ **Score-based modeling:** learn  $\nabla_{\mathbf{x}} \log p(\mathbf{x})$  (DDPM / NCSN / SDE).

# What We Covered: Part 1

Likelihood-based

- ▶ Autoregressive (L1):

$$p_{\theta}(\mathbf{x}) = \prod_{i=1}^n p_{\theta}(x_i | \mathbf{x}_{1:i})$$

- ▶ Normalizing Flows (L2, L10):

$$\mathbf{x} = \mathbf{f}_{\theta}(\mathbf{z}), \quad \log p_{\theta}(\mathbf{x}) = \log p(\mathbf{z}) + \log \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right|$$

- ▶ VAE (L3–L4):

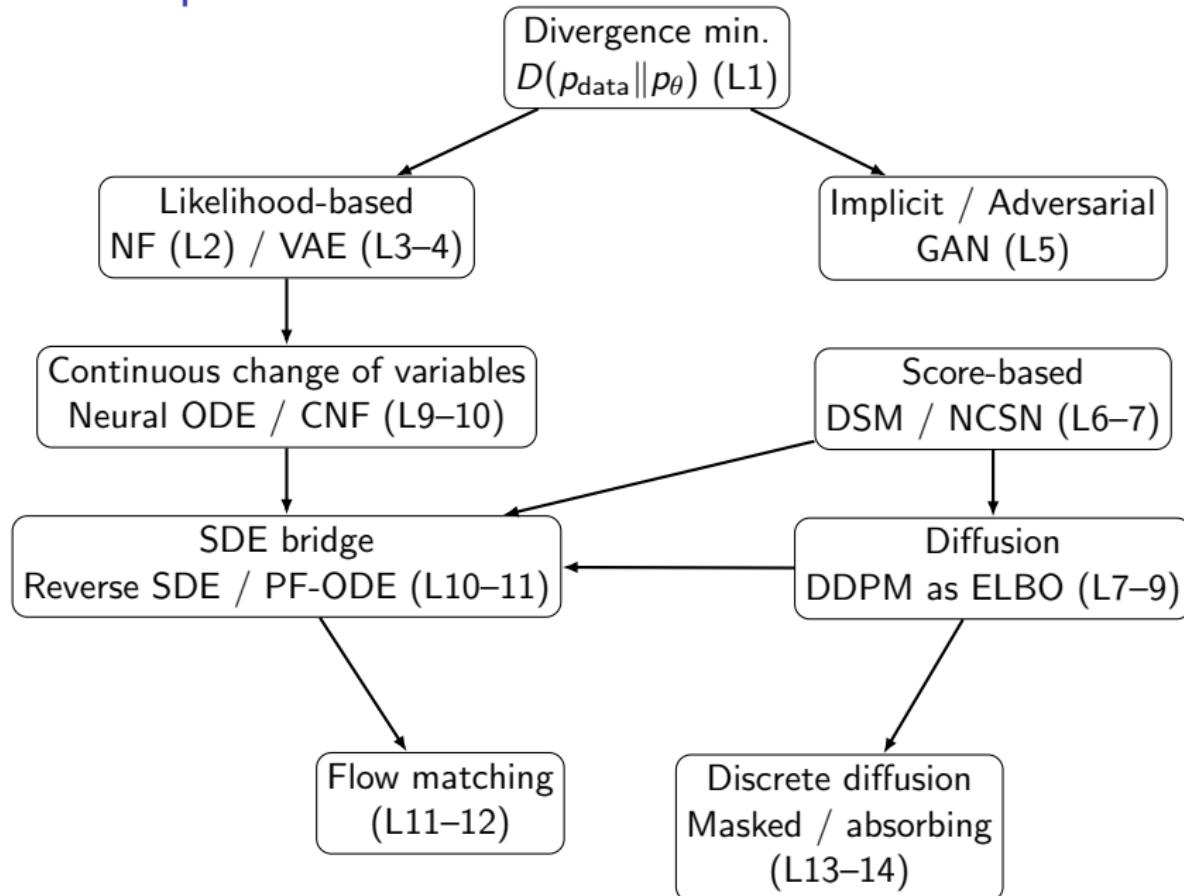
$$\log p_{\theta}(\mathbf{x}) \geq \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})] - \text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}))$$

## What We Covered: Part 2

### Implicit / Score-based

- ▶ **GAN (L5)**: implicit  $p_\theta$ , adversarial learning (close to JSD)
- ▶ **Score matching (L6)**: learn  $s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- ▶ **Diffusion / DDPM (L7–L9)**: forward noising  $q(\mathbf{x}_t | \mathbf{x}_{t-1})$  + reverse denoising
- ▶ **SDE / Flow matching (L10–L12)**: reverse SDE  $\leftrightarrow$  probability flow ODE, vector field (Neural ODE) / flow matching
- ▶ **Discrete diffusion (L13–L14)**: Markov chain on categorical states

# Mental Map



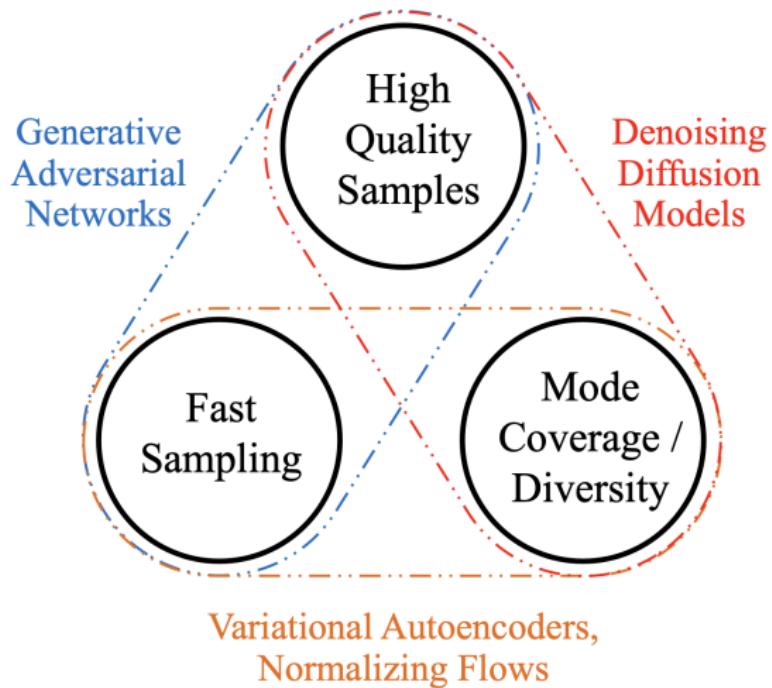
# Comparison Cheat-Sheet: Part 1

| Family       | Likelihood      | Training            |
|--------------|-----------------|---------------------|
| AR           | ✓               | stable CE           |
| NF           | ✓               | tricky architecture |
| VAE          | lower bound     | stable ELBO         |
| GAN          | ✗               | unstable/minimax    |
| Diffusion    | bound / est.    | stable              |
| FM / ODE     | est. / bound    | stable              |
| Discr. diff. | bound / CE-like | stable              |

## Comparison Cheat-Sheet: Part 2

| Family       | Sampling          | Best for                      |
|--------------|-------------------|-------------------------------|
| AR           | slow (sequential) | text / discrete               |
| NF           | fast (1 step)     | exact density, OOD            |
| VAE          | fast (1 step)     | latent modelling              |
| GAN          | fast (1 step)     | sharp images                  |
| Diffusion    | slow (many steps) | high fidelity + diversity     |
| FM / ODE     | medium–fast       | fewer steps + theory          |
| Discr. diff. | iterative         | sequences + bidirectional gen |

# Generative Learning Trilemma



# Generative Learning Trilemma

## Rule of Thumb

- ▶ **Likelihood & Coverage** ⇒ **AR / NF** exact density, no mode dropping, *slow sampling*
- ▶ **Likelihood & Fast Sampling** ⇒ **VAE** tractable bound, fast, *blurry samples*
- ▶ **Sample Quality & Fast Sampling** ⇒ **GAN** sharp samples, *no likelihood, mode collapse*
- ▶ **Quality & Coverage** ⇒ **Diffusion** stable training, high fidelity, *slow sampling*
- ▶ **Quality & Faster Sampling** ⇒ **FM / ODE** fewer steps, continuous flows, *approx. likelihood*
- ▶ **Discrete Structure & Coverage** ⇒ **Discrete Diffusion** stable CE training, parallel denoising, *iterative decoding*

## Summary

- ▶ For sequences, the forward process of discrete diffusion factorize over positions, but reverse process (the model  $p_\theta$ ) conditions on the whole context.
- ▶ In the absorbing case, tokens are either unchanged or masked; so only masked tokens contribute to the ELBO loss.
- ▶ The discrete ELBO reduces to a MLM objective.
- ▶ Reparameterizing time by the mask rate  $\lambda \in [0, 1]$  yields a continuous mixture of MLM losses.
- ▶ MDLM sampling performs iterative parallel refinement from fully masked to fully unmasked sequences.
- ▶ No generative model is strictly better than all others: different methods occupy different corners of the generative learning trilemma and come with unavoidable disadvantages.