

# Deep Generative Models

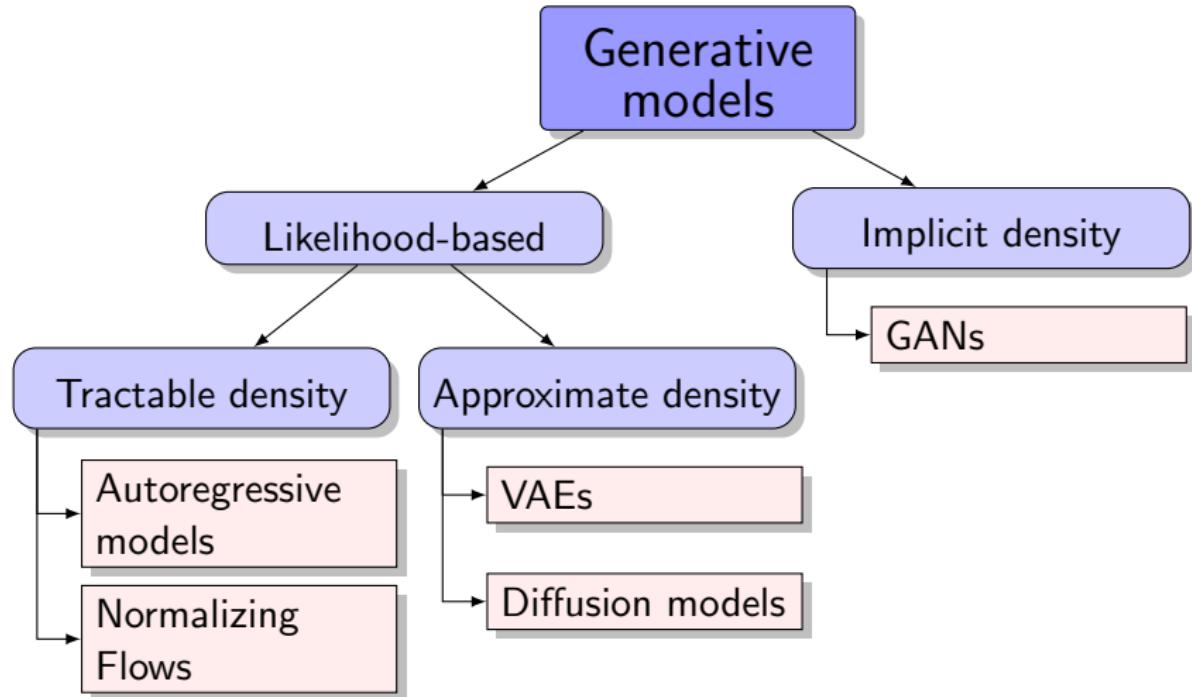
## Lecture 1

Roman Isachenko



2025, Spring

# Generative models zoo



# Outline

1. Generative models overview
2. Course tricks
3. Problem statement
4. Divergence minimization framework
5. Autoregressive modelling

# Outline

1. Generative models overview
2. Course tricks
3. Problem statement
4. Divergence minimization framework
5. Autoregressive modelling

## VAE – first scalable approach for image generation



# DCGAN – first convolutional GAN for image generation



---

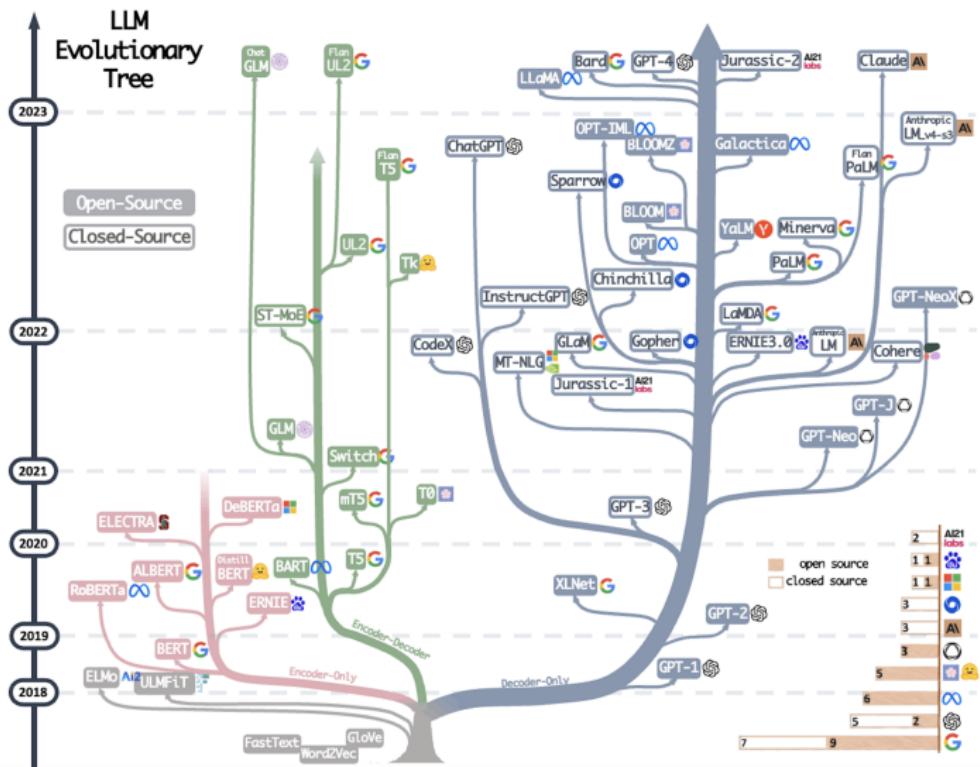
Radford A., Metz L., Chintala S. *Unsupervised representation learning with deep convolutional generative adversarial networks*, 2015

# StyleGAN – high quality generation of faces



Karras T., Laine S., Aila T. A style-based generator architecture for generative adversarial networks, 2018

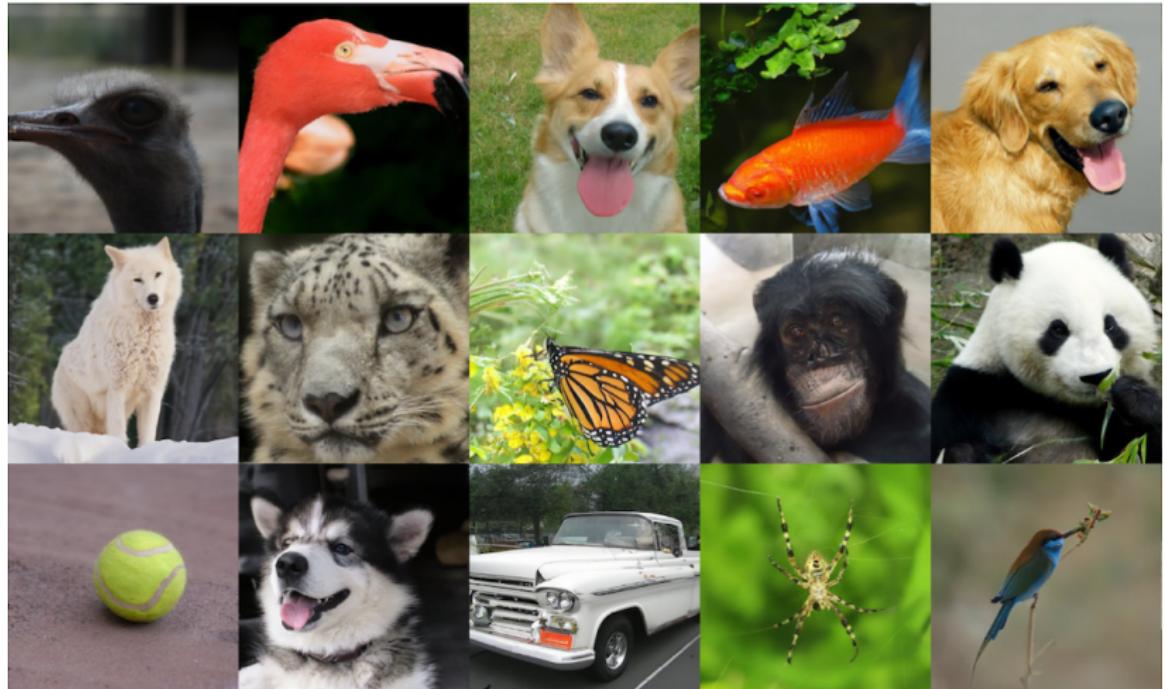
## Language modeling at scale



*image credit:*

<https://blog.biocomm.ai/2023/05/14/open-source-proliferation-l1m-evolutionary-tree/>

# Denoising Diffusion Probabilistic Model



# Midjourney - awesome text-to-image results



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

# Stable Diffusion 3 – flow matching



*image credit: <https://stability.ai/news/stable-diffusion-3>*

# Sora – video generation



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

# Outline

1. Generative models overview
2. Course tricks
3. Problem statement
4. Divergence minimization framework
5. Autoregressive modelling

# Course tricks 1

## Log-derivative trick

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

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

## Jensen's Inequality

Let  $\mathbf{x} \in \mathbb{R}^m$  be a continuous random variable with a density  $p(\mathbf{x})$  and  $f : \mathbb{R}^m \rightarrow \mathbb{R}$  be a convex function. 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 a density  $p(\mathbf{x})$  and let  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^d$  be a vector 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 2

### Change of variable theorem (CoV)

Let  $\mathbf{x}$  be a continuous random variable with a density  $p(\mathbf{x})$  and  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  is a differentiable, **invertible** function. If  $\mathbf{y} = \mathbf{f}(\mathbf{x})$ , then

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

### Proof (1d)

Assume  $f$  is a monotonically increasing function

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

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

## Course tricks 3

### Law of the unconscious statistician (LOTUS)

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

We could treat any deterministic variable  $\mathbf{x}_0$  as a random variable with density  $p(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_0)$ .

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

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

# Outline

1. Generative models overview
2. Course tricks
3. Problem statement
4. Divergence minimization framework
5. Autoregressive modelling

## Problem statement

We are given i.i.d. samples  $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^m$  from **unknown** distribution  $\pi(\mathbf{x})$ .

### Goal

We would like to learn a distribution  $\pi(\mathbf{x})$  for

- ▶ evaluating  $\pi(\mathbf{x})$  for new samples (how likely to get object  $\mathbf{x}$ ) – **density evaluation**;
- ▶ sampling from  $\pi(\mathbf{x})$  (to get new objects  $\mathbf{x} \sim \pi(\mathbf{x})$ ) – **generation**.

### Challenge

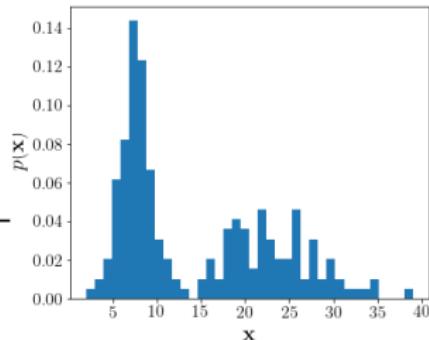
Data is complex and high-dimensional. E.g. the dataset of images lies in the space  $\mathbb{R}^{\text{width} \times \text{height} \times \text{channels}}$ . Curse of dimensionality does not allow us to find the exact density  $\pi(\mathbf{x})$ .

## Histogram as a generative model

Let  $x \sim \text{Categorical}(\pi)$ . The histogram is totally defined by

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

**Problem:** curse of dimensionality (number of bins grows exponentially).



**MNIST example:** 28x28 gray-scaled images, each image is  $\mathbf{x} = (x_1, \dots, x_{784})$ , where  $x_i \in \{0, 1\}$ .

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

Hence, the histogram will have  $2^{28 \times 28} - 1$  parameters to specify  $\pi(\mathbf{x})$ .

**Question:** How many parameters do we need in these cases?

$$\pi(\mathbf{x}) = \pi(x_1) \cdot \pi(x_2) \cdot \dots \cdot \pi(x_m);$$

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

## Problem statement

### Conditional model

In practice the popular task is to create a conditional model  $\pi(x|y)$ .

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

# Outline

1. Generative models overview
2. Course tricks
3. Problem statement
4. Divergence minimization framework
5. Autoregressive modelling

## Divergences

- ▶ Fix probabilistic model  $p(\mathbf{x}|\theta)$  – the set of parameterized distributions.
- ▶ Instead of searching true  $\pi(\mathbf{x})$  over all probability distributions, learn function approximation  $p(\mathbf{x}|\theta) \approx \pi(\mathbf{x})$ .

### What is a divergence?

Let  $\mathcal{P}$  be the set of all possible probability distributions. Then  $D : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}$  is 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 task

$$\min_{\theta} D(\pi||p),$$

where  $\pi(\mathbf{x})$  is a true data distribution,  $p(\mathbf{x}|\theta)$  is a model distribution.

## Forward KL vs Reverse KL

### Forward KL

$$KL(\pi||p) = \int \pi(\mathbf{x}) \log \frac{\pi(\mathbf{x})}{p(\mathbf{x}|\theta)} d\mathbf{x} \rightarrow \min_{\theta}$$

### Reverse KL

$$KL(p||\pi) = \int p(\mathbf{x}|\theta) \log \frac{p(\mathbf{x}|\theta)}{\pi(\mathbf{x})} d\mathbf{x} \rightarrow \min_{\theta}$$

What is the difference between these two formulations?

### Maximum likelihood estimation (MLE)

Let  $\{\mathbf{x}_i\}_{i=1}^n$  be the set of the given i.i.d. samples.

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

## Forward KL vs Reverse KL

### Forward KL

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

Maximum likelihood estimation is equivalent to minimization of the Monte-Carlo estimate of forward KL.

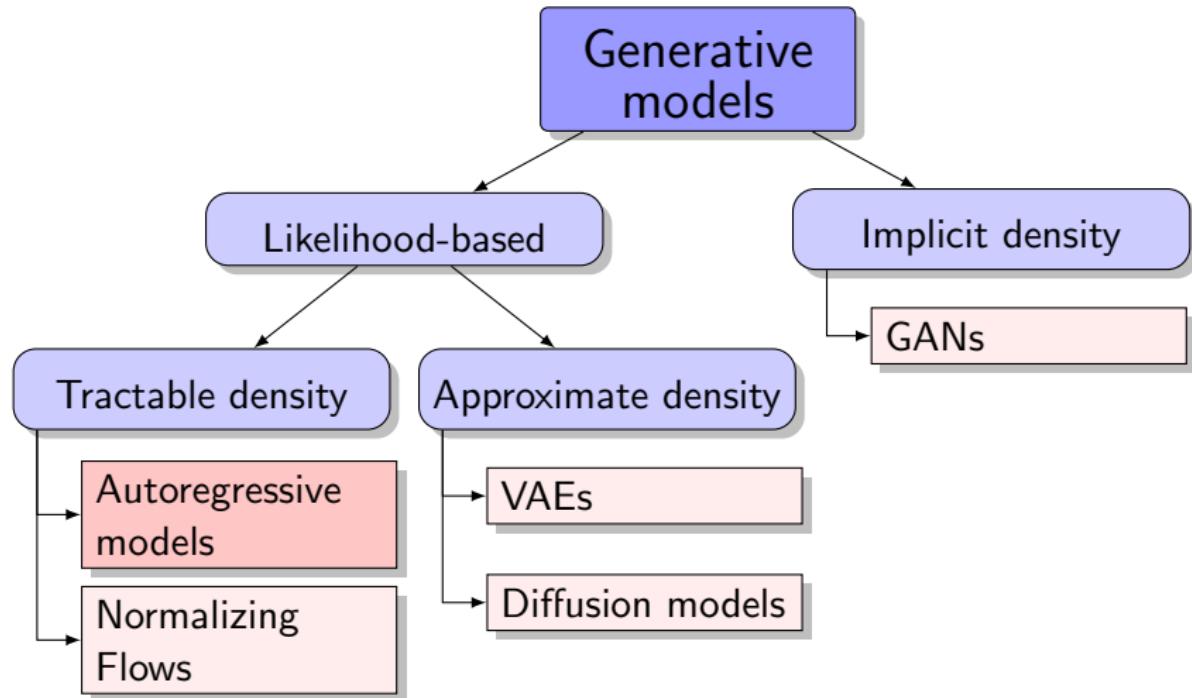
### Reverse KL

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

# Outline

1. Generative models overview
2. Course tricks
3. Problem statement
4. Divergence minimization framework
5. Autoregressive modelling

# Generative models zoo



# Autoregressive modelling

## MLE problem

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

- ▶ We would like to solve the problem using gradient-based optimization.
- ▶ We have to efficiently compute  $\log p(\mathbf{x} | \boldsymbol{\theta})$  and  $\frac{\partial \log p(\mathbf{x} | \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ .

## Likelihood as product of conditionals

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

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

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

## Autoregressive models

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

- ▶ Sampling is sequential:
  - ▶ sample  $\hat{x}_1 \sim p(x_1|\boldsymbol{\theta})$ ;
  - ▶ sample  $\hat{x}_2 \sim p(x_2|\hat{x}_1, \boldsymbol{\theta})$ ;
  - ▶ ...
  - ▶ sample  $\hat{x}_m \sim p(x_m|\hat{\mathbf{x}}_{1:m-1}, \boldsymbol{\theta})$ ;
  - ▶ new generated object is  $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_m)$ .
- ▶ Each conditional  $p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta})$  could be modeled by neural network.
- ▶ Modeling all conditional distributions separately is infeasible. Shared parameters  $\boldsymbol{\theta}$  across conditionals allow to avoid this problem.

## Autoregressive models: MLP

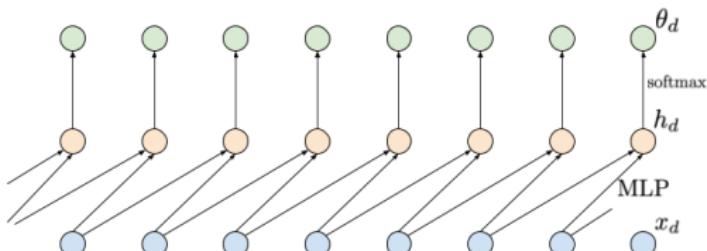
For large  $j$  the conditional distribution  $p(x_j | \mathbf{x}_{1:j-1}, \theta)$  could be infeasible. Moreover, the history  $\mathbf{x}_{1:j-1}$  has non-fixed length.

### Markov assumption

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

### Example

- ▶  $d = 2$ ;
- ▶  $x_j \in \{0, 255\}$ ;
- ▶  $\mathbf{h}_j = \text{MLP}_\theta(x_{j-1}, x_{j-2})$ ;
- ▶  $\pi_j = \text{softmax}(\mathbf{h}_j)$ ;
- ▶  $p(x_j | x_{j-1}, x_{j-2}, \theta) = \text{Categorical}(\pi_j)$       Is it possible to model continuous distributions instead of discrete one?



# Autoregressive models: LLM

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

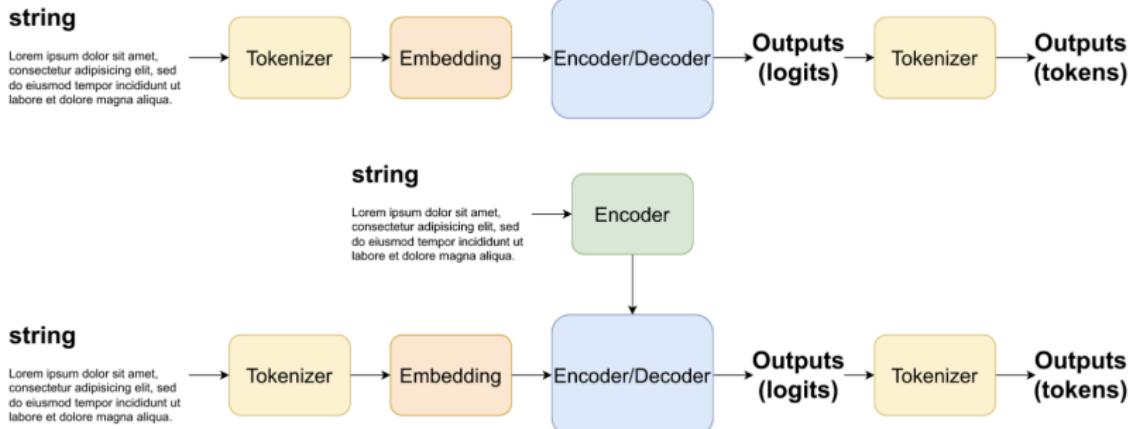


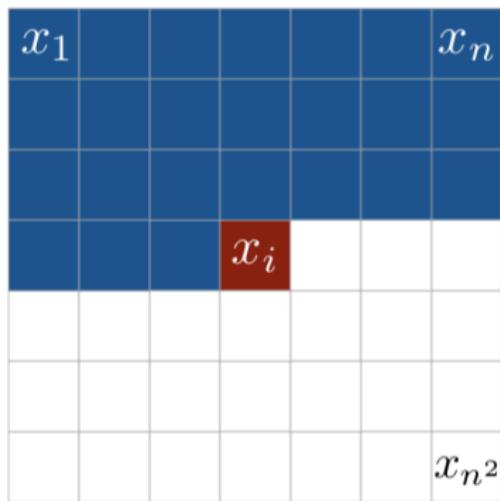
image credit: [https://jmtoyczak.github.io/blog/20/20\\_llms.html](https://jmtoyczak.github.io/blog/20/20_llms.html)

# Autoregressive models for images

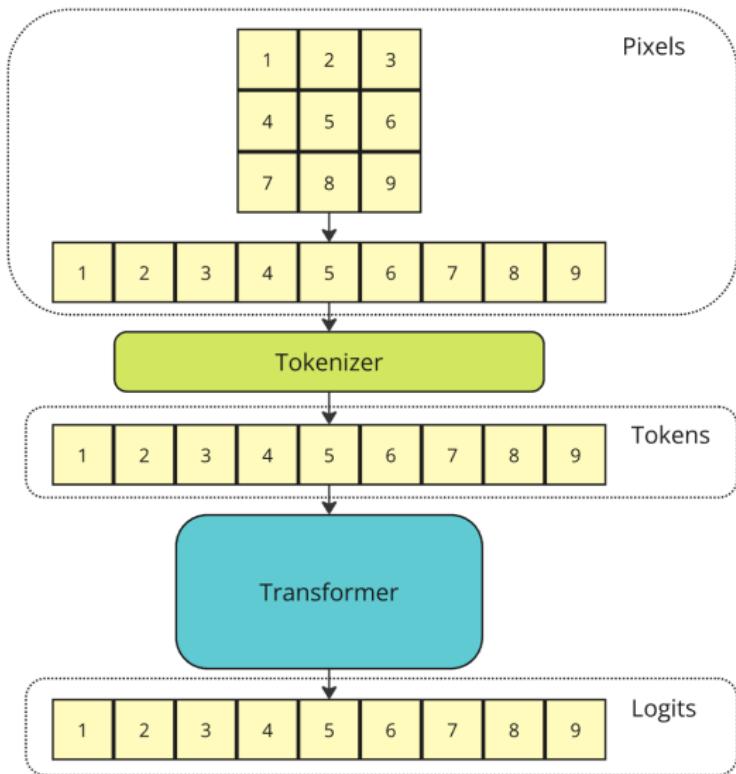
How to model the distribution  $\pi(\mathbf{x})$  of natural images?

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

- ▶ We need to introduce the order of image pixels. Raster ordering is the most straightforward way to do this.
- ▶ The image has RGB channels, these dependencies could be addressed.



# Autoregressive models: ImageGPT



## Summary

- ▶ We are trying to approximate the distribution of samples for density estimation and generation of new samples.
- ▶ To fit model distribution to the real data distribution one could use divergence minimization framework.
- ▶ Minimization of forward KL is equivalent to the MLE problem.
- ▶ Autoregressive models decompose the distribution to the sequence of the conditionals.
- ▶ Sampling from the autoregressive models is trivial, but sequential!
- ▶ To estimate density you need to multiply all conditionals  $p(x_j | \mathbf{x}_{1:j-1}, \theta)$ .
- ▶ ImageGPT uses raster pixel ordering to flatten image and to apply the transformer model to it.