# Deep Generative Models

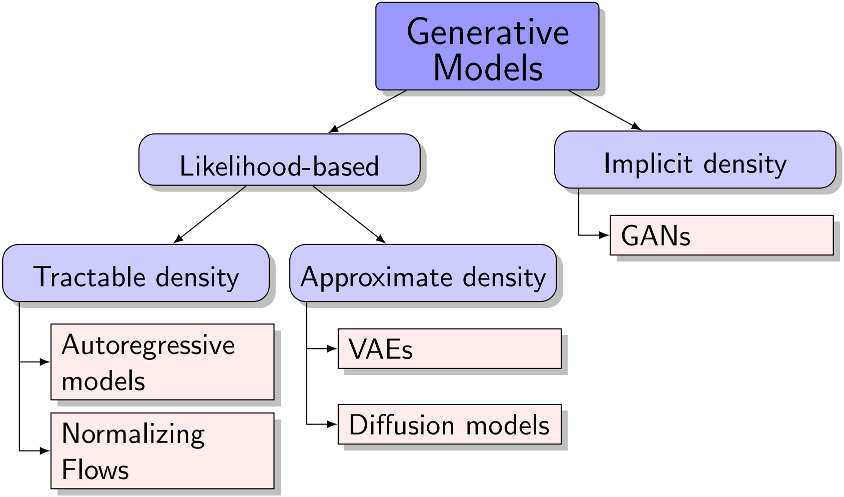
## Lecture 1

Roman Isachenko

Moscow Institute of Physics and Technology
Yandex School of Data Analysis

2025, Autumn

# Generative Models Zoo



# Outline

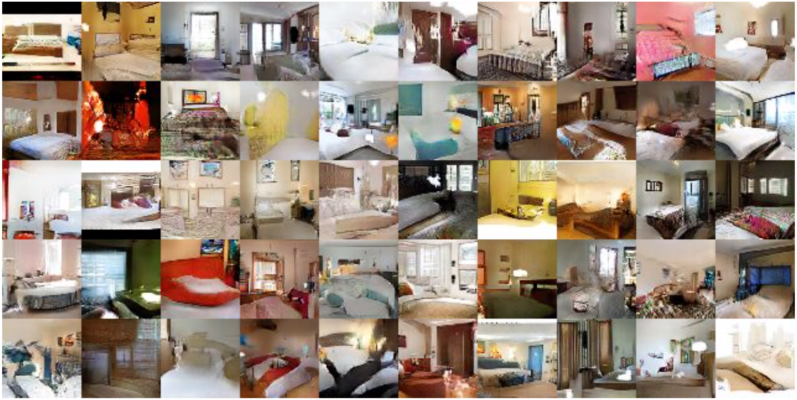
1. Generative Models Overview
2. Course Tricks
3. Problem Statement
4. Divergence Minimization Framework
5. Autoregressive Modeling

# Outline

# VAE – The First Scalable Approach for Image Generation

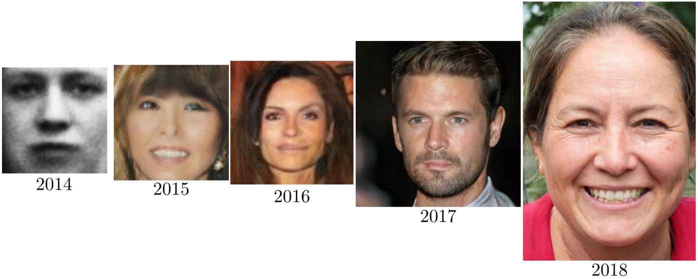
*Kingma D. P., Welling M. Auto-Encoding Variational Bayes, 2013*

# DCGAN – The 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 Face Generation




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

# Language Modeling at Scale

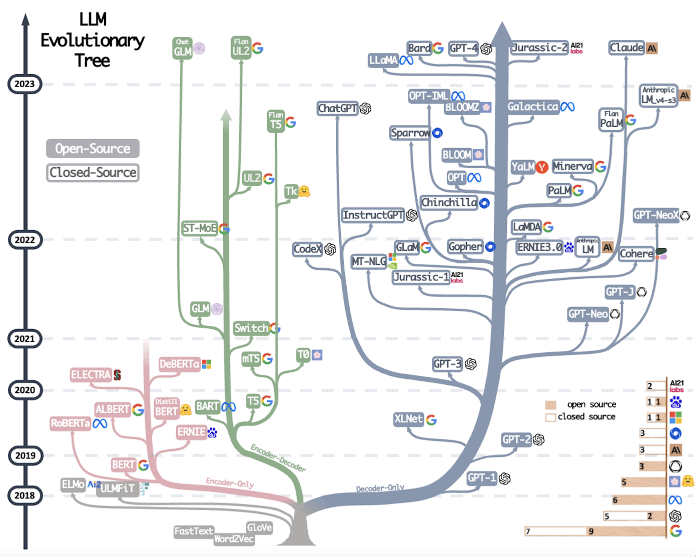



*Image credit:*
*https://blog.biocomm.ai/2023/05/14/open-source-proliferation-llm-evolutionary-tree/*

# Denoising Diffusion Probabilistic Model

*Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021*

# Midjourney – Impressive 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

# Course Tricks 1

## Log-Derivative Trick

Let $f : \mathbb{R}^m \to \mathbb{R}$ 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 density $p(\mathbf{x})$, and $f : \mathbb{R}^m \to \mathbb{R}$ 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 density $p(\mathbf{x})$ and $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^d$ be a 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 2

## Change of Variables Theorem (CoV)

Let $\mathbf{x}$ be a continuous random variable with density $p(\mathbf{x})$, and $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m$ be 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 density $p(\mathbf{x})$ and let $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m$ 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 can treat any deterministic variable $\mathbf{x}_0$ as a random variable with density $p(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_0)$.

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

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

# Outline

# Problem statement

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

## Objective

Our goal is to learn a distribution $\pi(\mathbf{x})$ that enables:

- evaluating $\pi(\mathbf{x})$ for new data (how likely is an object $\mathbf{x}$?) – **density estimation**;
- sampling from $\pi(\mathbf{x})$ (to generate new objects $\mathbf{x} \sim \pi(\mathbf{x})$) – **generation**.

## Challenge

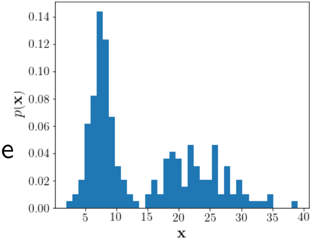
The data is complex and high-dimensional. For example, a dataset of images resides in the space $\mathbb{R}^{\text{width} \times \text{height} \times \text{channels}}$. The curse of dimensionality prevents us from finding the exact density $\pi(\mathbf{x})$.

# Histogram as a Generative Model

Assume $x \sim \mathrm{Categorical}(\boldsymbol{\pi})$. The histogram is completely determined by

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

**The curse of dimensionality:** the number of bins grows exponentially.




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

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

Hence, a full histogram would require $2^{28\times 28} - 1$ 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 Models

## Conditional Model

In practice, a common task is to construct a conditional model $\pi(\mathbf{x}|\mathbf{y})$.

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

# Outline

# Divergences

- Fix a probabilistic model $p(\mathbf{x}|\boldsymbol{\theta})$ – a family of parameterized distributions.
- Instead of searching for the true $\pi(\mathbf{x})$ among all possible probability distributions, we instead learn a function approximation $p(\mathbf{x}|\boldsymbol{\theta}) \approx \pi(\mathbf{x})$.

## What is a Divergence?

Let $\mathcal{P}$ be the set of all probability distributions. A function $D : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$ 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

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

where $\pi(\mathbf{x})$ is the true data distribution and $p(\mathbf{x}|\boldsymbol{\theta})$ is our model distribution.

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

## Forward KL

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

## Reverse KL

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

What is the practical difference between these two formulations?

## Maximum Likelihood Estimation (MLE)

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

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

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

## Forward KL

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

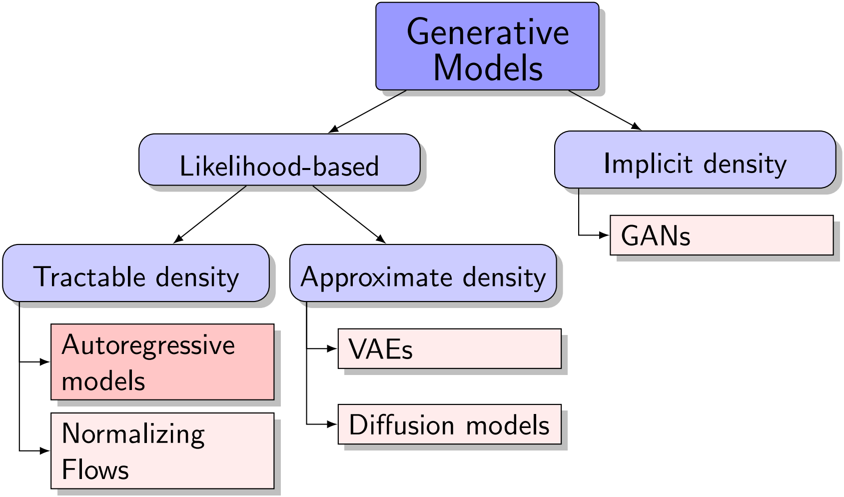
Maximum likelihood estimation is equivalent to minimizing the Monte Carlo estimate of the forward KL divergence.

## Reverse KL

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

# Outline

# Generative Models Zoo



# Autoregressive Modeling

## 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 seek to solve this maximization using gradient-based optimization.
- Thus, we need efficient computation of $\log p(\mathbf{x}|\boldsymbol{\theta})$ and its gradient $\frac{\partial \log p(\mathbf{x}|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$.

## Likelihood as a 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 $\hat{x}_m \sim p(x_m|\hat{\mathbf{x}}_{1:m-1}, \boldsymbol{\theta})$;
  - The generated object is $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_m)$.
- Each conditional $p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta})$ can be modeled by a neural network.
- Modeling all conditionals separately is infeasible. Sharing parameters $\boldsymbol{\theta}$ across all conditionals alleviates this issue.

# Autoregressive Models: MLP

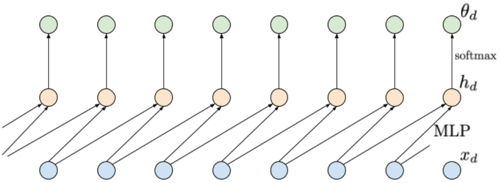
For large $j$, the conditional distribution $p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta})$ can become intractable. Furthermore, the history $\mathbf{x}_{1:j-1}$ has variable length.

## Markov Assumption

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

## Example

- $d = 2$;
- $x_j \in \{0, 255\}$;
- $\mathbf{h}_j = \mathrm{MLP}_{\boldsymbol{\theta}}(x_{j-1}, x_{j-2})$;
- $\boldsymbol{\pi}_j = \mathrm{softmax}(\mathbf{h}_j)$;
- $p(x_j|x_{j-1}, x_{j-2}, \boldsymbol{\theta}) = \mathrm{Categorical}(\boldsymbol{\pi}_j)$.




Is it possible to model continuous distributions instead of discrete ones?

*Image credit: https://jmtomczak.github.io/blog/2/2_ARM.html*

# Autoregressive Models: LLM

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

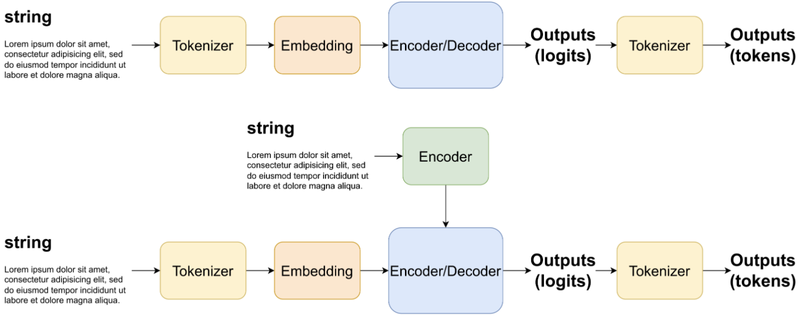


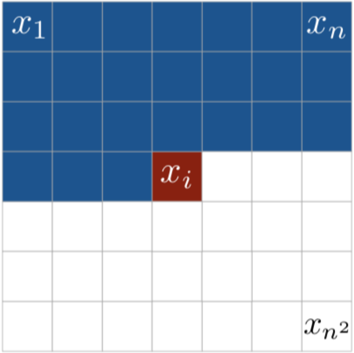

*Image credit: https://jmtomczak.github.io/blog/20/20_llms.html*

# Autoregressive Models for Images

How can we model the distribution $\pi(\mathbf{x})$ over natural images?

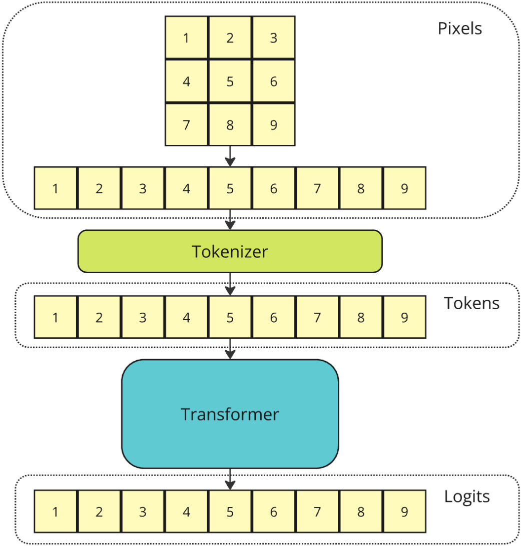
$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{j=1}^{\text{width}\times\text{height}} p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta}).$$

- We must specify an ordering of image pixels. Raster scan order is the most straightforward choice.
- RGB channel dependencies can also be explicitly modeled.




*Oord A., Kalchbrenner N., Kavukcuoglu K. Pixel Recurrent Neural Networks, 2016*

# Autoregressive Models: ImageGPT




*Chen M. et al. Generative Pretraining from Pixels, 2020*

# Summary

- We aim to approximate the data distribution for both density estimation and generation of new samples.
- Divergence minimization provides a general framework to fit a model distribution to the real data distribution.
- Minimizing the forward KL is equivalent to solving the MLE problem.
- Autoregressive models decompose the joint distribution into a sequence of conditionals.
- Sampling from autoregressive models is straightforward, but inherently sequential.
- To evaluate the density, multiply all conditionals $p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta})$.
- ImageGPT applies the transformer to raster-ordered image pixels.