

# Deep Generative Models

## Lecture 1

Roman Isachenko



Ozon Masters

Spring, 2021

# Logistics

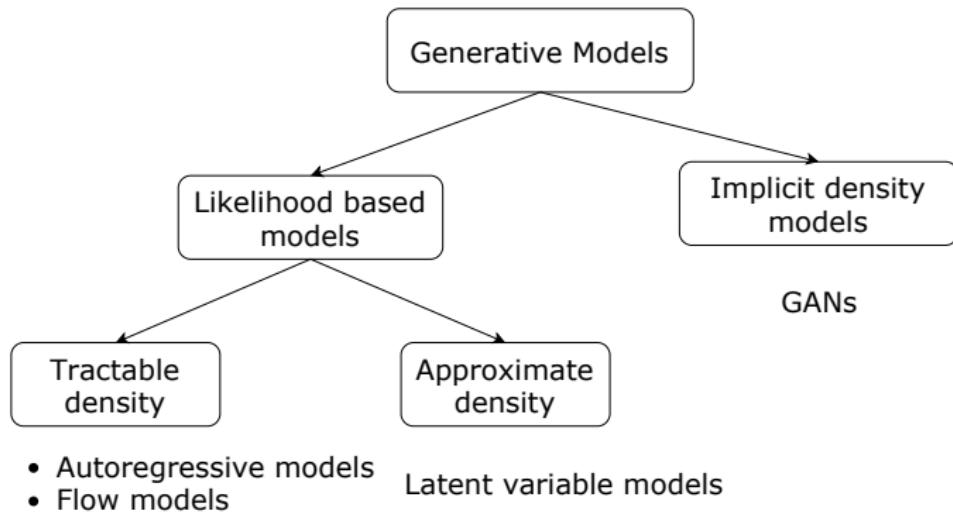
## How to win?

- ▶ homeworks: 78 points
  - ▶ hw1 - practice (13 points): autoregressive models
  - ▶ hw2 - theory (13 points): autoregressive + latent variable models
  - ▶ hw3 - practice (13 points): latent variable models
  - ▶ hw4 - theory (13 points): flow + adversarial models
  - ▶ hw5 - practice (13 points): flow models
  - ▶ hw6 - practice (13 points): adversarial models
- ▶ presentation: 26 points

Total: 104 points.  $\text{Mark} = \text{relu}\left(\frac{\#\text{points}}{8} - 2\right)$ .

- ▶ Repo:  
<https://github.com/r-isachenko/2021-DGM-Ozon-course>
- ▶ (!) **Anonymous feedback:**  
<https://forms.gle/Nzxedo6Vrp28p1717>

# Generative models zoo

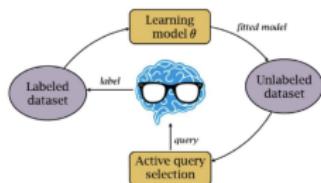


# Applications

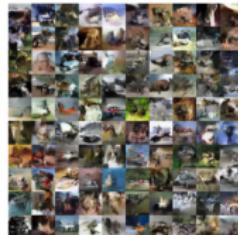
" i want to talk to you . "  
" i want to be with you . "  
" i do n't want to be with you . "  
" i do n't want to be with you . "  
she did n't want to be with him .

he was silent for a long moment .  
he was silent for a moment .  
it was quiet for a moment .  
it was dark and cold .  
there was a pause .  
it was my turn .

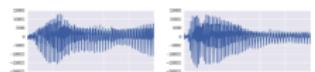
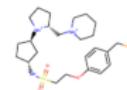
## Text analysis



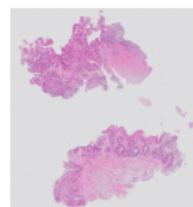
## Active Learning



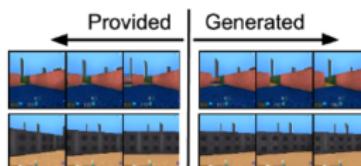
## Image analysis



## Audio analysis



## Medical data



## Reinforcement Learning

and more...

## Applications: Image generation (VAE)



# Applications: Image generation (DCGAN)



---

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

# Applications: SuperResolution (SRGAN)



---

Ledig C. et al. Photo-realistic single image super-resolution using a generative adversarial network, 2016

# Applications: Face generation (StyleGAN)



---

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

## Applications: Face generation (VQ-VAE-2)



---

Razavi A., Oord A., Vinyals O. Generating Diverse High-Fidelity Images with VQ-VAE-2, 2019

# Applications: Language modelling

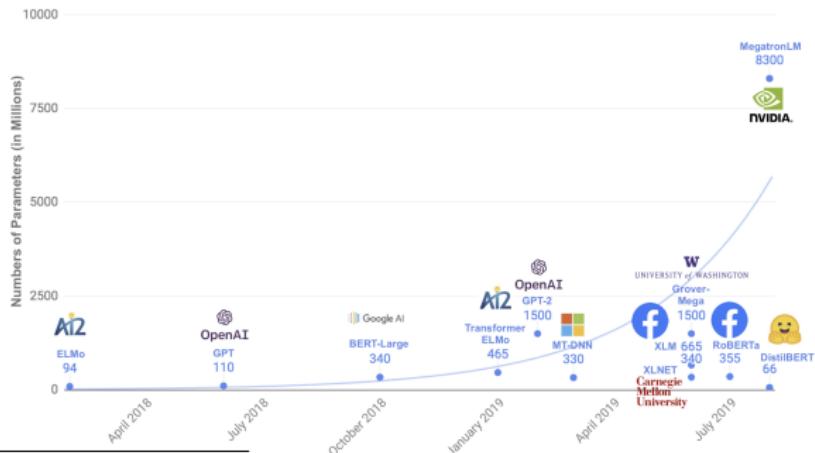
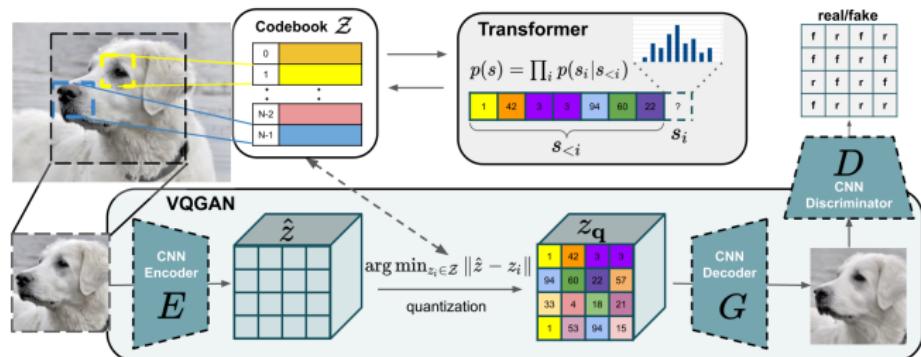


image credit: <http://jalammar.github.io/illustrated-gpt2>

*Sanh V. et al. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, 2019.*

# Applications: Image generation, new era



Esser P., Rombach R., Ommer B. Taming Transformers for High-Resolution Image Synthesis, 2020

# Problem Statement

We are given i.i.d. samples  $\{\mathbf{x}_i\}_{i=1}^n \in X$  (e.g.  $X = \mathbb{R}^m$ ) 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  $\mathbf{x}$ ?);
- ▶ sampling from  $\pi(\mathbf{x})$  (to get new objects  $\mathbf{x} \sim \pi(\mathbf{x})$ ).

## Challenge

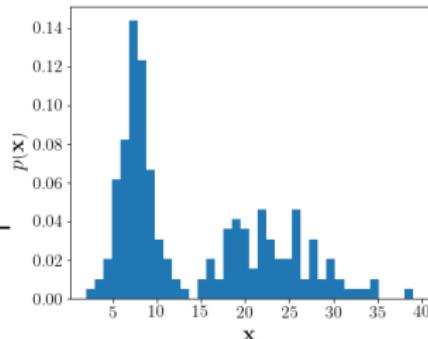
Data is complex and high-dimensional. Imagine the dataset of images which lies in the space  $X \subset \mathbb{R}^{\text{width} \times \text{height}}$ .

## Histogram as a generative model

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

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

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



MNIST: 28x28 gray-scaled images, each image is

$\mathbf{x} = (x_1, \dots, x_{784})$ , where  $x_i = \{0, 1\}$ .

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

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

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

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

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

# Maximum likelihood

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  $p(\mathbf{x}|\theta) \approx \pi(\mathbf{x})$ .

## MLE problem

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

The problem is solved with SGD.

## Requirements

- ▶ efficiently compute  $\log p(\mathbf{x}|\theta)$ ;
- ▶ efficiently compute gradient of  $\log p(\mathbf{x}|\theta)$ .

# Autoregressive model

## MLE problem

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

## Challenge

$p(\mathbf{x}|\theta)$  could be intractable.

## Likelihood as product of conditionals

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

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

**Example:**  $p(x_1, x_2, x_3) = p(x_2) \cdot p(x_1|x_2) \cdot p(x_3|x_1, x_2).$

## Autoregressive models

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

- ▶ Sampling is sequential:
  - ▶ sample  $\hat{x}_0 \sim p(x_1|\boldsymbol{\theta})$ ;
  - ▶ sample  $\hat{x}_1 \sim p(x_2|x_1, \boldsymbol{\theta})$ ;
  - ▶ ...
  - ▶ sample  $\hat{x}_n \sim p(x_n|\mathbf{x}_{1:n-1}, \boldsymbol{\theta})$ ;
  - ▶ new generated object is  $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n)$ .
- ▶ Each conditional  $p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta})$  could be modelled by neural network.
- ▶ Modelling all conditional distributions separately is infeasible and we would obtain separate models. To extend to high dimensions we could share parameters  $\boldsymbol{\theta}$  across conditionals.

# Autoregressive models

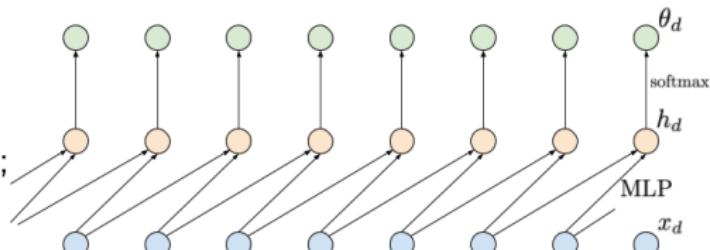
For large  $i$  the conditional distribution  $p(x_i | \mathbf{x}_{1:i-1}, \theta)$  could be infeasible. Moreover, the history  $\mathbf{x}_{1:i-1}$  has non-fixed length.

## Markov assumption

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

## Example

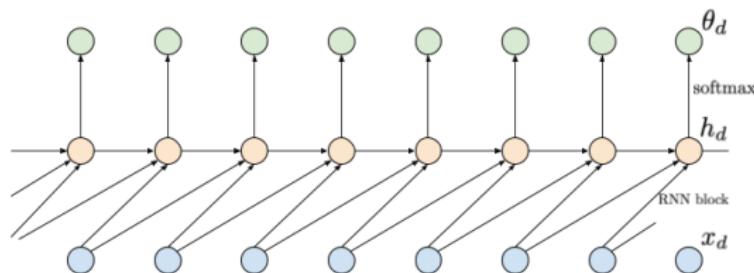
- ▶  $d = 2$ ;
- ▶  $x_i \in \{0, 255\}$ ;
- ▶  $\mathbf{h}_i = \text{MLP}_\theta(x_{i-1}, x_{i-2})$ ;
- ▶  $\mathbf{p}_i = \text{softmax}(\mathbf{h}_i)$ ;
- ▶  $p(x_i | x_{i-1}, x_{i-2}, \theta) = \text{Categorical}(\mathbf{p}_i)$ .



## Autoregressive models

- ▶ Previous model has **limited** memory  $d$ . It is insufficient for many modalities (e.g. for images and text).
- ▶ Recurrent NN fixes this problem and potentially could learn long-range dependencies:

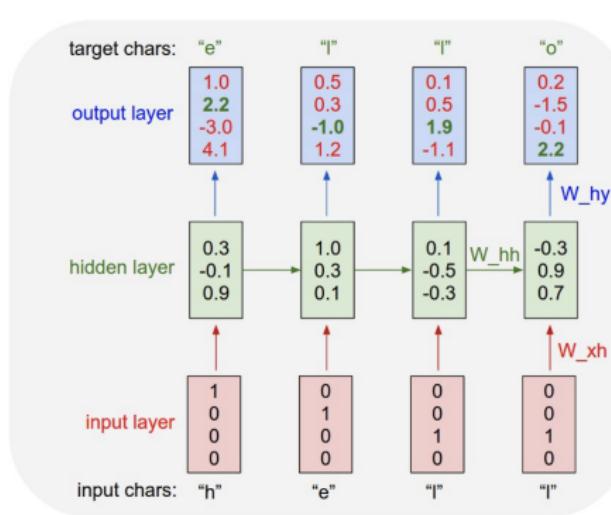
$$p(x_i | \mathbf{x}_{1:i-1}, \theta) = p(x_i | \mathbf{h}_i, \theta), \quad \mathbf{h}_i = \text{RNN}(\mathbf{x}_{i-1}, \mathbf{h}_{i-1})$$



- ▶ Sequential computation of all conditionals  $p(x_i | \mathbf{x}_{1:i-1}, \theta)$ , hence, the training is slow.
- ▶ RNN suffers from vanishing and exploding gradients.

# Char RNN

Model tries to predict the next token (single letter) from previous context.



#### PANDARUS:

Alas, I think he shall be come approached and the day  
When little strain would be attain'd into being never fed,  
And who is but a chain and subjects of his death,  
I should not sleep.

#### Second Senator:

They are away this miseries, produced upon my soul,  
Breaking and strongly should be buried, when I perish  
The earth and thoughts of many states.

#### DUKE VINCENTIO:

Well, your wit is in the care of side and that.

#### Second Lord:

They would be ruled after this chamber, and  
my fair nues begun out of the fact, to be conveyed,  
Whose noble souls I'll have the heart of the wars.

#### Clown:

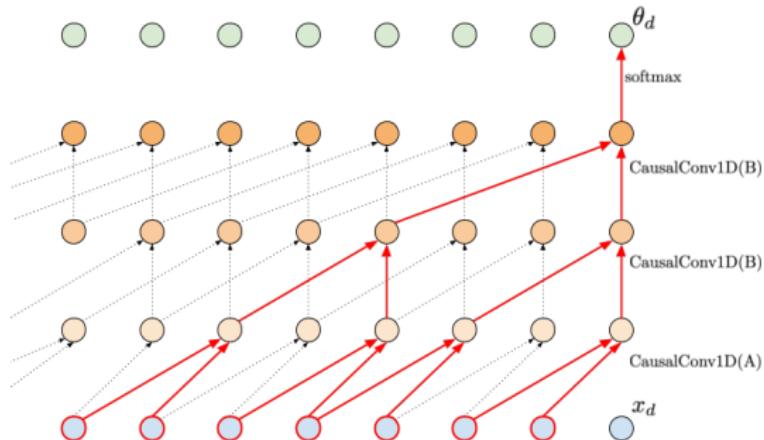
Come, sir, I will make did behold your worship.

#### VIOLA:

I'll drink it.

## Autoregressive models

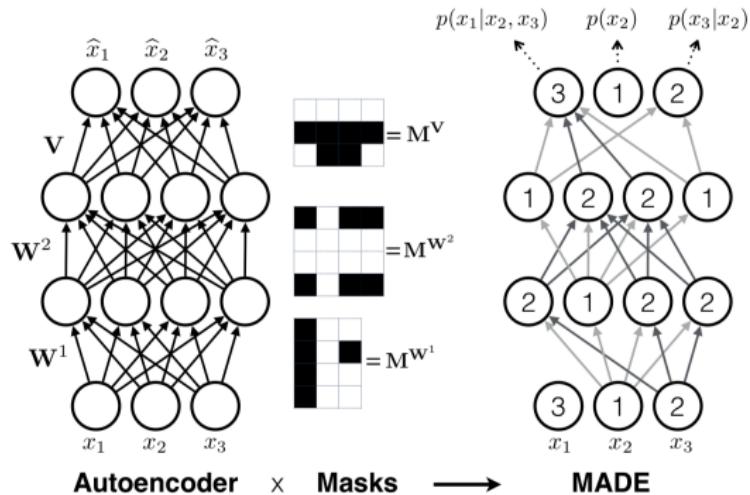
- ▶ Convolutions could be used for autoregressive models, but they have to be **causal**.
- ▶ Try to find and understand the difference between Conv A/B.



- ▶ Could learn long-range dependencies.
- ▶ Do not suffer from gradient issues.
- ▶ Easy to estimate probability for given input, but hard generation of new samples (the sequential process).

# MADE

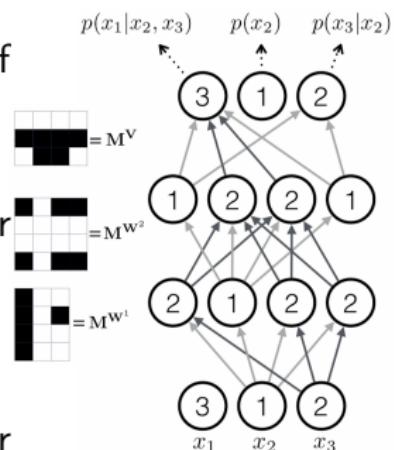
- ▶ Vanila autoencoder is not a generative model. Why?
- ▶ Let mask the weight matrices to make the model generative:  
 $\mathbf{W}_M = \mathbf{W} \cdot \mathbf{M}$ .



- ▶ The question is how to create matrices  $\mathbf{M}$  which produce the autoregressive property?

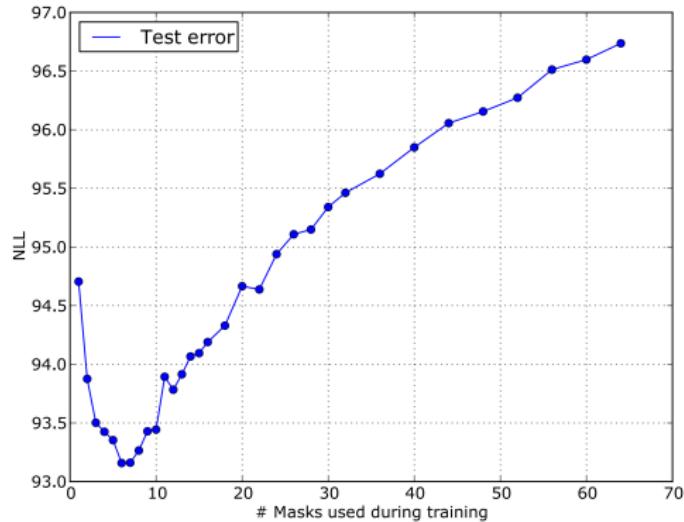
## Masks generation

- ▶ Define the ordering of input elements from 1 to  $n$ .
- ▶ Assign the random number  $m$  from 1 to  $n - 1$  to each hidden unit. The number gives the maximum number of input units to which the unit can be connected.
- ▶ Connect each hidden unit with number  $m$  with the previous layer units which has the number is **less or equal** than  $m$ .
- ▶ Connect each output unit with number  $m$  with the previous layer units which has the number is **less** than  $m$ .



## Possible variations

- ▶ Order agnostic training (missing values in partially observed input vectors can be imputed efficiently);
- ▶ Connectivity-agnostic training (cheap ensembling).



# WaveNet

## Goal

Efficient generation of raw audio waveforms with natural sounds.



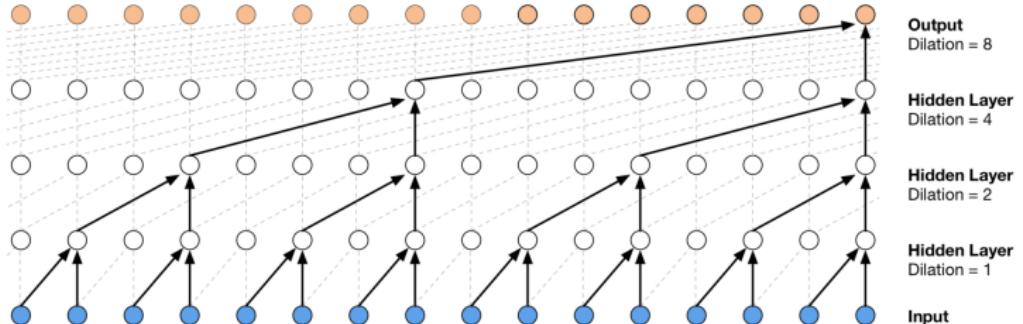
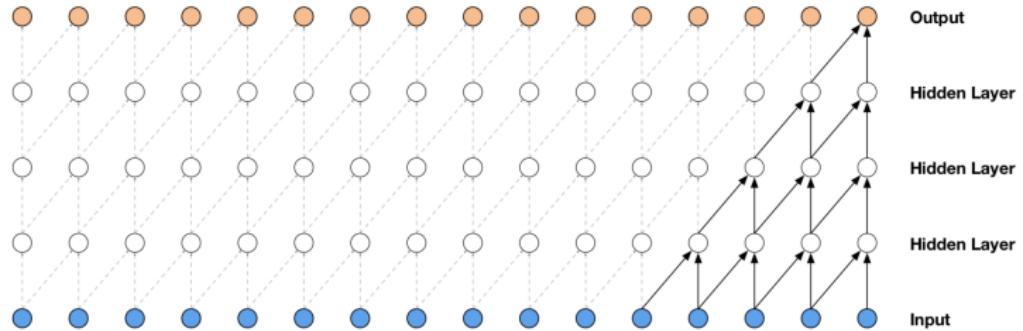
## Solution

Autoregressive model

$$p(\mathbf{x}|\theta) = \prod_{t=1}^T p(x_t|\mathbf{x}_{1:t-1}, \theta).$$

- ▶ Each conditional  $p(x_t|\mathbf{x}_{1:t-1}, \theta)$  models the distribution for the timestamp  $t$ .
- ▶ The model uses **causal** dilated convolutions.

# WaveNet



# PixelCNN

## Goal

Model a distribution of natural images.

## Solution

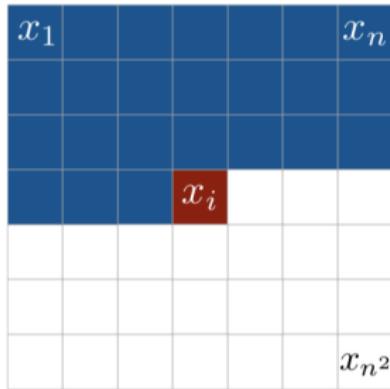
Autoregressive model on 2D pixels

$$p(\mathbf{x}|\theta) = \prod_{i=1}^{n^2} p(x_i|\mathbf{x}_{1:i-1}, \theta).$$

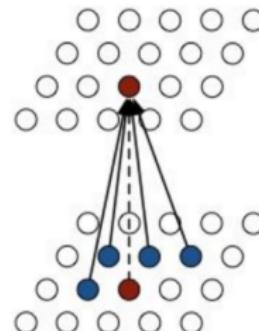
- ▶ We need to introduce the ordering of image pixels.
- ▶ The convolution should be **masked** to make them causal.
- ▶ The image has RGB channels, these dependencies could be addressed.

# PixelCNN

Raster ordering

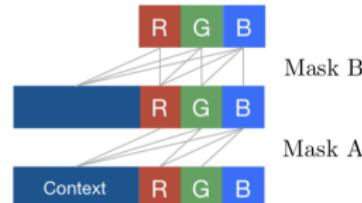
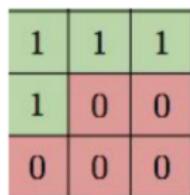


Dependencies between pixels



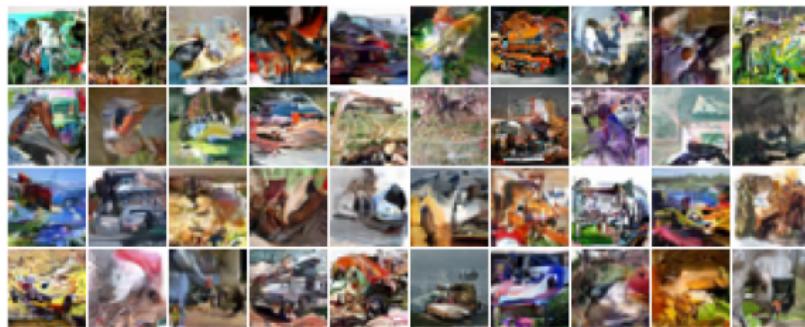
PixelCNN

Masked convolution kernel



# PixelCNN

## CIFAR-10 generated samples



## CIFAR-10 performance

Model	NLL Test (Train)
Uniform Distribution:	8.00
Multivariate Gaussian:	4.70
NICE [1]:	4.48
Deep Diffusion [2]:	4.20
Deep GMMs [3]:	4.00
RIDE [4]:	3.47
PixelCNN:	3.14 (3.08)
Row LSTM:	3.07 (3.00)
Diagonal BiLSTM:	<b>3.00</b> (2.93)

## Summary

- ▶ We are trying to approximate the distribution of samples for density estimation and generation of new samples.
- ▶ Autoregressive models decompose the distribution to the sequence of conditionals.
- ▶ Sampling from autoregressive models is trivial, but sequential (this is a main drawback)
  - ▶ sample  $x_0 \sim p(x_0)$ ;
  - ▶ sample  $x_1 \sim p(x_1|x_0)$ ;
  - ▶ ...
- ▶ Density estimation:

$$p(\mathbf{x}) = \prod_{i=1}^m p(x_i|\mathbf{x}_{1:i-1}).$$

- ▶ Autoregressive models work on both continuous and discrete data.
- ▶ There is no natural way to do unsupervised learning (get latent representations) for autoregressive models.