# Deep Generative Models

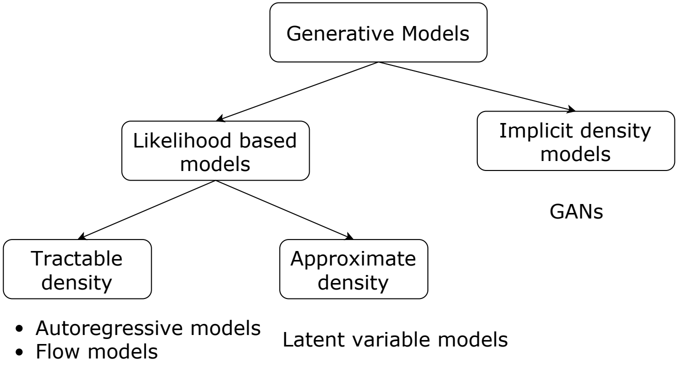
## Lecture 1


Roman Isachenko

Moscow Institute of Physics and Technology


Autumn, 2021

# Generative models zoo



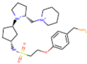
# Applications

" 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 .

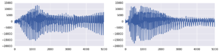
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

**Text analysis**

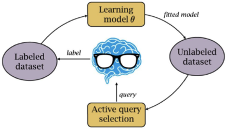
**Image analysis**



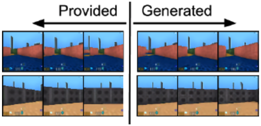
**Graph analysis**



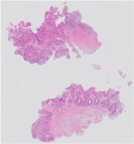
**Audio analysis**




**Active Learning**




**Reinforcement Learning**



**Medical data**

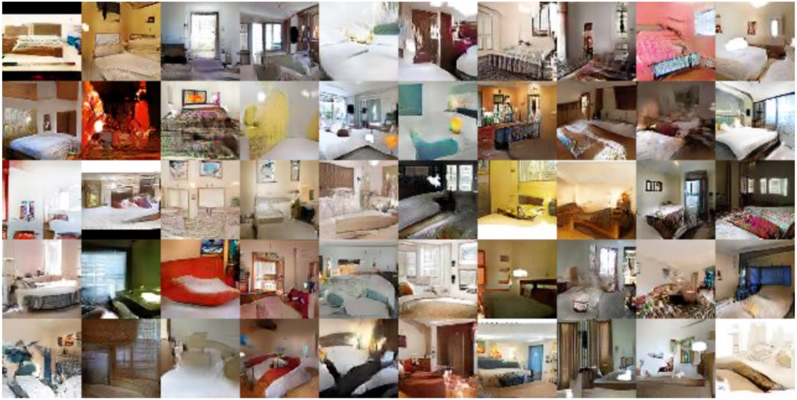
**and more...**

image credit: *https://jmtomczak.github.io/blog/1/1_introduction.html*

# Applications: Image generation (VAE)

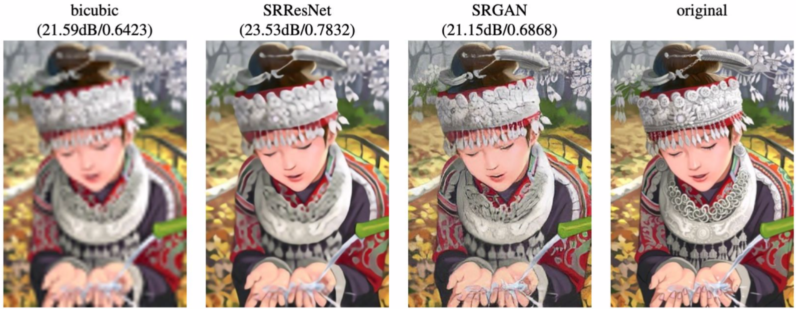
*Kingma D. P., Welling M. Auto-encoding variational bayes, 2013*

# Applications: Image generation (DCGAN)



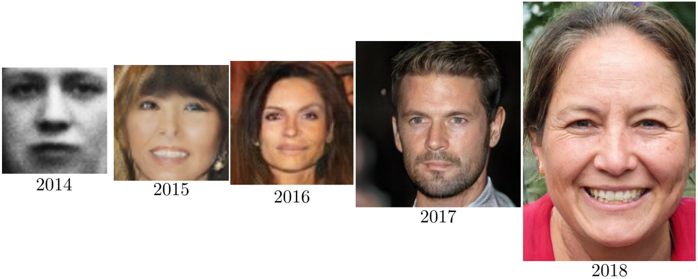
*Radford A., Metz L., Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015*

# Applications: SuperResolution (SRGAN)




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

# Applications: Face generation (StyleGAN)

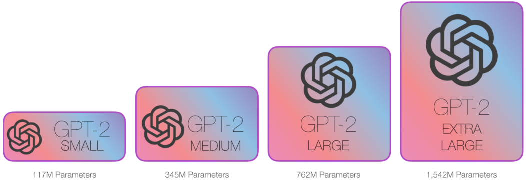



*Karras T., Laine S., Aila T. A style-based generator architecture for generative adversarial networks, 2018*

# 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



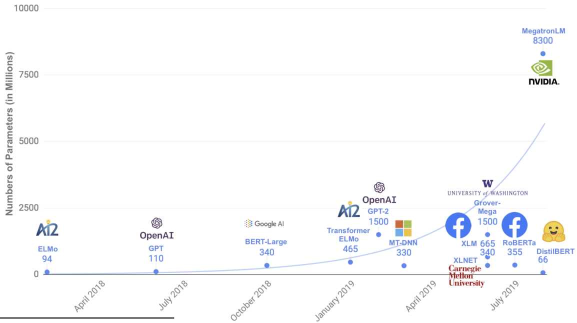
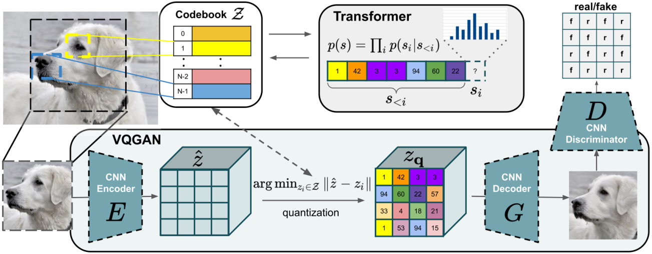




*image credit: http://jalammar.github.io/illustrated-gpt2*
*Sanh V. et al. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, 2019.*

# 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 $\mathbf{x} \sim \pi(\mathbf{x})$).

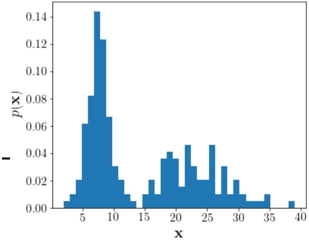
## Challenge

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

# Histogram as a generative model

Let $x \sim$ 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 $p(\mathbf{x}|\boldsymbol{\theta})$ – the set of parameterized distributions .
Instead of searching true $\pi(\mathbf{x})$ over all probability distributions, learn function approximation $p(\mathbf{x}|\boldsymbol{\theta}) \approx \pi(\mathbf{x})$.

## MLE problem

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

The problem is solved with SGD.

## Requirements

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

# Autoregressive model

## MLE problem

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

## Challenge

$p(\mathbf{x}|\boldsymbol{\theta})$ 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

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

**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 $\hat{x}_1 \sim p(x_1|\boldsymbol{\theta})$;
  - sample $\hat{x}_2 \sim p(x_2|\hat{x}_1, \boldsymbol{\theta})$;
  - ...
  - sample $\hat{x}_m \sim p(x_n|\hat{\mathbf{x}}_{1:m-1}, \boldsymbol{\theta})$;
  - new generated object is $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_m)$.
- Each conditional $p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta})$ 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 $p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta})$ could be infeasible. Moreover, the history $\mathbf{x}_{1:i-1}$ has non-fixed length.

## Markov assumption

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

## Example

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

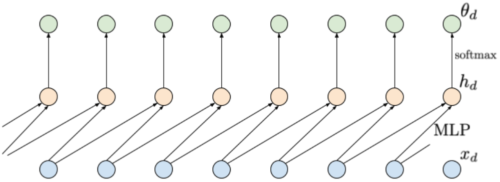
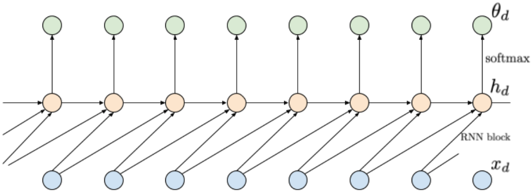



*image credit: https://jmtomczak.github.io/blog/2/2_ARM.html*

# 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:

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

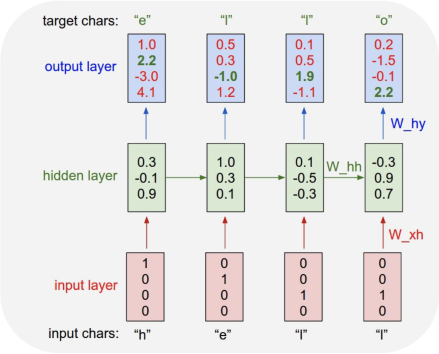



- Sequential computation of all conditionals $p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta})$, hence, the training is slow.
- RNN suffers from vanishing and exploding gradients.

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

# Char RNN

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

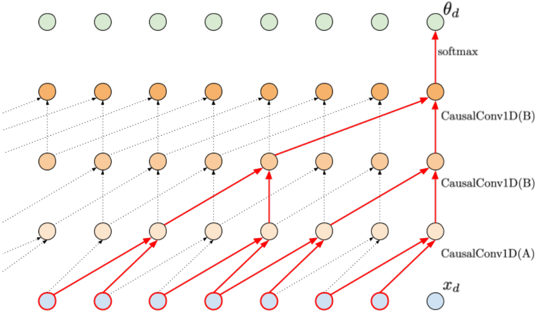



```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

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.
```

*image credit: http://karpathy.github.io/2015/05/21/rnn-effectiveness*

# 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.



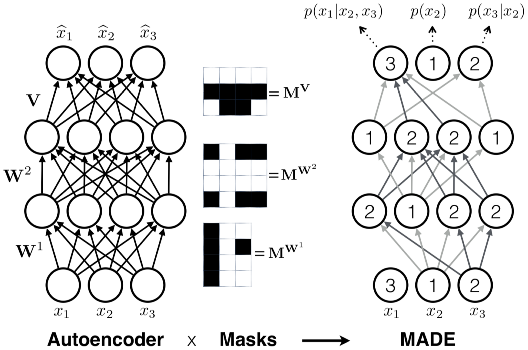

- Could learn long-range dependecies.
- Do not suffer from gradient issues.
- Easy to estimate probability for given input, but hard generation of new samples (the sequential process).

---

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

# 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 **M** which produce the autoregressive property?

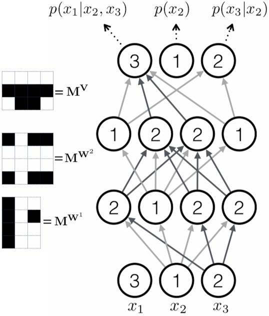
---

*Germain M. et al. Made: Masked autoencoder for distribution estimation, 2015*

# MADE

## Masks generation

- Define the ordering of input elements from 1 to $m$.
- Assign the random number $k$ from 1 to $m-1$ 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 $k$ with the previous layer units which has the number is **less or equal** than $k$.
- Connect each output unit with number $k$ with the previous layer units which has the number is **less** than $k$.

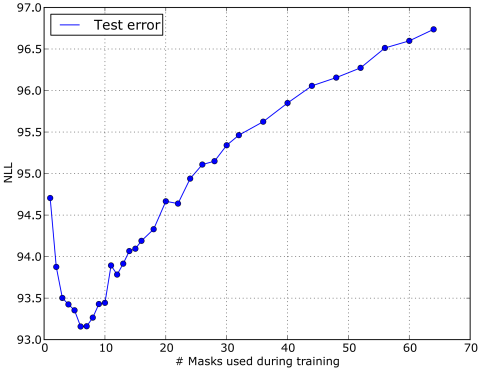


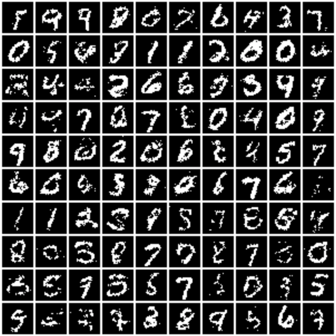

*Germain M. et al. Made: Masked autoencoder for distribution estimation, 2015*

# MADE

## Possible variations

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



*Germain M. et al. Made: Masked autoencoder for distribution estimation, 2015*

# WaveNet

## Goal

Efficient generation of raw audio waveforms with natural sounds.

## Solution

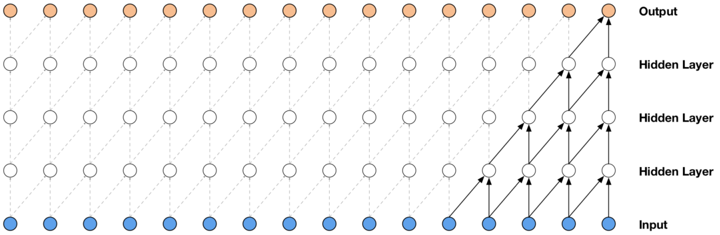
Autoregressive model

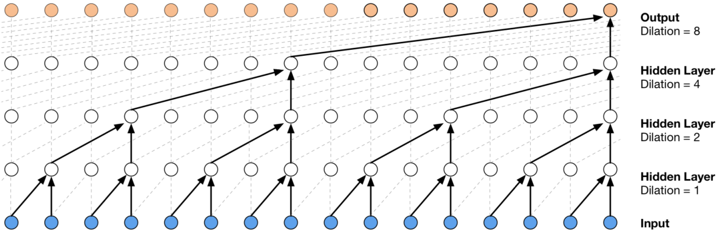
$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{t=1}^{T} p(x_t|\mathbf{x}_{1:t-1}, \boldsymbol{\theta}).$$

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

*Oord A. et al. Wavenet: A generative model for raw audio, 2016*

# WaveNet




*Oord A. et al. Wavenet: A generative model for raw audio, 2016*

# PixelCNN

## Goal

Model a distribution of natural images.

## Solution

Autoregressive model on 2D pixels

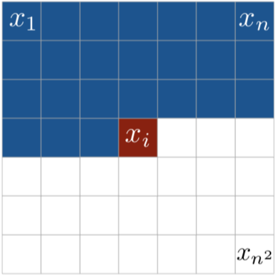
$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{i=1}^{\text{width} \times \text{height}} p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta}).$$

- 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.

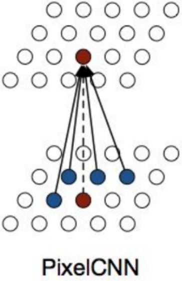
*Oord A., Kalchbrenner N., Kavukcuoglu K. Pixel recurrent neural networks, 2016*

# PixelCNN

## Raster ordering

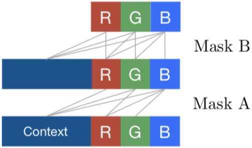



## Dependencies between pixels




## Masked convolution kernel

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

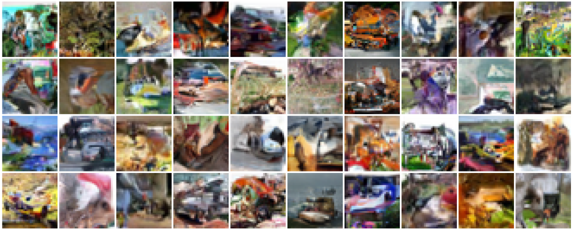



---

*Oord A., Kalchbrenner N., Kavukcuoglu K. Pixel recurrent neural networks, 2016*

# PixelCNN

## CIFAR-10 generated samples



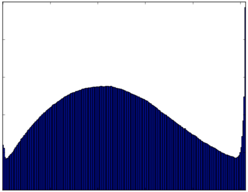
## CIFAR-10 perfomance

| 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: | **3.00** (2.93) |

*Oord A., Kalchbrenner N., Kavukcuoglu K. Pixel recurrent neural networks, 2016*

# PixelCNN++

## CIFAR-10 pixel values distribution



- Standard PixelCNN outputs softmax probabilities for values $\{0, 255\}$ (256 outputs feature maps).
- Categorical distribution do not know anything about numerical relationships (220 is close to 221 and far from 15).
- If pixel value is not presented in the training dataset , it won't be predicted.
- (Look at the edges of the distributions: they have higher probability mass).

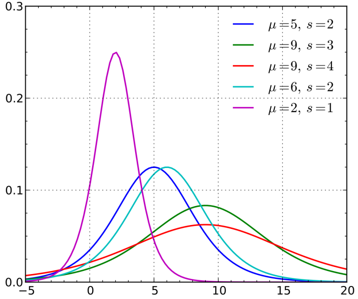
*Salimans T. et al. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications, 2017*

# PixelCNN++

## Mixture of logistic distributions

$$p(\nu|\mu, s) = \frac{\exp^{-(\nu-\mu)/s}}{s(1+\exp^{-(\nu-\mu)/s})^2};$$

$$p(\nu|\boldsymbol{\mu}, \mathbf{s}, \boldsymbol{\pi}) = \sum_{i=1}^{K} \pi_k p(\nu|\mu_k, s_k);$$




To adopt probability calculation to discrete values:

$$P(x|\boldsymbol{\mu}, \mathbf{s}, \boldsymbol{\pi}) = P(x+0.5|\boldsymbol{\mu}, \mathbf{s}, \boldsymbol{\pi}) - P(x-0.5|\boldsymbol{\mu}, \mathbf{s}, \boldsymbol{\pi})$$

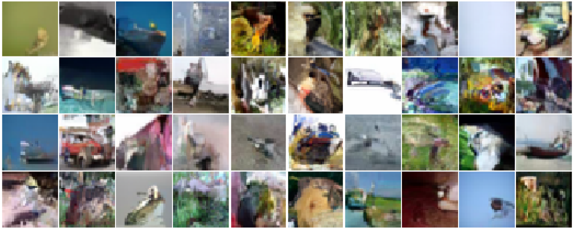
For the edge case of 0, replace $x - 0.5$ by $-\infty$, and for 255 replace $x + 0.5$ by $+\infty$.

---

*Salimans T. et al. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications, 2017*

# PixelCNN++

## CIFAR-10 generated samples



## CIFAR-10 perfomance

| Model | Bits per sub-pixel |
|---|---|
| Deep Diffusion (Sohl-Dickstein et al., 2015) | 5.40 |
| NICE (Dinh et al., 2014) | 4.48 |
| DRAW (Gregor et al., 2015) | 4.13 |
| Deep GMMs (van den Oord & Dambre, 2015) | 4.00 |
| Conv DRAW (Gregor et al., 2016) | 3.58 |
| Real NVP (Dinh et al., 2016) | 3.49 |
| PixelCNN (van den Oord et al., 2016b) | 3.14 |
| VAE with IAF (Kingma et al., 2016) | 3.11 |
| Gated PixelCNN (van den Oord et al., 2016c) | 3.03 |
| PixelRNN (van den Oord et al., 2016b) | 3.00 |
| **PixelCNN++** | **2.92** |

*Salimans T. et al. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications, 2017*

# 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.