NỀN TẢNG AI TẠO SINH

(IT5410 – Foundation of Generative AI)

MÔ HÌNH TỰ HỒI QUY (Autoregressive Models)

Thân Quang Khoát

Trường CNTT&TT, ĐHBKHN

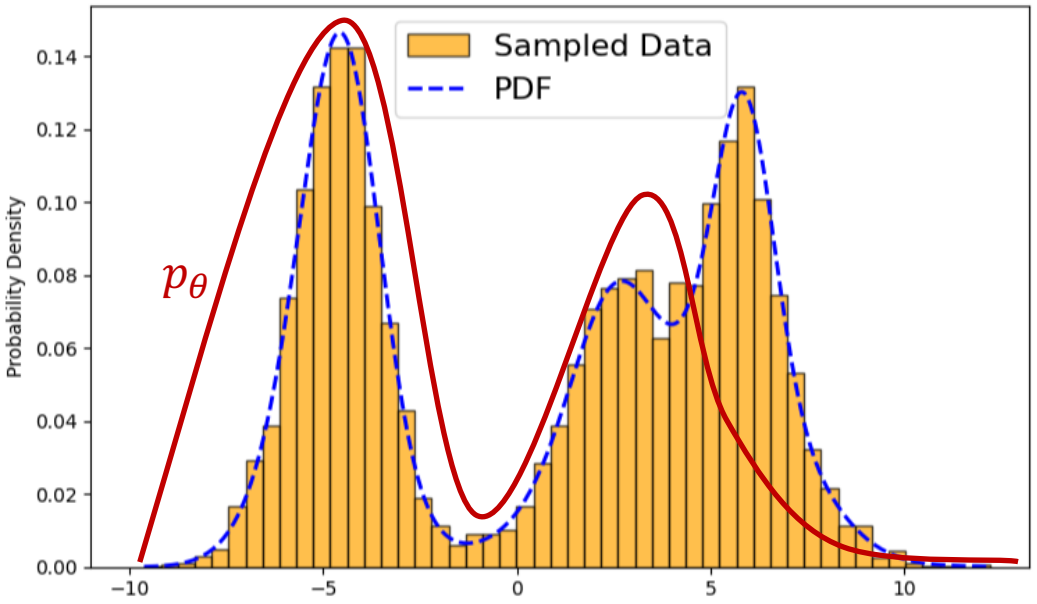
2025

Nội dung

- Mở đầu
- Một số vấn đề của Học sâu
- Một số kiến trúc mạng nơron
- **Mô hình sinh sâu**
- Đánh giá chất lượng
- Học tăng cường

Learning a generative model

- Given a training set of examples, e.g., images of dogs
- We want to learn a probability distribution P(**x**) over image **x** such that
  - **Generation:** If we sample $\boldsymbol{x}_{new} \sim P(\boldsymbol{x})$, $\boldsymbol{x}_{new}$ should look like a dog (*sampling*)
  - **Density estimation:** P(**x**)
  - **Unsupervised representation learning:** We should be able to learn what these images have in common, e.g., ears, tail, …
- In practice
  - We often find a $P_\theta(\boldsymbol{x})$ to **approximate** $P(\boldsymbol{x})$
- How to choose a good model family?
  - $P_\theta(\boldsymbol{x})$



Bayesian networks vs neural models

- By chain rule: $p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3)$
- **Bayesian networks:** approximate

$$p(x_1, x_2, x_3, x_4) \approx p(x_1)p(x_2|x_1)p(x_3|\cancel{x_1,} x_2)p(x_4|\cancel{x_1, x_2,} x_3)$$

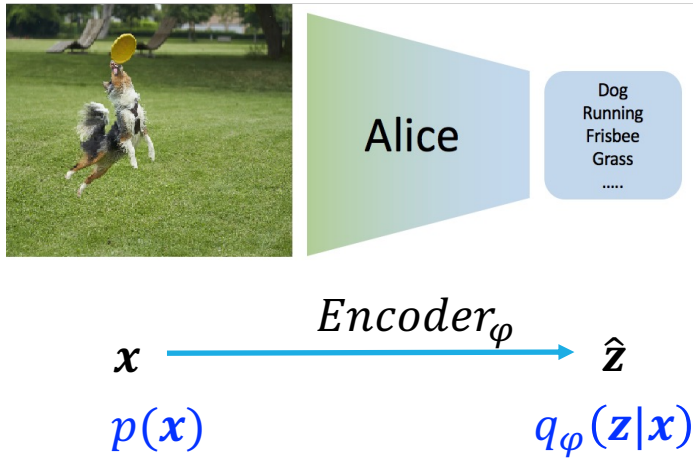
  - We estimate $p(x_1), p(x_2|x_1), p(x_3|\cancel{x_1,} x_2)p(x_4|\cancel{x_1, x_2,} x_3)$
    (thường dùng một bảng để lưu các giá trị xác suất có điều kiện)
  - Often consider **discrete data**
  - Assume the conditional independence (thường giả thuyết điều kiện độc lập có điều kiện)
- **Neural models:**

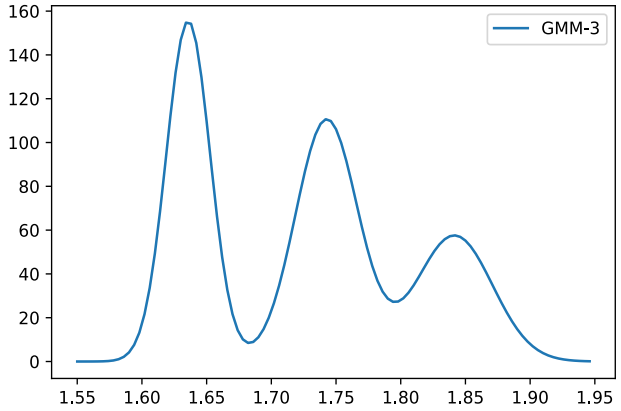
$$p(x_1, x_2, x_3, x_4) \approx p(x_1)p_{\text{NN}}(x_2|x_1)p_{\text{NN}}(x_3|x_1, x_2)\, p_{\text{NN}}(x_4|x_1, x_2, x_3)$$

  - Do not assume conditional independence (Không có giả thuyết độc lập)
  - But require specific functional forms of the conditionals
    (cần biết dạng cụ thể về phân bố xác suất có điều kiện)

Neural models: example

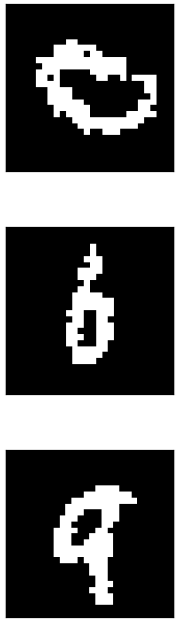
- Specific functional forms:
  - Assume $P_{\mathrm{NN}}(x_4|x_1, x_2, x_3)$ to be a Gaussian distribution with mean μ and variance σ
    (giả sử $P_{\mathrm{NN}}(x_4|x_1, x_2, x_3)$ là phân bố chuẩn với kỳ vọng μ và phương sai σ)
  - μ and σ are the output of a neural network NN(x_1, x_2, x_3; φ), with weights φ, when receiving an input (x_1, x_2, x_3), i.e., (μ, σ) = NN(x_1, x_2, x_3; φ)
    (μ và σ là đầu ra từ một mạng nơron khi nhận đầu vào (x_1, x_2, x_3))
- Note that $p_{\mathrm{NN}}(x_4|x_1, x_2, x_3)$ can be simple, but $p(x_1, \mathrm{x}_2, \mathrm{x}_3, x_4)$ may be complex





GMM with 3 components

Autoregressive Models (mô hình tự hồi quy)



- We can write

$$p(x_1, \dots, x_n) = p(x_1)p(x_2|x_1) \cdots p(x_n|x_1, \dots, x_{n-1})$$

  - You can choose your own order of the variables
- We can choose an **Autoregressive model**

$$p_\theta(x_1, \dots, x_n) = p_\theta(x_1; \theta_1)p_\theta(x_2|x_1; \theta_2) \cdots p_\theta(x_n|x_1, \dots, x_{n-1}; \theta_n)$$

- E.g., consider a set of MNIST binarized images
  - Each pixel has a value of 0 or 1
  - Each variable x_i represents one pixel
- The log-likelihood:

$$\log p_\theta(x_1, \dots, x_n) = \log p_\theta(x_1; \theta_1) + \sum_{i=2}^{n-1} \log p_\theta(x_i|x_1, \dots, x_{i-1}; \theta_i)$$

1. **Easy to evaluate likelihood**
2. **MLE is good**
3. **Expressive**

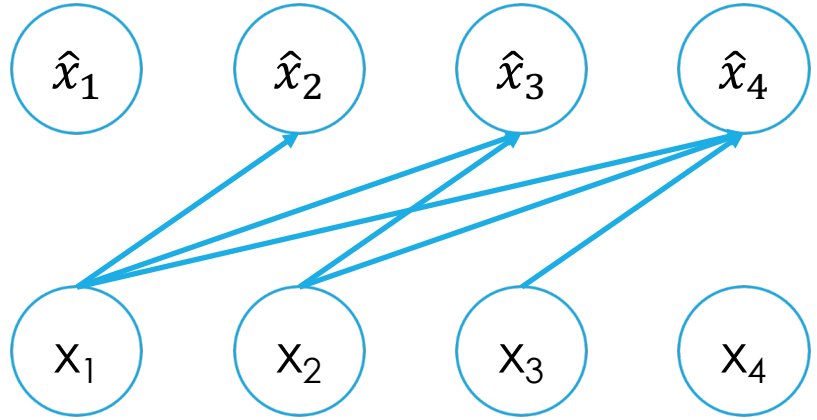
Autoregressive Models

- Example: when $x_1, \dots, x_n$ are 0 or 1, you can choose
  - $p_\theta(X_1 = 1; \theta_1) = \theta_1$ and $p_\theta(X_1 = 0; \theta_1) = 1 - \theta_1$
  - $p_\theta(X_2 = 1 | x_1; \theta_2) = sigmoid(\theta_{2,0} + \theta_{2,1} x_1), \dots$
  - $p_\theta(X_n = 1 | x_1, \dots, x_{n-1}; \theta_n) = sigmoid(\theta_{n,0} + \theta_{n,1} x_1 + \cdots + \theta_{n,n-1} x_{n-1})$
- How to sample from $p(x_1, \dots, x_n)$?
  - Sample $\hat{x}_1 \sim P_\theta(X_1; \theta_1)$
  - Sample $\hat{x}_2 \sim P_\theta(X_2 | X_1 = \hat{x}_1; \theta_1)$
  - Sample $\hat{x}_3 \sim P_\theta(X_3 | X_1 = \hat{x}_1, X_2 = \hat{x}_2; \theta_1)$
  - …
- Data generation: Sequential ➔ Slow
- Naive implementation: conditionals do not share information ➔ Too many params

This is a modeling assumption or your choice

Sigmoid Belief Network (SBN)

- The conditional variables $X_i|X_1, \dots, X_{i-1}$ are Bernoulli with parameters
  - $\hat{x}_i = p_\theta(X_i = 1|x_1, \dots, x_{i-1}; \theta_i) = sigmoid(\theta_{i,0} + \theta_{i,1}x_1 + \cdots + \theta_{i,i-1}x_{i-1})$
  - $\hat{x}_1 = p_\theta(X_1 = 1)$
- Learning: find $\boldsymbol{\theta} = \{\theta_1, \dots, \theta_n\}$
- How to sample from $p(x_1, \dots, x_n)$?
  - Sample $\hat{x}_1 \sim P_\theta(X_1)$
  - Sample $\hat{x}_2 \sim P_\theta(X_2|X_1 = \hat{x}_1)$
  - Sample $\hat{x}_3 \sim P_\theta(X_3|X_1 = \hat{x}_1, X_2 = \hat{x}_2)$, …
- Generate an image in a pixel-wise manner (sinh từng điểm ảnh một cách tuần tự)
- How many parameters?



$$1 + 2 + \cdots + n = O(n^2)$$

➔ Too many!!!

Deep Sigmoid Belief Network: some results

- Stack many SBN layers (xếp chồng nhiều tầng SBN lại)

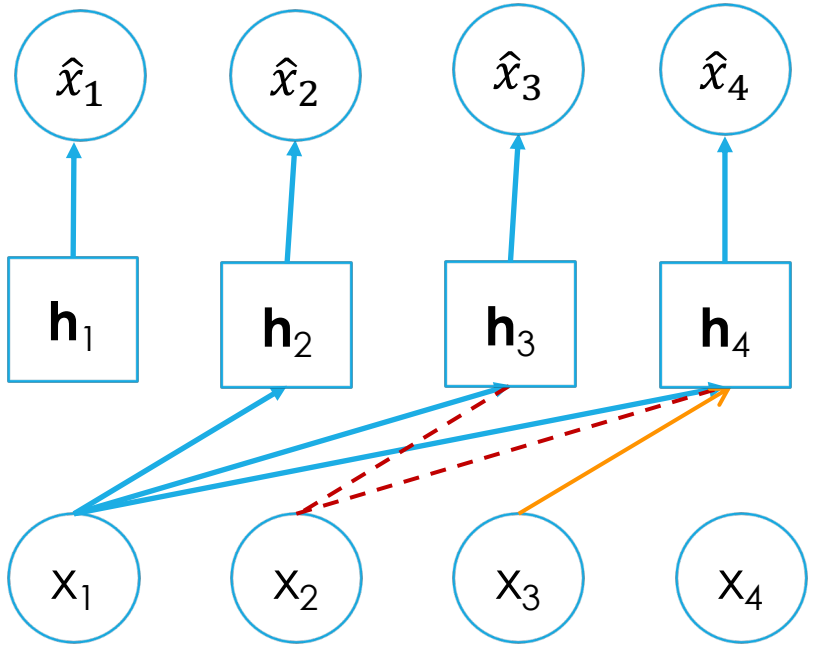
Training data

Generated data

Gan, Z., Henao, R., Carlson, D., & Carin, L. (2015). Learning deep sigmoid belief networks with data augmentation. In *Artificial Intelligence and Statistics*.

NADE: Neural Autoregressive Distribution Estimation

- Use one-layer neural network to connect from input **x** to output **a**
- The conditional variables $X_i|X_1, \dots, X_{i-1}$ are Bernoulli with parameters
  - $\boldsymbol{h}_i = sigmoid(\boldsymbol{W}_i \boldsymbol{x}_{<i} + \boldsymbol{c}_i)$
  - $\hat{x}_i = p_\theta(x_i|\boldsymbol{x}_{<i}; \boldsymbol{W}_i, \boldsymbol{V}_i, \boldsymbol{c}_i) = sigmoid(\boldsymbol{V}_i \boldsymbol{h}_i + b_i)$
  - where $\boldsymbol{x}_{<i} = (x_1, \dots, x_{i-1})^T, \quad \boldsymbol{h}_i \in \mathbb{R}^d$
  - The connections from each input x_i to $\boldsymbol{h}_i$'s **share the same weight** (một vài trọng số được chia sẻ)
- Params: $\boldsymbol{W}_i, \boldsymbol{V}_i, \boldsymbol{c}_i, b_i$
- Number of params: O(nd)

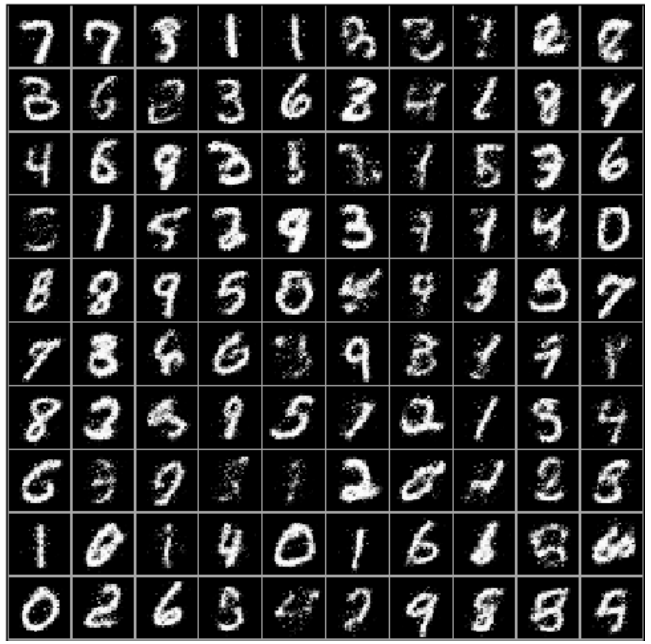


Efficient

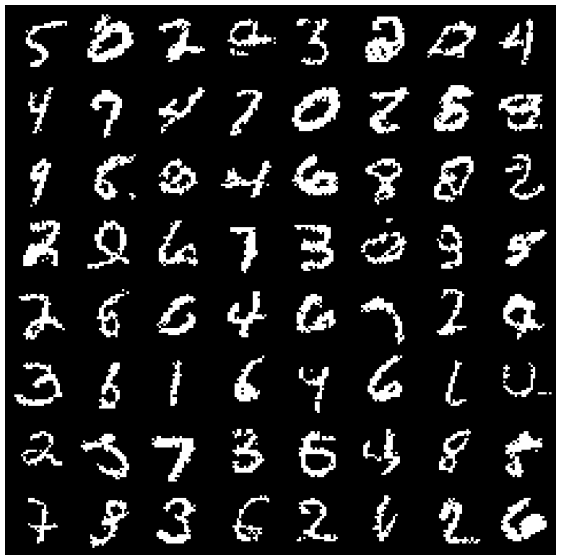
Uria, B., Côté, M.A., Gregor, K., Murray, I. and Larochelle, H., 2016. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*.

NADE: result on MNIST

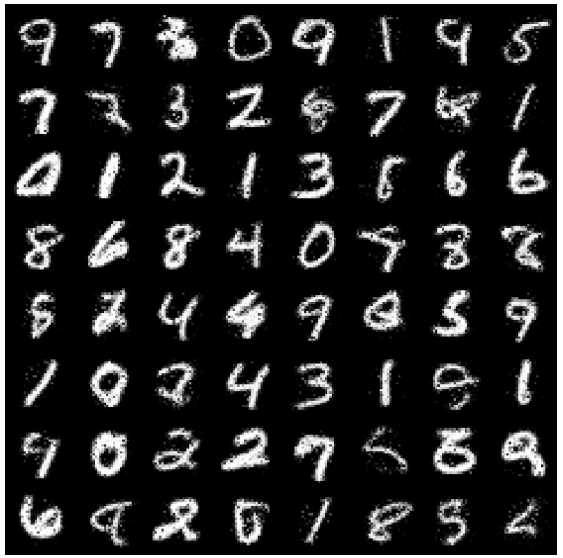
Samples from NADE



Conditional probabilities $\hat{x}_i$



ConvNADE+DeepNADE



Conditional probabilities $\hat{x}_i$

Model	$-\log p$
DRAW (without attention)	
DRAW	
Pixel RNN	**79.20**
ConvNADE+DeepNADE (no input masks)	85.25
ConvNADE	81.30
ConvNADE+DeepNADE	80.82

Negative log-likelihood

Some variants of NADE can work with continuous inputs

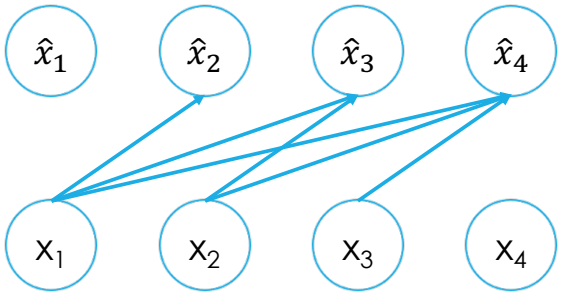
Uria, B., Côté, M.A., Gregor, K., Murray, I. and Larochelle, H., 2016. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*.

Autoregressive models vs. autoencoders

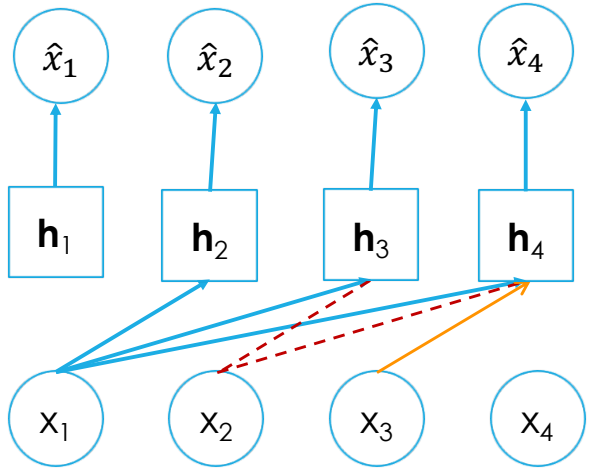
- SBN and NADE seems to be similar with Autoencoders
  - *Encoder:* produce an embedding $\boldsymbol{h}_i$, e.g., $\boldsymbol{h}_i = sigmoid(\boldsymbol{W}_i \boldsymbol{x}_{<i} + \boldsymbol{c}_i)$
  - *Decoder:* reconstruct the input, e.g., $\hat{x}_i = sigmoid(\boldsymbol{V}_i \boldsymbol{h}_i + b_i)$
  - MLE will lead to the loss for dataset **D**:

$$-\sum_{\boldsymbol{x} \in \boldsymbol{D}} \sum_{i} [x_i \log \hat{x}_i + (1 - x_i) \log(1 - \hat{x}_i)]$$

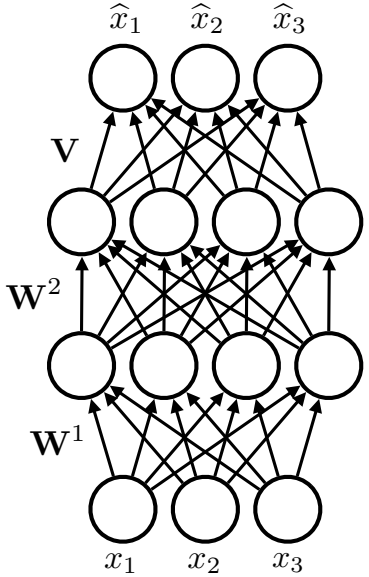
- Vanilla autoencoders cannot generate data



SBN



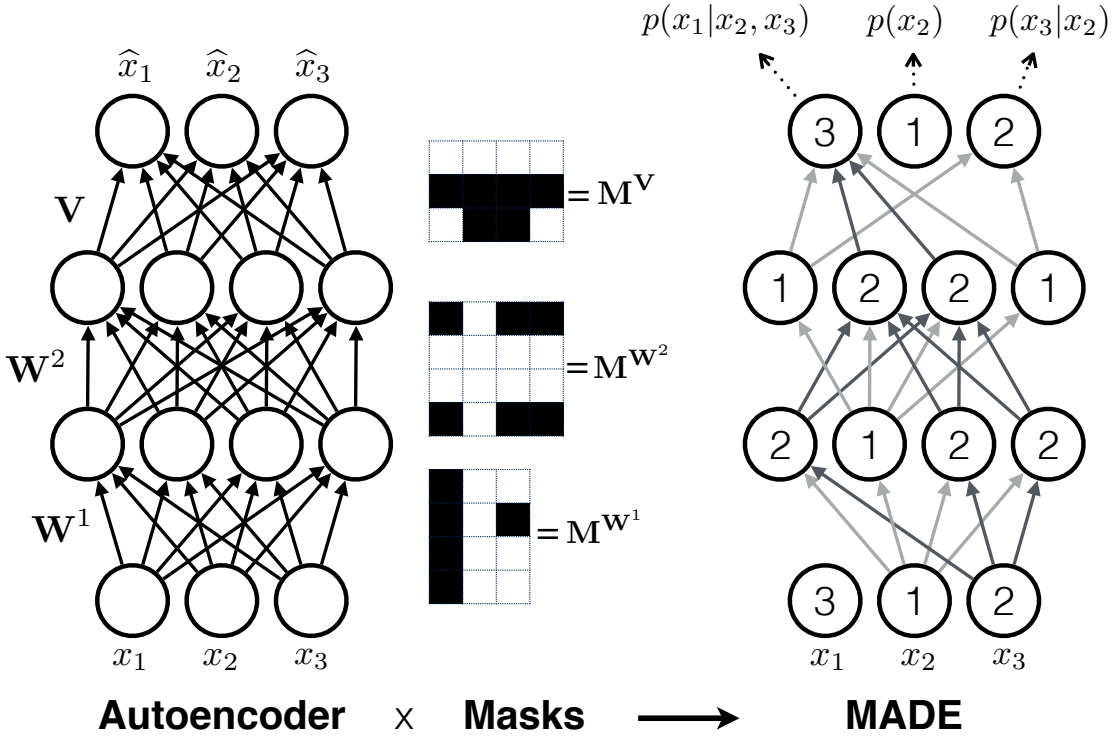
NADE



Autoencoders

MADE: Masked autoencoder for distribution estimation

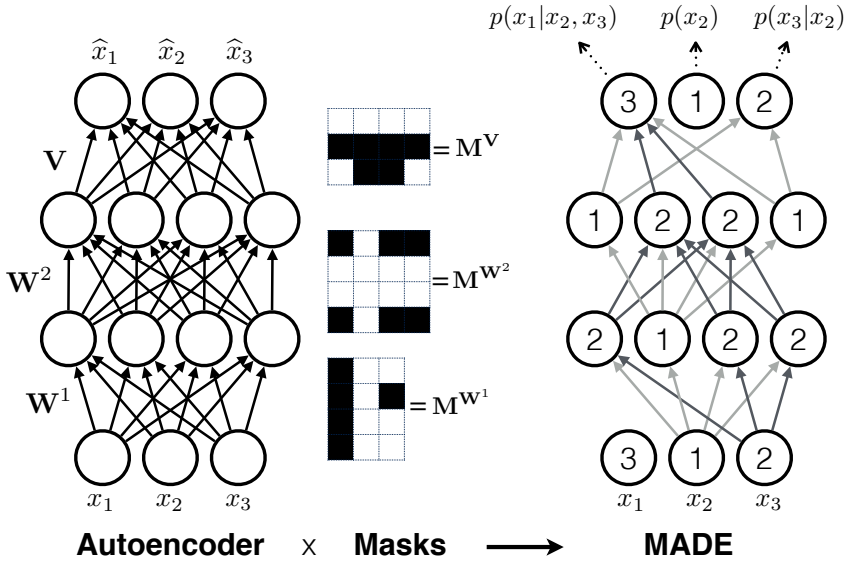
- Can we enable an autoencoder to be a generative model?
- **Solution:** use masks to disallow certain paths from input to output
  (dùng mặt nạ để tắt một số đường đi từ đầu vào đến đầu ra)
- Given an ordering x_2, x_3, x_1, we need to know $p(x_2), p(x_3|x_2), p(x_1|x_2, x_3)$
  - $p(x_2)$ does not receive any input, $p(x_3|x_2)$ receives input x_2 only, …
  - Each hidden unit has a count i randomly picked in $[1, n-1]$, and can receive the first i inputs only (mỗi nơron ẩn chỉ nhận nhiều nhất *i* tín hiệu đầu vào, *i* được sinh ngẫu nhiên)
  - Connect to all units in previous layer with smaller or equal assigned count, using a mask



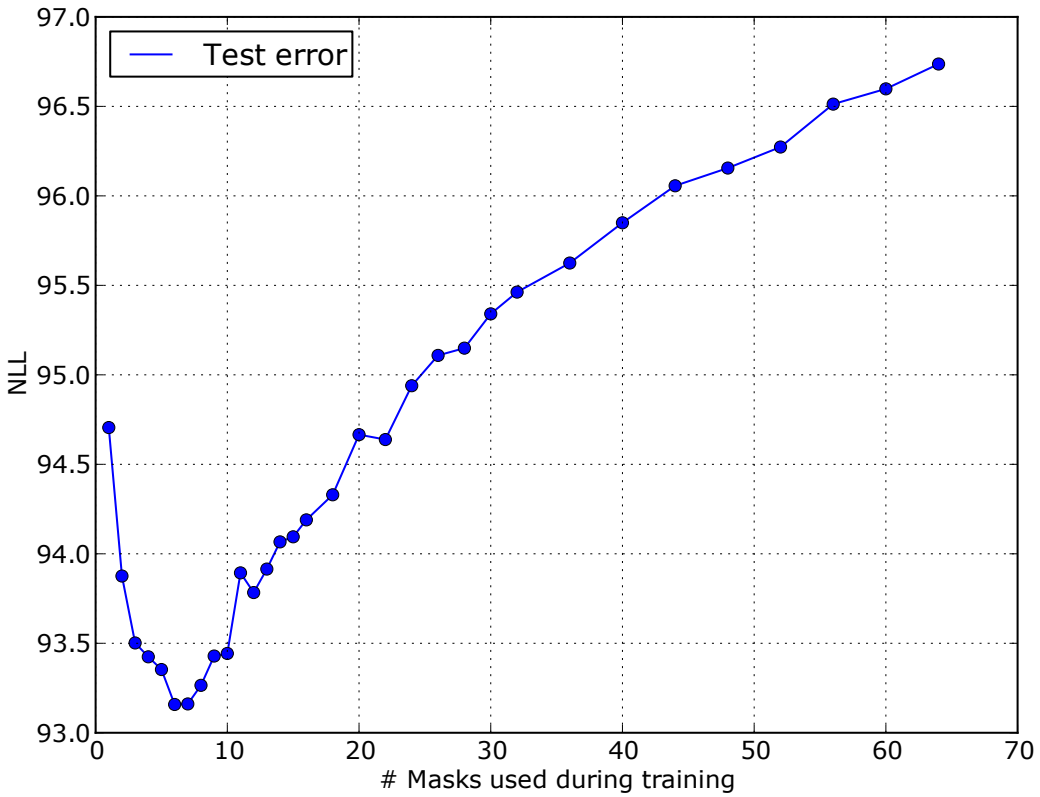
Germain, M., Gregor, K., Murray, I., & Larochelle, H. (2015). MADE: Masked autoencoder for distribution estimation. In *ICML*

MADE

- Let $\boldsymbol{M}^{v}, \boldsymbol{M}^{w}$ be the mask matrices, then
  - $\boldsymbol{h}_i = sigmoid((\boldsymbol{M}^{w} \odot \boldsymbol{W}_i)\boldsymbol{x}_{<i} + \boldsymbol{c}_i)$
  - $\hat{x}_i = p_\theta(x_i|\boldsymbol{x}_{<i}; \boldsymbol{W}_i, \boldsymbol{V}_i, \boldsymbol{c}_i) = sigmoid((\boldsymbol{M}^{v} \odot \boldsymbol{V}_i)\boldsymbol{h}_i + b_i)$
  - where $\odot$ denotes the element-wise product
- The ordering:
  - Need not being fixed
  - Can be generated according to the training process
    (thức tự của các biến có thể được chọn nhiều lần trong quá trình học)
    ➔ amount to training many different models
- Complexity to compute p(x): $O(n^2)$
  - Worse than NADE
  - How about VAE?



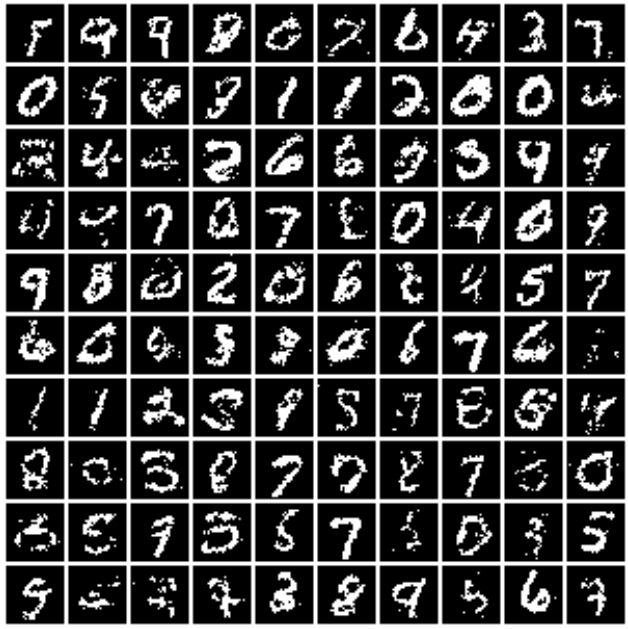
MADE: results



NLL - Negative log-likelihood

Model	Adult	Connect4	DNA
MoBernoullis	20.44	23.41	98.19
RBM	16.26	22.66	96.74
FVSBN	**13.17**	12.39	83.64
NADE (fixed order)	**13.19**	11.99	84.81
EoNADE 1hl (16 ord.)	**13.19**	12.58	82.31
DARN	13.19	11.91	81.04
MADE	**13.12**	**11.90**	83.63
MADE mask sampling	**13.13**	**11.90**	**79.66**

Too many masks may not be good

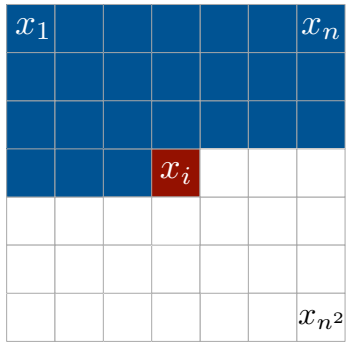


PixelRNN

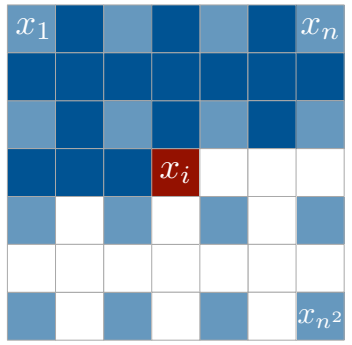
- MADE uses a feed-forward neural network (MLP)
- **PixelRNN** uses a recurrent neural network (RNN)
  - Model images pixel by pixel by using a raster scan order
  - Each pixel conditional $\mathrm{p}(x_i|x_{1:i-1})$ needs to specify 3 colors

$$\mathrm{p}(x_i|x_{1:i-1}) = p\left(x_i^{red}\middle|x_{1:i-1}\right)p\left(x_i^{green}\middle|x_{1:i-1}, x_i^{red}\right)p\left(x_i^{blue}\middle|x_{1:i-1}, x_i^{red}, x_i^{green}\right)$$

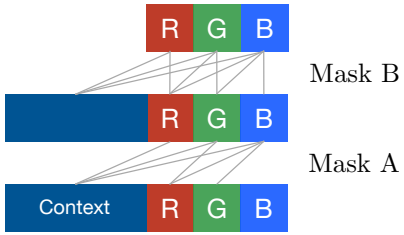
  - where $p\left(x_i^{red}\middle|x_{1:i-1}\right) = categorical(\theta_1^r, \ldots, \theta_{256}^r)$, and the same for other conditionals
  - *Share the same weight for every i*
- We can use RNN variants
  - LSTM + masking
- CNN can also be used: **PixelCNN**



Context



Multi-scale context



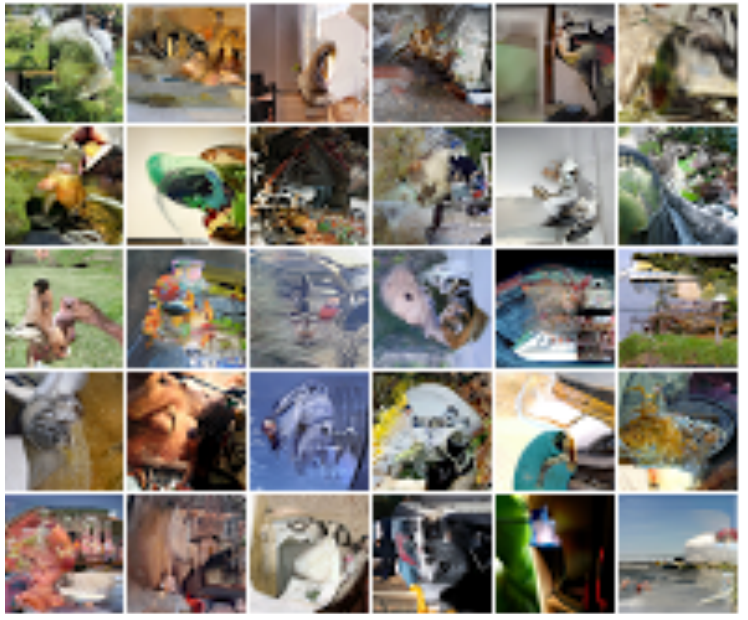
Van Den Oord, A., Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *ICML*.

PixelRNN + PixelCNN

- Better than prior models

Model	NLL Test
DBM 2hl [1]:	$\approx$ 84.62
DBN 2hl [2]:	$\approx$ 84.55
NADE [3]:	88.33
EoNADE 2hl (128 orderings) [3]:	85.10
EoNADE-5 2hl (128 orderings) [4]:	84.68
DLGM [5]:	$\approx$ 86.60
DLGM 8 leapfrog steps [6]:	$\approx$ 85.51
DARN 1hl [7]:	$\approx$ 84.13
MADE 2hl (32 masks) [8]:	86.64
DRAW [9]:	$\leq$ 80.97
PixelCNN:	81.30
Row LSTM:	80.54
Diagonal BiLSTM (1 layer, $h = 32$):	**80.75**
Diagonal BiLSTM (7 layers, $h = 16$):	**79.20**

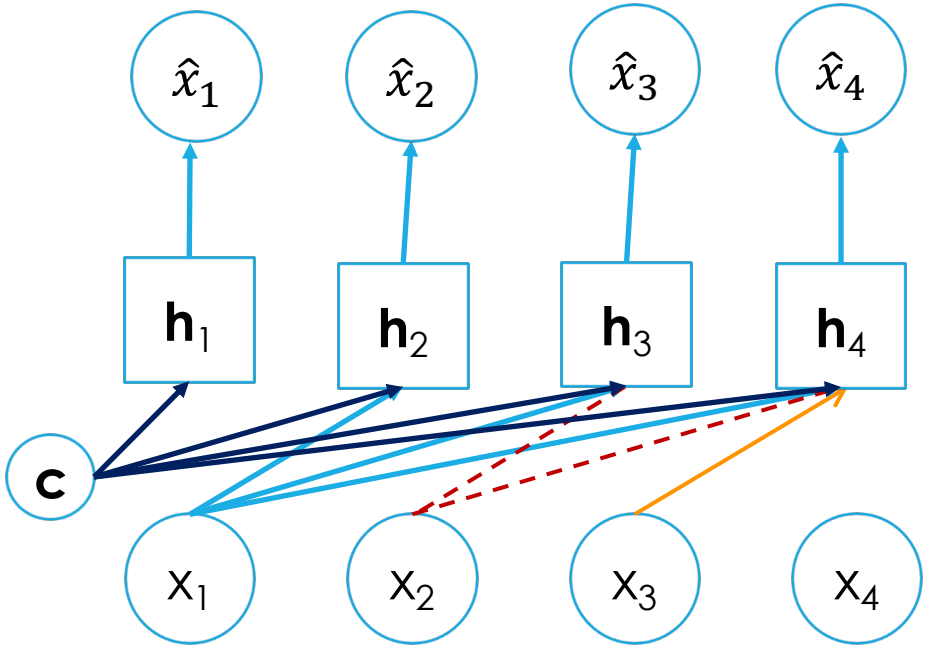
PixelCNN	Row LSTM	Diagonal BiLSTM
7×7 conv mask A		
Multiple residual blocks: (see fig 5)		
Conv 3×3 mask B	Row LSTM i-s: 3×1 mask B s-s: 3×1 no mask	Diagonal BiLSTM i-s: 1×1 mask B s-s: 1×2 no mask
ReLU followed by 1×1 conv, mask B (2 layers)		
256-way Softmax for each RGB color (Natural images) or Sigmoid (MNIST)		



Conditional models

- How to exploit more information? Class labels, tags, keywords, …
- Given information represented by a latent vector **c**, we need to know $p(\boldsymbol{x}|\boldsymbol{c})$
- Solution:
  - Whenever $\boldsymbol{Wx} + \boldsymbol{w_0}$ appears, replace it with $\boldsymbol{Wx} + \boldsymbol{Ac}$
  - **A** is trainable
  - Can do the same for the hidden layers
- Class-Conditional PixelCNN
  - IN: One-hot encoding of the labels
  - THEN: multiplying by different learned weight matrices in each convolutional layer, and added as a bias channel-wise and broadcasted spatially

A small hedgehog holding a piece of watermelon

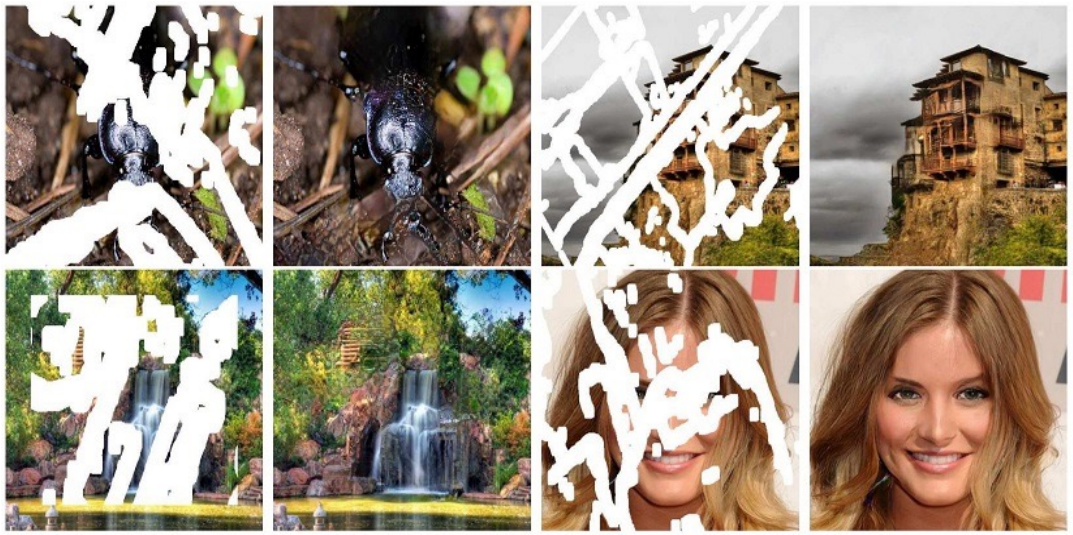


Conditional models: some examples

P(high resolution | low resolution)

Menon et al, 2020

P(full image | mask)



Liu al, 2018

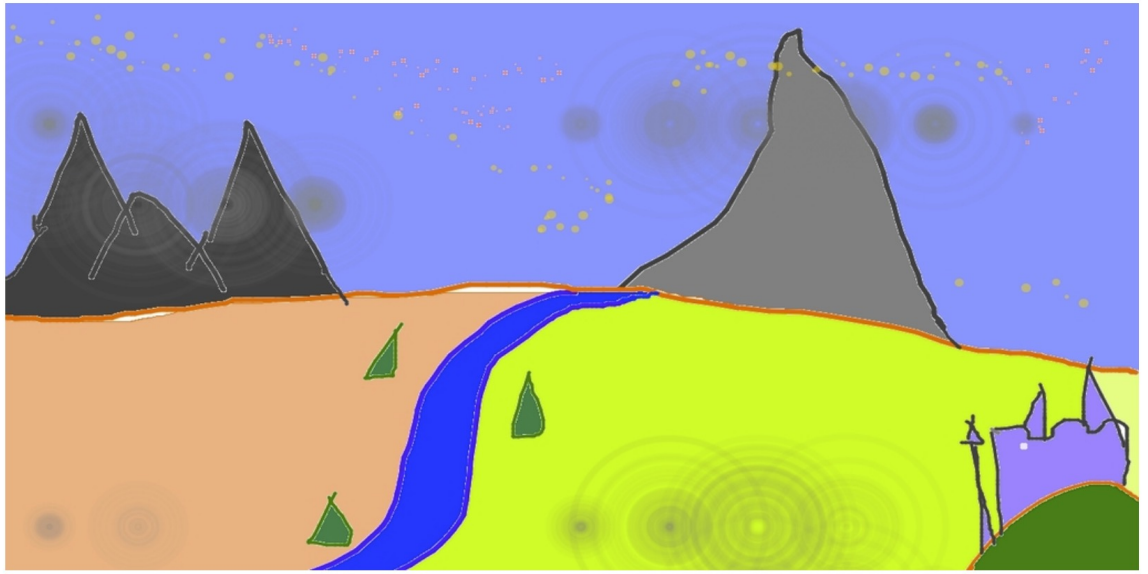
P(color image | greyscale)

Antic, 2020

Conditional models: some examples

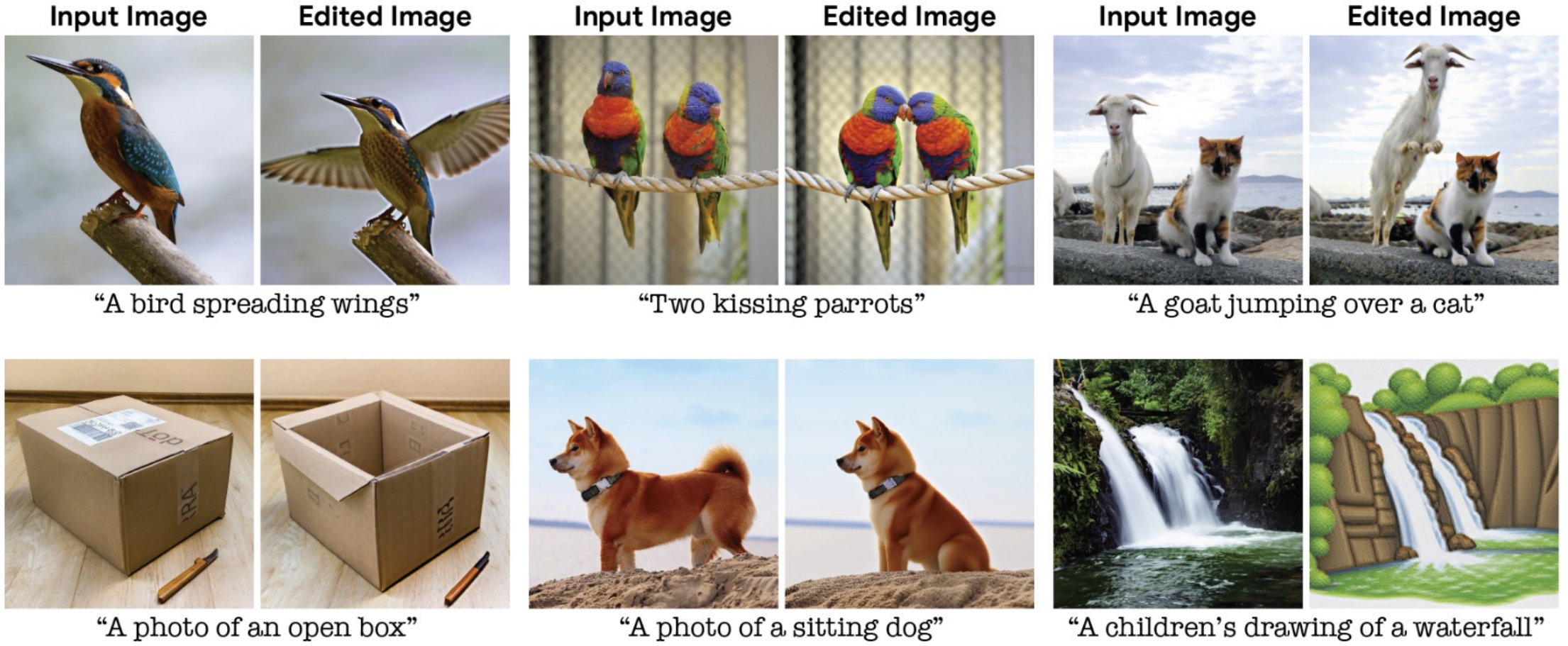
- Input

Sketch



Output

Conditional models: some examples



Kawar et al., 2023

Neural autoregressive models: summary

- Pros:
  - Best in class modelling performance
  - Expressivity: autoregressive factorization is general, NNs are universal approximators
  - Generalization: meaningful parameter sharing has good inductive bias
- Cons:
  - Slow, due to sequential generation, e.g., pixel-by-pixel, character-by-character

More

- Character RNN
- Transformer
- GPTs
- WaveNet
- …