Generative Modeling

Autoregressive models

Denis Derkach, Artem Ryzhikov, Maxim Artemev

Laboratory for methods of big data analysis

LAMBDA • HSE

Spring 2022

In this Lecture

- Motivation
  - LAN
- Neural Autoregressive Networks:
  - NAN
  - NADE, NADE-k
  - RNADE
- Masked Autoencoders:
  - MADE
- RNN:
  - PixelRNN

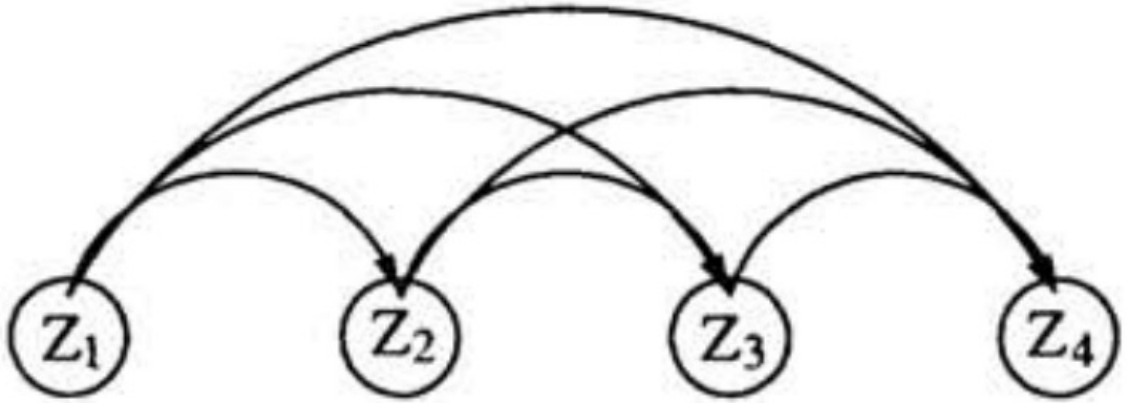
Motivation

Reminder: Chain Rule

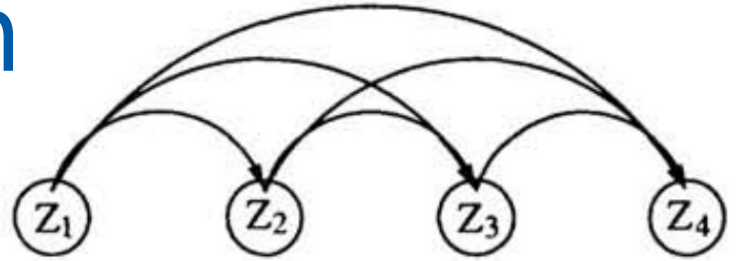
- Probability Chain Rule:

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

- Autoregressive modelling: observations from the previous steps are used to predict the value at the current step



Autoregressive Models: discussion



- Probability Chain Rule:

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

- Need to choose ordering of dimensions (or use time-series definition).
- Properties:
  - **pros**: $p(x)$ is tractable, thus easy to train (maximum likelihood) and easy to sample (may be slower);
  - **cons**: do not natural latent representation.

Frey B.J. Adaptive Computation and Machine Learning Graphical Models for Machine Learning and Digital Communication (1998), MIT Press

Logistic Autoregressive Network

- Need to generate 2D distribution with only 2 outcomes possible per observable.
- The chain rule will become:

$$p(x_1, x_2) = p(x_0 = 1, x_1, x_2) = p(x_0)p(x_1|x_0)p(x_2|x_0, x_1).$$

- What is the distribution for $p(x_1|x_0)$? Bernoulli?
- Parametrize it with logistic regression:

$$p(X_1 = 1|x_0) = \sigma\left(a_1 + a_{1,0}p(x_0)\right).$$

- And so on:

$$p(X_2 = 1|x_0, x_1) = \sigma\left(a_2 + a_{2,1}p(x_1) + a_{2,0}p(x_0)\right).$$

Logistic Autoregressive Network: MNIST

- Binarized version MNIST

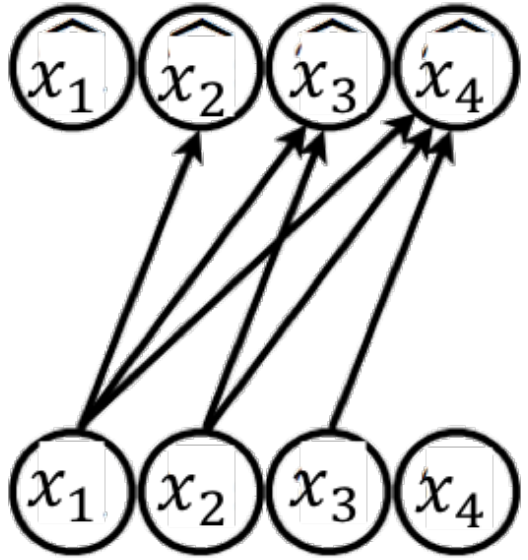
- In case of MNIST n = 28×28 = 784 pixels, each pixel is black or white. The conditional variables $X_i|X_1, \dots, X_{i-1}$ are Bernoulli with parameters.

$$\hat{x}_i = p\left(X_i = 1 \middle| x_1, \dots, x_{i-1}; \alpha^i\right) = \sigma(\alpha_0^i + \sum \alpha_j^i x_j)\,.$$

Logistic Autoregressive Network: properties

We can evaluate by computing corresponding conditional probabilities:

$$p(X_1 = 0, X_2 = 1, X_3 = 1, X_4 = 0) =$$
$$= (1 - \hat{x}_1) \times \hat{x}_2 \times \hat{x}_3 \times (1 - \hat{x}_4) =$$
$$= (1 - \hat{x}_1) \times \hat{x}_2(X_1 = 0) \times$$
$$\times x^3(X_1 = 0, X_2 = 1) \times (1 - \hat{x}_4(X_1 = 0, X_2 = 1, X_3 = 1)).$$

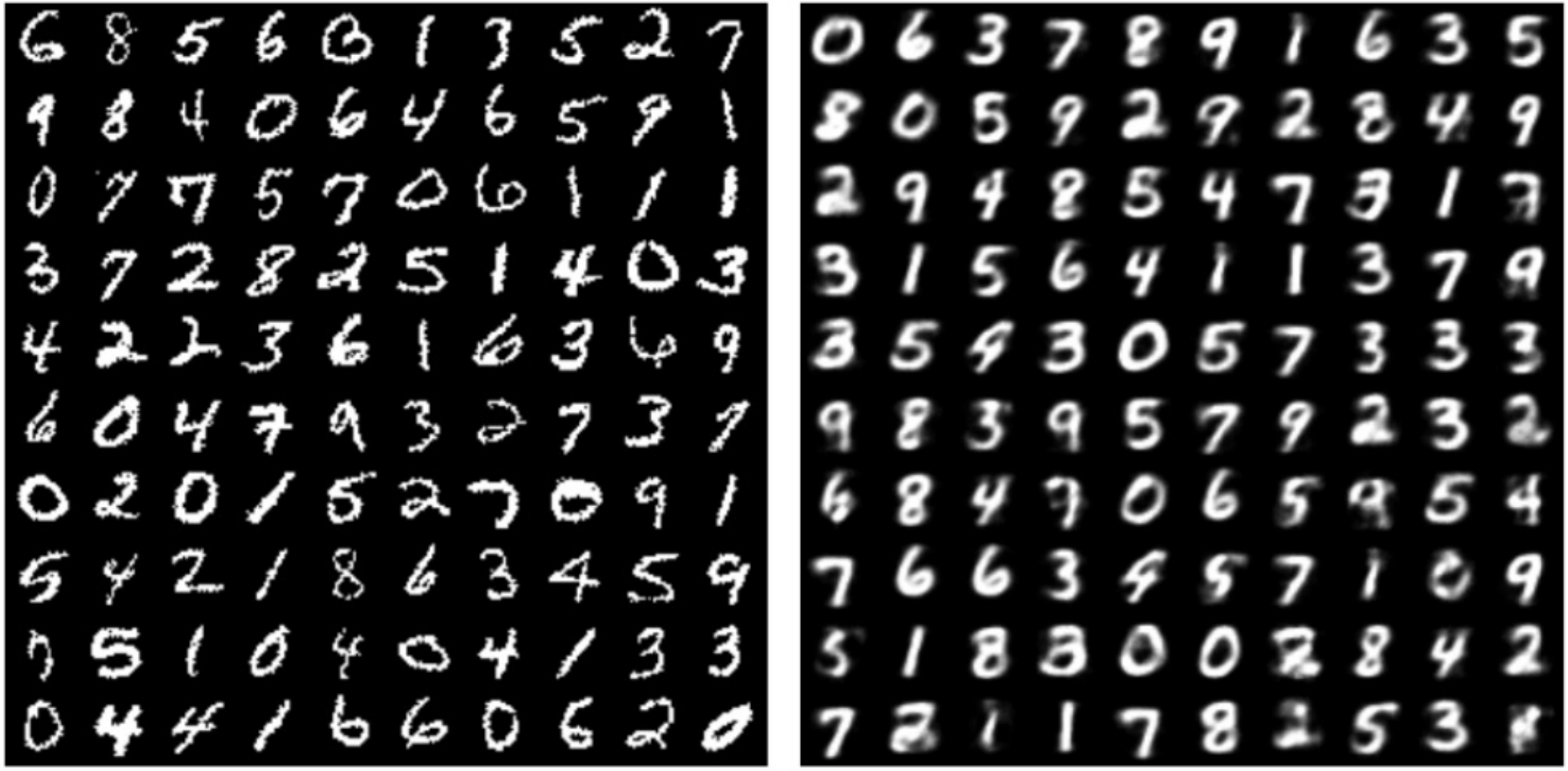


FVSBN

Consecutive sampling $p(x_1, \ldots, x_{784})$:

1. Sample $\bar{x}_1 \sim p(x_1)$.
2. Sample $\bar{x}_2 \sim p(x_2|x_1 = \bar{x}_1)$
3. Sample $\bar{x}_3 \sim p(x_3|x_1 = \bar{x}_1, x_2 = \bar{x}_2)$

FVSBN: results



Training data on the left (MNIST binarized). Samples from the model on the right.

Z. Gan et al. Learning Deep Sigmoid Belief Networks with Data Augmentation AISTATS 2015

LAN: discussion

- easy to sample;
- easy to evaluate;
- cannot obtain additional capacity;
- hard to scale the capacity, still can suffer from curse of dimensionality;
- number of parameters $d^2/2$.

Neural parameterization

- Probability Chain Rule:

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

- What if we can approximate factor by a neural net?

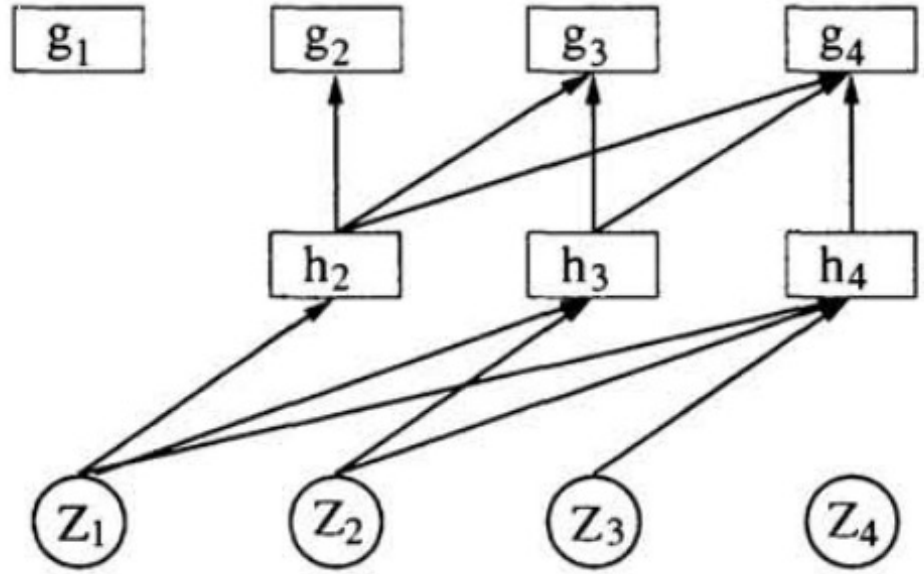
$$p(x_1, x_2, \ldots, x_n) = p(x_1)p_{NN}(x_2|x_1)\ldots p_{NN}(x_n|x_1, \ldots, x_{n-1})$$

- Should we be able to estimate it by using ever deeper network?

Neural Autoregressive Network

Neural Autoregressive Network

- Increase the capacity of model by inserting a hidden layer neural network instead of sigmoid.
- NN is more capable of catching better parameterization.



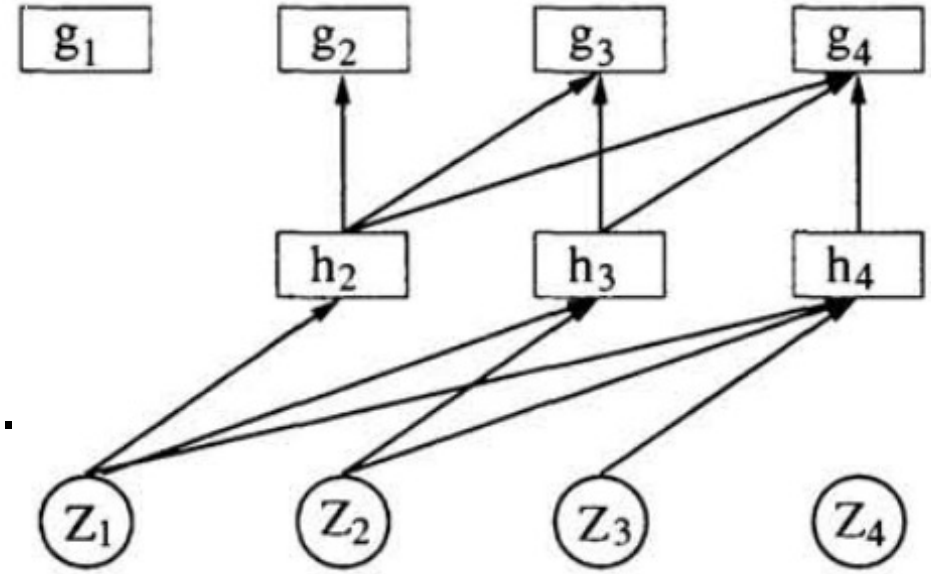
Y. Bengio et al., Modeling High-Dimensional Discrete Data with Multi-Layer Neural Networks, NIPS 2000

Neural Autoregressive Network

- One-layer neural network instead of logistic regression:

$$h_i = \sigma(W_i x_{<i} + b)$$

$$\hat{x}_i = p(X_i = 1 | x_1, \dots x_{i-1}; W, b, \alpha) = \sigma(\alpha_0^i + \sum \alpha_j^i x_j)).$$



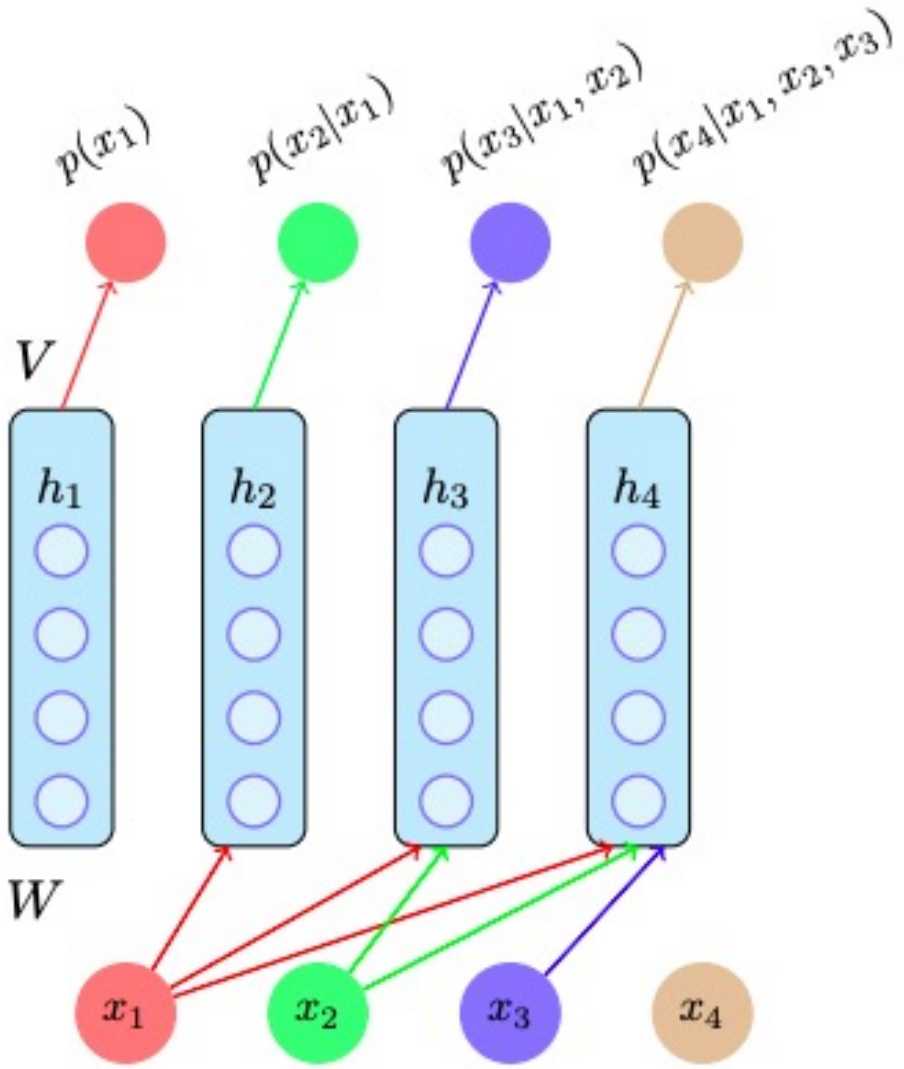
- NNs are different:

$$h_2 = \sigma\left(\left(\dots\right) + \left(\dots\right) x_1\right); h_3 = \sigma\left(\left(\dots\right) + \left(\dots\dots\right)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right)$$

Y. Bengio et al., Modeling High-Dimensional Discrete Data with Multi-Layer Neural Networks, NIPS 2000

NADE: Neural Autoregressive Density Estimation

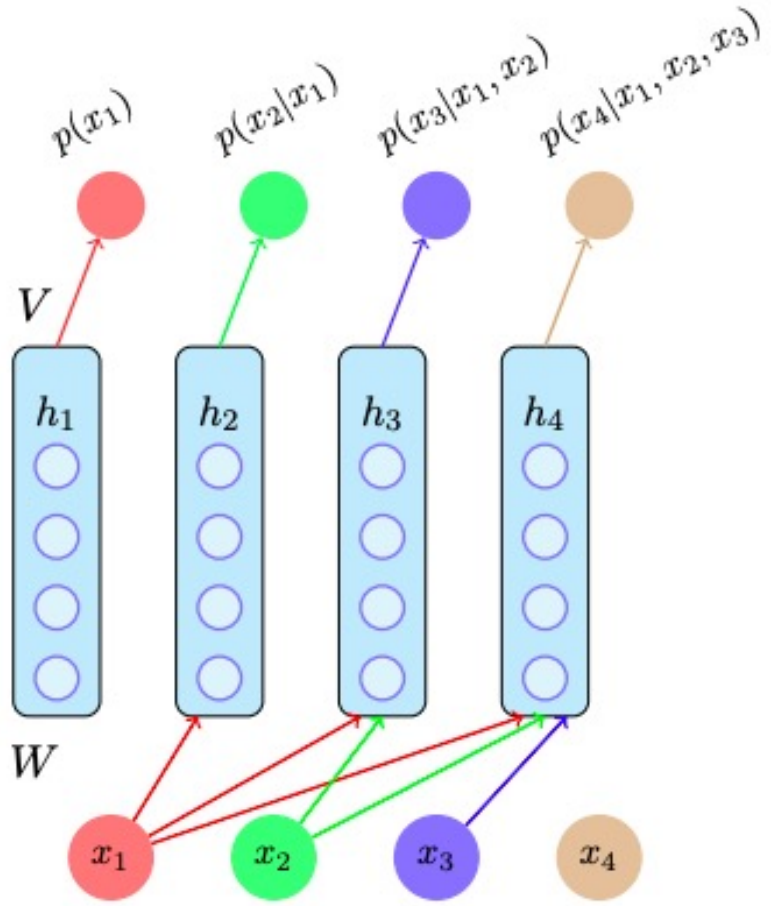
Once can go even further: tie weights to reduce the number of parameters and speed up computation.



NADE: math

The amount of parameters will be reduced due to the shared weights:

$$h_2 = \sigma\left(\left(\ c\ \right) + \left(W_1\right)x_1\right); h_3 = \sigma\left(\left(\ c\ \right) + \left(W_1 W_2\right)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right)$$



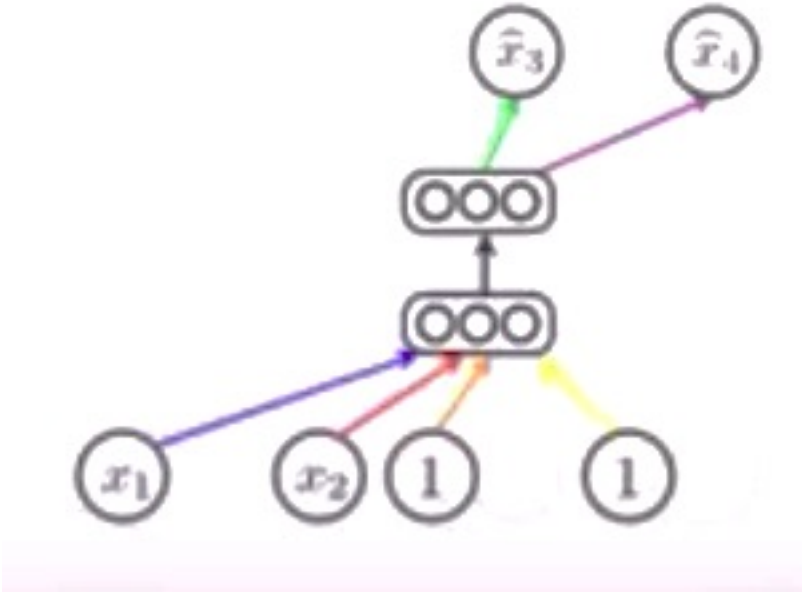
H. Larochelle et al., The Neural Autoregressive Distribution Estimator, AISTATS 2011

NADE: results and discussion

- NADE allows for better image generation.
- The idea of calculations is closely related to Restricted Boltzman Machines.
- The complexity is O(HD).
- Not designed for representation learning.
- Originally only allows for binary distribution and single hidden layer network.

Deep NADE

- Two problems to be solved:
  - ordering dependence;
  - deep architectures.
- Solution:
  - consider ordering as a part of loss function;
  - condition on whether the input is observed using masked inputs;
  - stochastic update to train all conditional subnets.



B. Uria et al. A Deep and Tractable Density Estimator, ICML 2014

Deep NADE: results

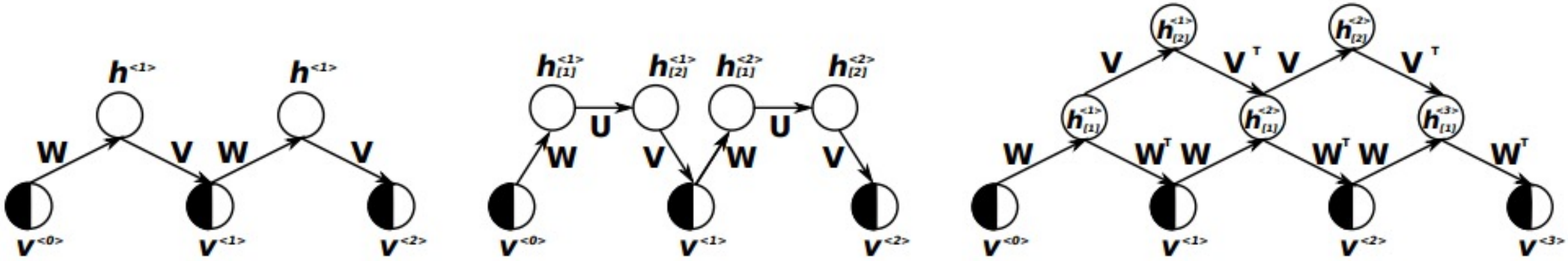
- Using same network brings low complexity while training

$$O(DH + H^2L).$$

- Sampling is complicated as we need to pass through all available ordering.

k-steps training (NADE-k)

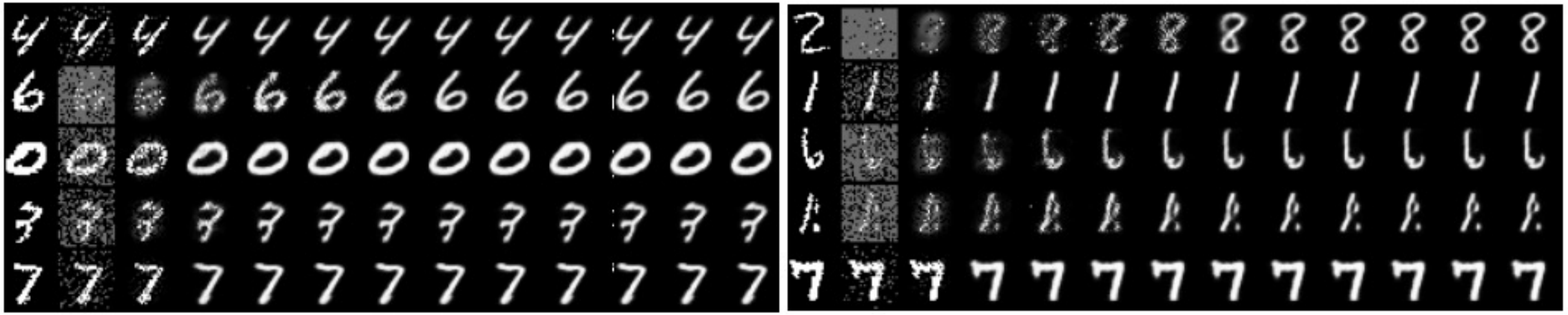
- Can use iterative procedure with k steps in order to have better final precision.



T. Raiko et al., Iterative Neural Autoregressive Distribution Estimator (NADE-k), NIPS 2014

NADE-k results

- $p(x_1)$ choice is random.
- Procedure to take ordering into account.
- Can use iterative procedure with k steps



T. Raiko et al., Iterative Neural Autoregressive Distribution Estimator (NADE-k), NIPS 2014

Real valued NADE (RNADE)

- Outputs the parameters of a mixture model for $p(x_k|x_{<k})$:

$$
\begin{aligned}
K \text{ mixing fractions}, \quad & \boldsymbol{\alpha}_d = \operatorname{softmax}\left(\boldsymbol{V}_d^{\alpha\top}\boldsymbol{h}_d + \boldsymbol{b}_d^{\alpha}\right) \\
K \text{ component means}, \quad & \boldsymbol{\mu}_d = \boldsymbol{V}_d^{\mu\top}\boldsymbol{h}_d + \boldsymbol{b}_d^{\mu} \\
K \text{ component standard deviations}, \quad & \boldsymbol{\sigma}_d = \exp\left(\boldsymbol{V}_d^{\sigma\top}\boldsymbol{h}_d + \boldsymbol{b}_d^{\sigma}\right),
\end{aligned}
$$

- W remains the same.

Table 1: Average test-set log-likelihood per datapoint for 4 different models on five UCI datasets. Performances not in bold can be shown to be significantly worse than at least one of the results in bold as per a paired t-test on the ten mean-likelihoods, with significance level 0.05.

Dataset	dim	size	Gaussian	MFA	FVBN	RNADE-MoG	RNADE-MoL
Red wine	11	1599	−13.18	−10.19	−11.03	**−9.36**	**−9.46**
White wine	11	4898	−13.20	−10.73	−10.52	**−10.23**	−10.38
Parkinsons	15	5875	−10.85	−1.99	**−0.71**	**−0.90**	−2.63
Ionosphere	32	351	−41.24	−17.55	−26.55	**−2.50**	**−5.87**
Boston housing	10	506	−11.37	−4.54	**−3.41**	**−0.64**	−4.04

B. Uria et al. RNADE: The real-valued neural autoregressive density-estimator

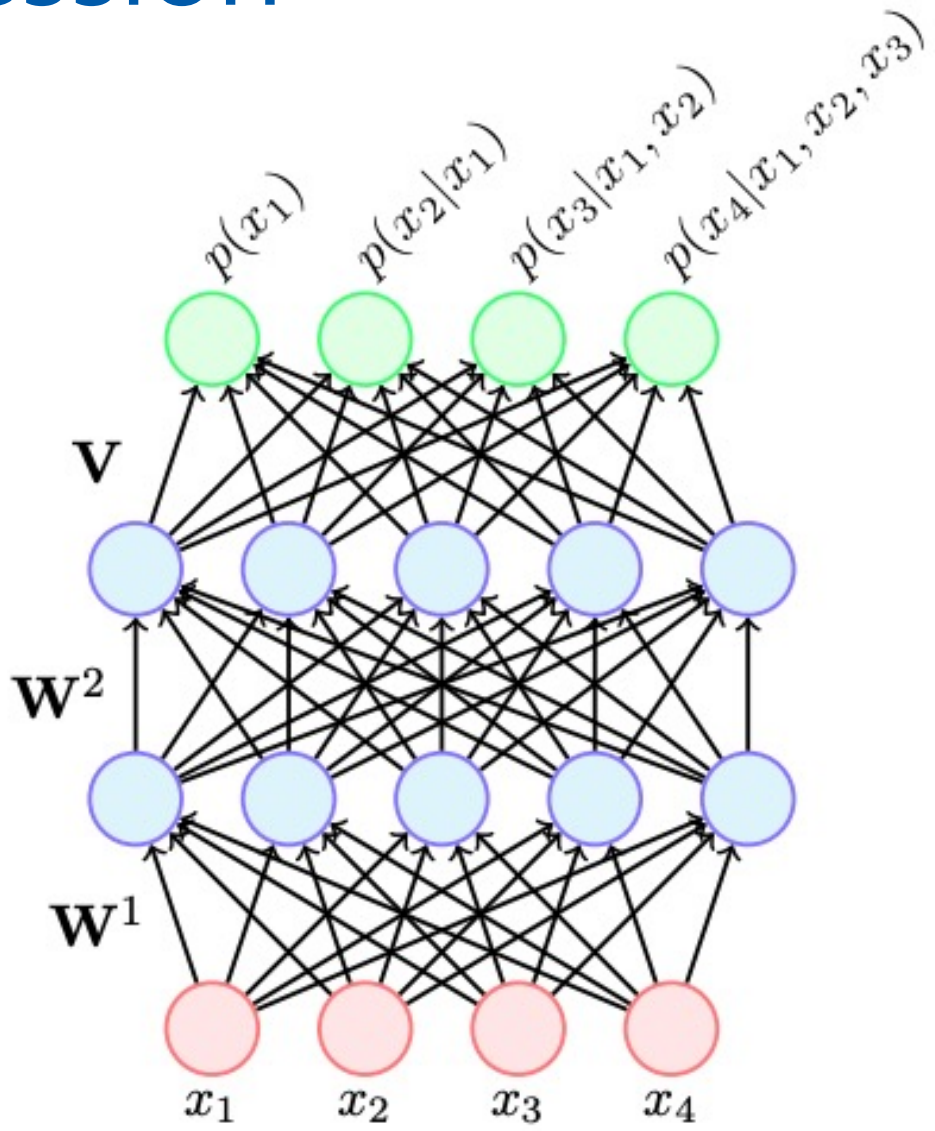
RNADE: conclusions

- deals with real numbered observables;
- outperforms mixture models on many datasets;
- tends to get complicated with many dimensions.

Masked Autoencoders

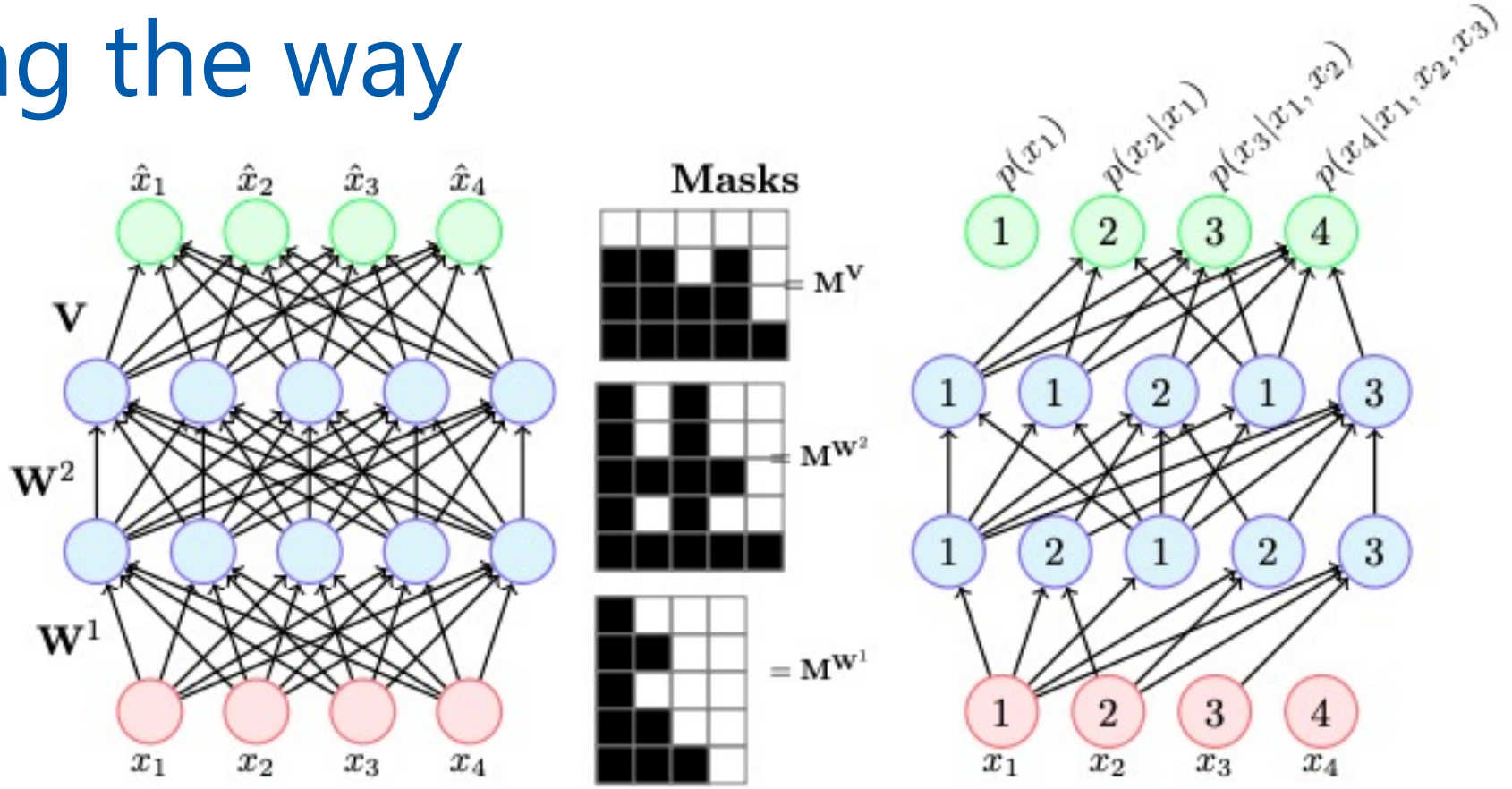
Autoencoders for autoregression

- Very similar regressive models.
- In a standard autoencoder with fully connected layers the k-th unit obviously depends on all the input units.
- How to ensure that the structure is correctly defined?



Mathieu Germain et al., MADE: Masked Autoencoder for Distribution Estimation

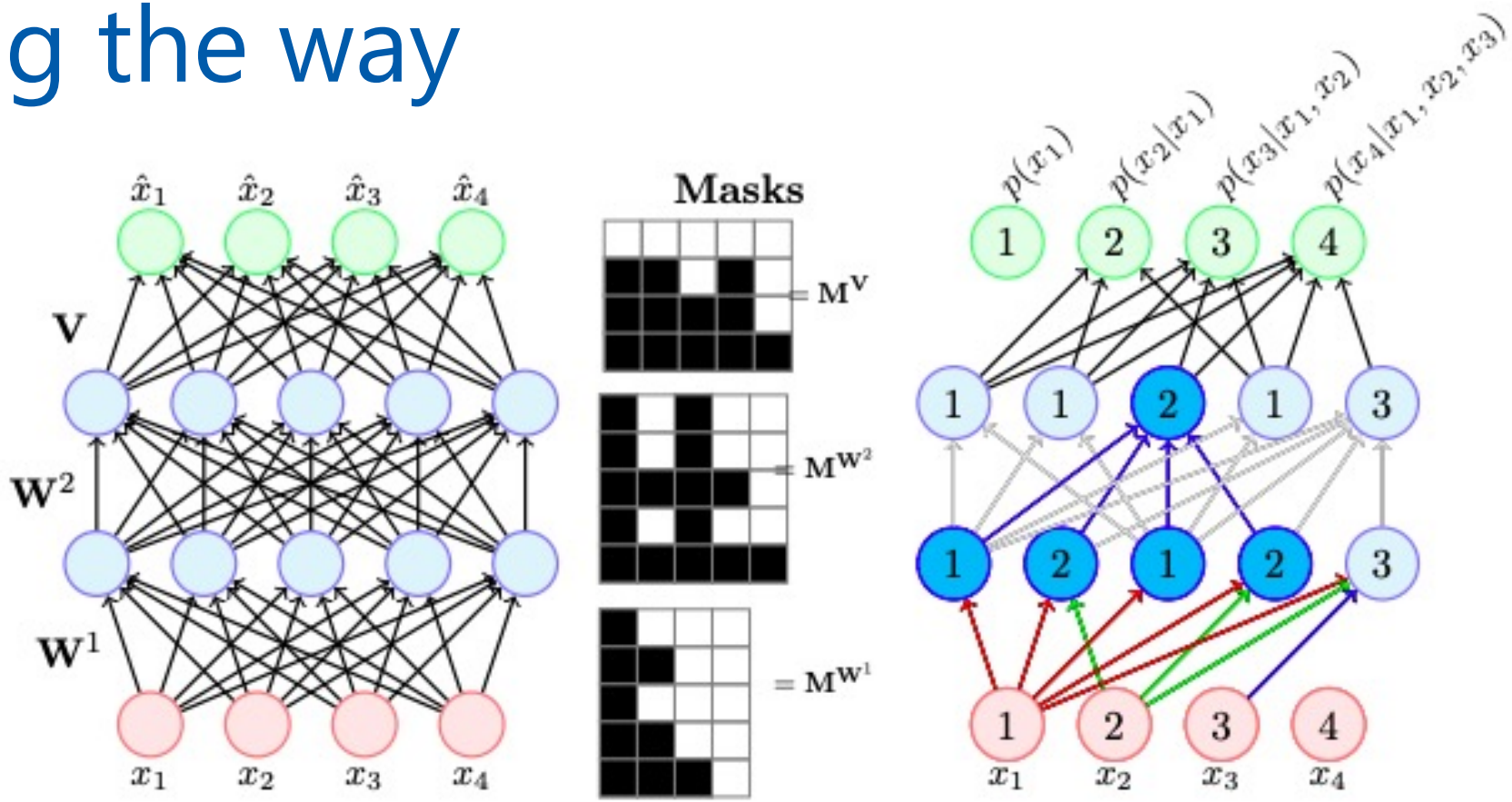
Masking the way



- We can create a random mask for each layer to ensure that the autoregressive condition is obtained:
  - Assign **random** integer [1; D-1] for each hidden unit.
  - This number corresponds to the possible connections of the hidden unit. And can be used to construct masks.

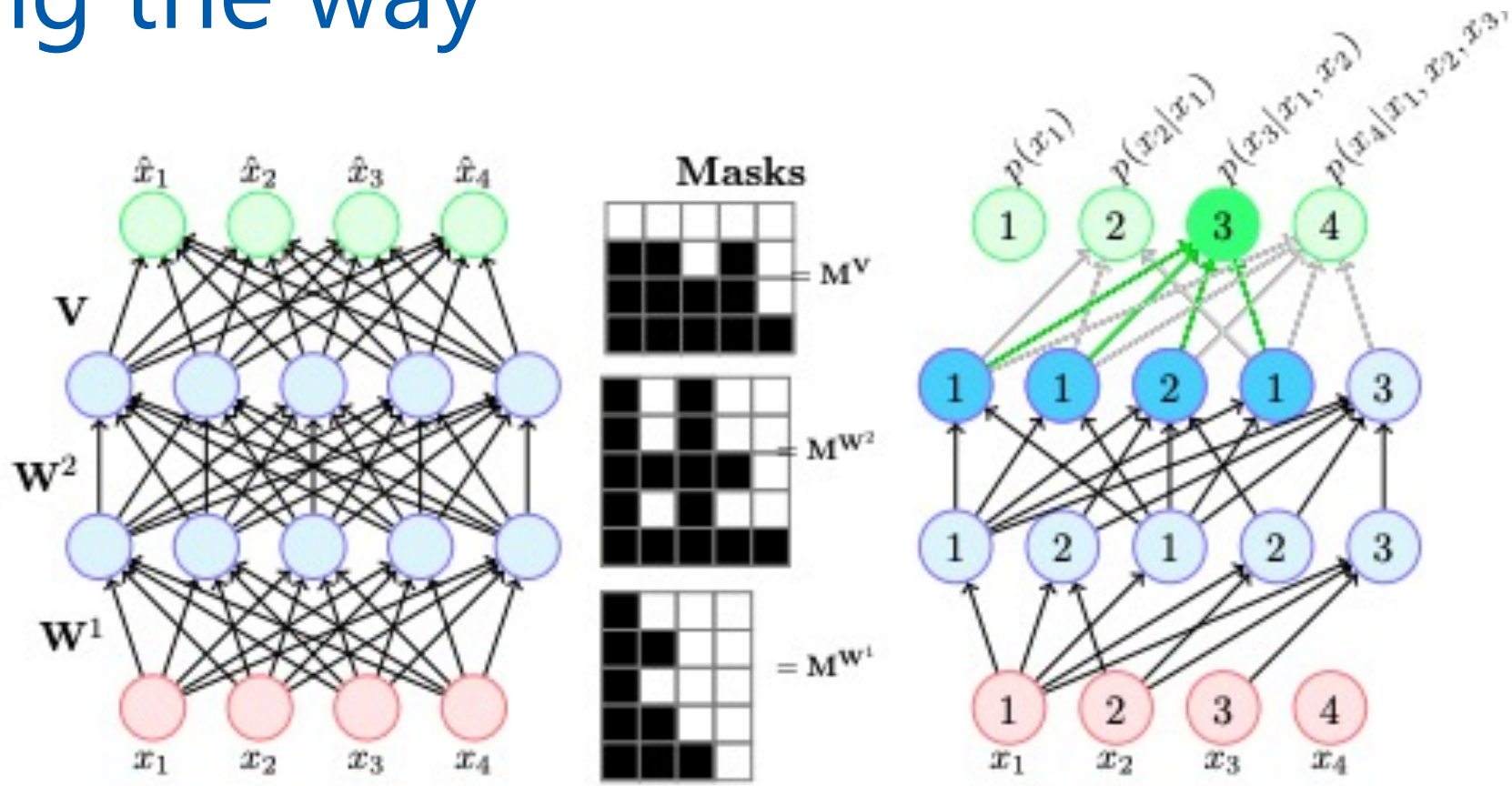
Figs: M. Kharpa

Masking the way



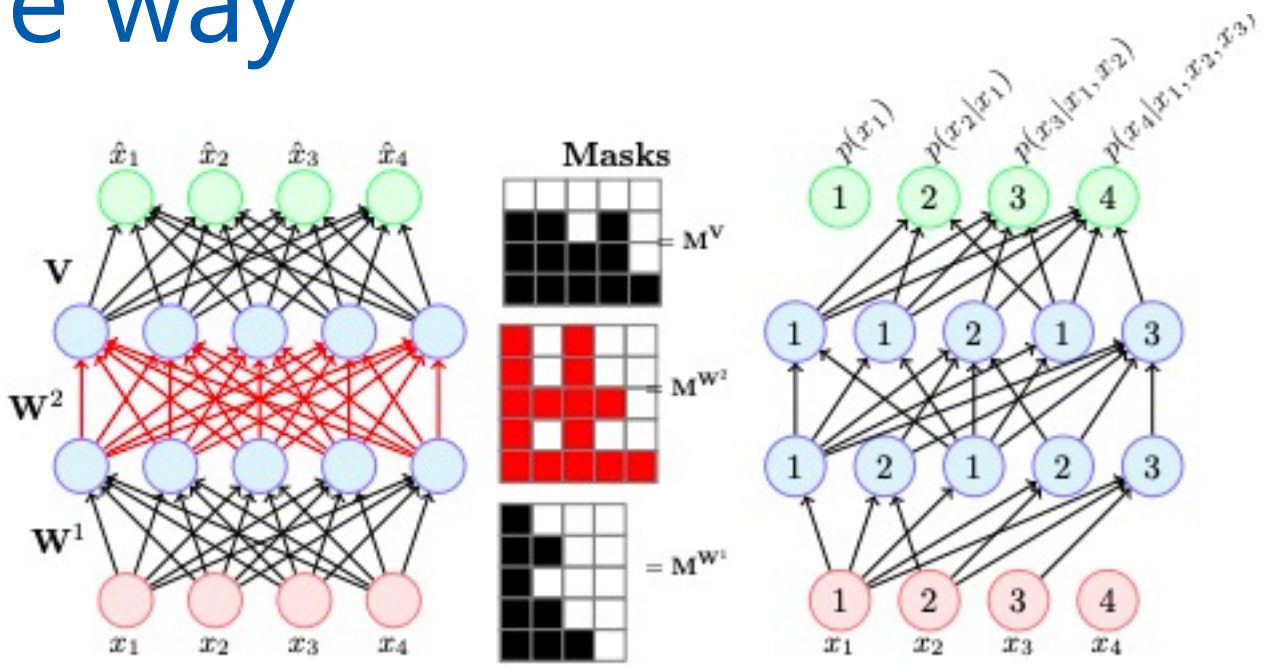
- The inner layer must be connected only to the number of input shown in cirlce.

Masking the way



- The output layer must be connected only to the inner layers that use appropriate inputs.

Masking the way

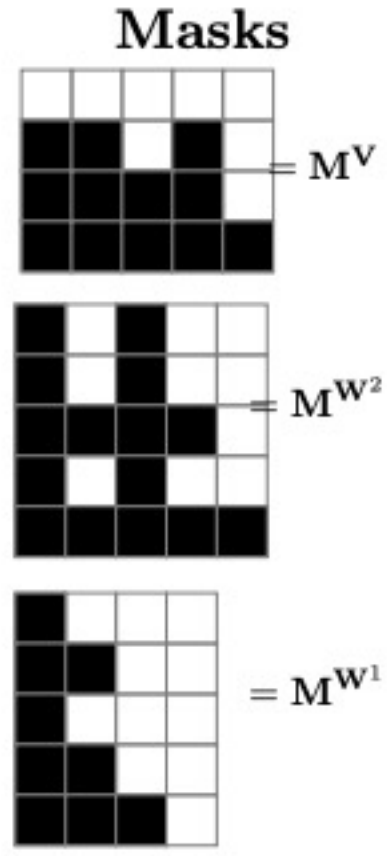


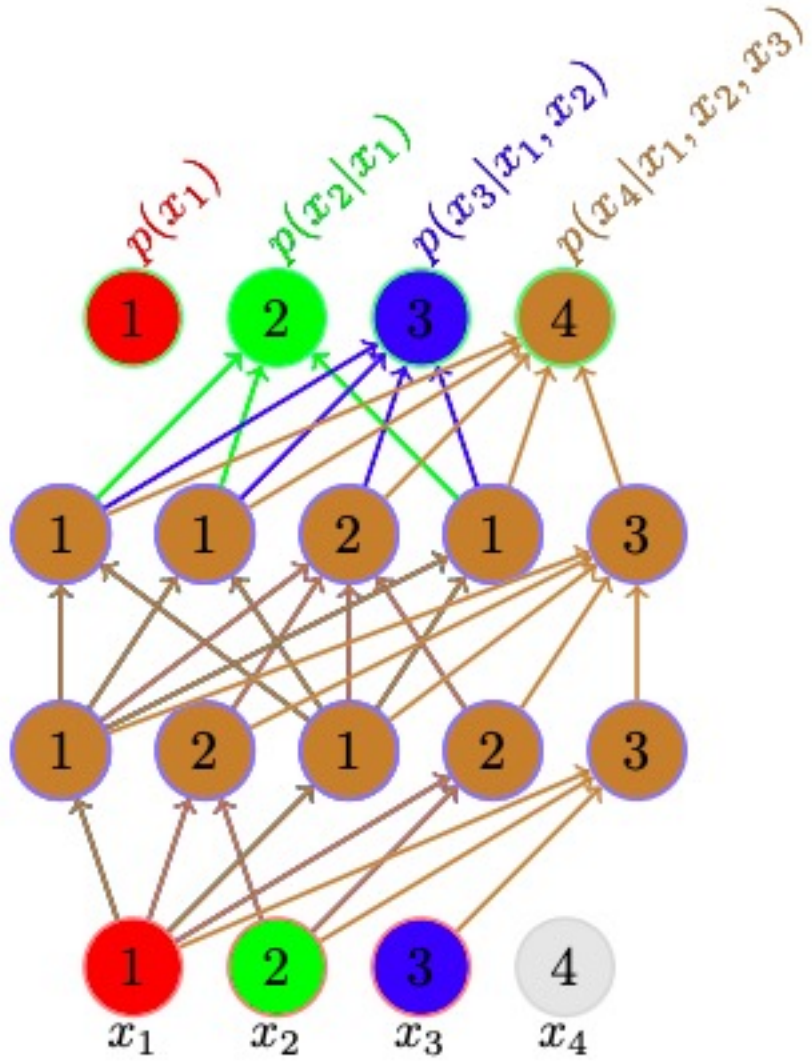
For example we can apply the following mask at layer 2

$$\begin{bmatrix} W^2_{11} & W^2_{12} & W^2_{13} & W^2_{14} & W^2_{15} \\ W^2_{21} & W^2_{22} & W^2_{23} & W^2_{24} & W^2_{25} \\ W^2_{31} & W^2_{32} & W^2_{33} & W^2_{34} & W^2_{35} \\ W^2_{41} & W^2_{42} & W^2_{43} & W^2_{44} & W^2_{45} \\ W^2_{51} & W^2_{52} & W^2_{53} & W^2_{54} & W^2_{55} \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Masking the way

- The objective function is a sum of cross entropies.
- The network can be trained using backpropagation
- Errors will only be propagated along the active (unmasked) connections (similar to what happens in dropout).





MADE: results

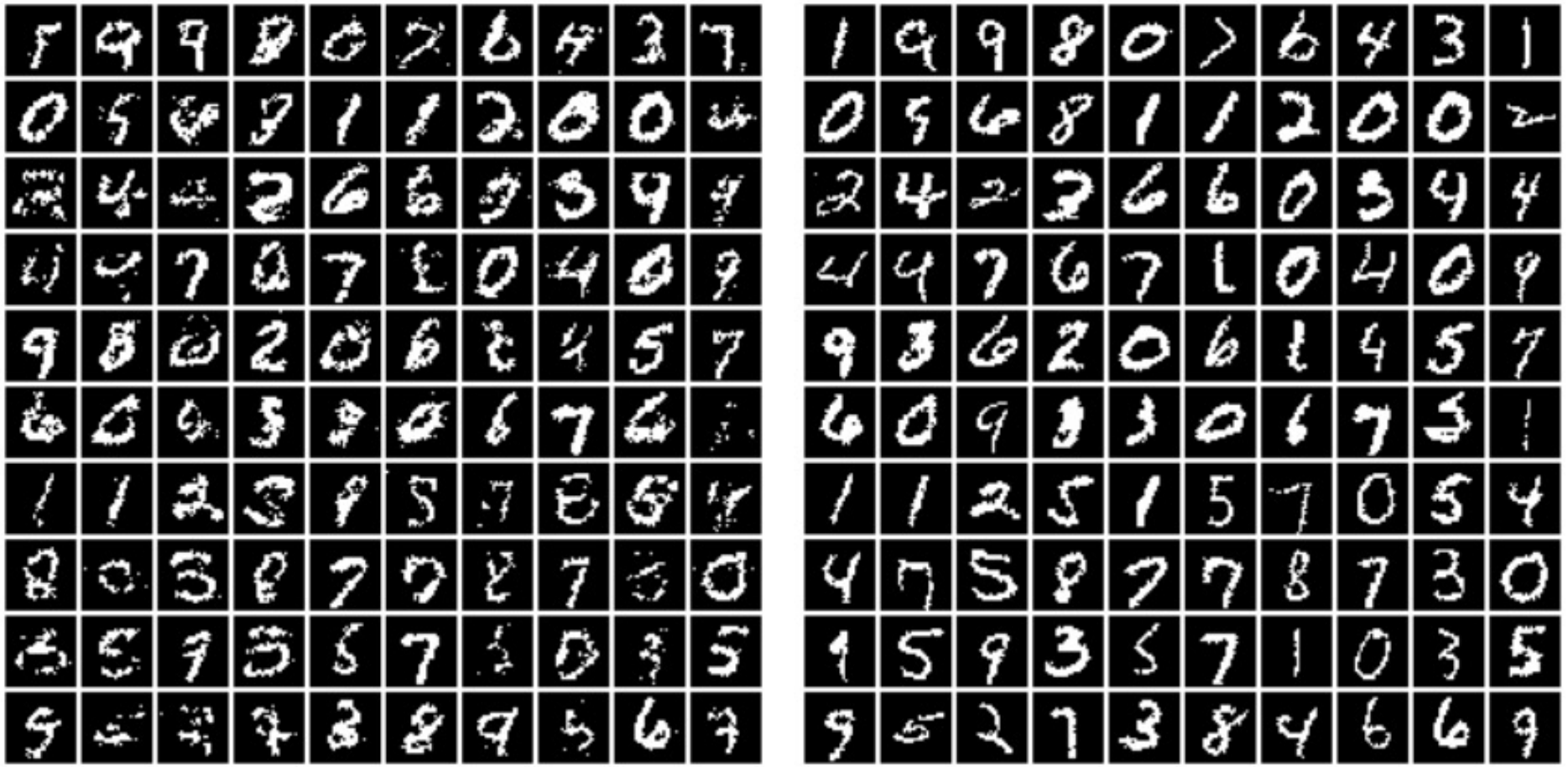
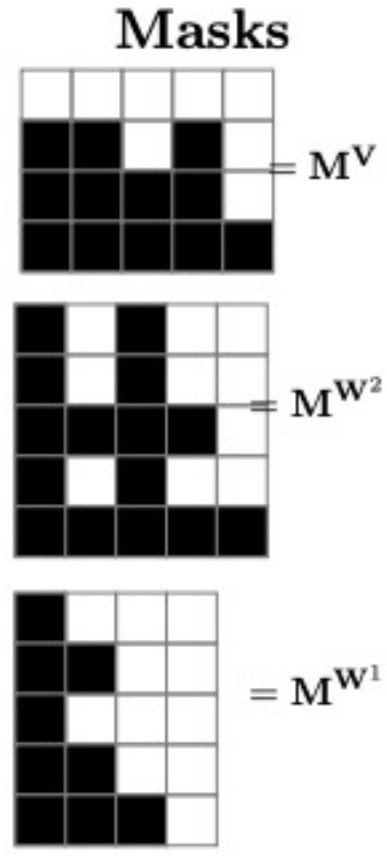
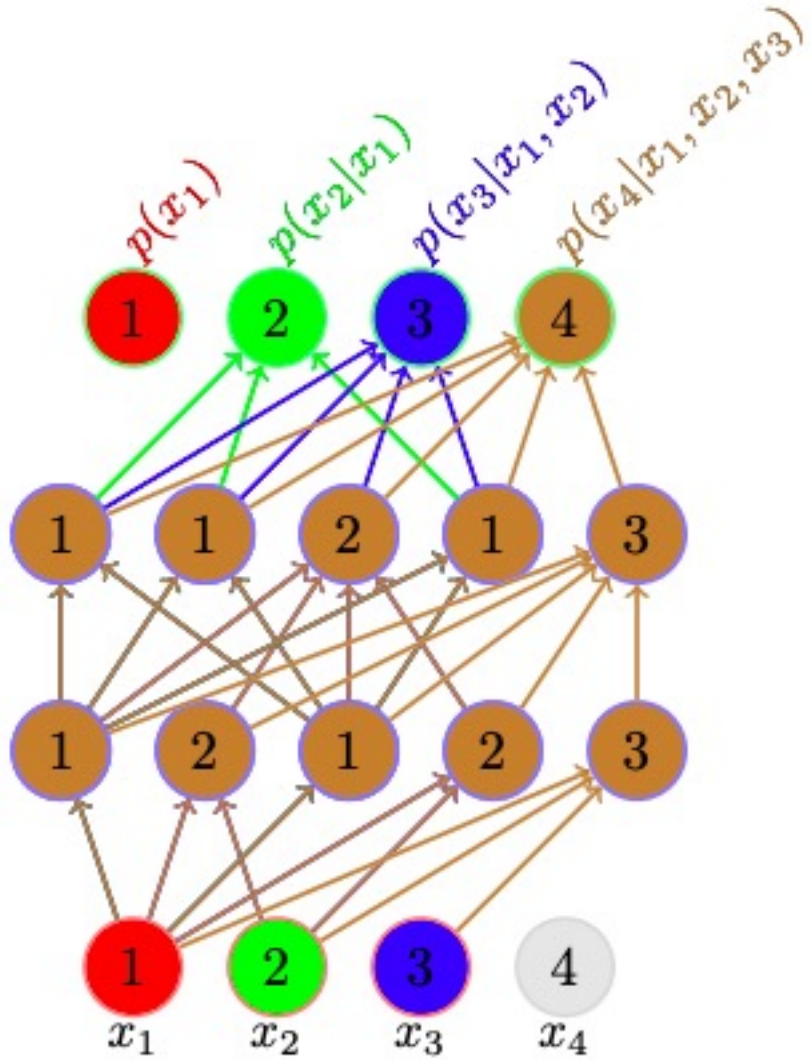


Figure 3. Left: Samples from a 2 hidden layer MADE. Right: Nearest neighbour in binarized MNIST.

MADE: discussion

- Can use deep NNs naturally.
- Ordering can be addressed in the same way as in NADE-k.
- Only one forward pass to evaluate likelihood.
- D passes to sample.

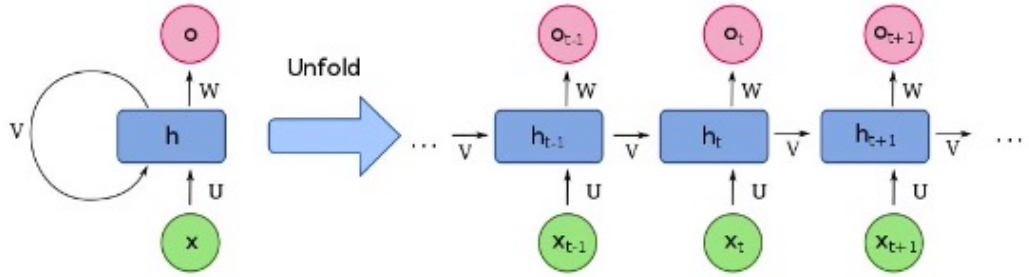




RNN based generation

Motivation and idea

- Autoregressive models have problems with large dimensions due “tail” of previous dimension information, should we just keep summary?

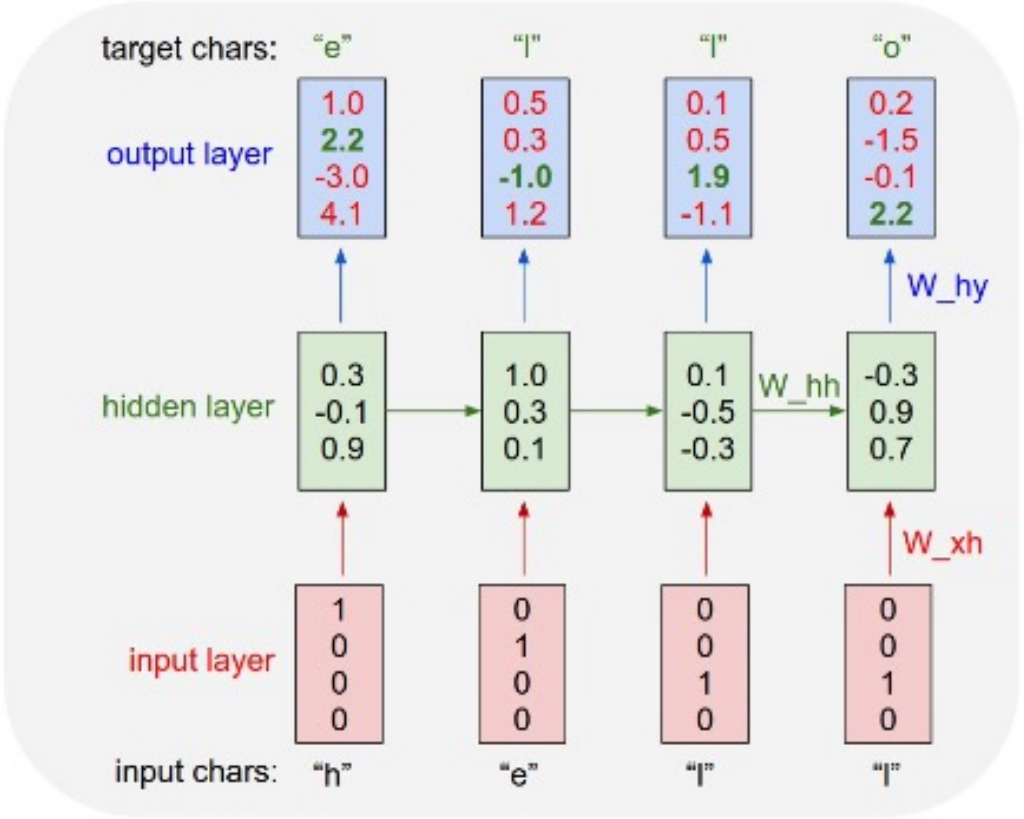


The update rule will thus be: $h_{t+1} = \tanh(W_{hh}h_t + W_{xh}x_{t+1})$.

The prediction: $o_{t+1} = W_{hy}h_{t+1}$.

- hidden layer h_t is a summary of the inputs seen till time t;
- output layer o_{t1} specifies parameters for conditional $p(x_t|x_{1:t1})$;
- Parameterized by b_0 (initialization), and matrices W_{hh}, W_{xh}, W_{hy}. Constant number of parameters w.r.t n.

Simple RNN for generation



- $xi \in \{h, e, l, o\}$. One can use one-hot encoding. We want to build "Hello".
- Autoregressive rule:

$$p(x = hello) = p(x_1 = h)p(x_2 = e|x_1 = h)p(x_3 = l|x_1 = h, x_2 = e) \dots p(x_5 = o|x_1 = h, x_2 = e, x_3 = l, x_4 = l).$$

- $p(x_2 = e|x_1 = h) = softmax(o_1);$

$$o_1 = W_{hy}h_1;$$

$$h_1 = tanh(W_{hh}h_0 + W_{xh}x_1).$$

A. Karpathy's blog

Problem of Long-Term Dependencies

- When dealing with longer sequences there might be some problems. Imagine we try to predict the last word of the sentence (and we trained on words:

the clouds are in the sky

- The RNN works perfectly.
- The longer sequence, however, might suffer from the problems with long-term memory:

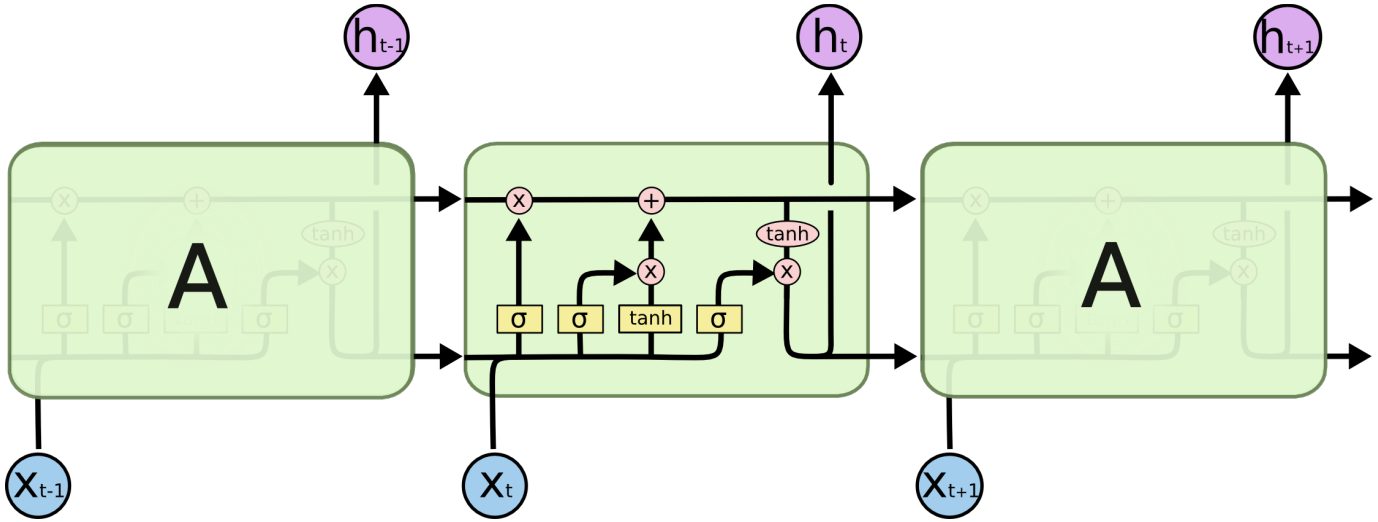
I grew up in France… I speak fluent French

- Some problems can be obtained due to forgetfulness of the RNN.

Y. Bengio et al. Learning long-term dependencies with gradient descent is difficult, IEEE-TNN 1994

Idea: use LSTM cells

- Tell RNN what features you might find important.



C. Olah's blog

Generation of Characters

Addition of LSTM layers creates a possibility to generate long texts.

Train 3-layer RNN with 512 hidden nodes on all the works of Shakespeare.
Then sample from the model:

> KING LEAR: O, if you were a feeble sight, the courtesy of your law,
> Your sight and several breath, will wear the gods
> With his heads, and my hands are wonder'd at the deeds,
> So drop upon your lordship's head, and your opinion
> Shall be against your honour.

Note: generation happens **character by character**. Needs to learn valid words, grammar, punctuation, etc.

Generation of LaTeX Markup

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{étale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

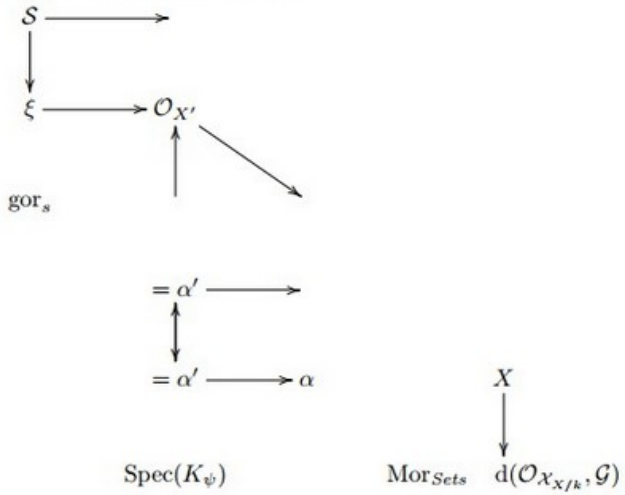
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-}1(\mathcal{O}_{X_{étale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

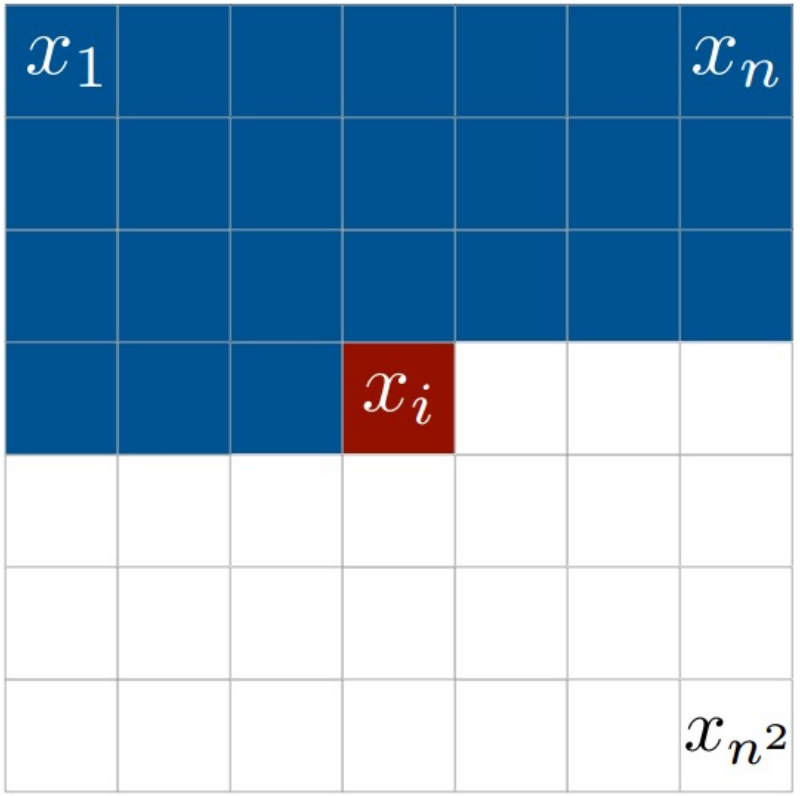
If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

RNN summary

- Pros:
  - Can be applied to sequences of arbitrary length.
  - Very general.
  - Generalized for long sequences with LSTM.
- Cons:
  - Still requires ordering.
  - Sequential likelihood evaluation (very slow for training).
  - Sequential generation (unavoidable in an autoregressive model).
  - Can be difficult to train (vanishing/exploding gradients).

pixelRNN

- The idea can also be applied to generate images. We can model images pixel by pixel using raster scan order.
- We need to make a conditional probability based on all colors of previous pixels and also RGB direction of current pixel.

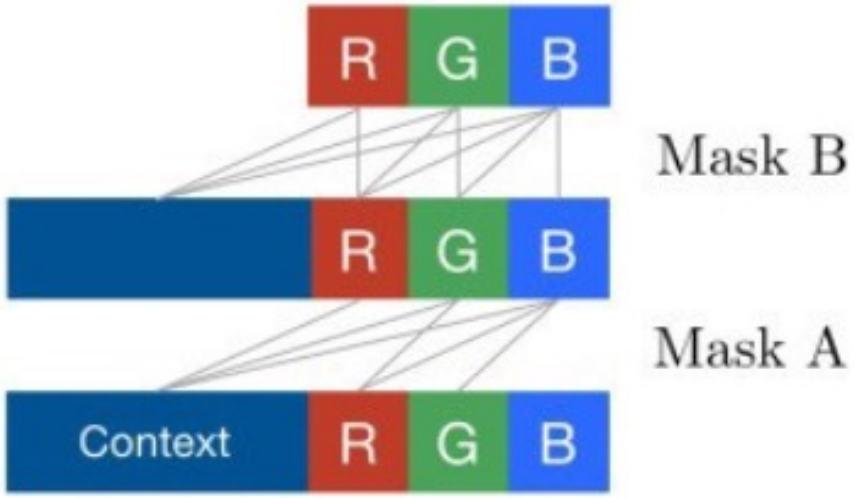


A. van den Oord et al. Pixel Recurrent Neural Networks, ICML 2016

Masked Convolutions

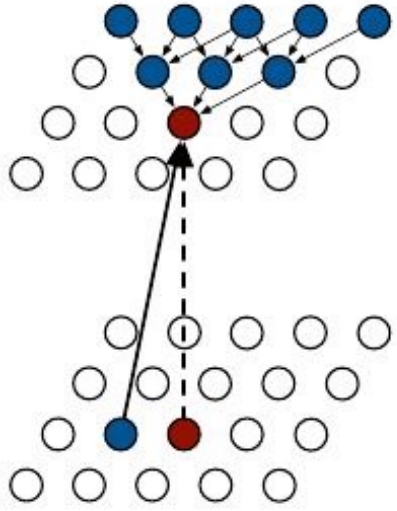
We use two types of masks :

- Type A : this mask is only applied to the first convolutional layer and restricts connections to those colors in current pixels that have already been predicted.
- Type B : this mask is applied to other layers and allows connections to predicted colors in the current pixels

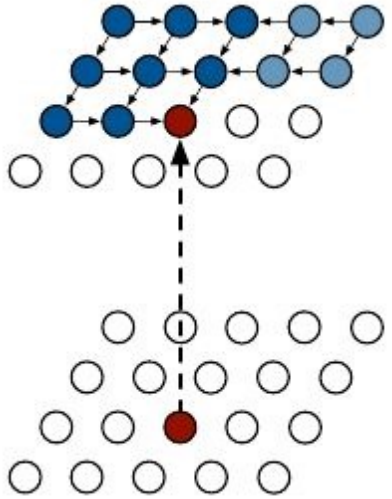


LSTM for PixelRNN

We can use two types of spatial LSTMs that concentrate on different structure



Row LSTM



Diagonal BiLSTM

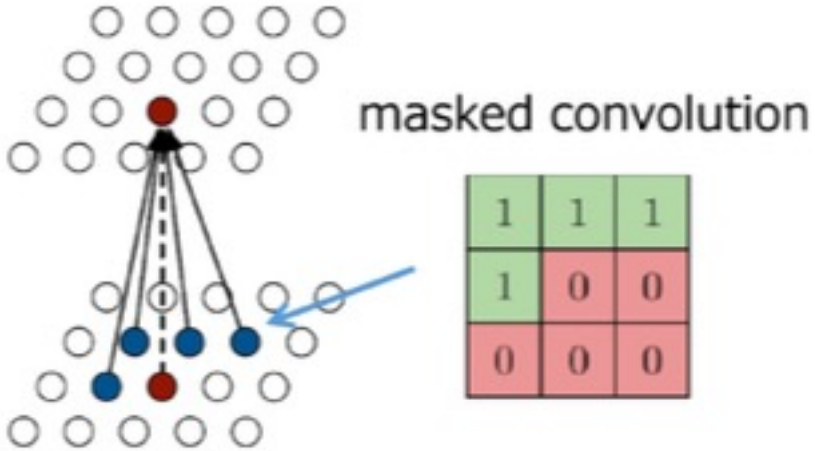
PixelRNN: results

Figure 1. Image completions sampled from a PixelRNN.

- The evaluation is very slow.
- Basic features of images are caught.

pixelCNN

- Convolutions are natural for image data and easy to parallelize on modern hardware.
- We can use convolutional architecture to predict next pixel given context (a neighborhood of pixels).
- Since masked convolutions preserve raster scan order, need additional masking for colors order.



pixelCNN

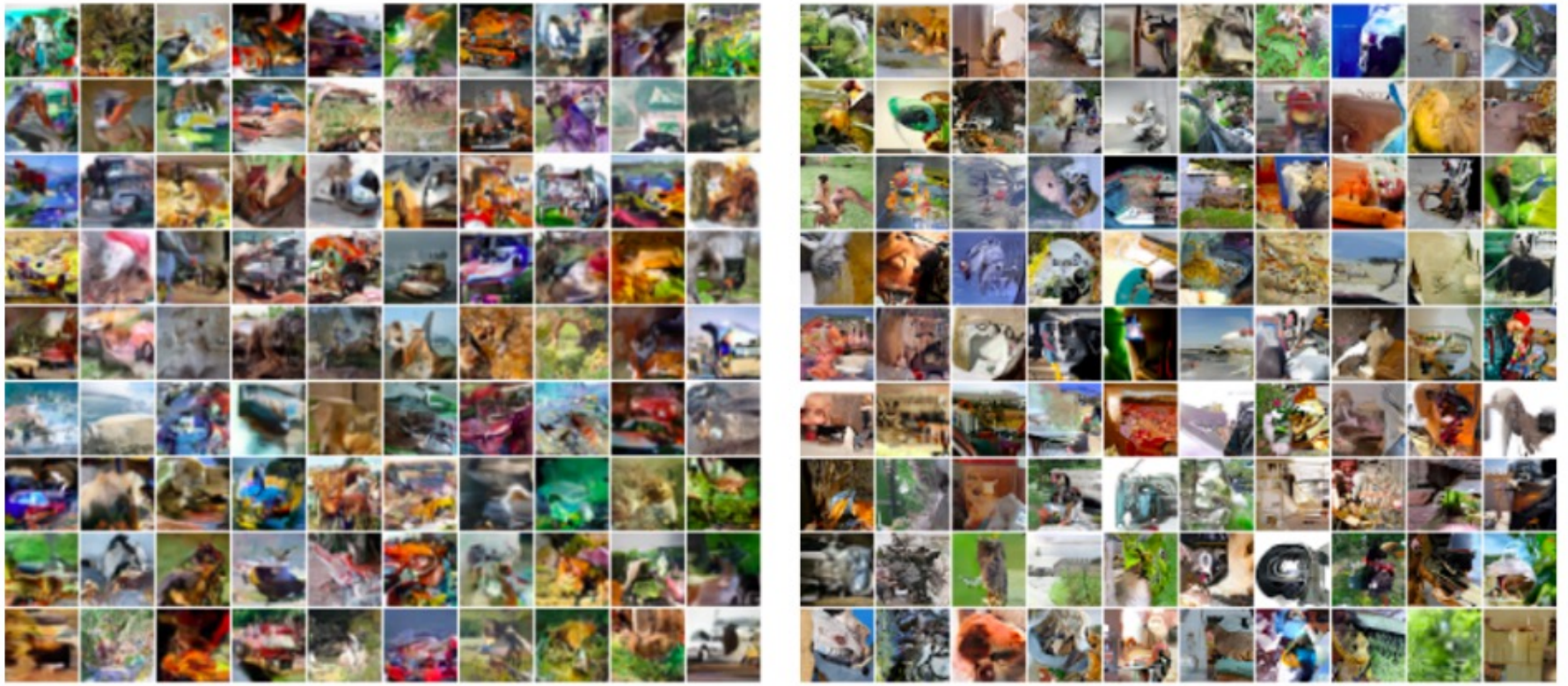


Figure 7. Samples from models trained on CIFAR-10 (left) and ImageNet 32x32 (right) images. In general we can see that the models capture local spatial dependencies relatively well. The ImageNet model seems to be better at capturing more global structures than the CIFAR-10 model. The ImageNet model was larger and trained on much more data, which explains the qualitative difference in samples.

Similar performance to PixelRNN, but much faster.

Summary for Autoregressive models

- Easy to sample from (but slow due to ancestral sampling).
- Easy to compute probability (sometimes even fast).
- No natural way to get representation.