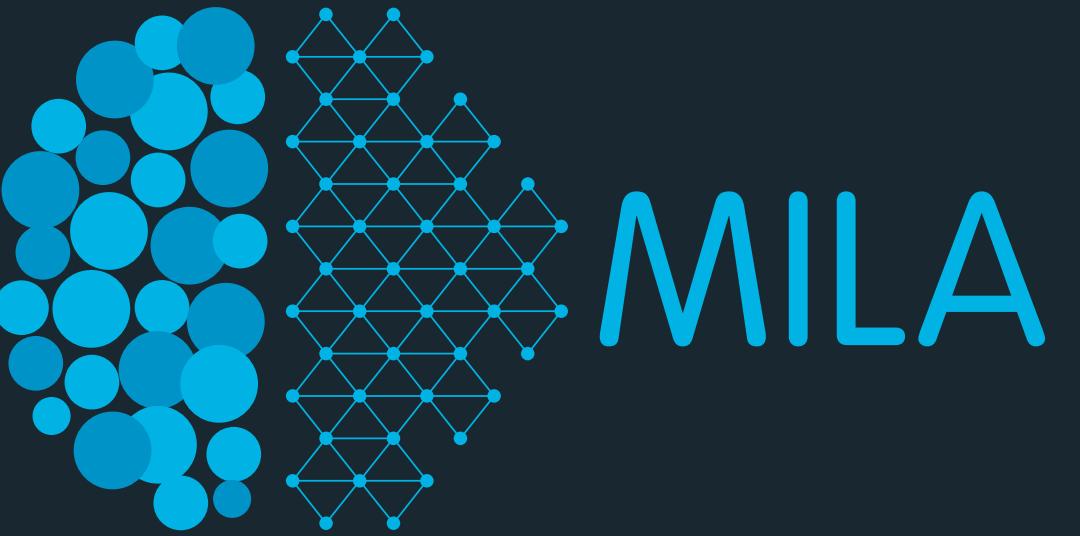


Institut  
des algorithmes  
d'apprentissage  
de Montréal



# Autoregressive Generative Models

Aaron Courville  
Université de Montréal

Slides are drawn from Hugo Larochelle, Vincent Dumoulin and Aaron Courville

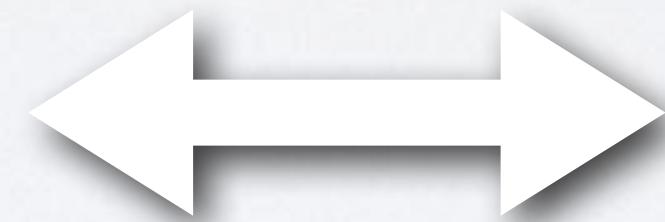
# WHY GENERATIVE MODELS

- Useful learning signal for semi-supervised learning
  - ▶ expect a good model to distinguish between real and fake data



real image

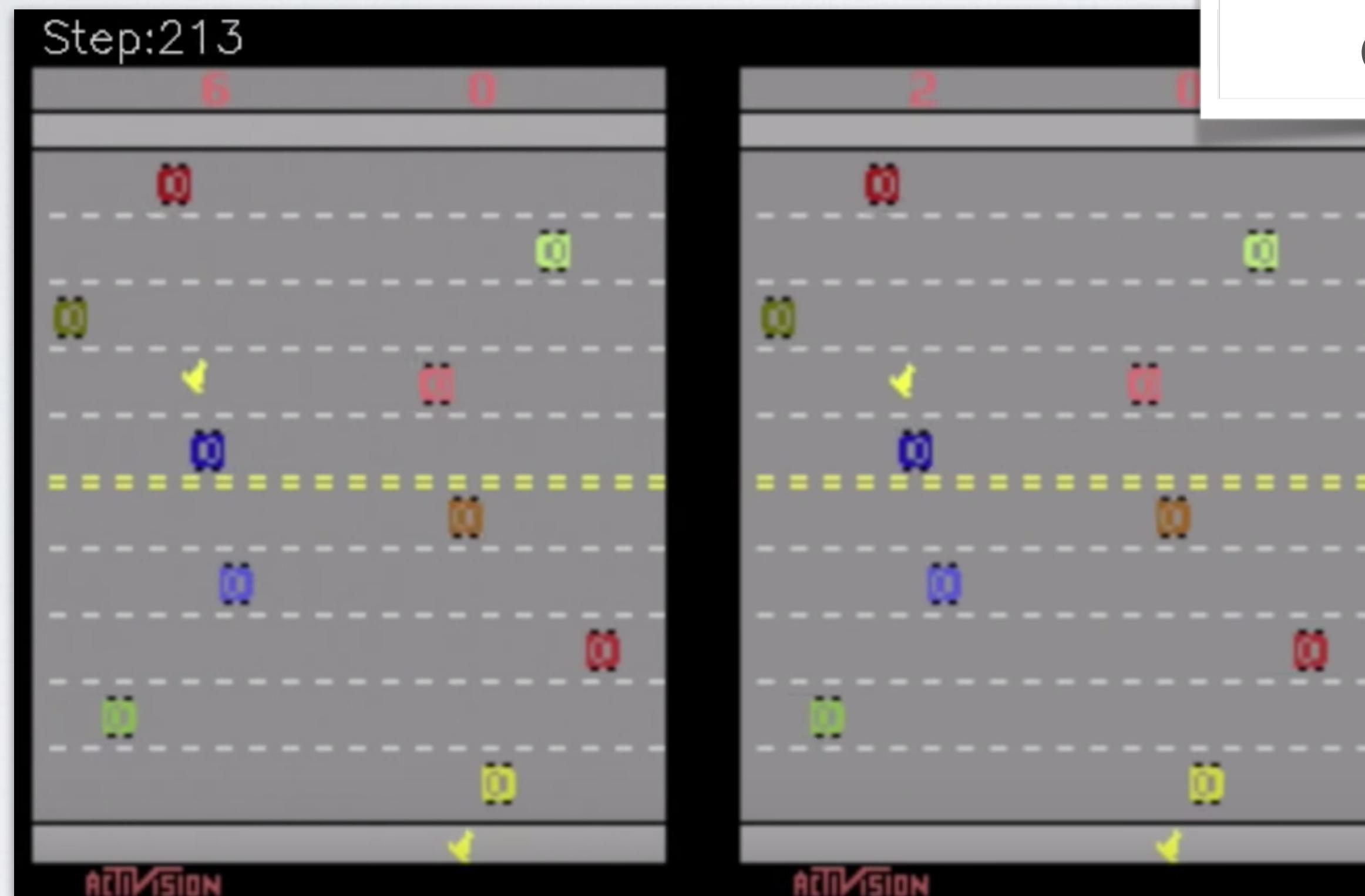
Why is one  
a character  
and not the  
other?



random image

# WHY GENERATIVE MODELS

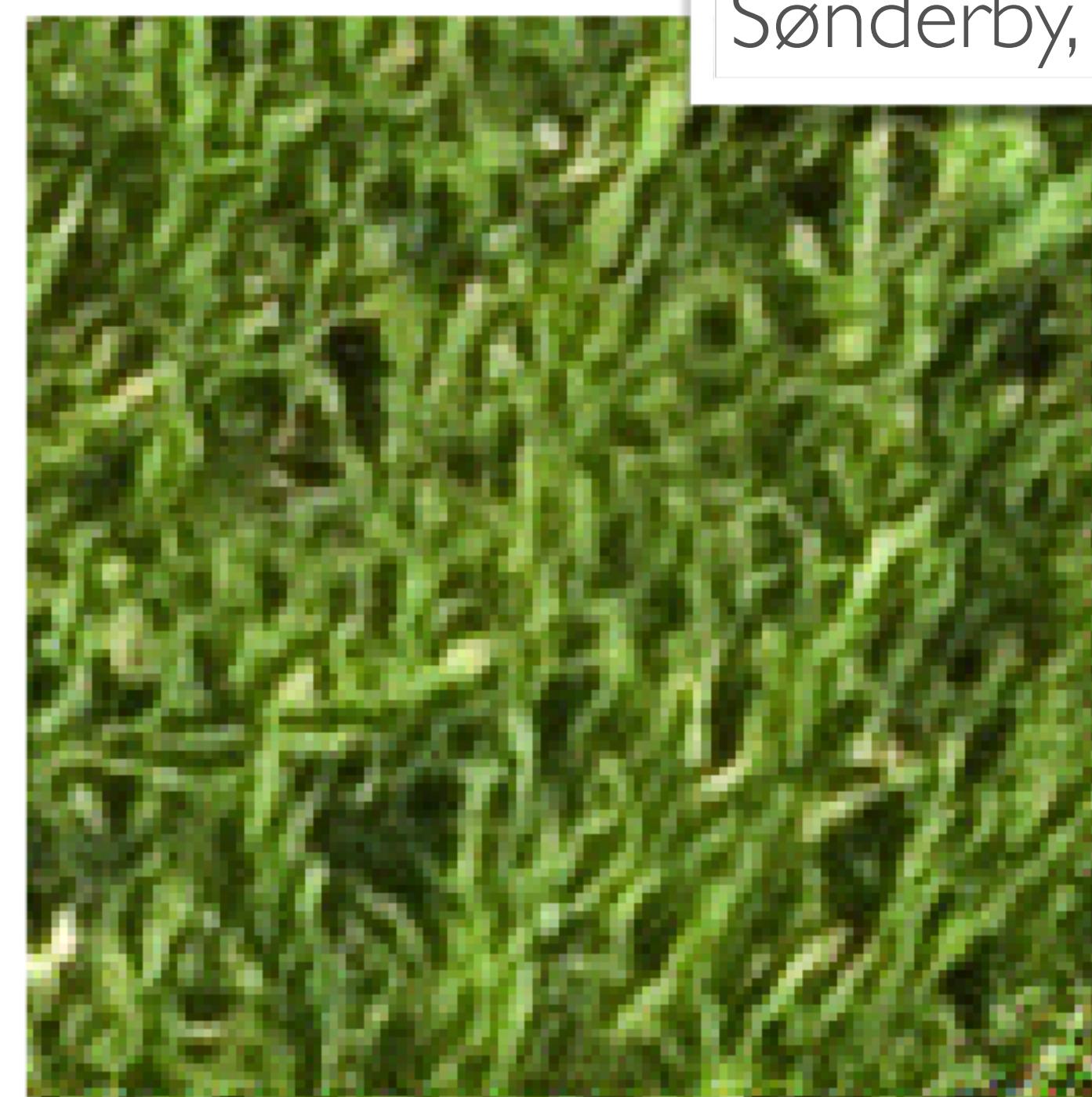
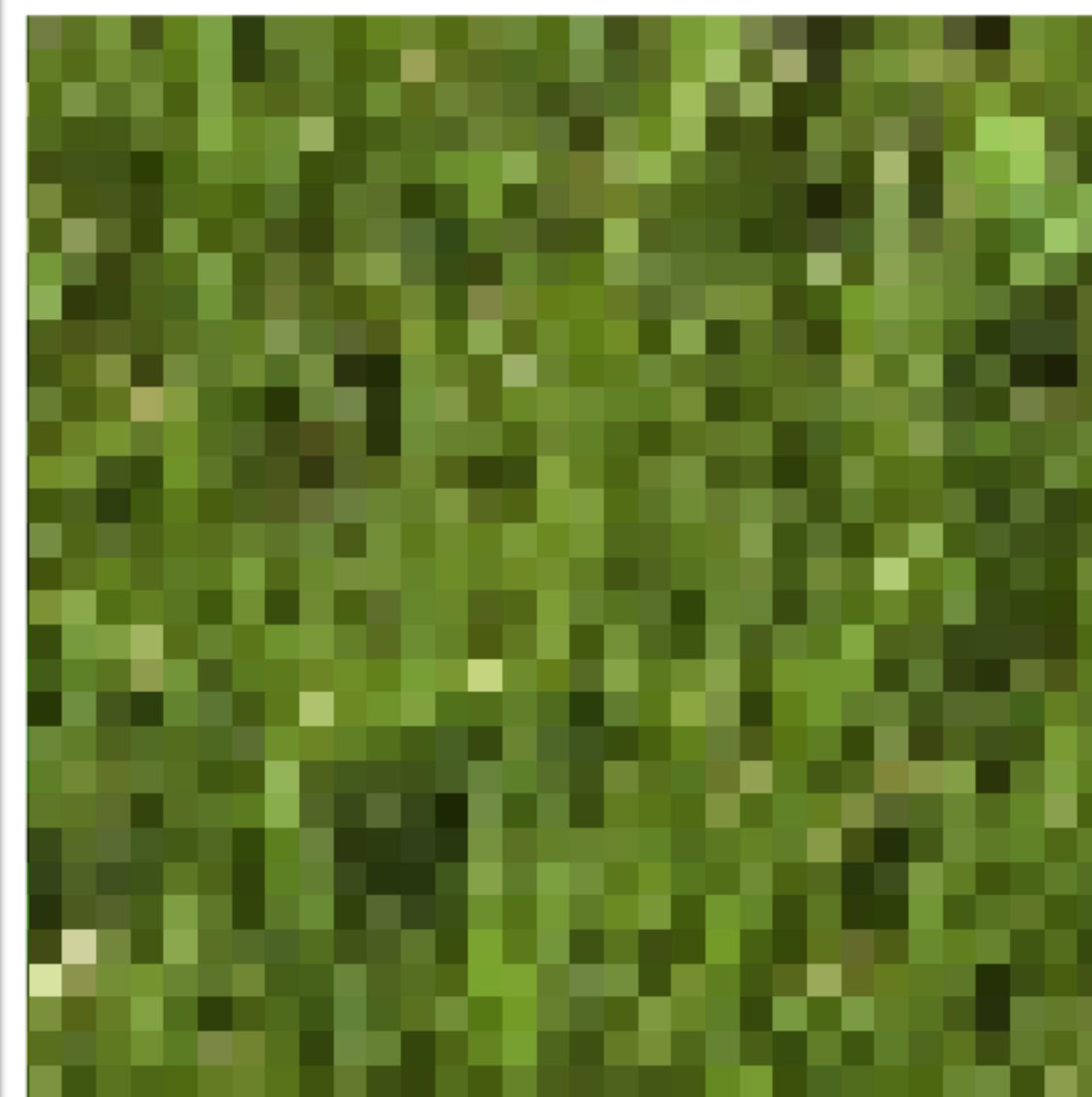
- To synthesize new observations
  - ▶ useful for planning in a visual environment



Action-Conditional Video Prediction  
using Deep Networks in Atari Games  
Oh, Guo, Lee, Singh, Lewis. NIPS 2015

# WHY GENERATIVE MODELS

- As a prior over *real* observations
  - ▶ useful for denoising or super-resolution



Amortised MAP Inference for  
Image Super-resolution

Sønderby, Caballero, Theis, Shi, Huszár. arXiv 2016

# FAMILY OF GENERATIVE MODELS

- **Directed graphical models**

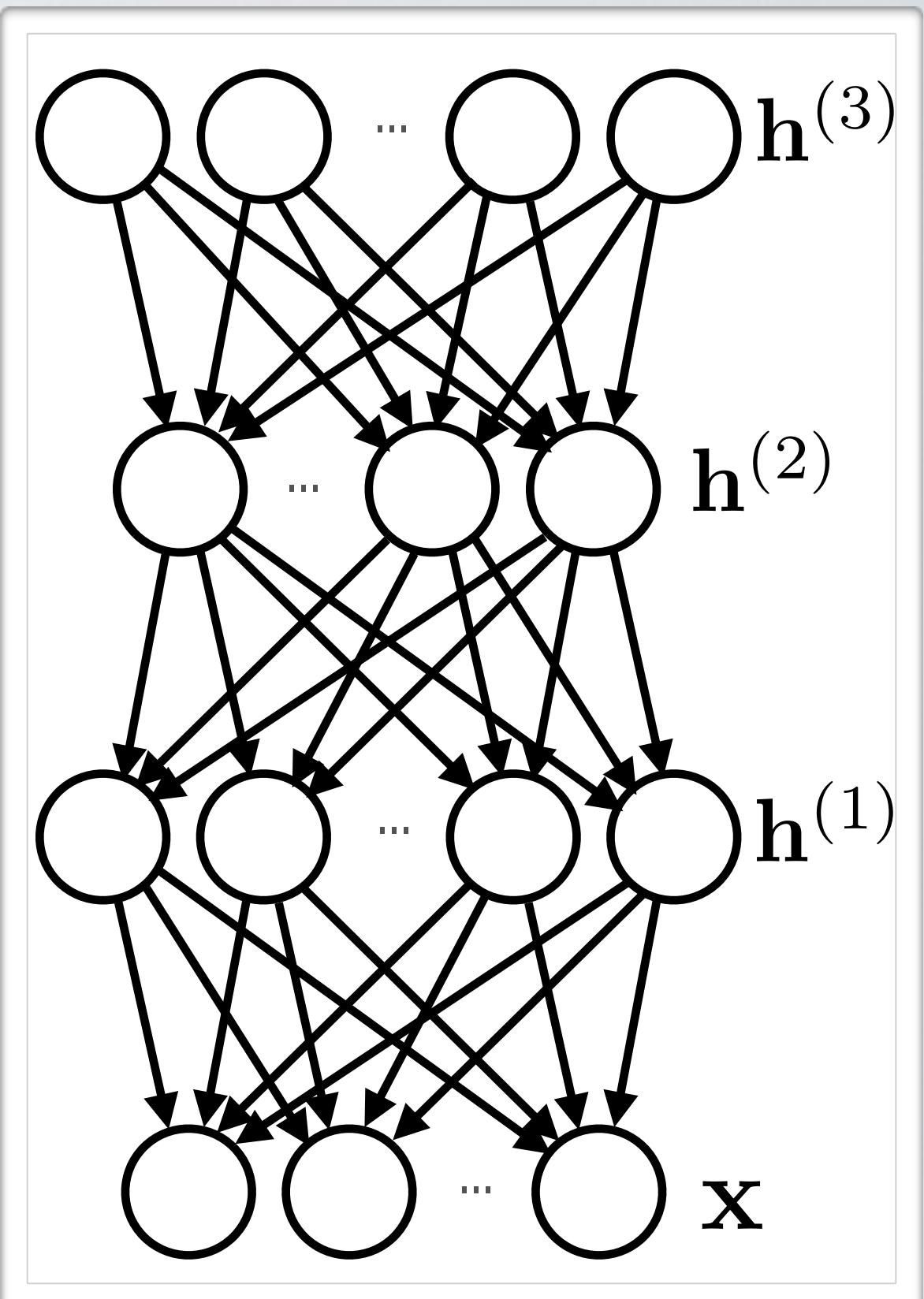
- ▶ define prior over top-most latent representation
- ▶ define conditionals from top latent representation to observation

$$p(\mathbf{x}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = p(\mathbf{x}|\mathbf{h}^{(1)})p(\mathbf{h}^{(1)}|\mathbf{h}^{(2)})p(\mathbf{h}^{(2)}|\mathbf{h}^{(3)})p(\mathbf{h}^{(3)})$$

- ▶ examples: variational autoencoders (VAE), generative adversarial networks (GAN), sparse coding, helmholtz machines

- Properties

- ▶ pros: easy to sample from (ancestral sampling)
- ▶ cons:  $p(\mathbf{x})$  is intractable, so hard to train



# FAMILY OF GENERATIVE MODELS

- **Undirected graphical models (Energy-based)**

- ▶ define a joint energy function

$$E(\mathbf{x}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = -\mathbf{x}\mathbf{W}^{(1)}\mathbf{h}^{(1)} - \mathbf{h}^{(2)}\mathbf{W}^{(2)}\mathbf{h}^{(3)} - \mathbf{h}^{(3)}\mathbf{W}^{(3)}\mathbf{h}^{(4)}$$

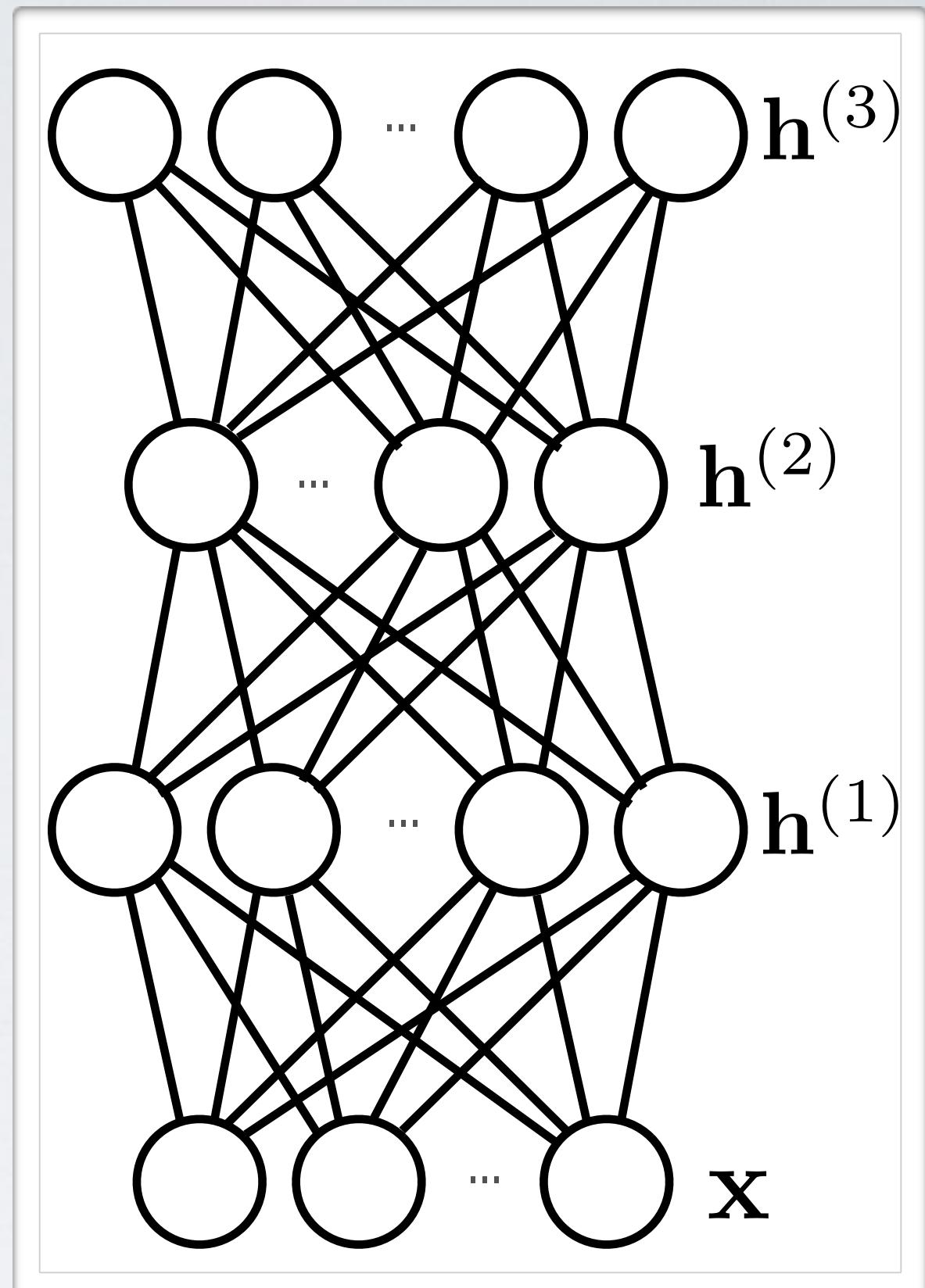
- ▶ exponentiate and normalize

$$p(\mathbf{x}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \exp\left(-E(\mathbf{x}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)})\right) / Z$$

- ▶ examples: deep Boltzmann machines (DBM), deep energy models

- Properties

- ▶ pros: can compute  $p(\mathbf{x})$  up to a multiplicative factor (true for RBMs not general BMs)
- ▶ cons: hard to sample from (MCMC),  $p(\mathbf{x})$  is intractable, so hard to train



# FAMILY OF GENERATIVE MODELS

- **Autoregressive generative models**

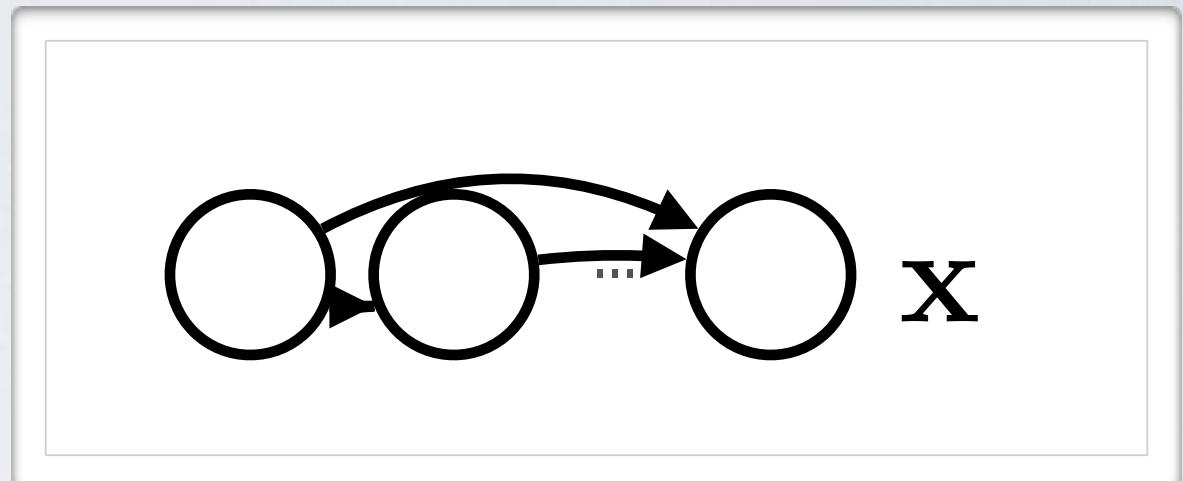
- ▶ choose an ordering of the dimensions in  $\mathbf{x}$
- ▶ define the conditionals in the product rule expression of  $p(\mathbf{x})$

$$p(\mathbf{x}) = \prod_{k=1}^D p(x_k | \mathbf{x}_{<k})$$

- ▶ examples: masked autoencoder distribution estimator (MADE), pixelCNN  
neural autoregressive distribution estimator (NADE), spatial LSTM, pixelRNN

- Properties

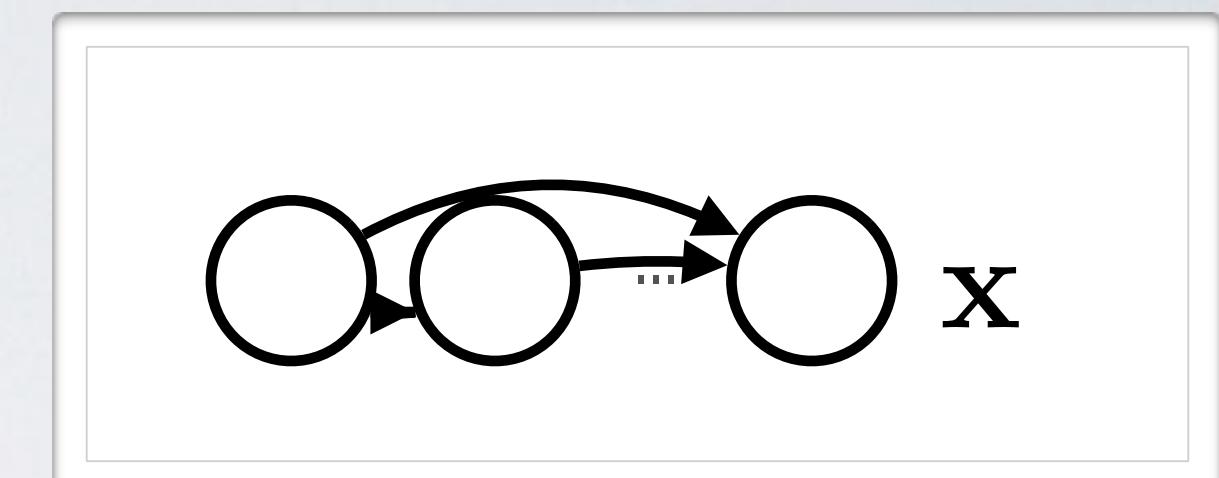
- ▶ pros:  $p(\mathbf{x})$  is tractable, so easy to train, easy to sample (though slower)
- ▶ cons: doesn't have a natural latent representation



# FAMILY OF GENERATIVE MODELS

- **Autoregressive generative models**

- ▶ autoregressive models are well known for sequence data  
(language modeling, time series, etc.)
- ▶ less obviously applicable to arbitrary (non-sequential) observations



- **Little history**

logistic regression for the conditionals (Frey et al., 1996)



neural networks for the conditionals (Bengio and Bengio, 2000)

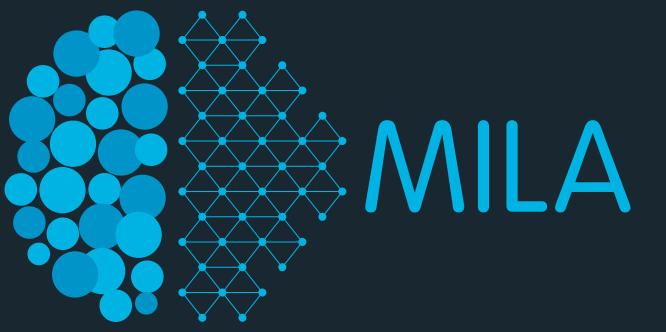


idem, with new weight sharing (NADE) (Larochelle and Murray, Gregor and Lecun, 2011)



Deep NADE, Spatial LSTM, PixelRNN, PixelCNN, WaveNet, Video Pixel Network, etc.

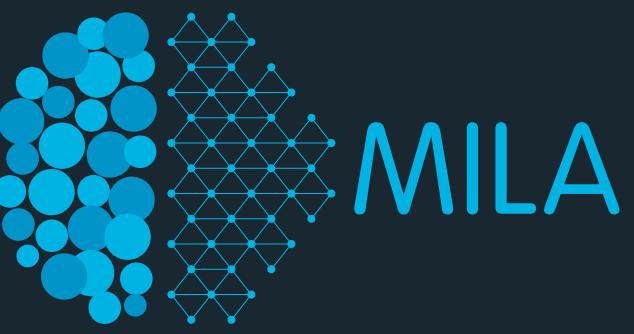
# Autoregressive Generative Models



On the menu:

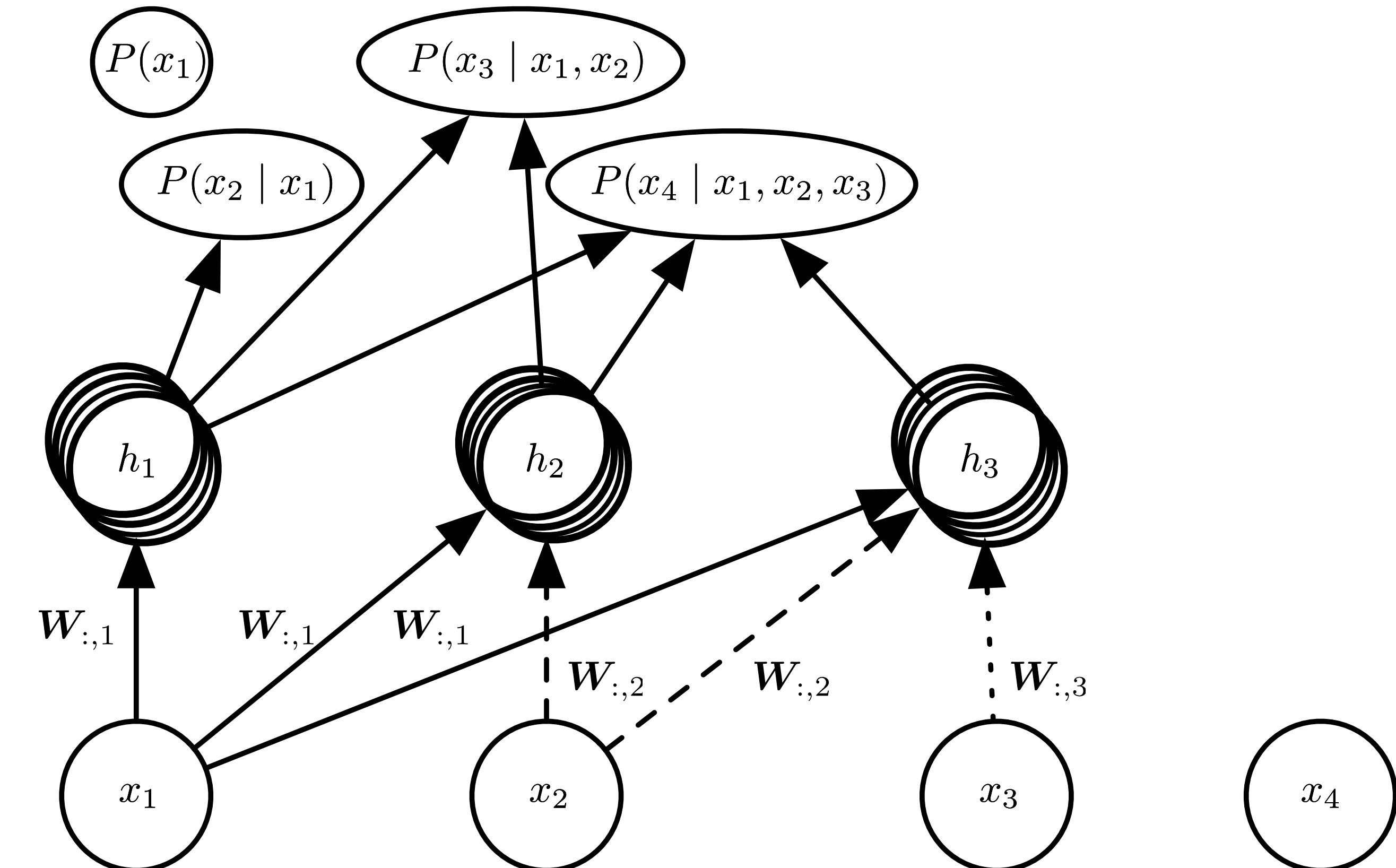
- NADE: Neural Autoregressive Density Estimator
- MADE: Masked Autoencoder for Density Estimator
- PixelCNN Autoregressive CNN
- Extras (we won't discuss)

# NADE (Larochelle and Murray, AISTATS 2011)



- NADE: connectivity is the same as for the original neural auto-regressive network of Bengio and Bengio (NIPS 2000)
- NADE introduces an additional parameter sharing scheme:
  - weights  $\mathbf{W}'_{j,k,i}$  from the  $i$ -th input  $x_i$  to the  $k$ -th element of the  $j$ -th group of hidden unit  $h_k^{(j)}$  ( $j \geq i$ ) are shared among the groups:

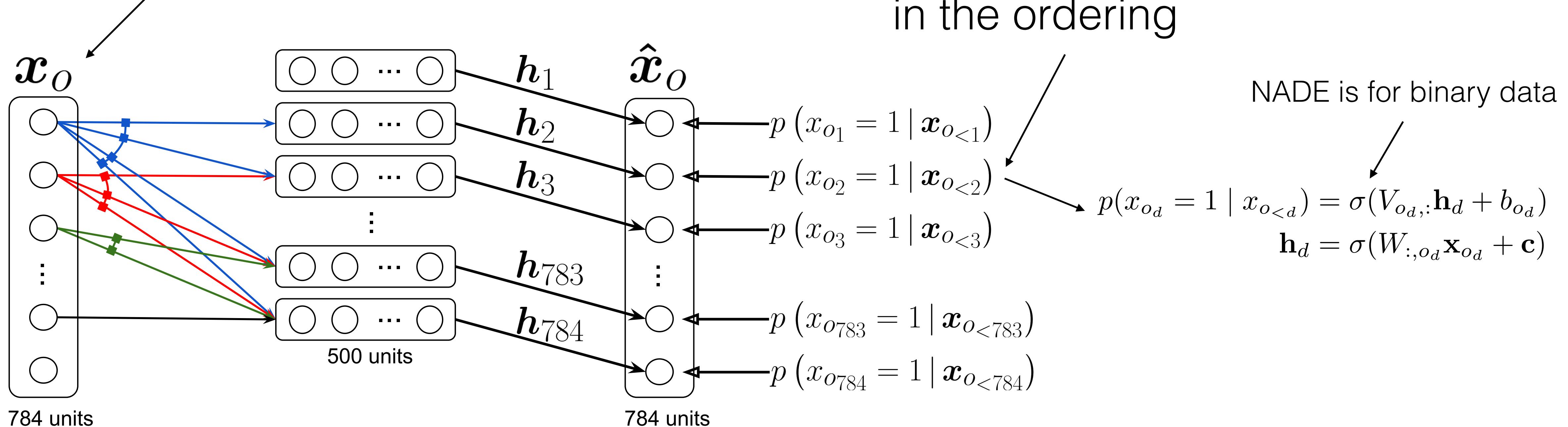
$$\mathbf{W}'_{j,k,i} = \mathbf{W}_{k,i}$$



# Neural autoregressive distribution estimator (NADE)

some ordering of  $\mathbf{x}$

given previous variables  
in the ordering



# Neural autoregressive distribution estimator (NADE)

Negative  
log-likelihood

$$\mathcal{L}(\mathbf{x}) = -\log p(\mathbf{x}) = -\sum_{i=1}^{|\mathbf{x}|} \log p(x_{o_i} \mid x_{o_{<i}})$$

$$p(x_{o_d} = 1 \mid x_{o_{<d}}) = \sigma(V_{o_d,:}\mathbf{h}_d + b_{o_d})$$

$$\mathbf{h}_d = \sigma(W_{:,o_d}\mathbf{x}_{o_d} + \mathbf{c})$$

# NADE Training

NADE training is done using  
**“teacher forcing”**

- Ground truth values of the pixels are used for conditioning when predicting subsequent values

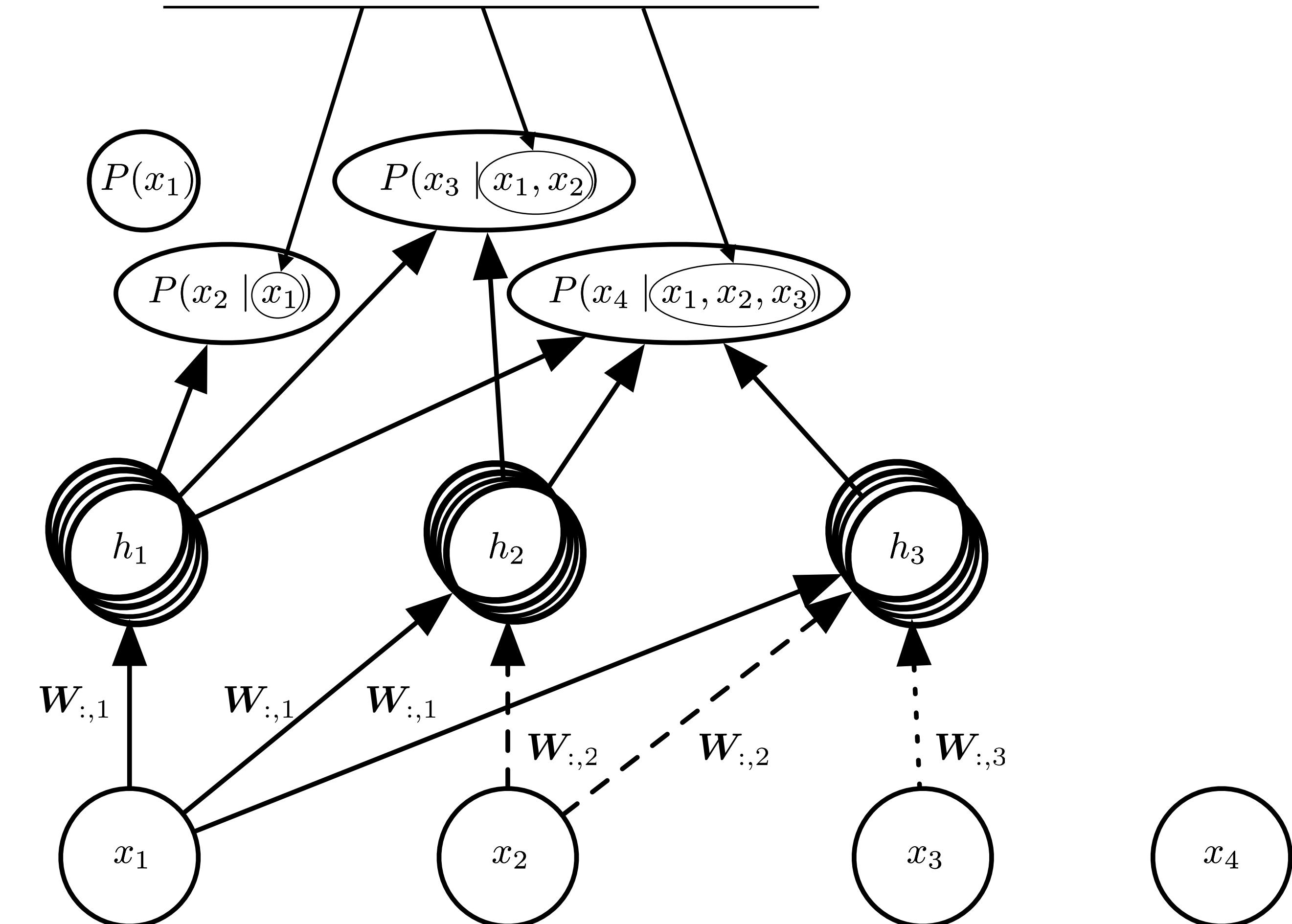
Negative log-likelihood

$$\mathcal{L}(\mathbf{x}) = -\log p(\mathbf{x}) = -\sum_{i=1}^{|\mathbf{x}|} \log p(x_{o_i} | x_{o_{<i}})$$

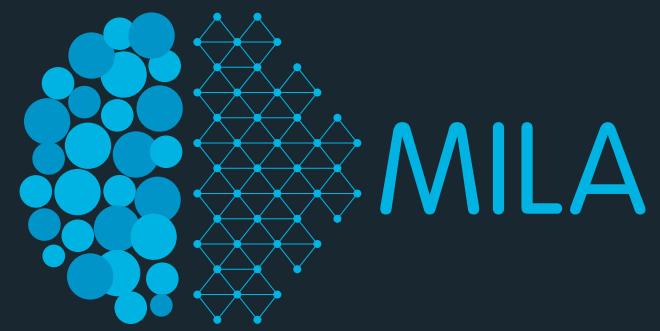
$$p(x_{o_d} = 1 | x_{o_{<d}}) = \sigma(V_{o_d,:}\mathbf{h}_d + b_{o_d})$$

$$\mathbf{h}_d = \sigma(W_{:,o_d}\mathbf{x}_{o_d} + \mathbf{c})$$

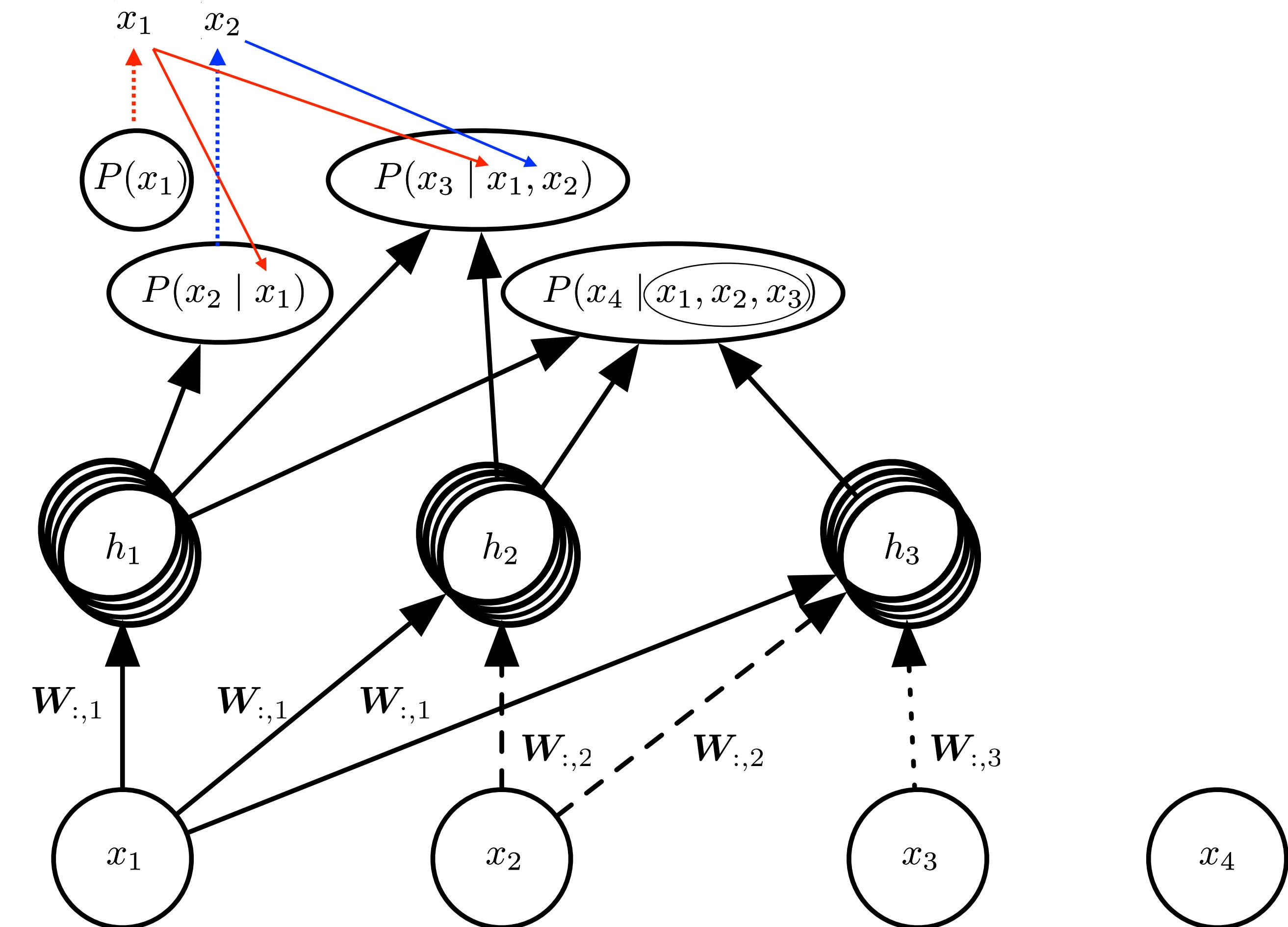
Ground truth values  
(i.e. from training set)



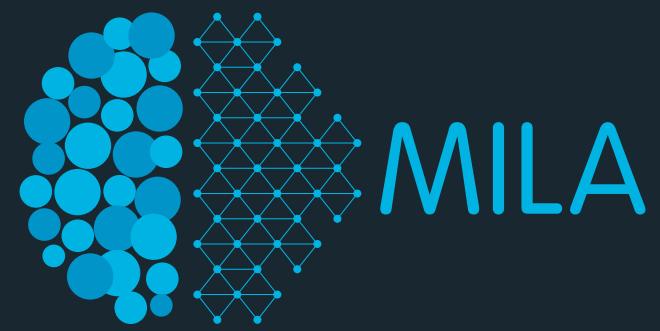
# NADE Generation



NADE generation (at “test time”) is done by conditioning on values previously sampled from the model.



# NADE (Larochelle and Murray, AISTATS 2011)



- NADE for images: must choose an arbitrary ordering over pixels for conditional sampling.

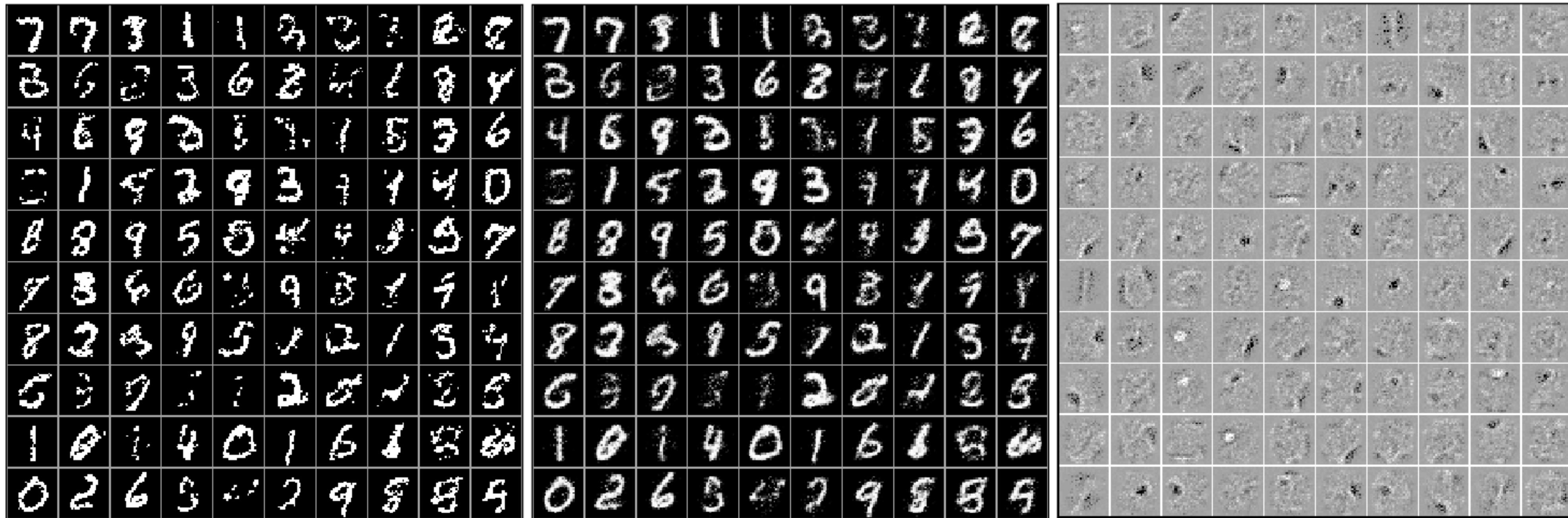


Figure 2: **(Left)**: samples from NADE trained on a binary version of MNIST. **(Middle)**: probabilities from which each pixel was sampled. **(Right)**: visualization of some of the rows of  $\mathbf{W}$ . This figure is better seen on a computer screen.

# Neural autoregressive distribution estimator (NADE)

## Extensions

- Real-valued NADE (RNADE): conditionals are modelled by a mixture of gaussians.
- Orderless and deep NADE (DeepNADE): a single deep neural network is trained to assign a conditional distribution to any variable given any subset of the others.
- Convolutional NADE (ConvNADE)

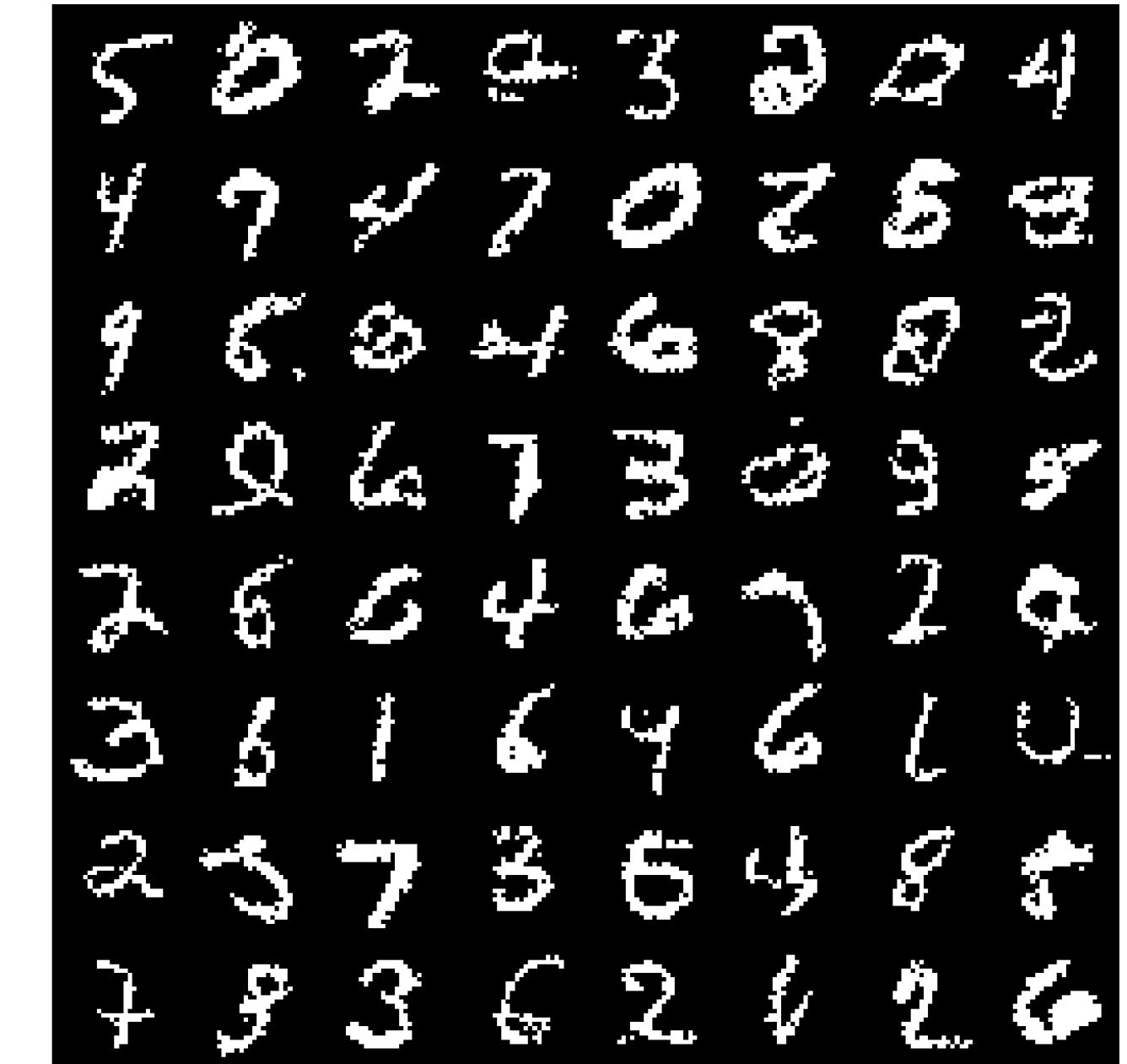
# Neural autoregressive distribution estimator (NADE)



Binarized MNIST  
samples (NADE)



Binarized MNIST  
samples (DeepNADE)



Binarized MNIST  
samples (ConvNADE)

*Uria, Benigno, et al. "Neural Autoregressive Distribution Estimation."  
Journal of Machine Learning Research 17.205 (2016): 1-37.*

- **MADE:** Masked Autoencoder for Density Estimator
- **Question:** How do you construct an autoregressive autoencoder?
  - Specifically: How to modify the autoencoder so as to satisfy the autoregressive property: where prediction of  $x_d$  depends only on the preceding inputs  $x_{<d}$ , *relative to some (arbitrary) ordering.*
  - I.e. there must be no computational path between output unit  $x_d$  and any of the input units  $x_d, \dots, x_D$ , *again relative to some ordering.*
  - I.e. For each of these paths, at least one connection in the weight matrix must be 0.

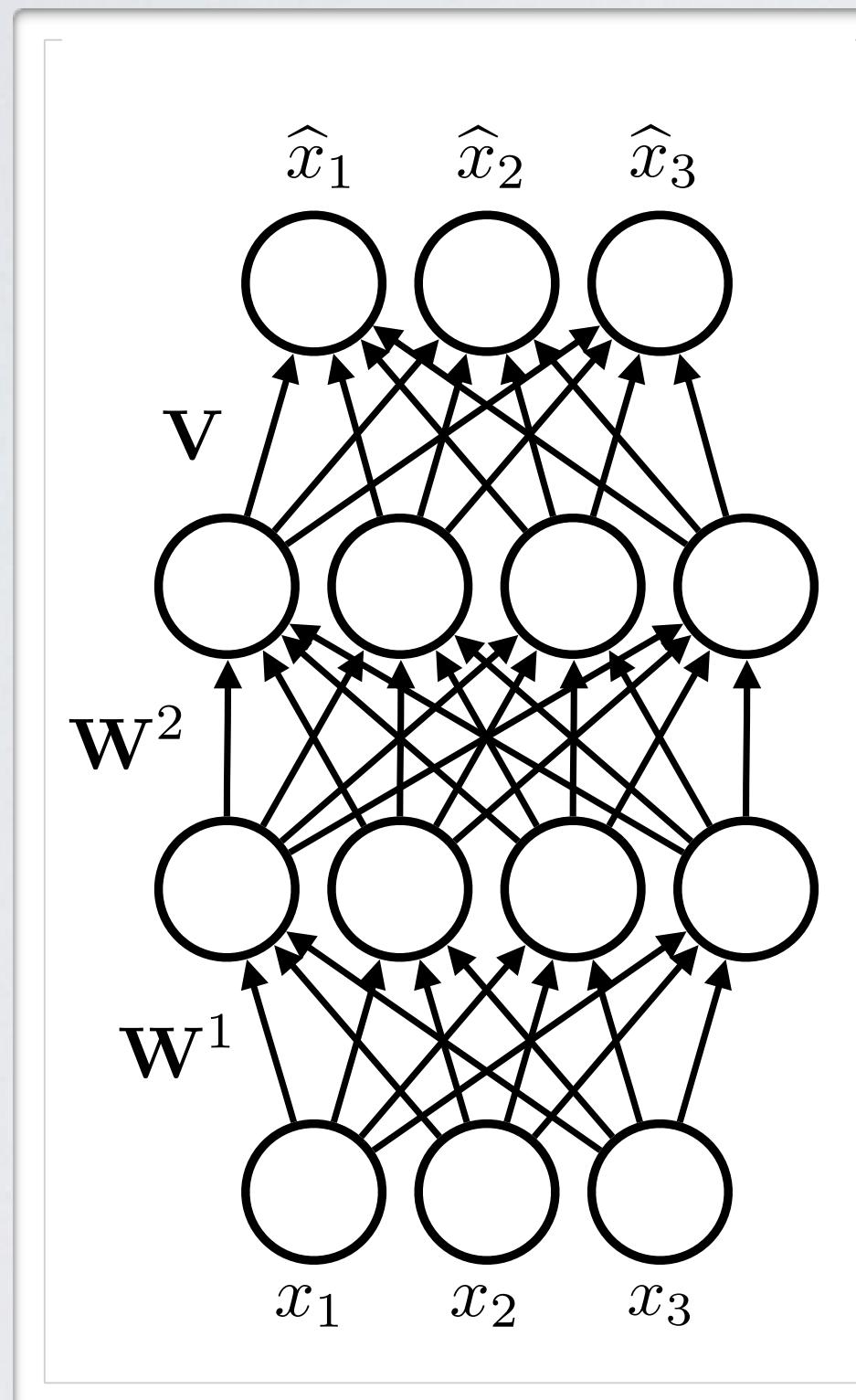
- **Question:** How do you construct an autoregressive autoencoder?
- Convenient way of zeroing connections is to elementwise-multiply each matrix by a binary mask matrix  $\mathbf{M}$ , whose entries that are set to 0 correspond to the connections we wish to remove.
- For a single hidden layer autoencoder:

$$\begin{aligned} \mathbf{h}(\mathbf{x}) &= \mathbf{g}(\mathbf{b} + (\mathbf{W} \odot \mathbf{M}^{\mathbf{W}})\mathbf{x}) \\ \hat{\mathbf{x}} &= \text{sigm}(\mathbf{c} + (\mathbf{V} \odot \mathbf{M}^{\mathbf{V}})\mathbf{h}(\mathbf{x})) \end{aligned}$$

# MASKED AUTOENCODER DISTRIBUTION ESTIMATION

**Topics:** MADE (Germain et al. 2015)

- Idea: constrain output so can be used for the conditionals  $p(x_k | \mathbf{x}_{<k})$

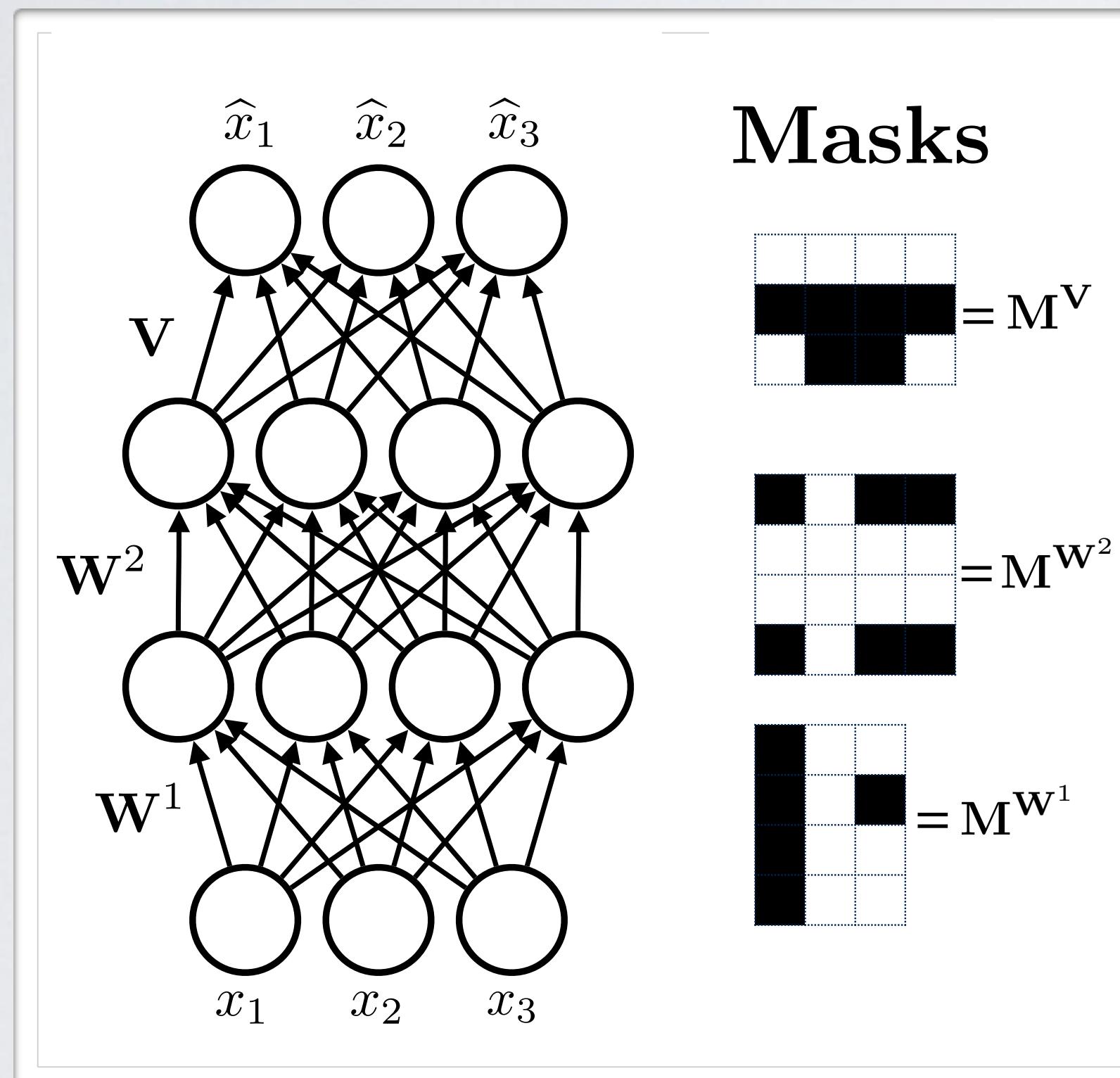


- Generalization of the work from Bengio and Bengio (2000)

# MASKED AUTOENCODER DISTRIBUTION ESTIMATION

**Topics:** MADE (Germain et al. 2015)

- Idea: constrain output so can be used for the conditionals  $p(x_k | \mathbf{x}_{<k})$

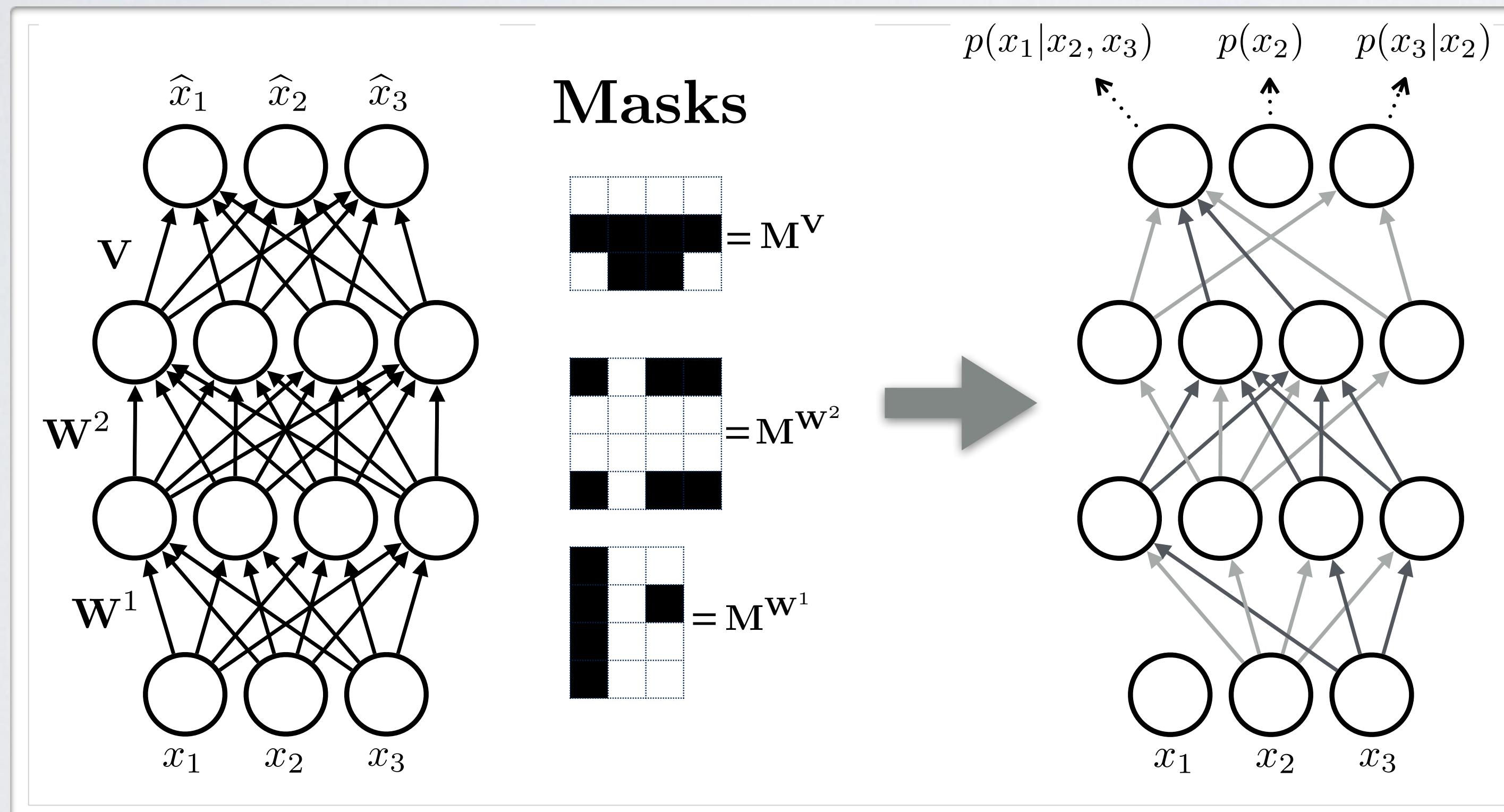


- Generalization of the work from Bengio and Bengio (2000)

# MASKED AUTOENCODER DISTRIBUTION ESTIMATION

**Topics:** MADE (Germain et al. 2015)

- Idea: constrain output so can be used for the conditionals  $p(x_k | \mathbf{x}_{<k})$

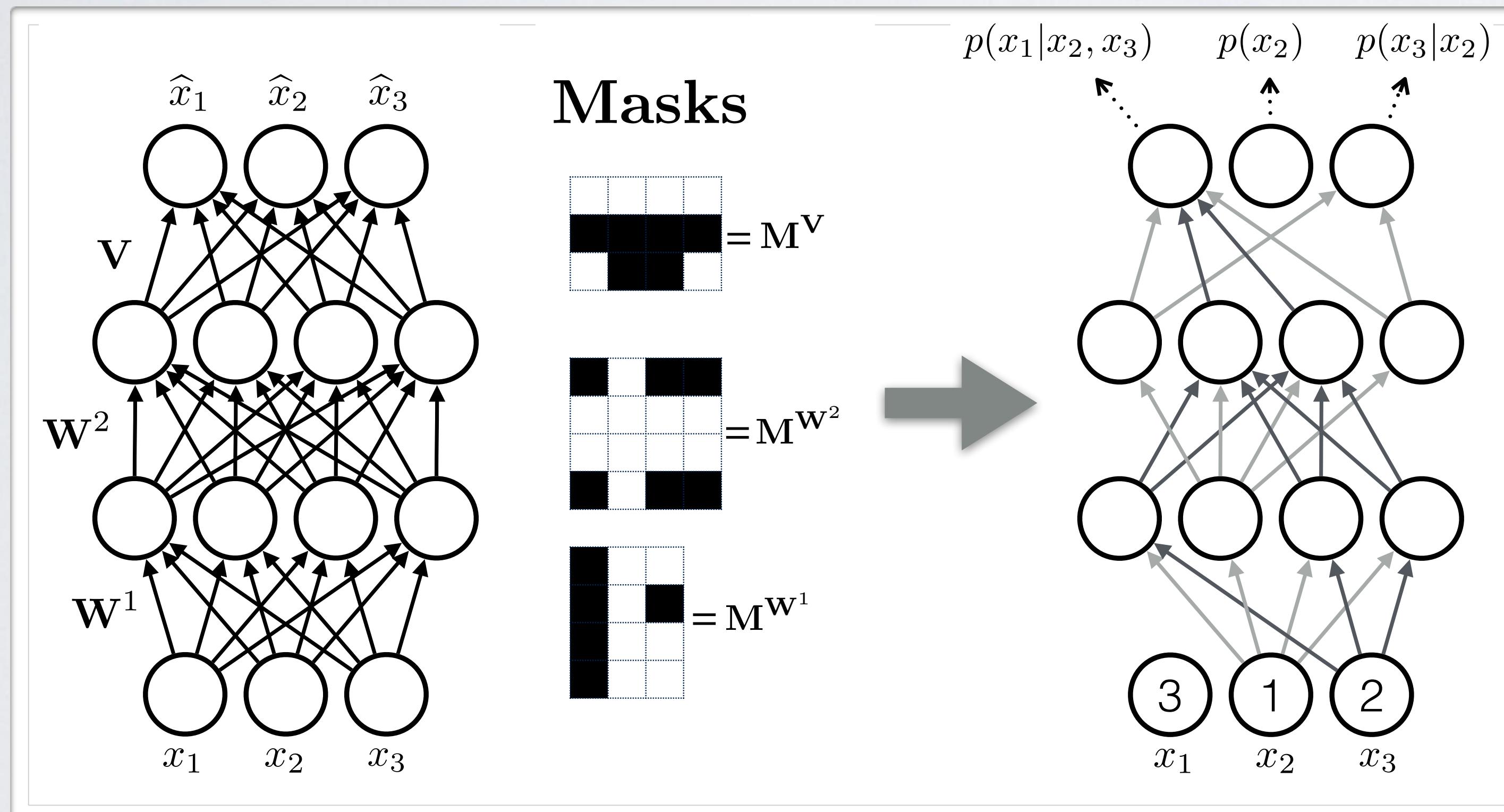


- Generalization of the work from Bengio and Bengio (2000)

# MASKED AUTOENCODER DISTRIBUTION ESTIMATION

**Topics:** MADE (Germain et al. 2015)

- Idea: constrain output so can be used for the conditionals  $p(x_k | \mathbf{x}_{<k})$

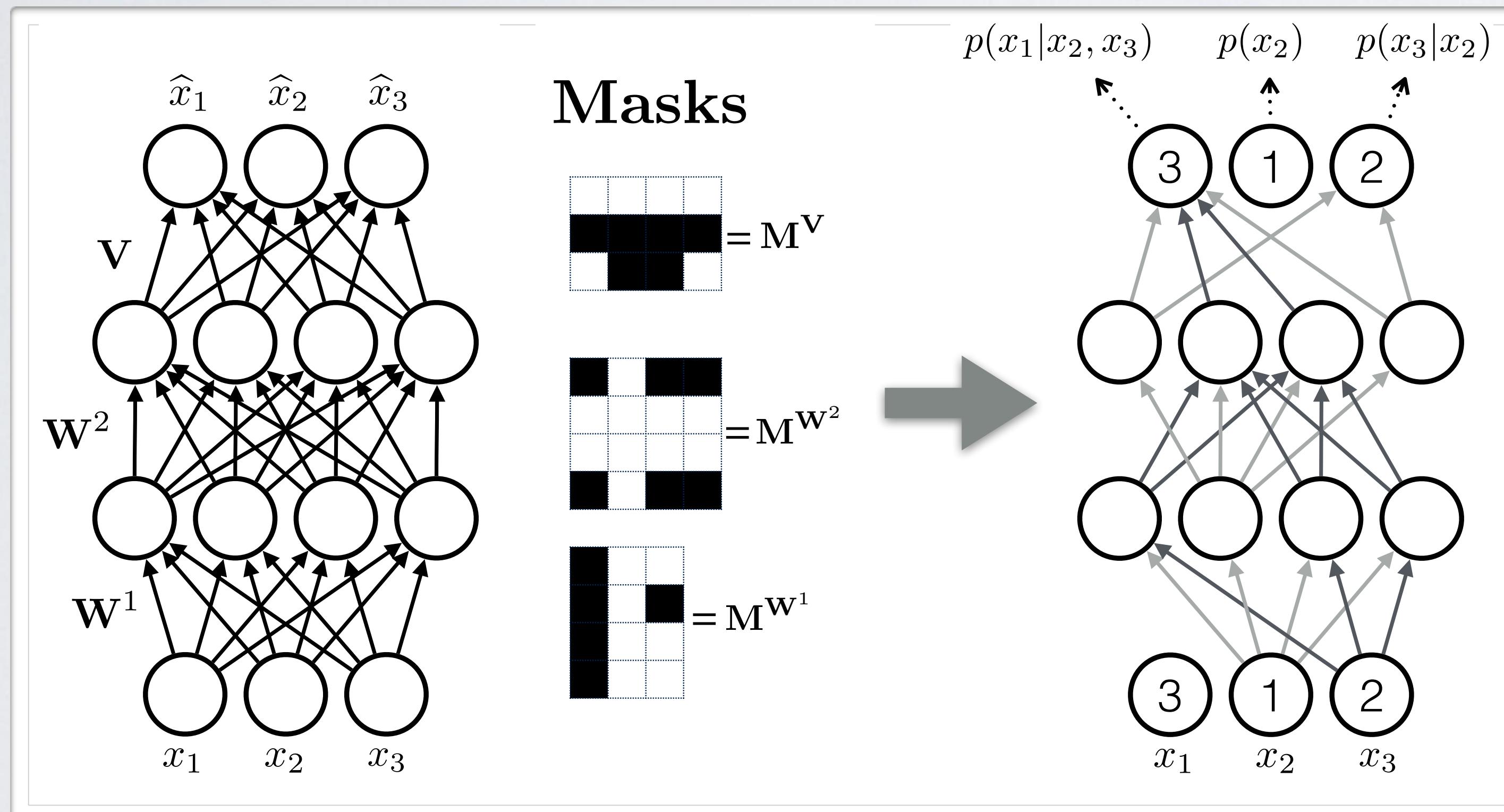


- Generalization of the work from Bengio and Bengio (2000)

# MASKED AUTOENCODER DISTRIBUTION ESTIMATION

**Topics:** MADE (Germain et al. 2015)

- Idea: constrain output so can be used for the conditionals  $p(x_k | \mathbf{x}_{<k})$

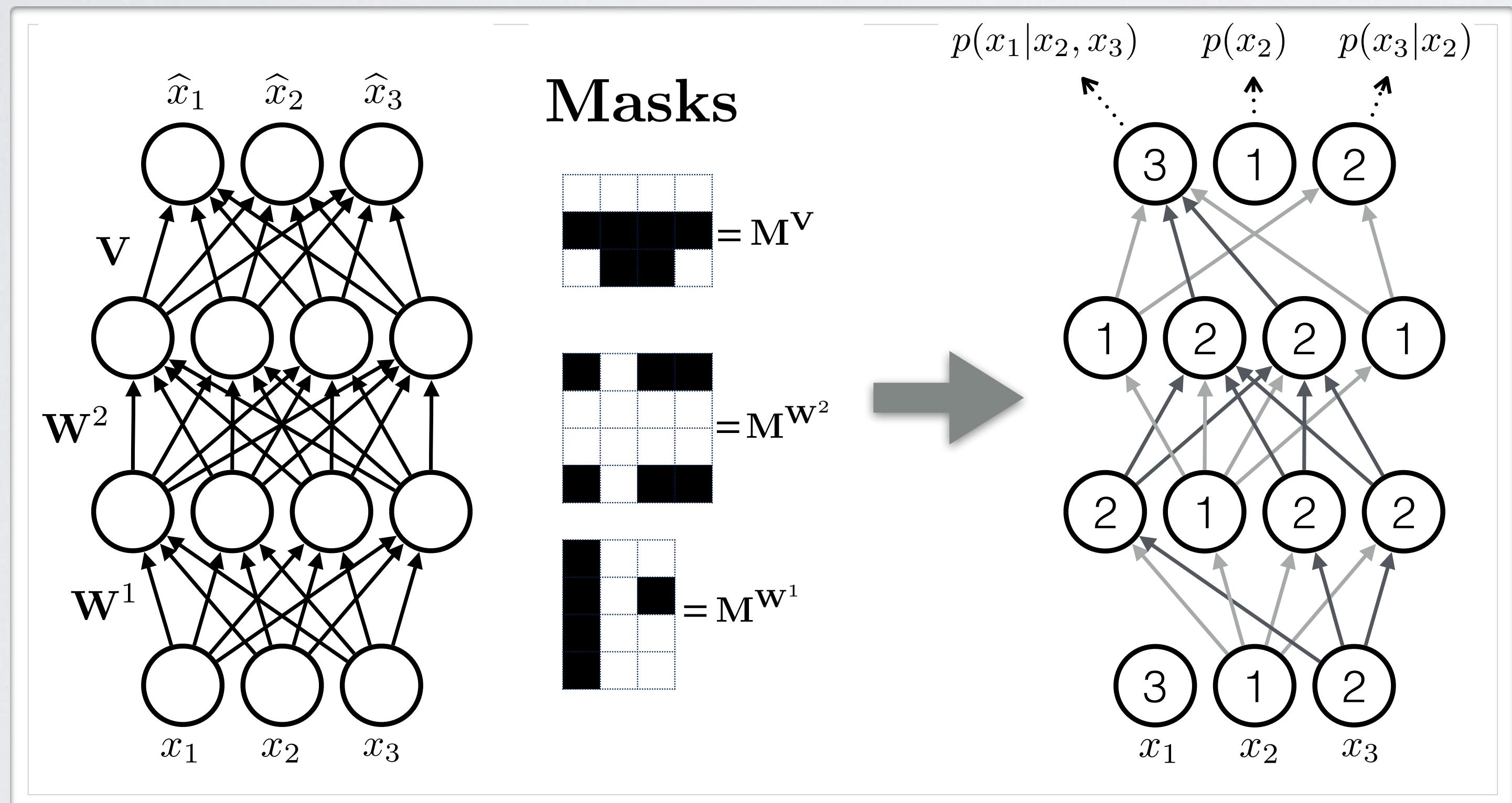


- Generalization of the work from Bengio and Bengio (2000)

# MASKED AUTOENCODER DISTRIBUTION ESTIMATION

**Topics:** MADE (Germain et al. 2015)

- Idea: constrain output so can be used for the conditionals  $p(x_k | \mathbf{x}_{<k})$



$$M_{k',k}^{\mathbf{W}^l} = 1_{m^l(k') \geq m^{l-1}(k)}$$

$$M_{d,k}^{\mathbf{V}} = 1_{d > m^L(k)}$$

- Generalization of the work from Bengio and Bengio (2000)

# MASKED AUTOENCODER DISTRIBUTION ESTIMATION

**Topics:** MADE (Germain et al. 2015)

- Training has the same complexity as regular autoencoders
- Computing  $p(\mathbf{x})$  is just a matter of performing a forward pass
- Sampling however requires  $D$  forward passes
- In practice, very large hidden layers may be required
  - ▶ not all hidden units can contribute to each conditional

# Masked Autoencoder for Distribution Estimation (MADE)

reconstruction



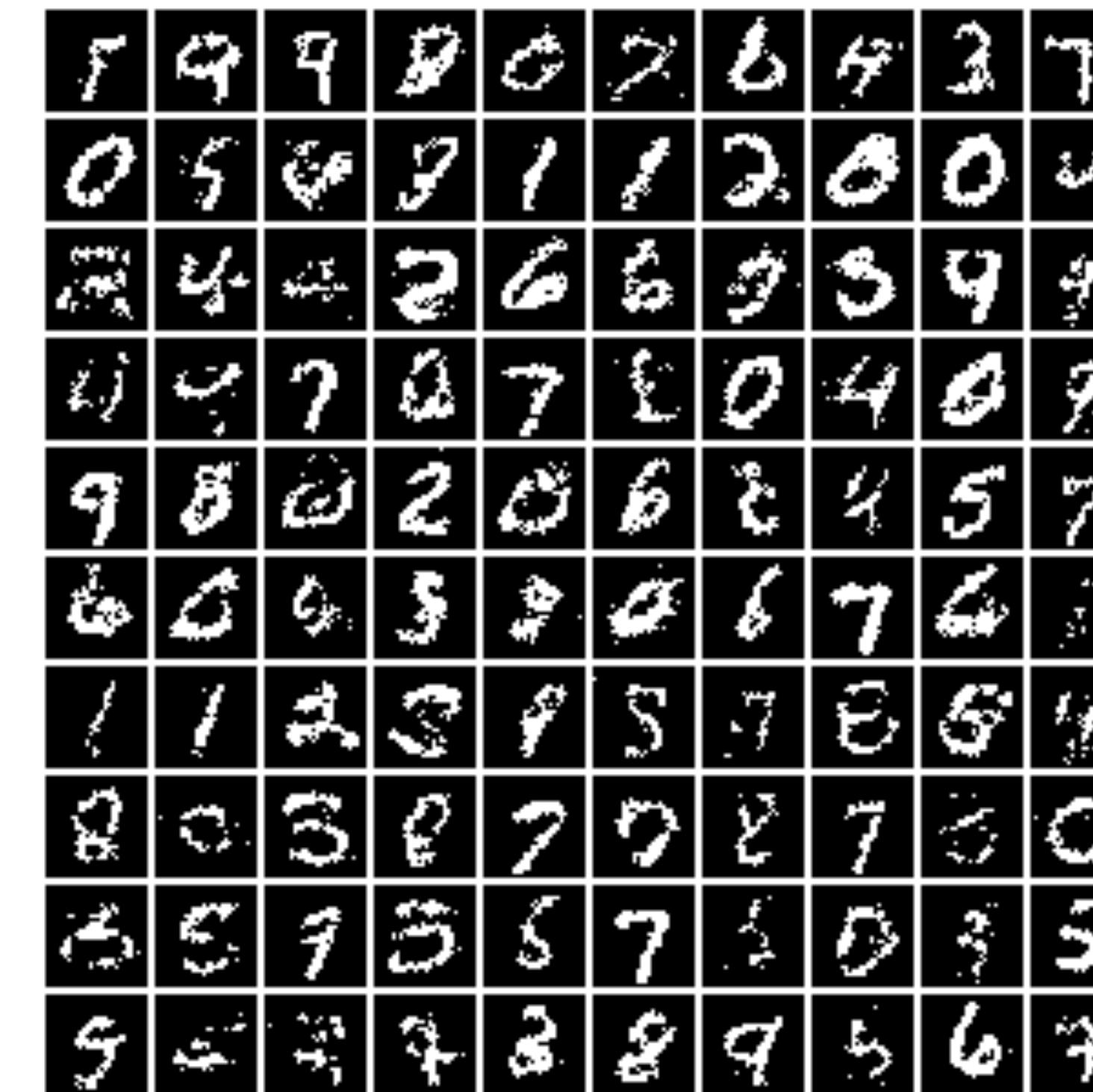
$$\hat{\mathbf{x}} = \text{decode}(\text{encode}(\mathbf{x}))$$

$$\mathcal{L}(\mathbf{x}) = - \sum_{i=1}^{|\mathbf{x}|} \left( x_i \log \hat{x}_i + (1 - x_i) \log(1 - \hat{x}_i) \right)$$



NLL criterion for a binary  $\mathbf{x}$

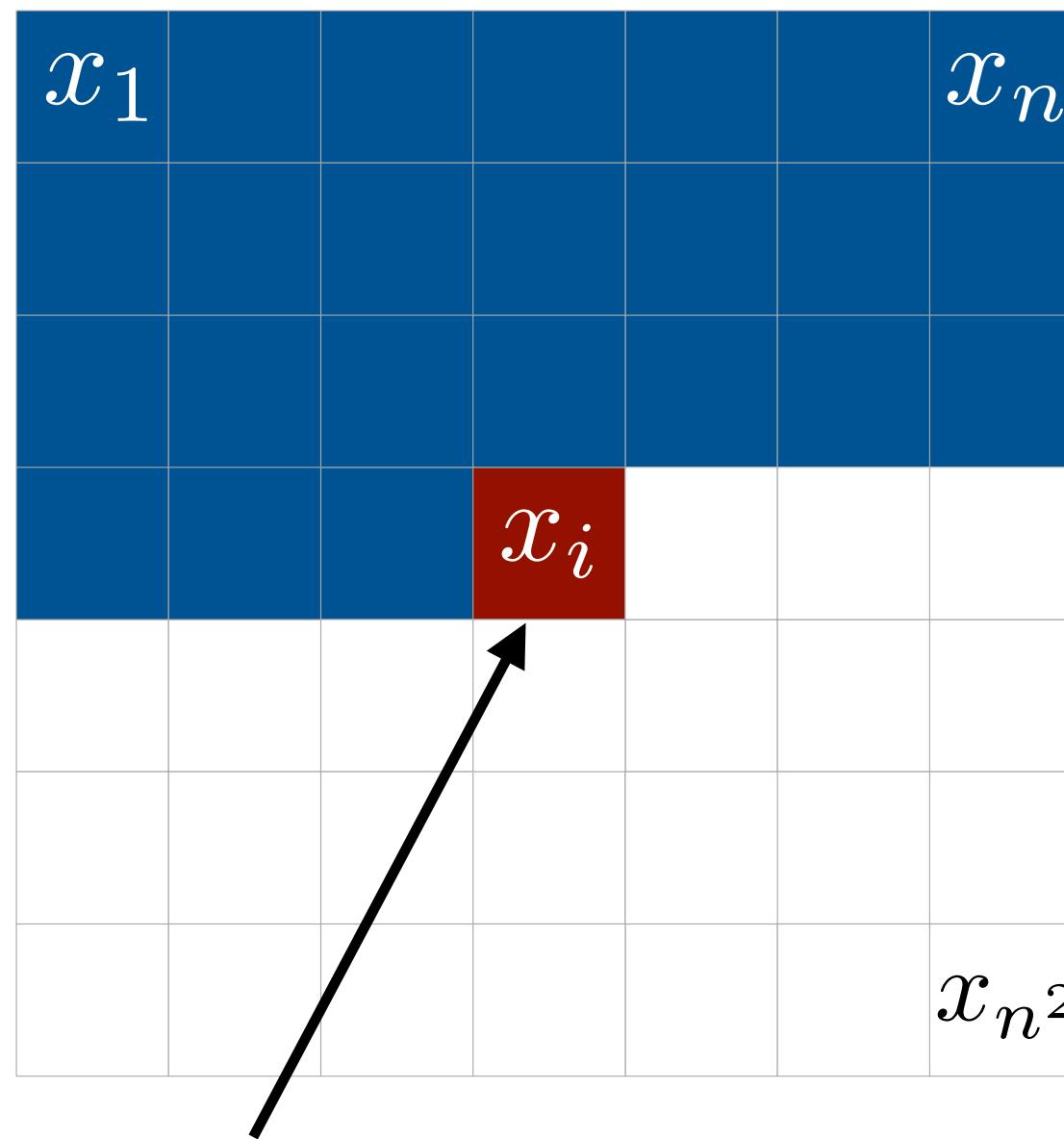
# Masked Autoencoder for Distribution Estimation (MADE)



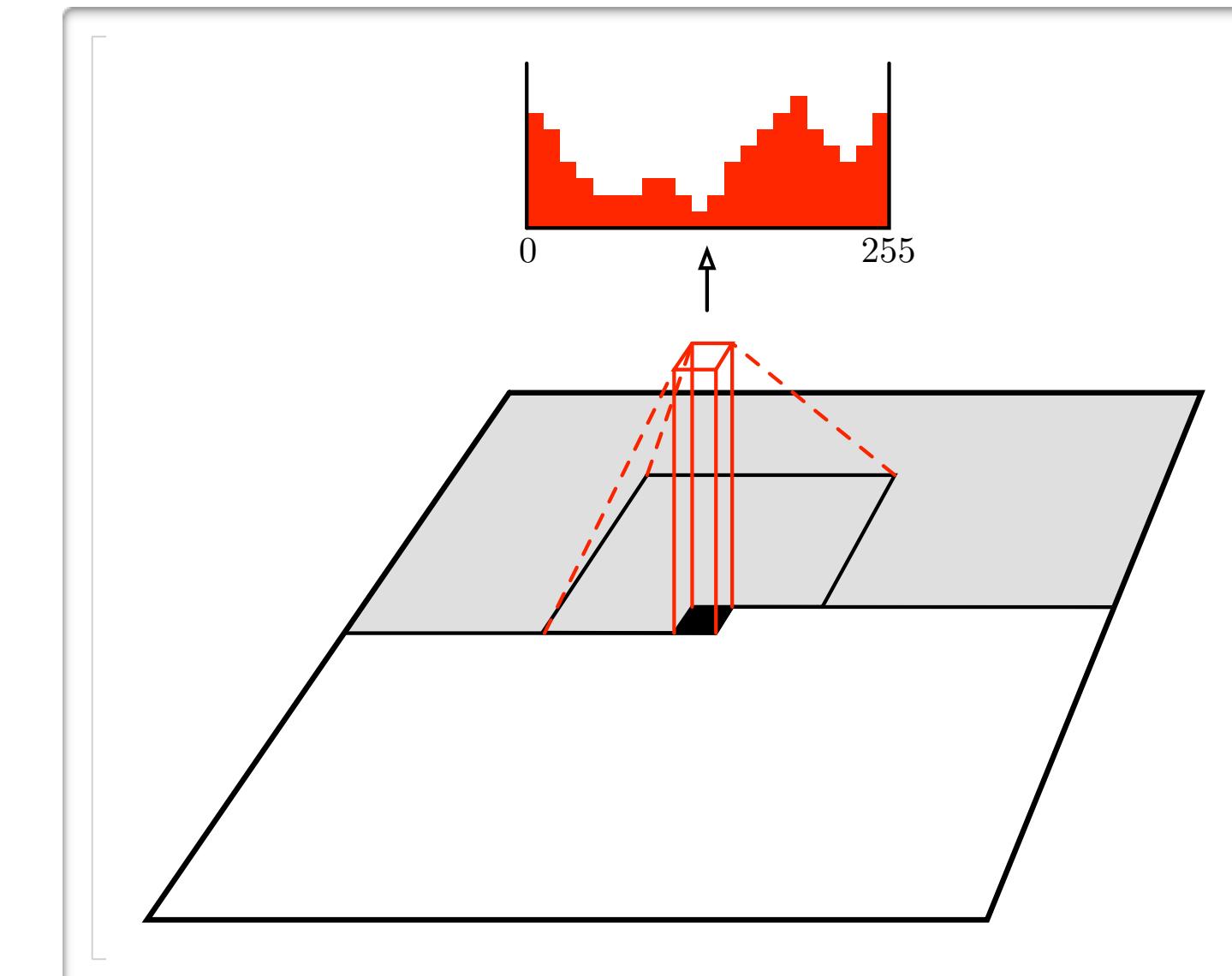
Binarized MNIST samples

# PixelCNN

Idea: use masked convolutions to enforce the autoregressive relationship



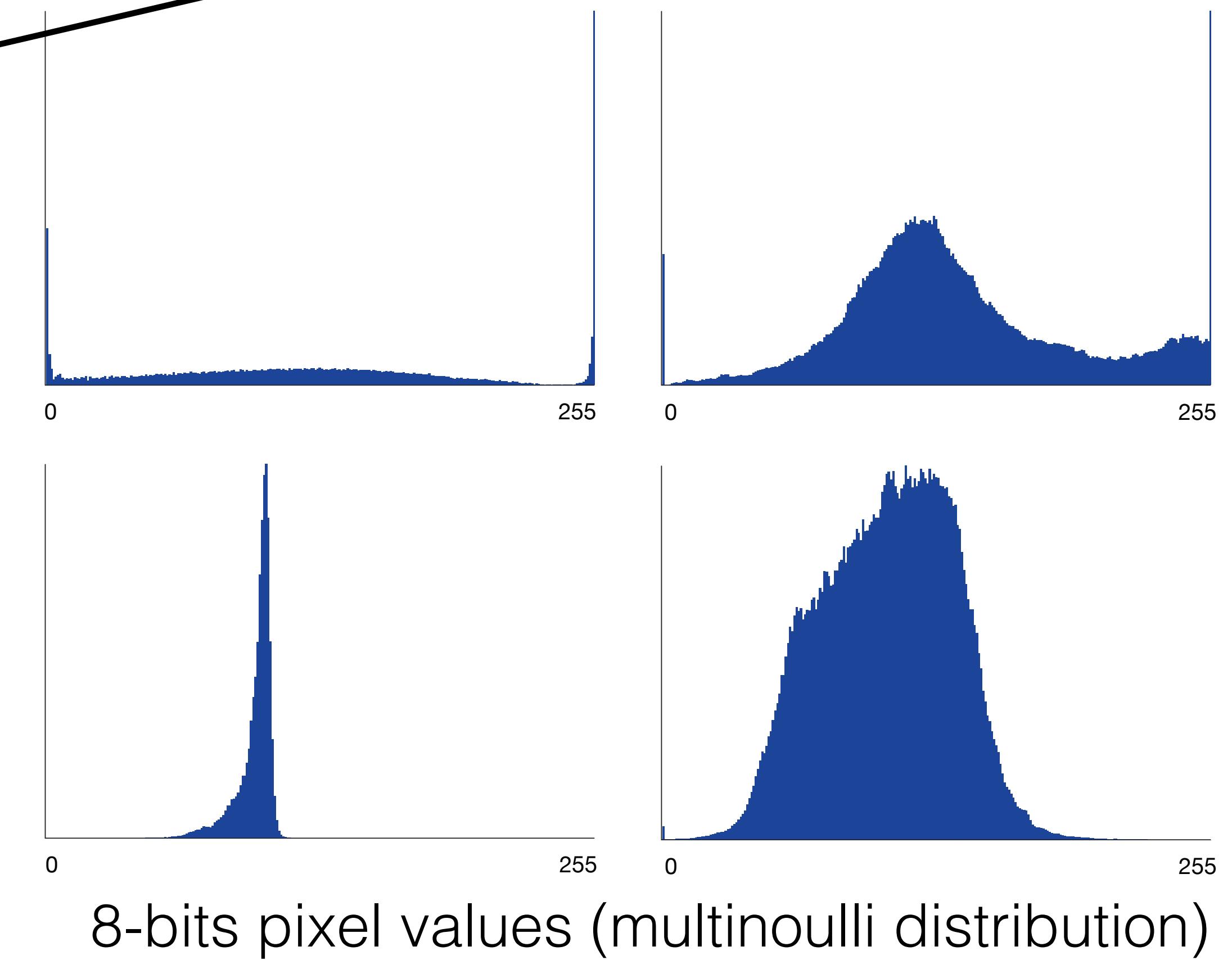
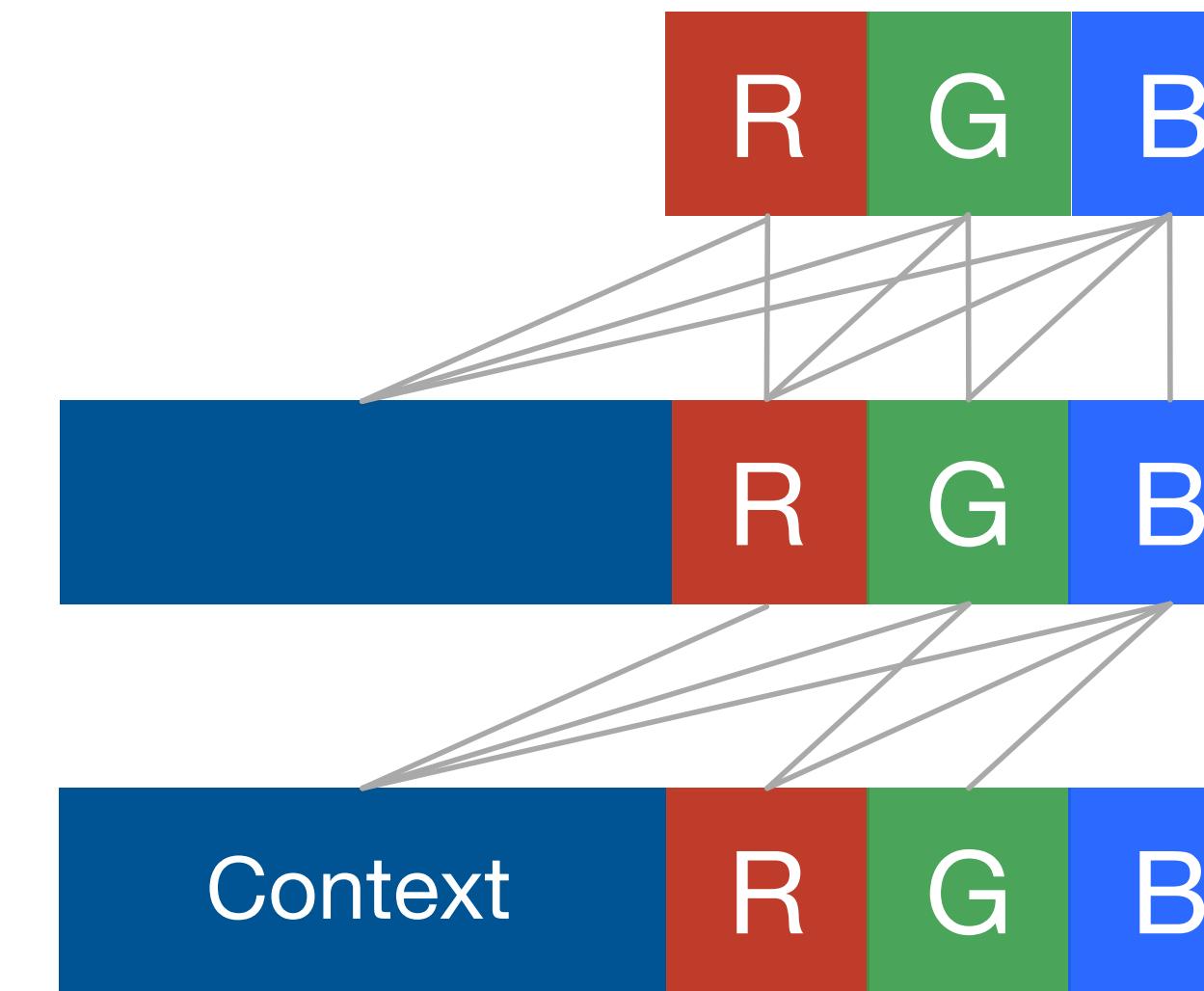
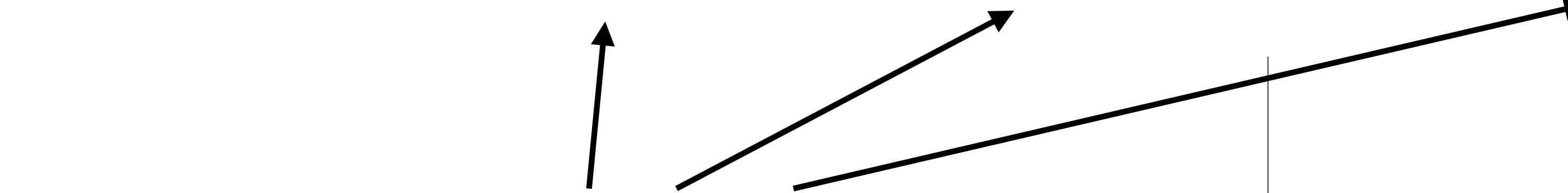
$$p(x_i \mid \mathbf{x}_{<i})$$



Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." arXiv preprint arXiv:1601.06759 (2016).

# PixelCNN

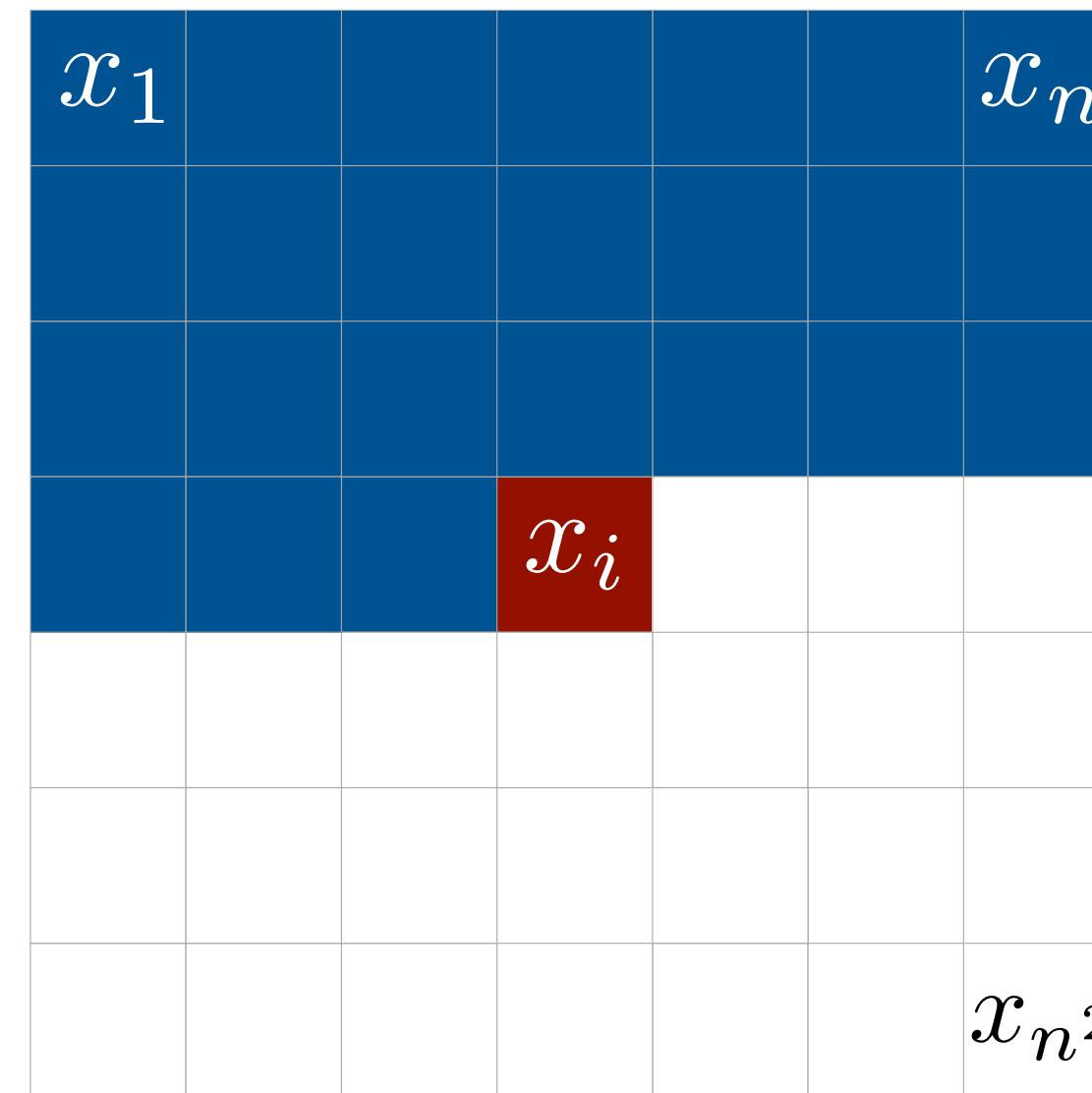
$$p(x_i \mid \mathbf{x}_{<i}) = p(x_{i,R} \mid \mathbf{x}_{<i})p(x_{i,G} \mid x_{i,R}, \mathbf{x}_{<i})p(x_{i,B} \mid x_{i,R}, x_{i,G}, \mathbf{x}_{<i})$$



# PixelCNN

*How can convolutions make  
this raster scan faster?*

Use a stack of masked convolutions

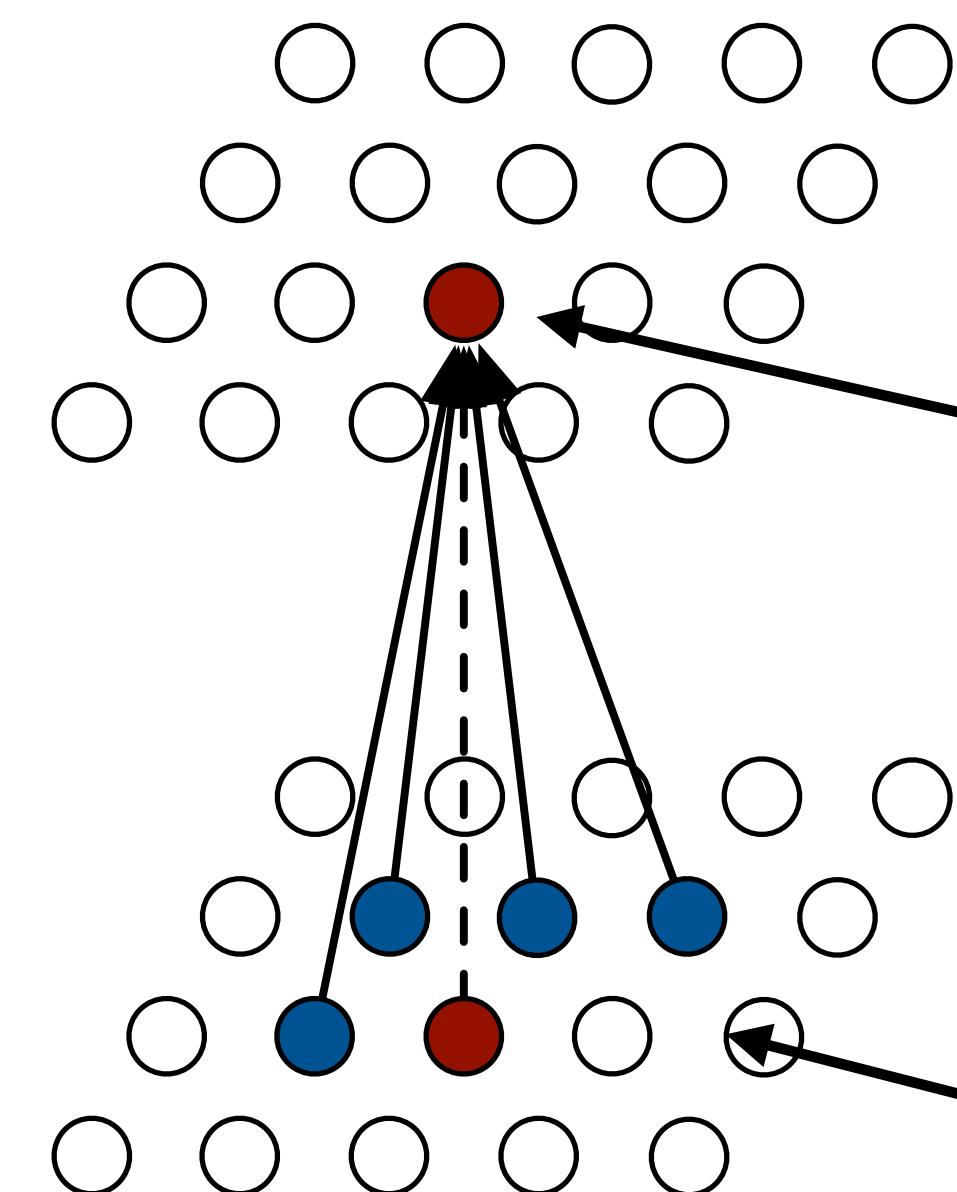


1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

Training can be parallelized, though generation is still a sequential operation over pixels

*van den Oord, Aaron, et al. "Conditional image generation with PixelCNN decoders."  
Advances in Neural Information Processing Systems. 2016.*

# PixelCNN



composing multiple  
layers increases the  
context size

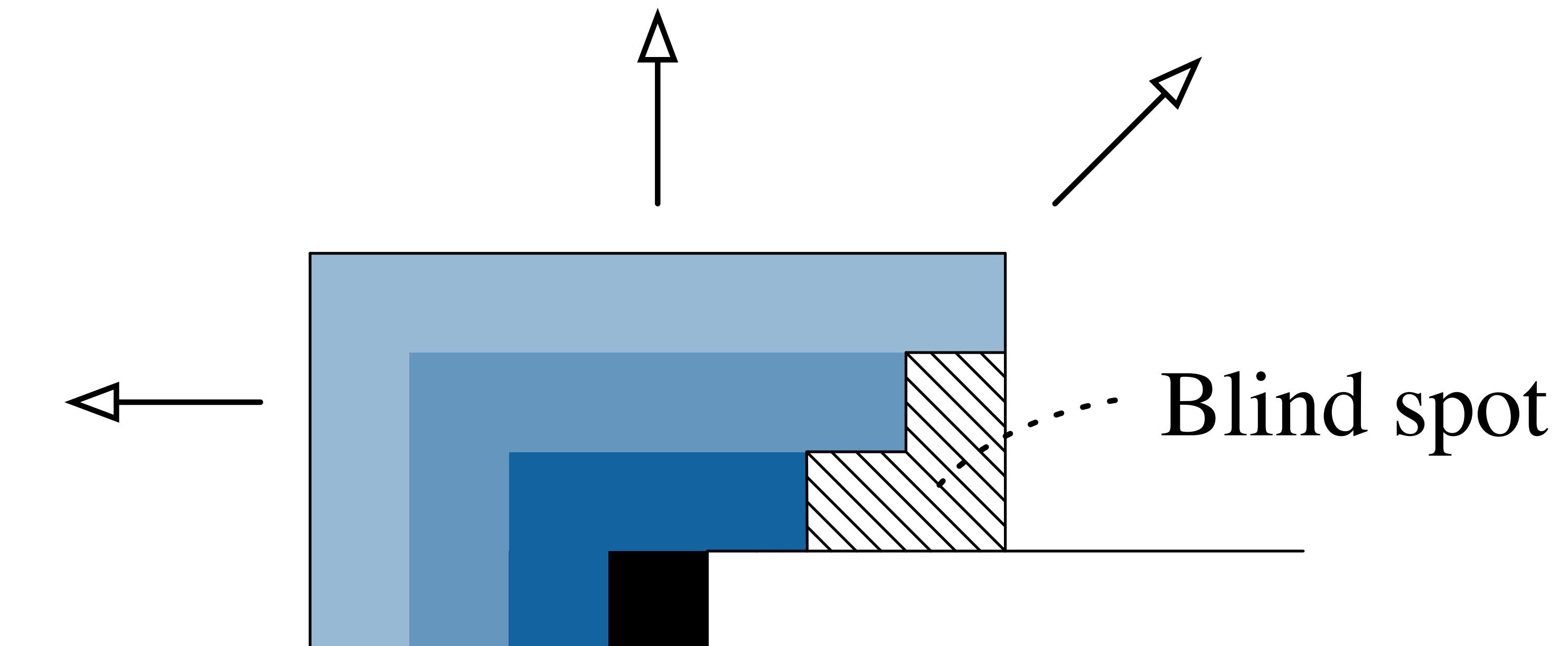
only depends on pixel  
above and to the left

masked convolution

# Improving PixelCNN

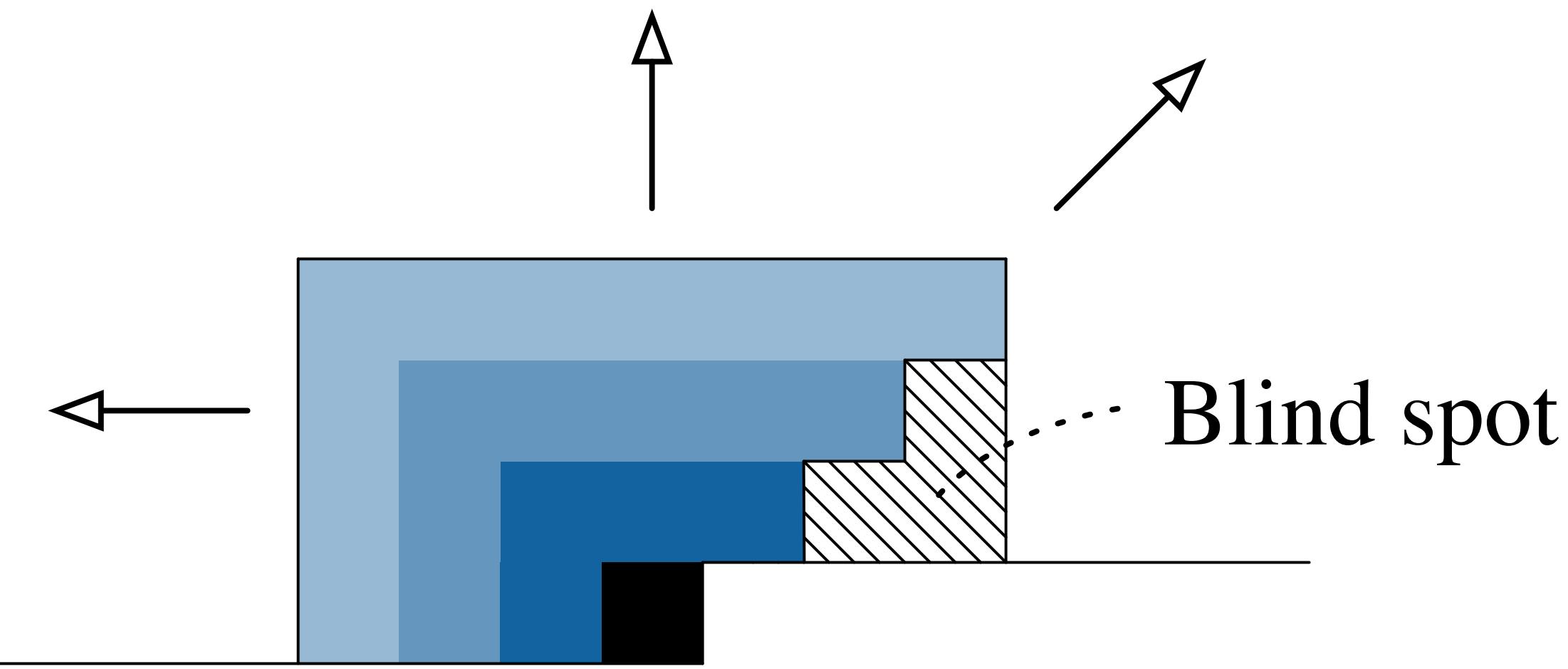
*There is a problem with this form of masked convolution.*

1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

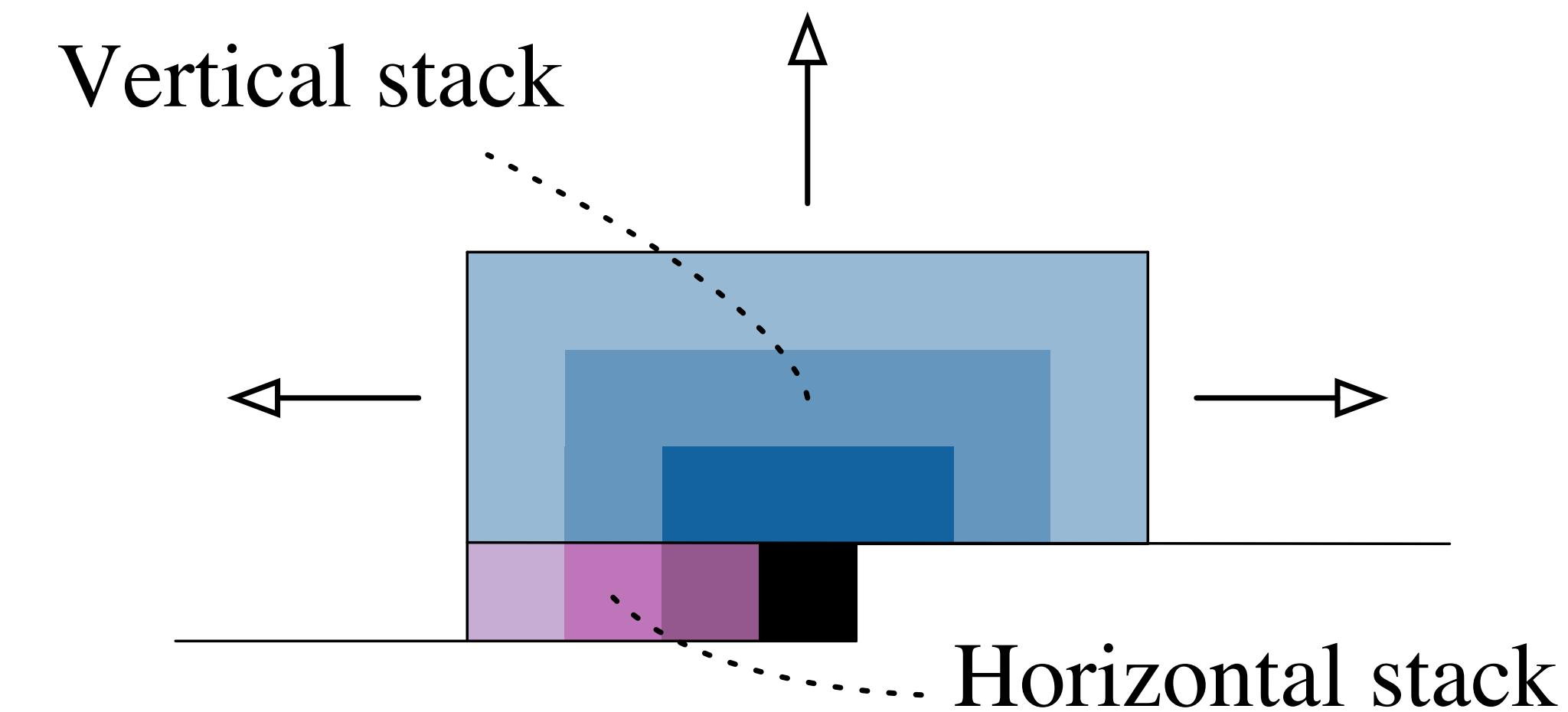


*Stacking layers of masked convolution creates a blindspot*

# Improving PixelCNN I



*Stacking layers of masked convolution creates a blindspot*

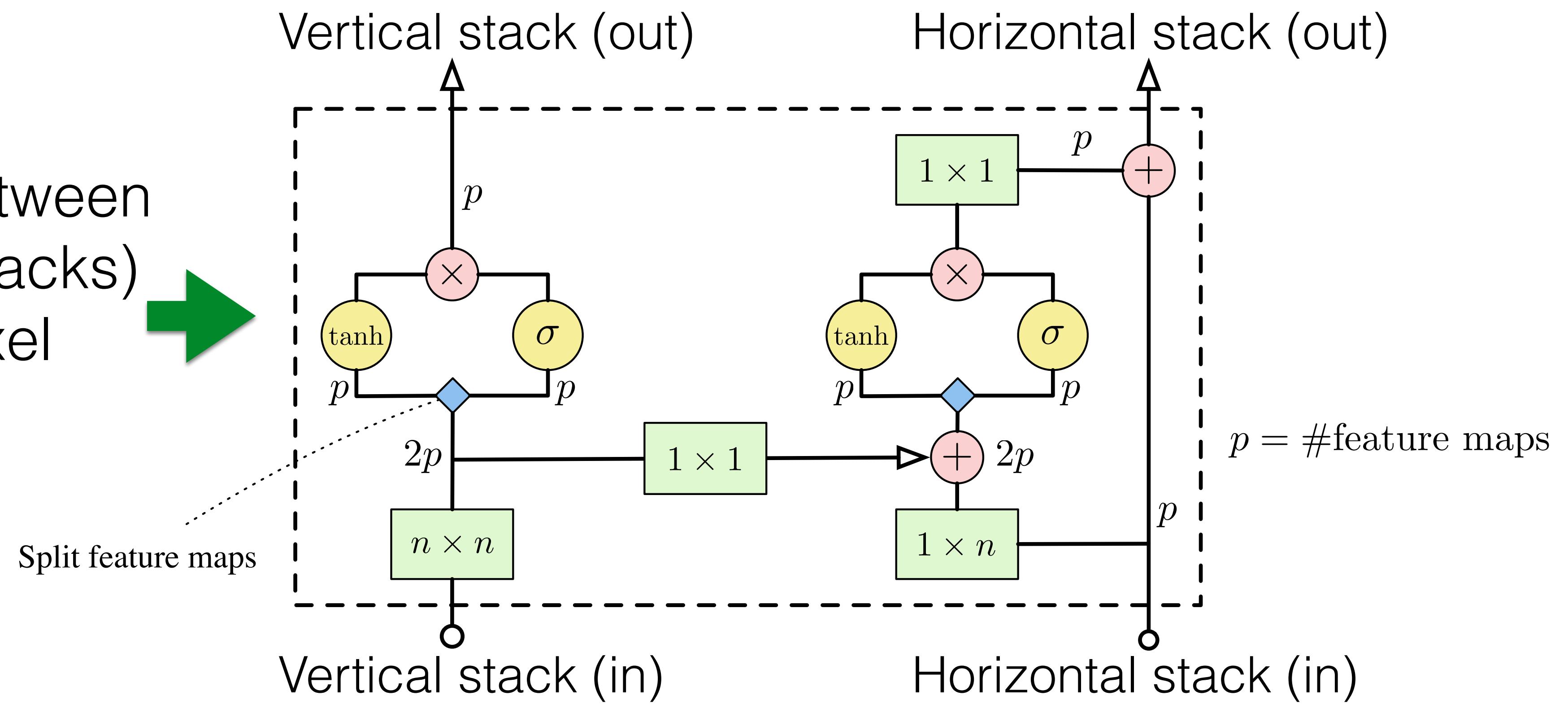


*Solution: use two stacks of convolution, a vertical stack and a horizontal stack*

# Improving PixelCNN II

Use more expressive nonlinearity:  $\mathbf{h}_{k+1} = \tanh(W_{k,f} * \mathbf{h}_k) \odot \sigma(W_{k,g} * \mathbf{h}_k)$

This information flow (between vertical and horizontal stacks)  
preserves the correct pixel dependencies



# EXPERIMENTAL RESULTS

**Topics:** CIFAR-10

*Conditional Image Generation with PixelCNN Decoders*  
van den Oord, Kalchbrenner, Vinyals, Espeholt, Graves, Kavukcuoglu, NIPS 2016

- Performance measured in bits/dim

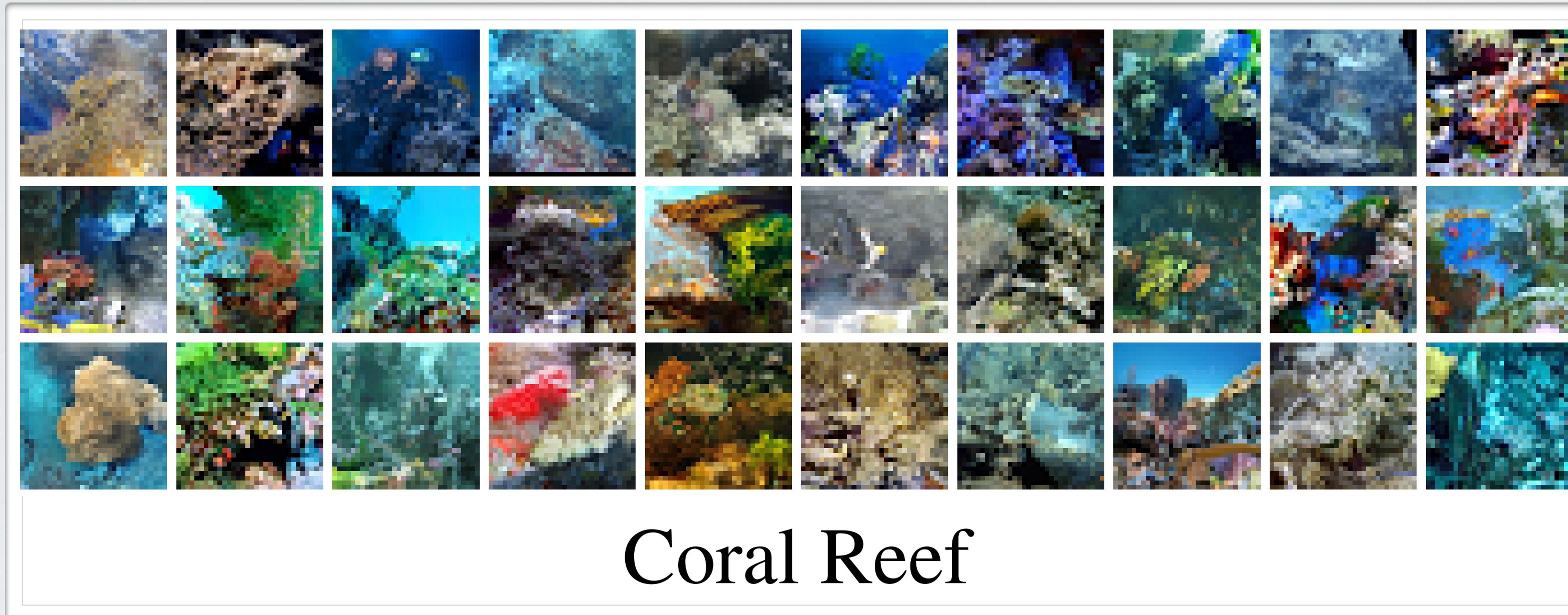
Model	NLL Test (Train)
Uniform Distribution: [30]	8.00
Multivariate Gaussian: [30]	4.70
NICE: [4]	4.48
Deep Diffusion: [24]	4.20
DRAW: [9]	4.13
Deep GMMs: [31, 29]	4.00
Conv DRAW: [8]	3.58 (3.57)
RIDE: [26, 30]	3.47
PixelCNN: [30]	3.14 (3.08)
PixelRNN: [30]	3.00 (2.93)
<b>Gated PixelCNN:</b>	<b>3.03 (2.90)</b>

# EXPERIMENTAL RESULTS

**Topics:** CIFAR-10

*Conditional Image Generation with PixelCNN Decoders*  
van den Oord, Kalchbrenner, Vinyals, Espeholt, Graves, Kavukcuoglu, NIPS 2016

- Samples from a class-conditional PixelCNN



# EXPERIMENTAL RESULTS

**Topics:** CIFAR-10

*Conditional Image Generation with PixelCNN Decoders*  
van den Oord, Kalchbrenner, Vinyals, Espeholt, Graves, Kavukcuoglu, NIPS 2016

- Samples from a class-conditional PixelCNN



# EXPERIMENTAL RESULTS

**Topics:** CIFAR-10

*Conditional Image Generation with PixelCNN Decoders*  
van den Oord, Kalchbrenner, Vinyals, Espeholt, Graves, Kavukcuoglu, NIPS 2016

- Samples from a class-conditional PixelCNN

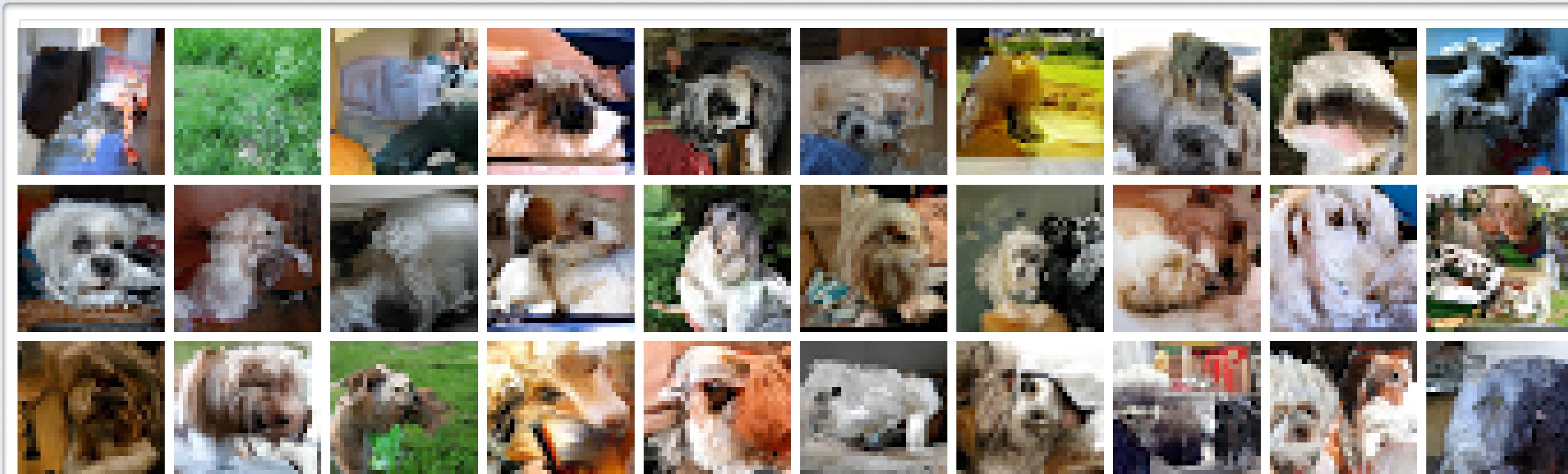


# EXPERIMENTAL RESULTS

**Topics:** CIFAR-10

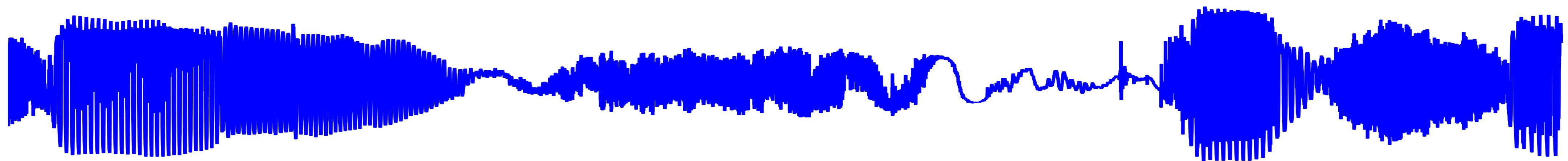
*Conditional Image Generation with PixelCNN Decoders*  
van den Oord, Kalchbrenner, Vinyals, Espeholt, Graves, Kavukcuoglu, NIPS 2016

- Samples from a class-conditional PixelCNN



Lhasa Apso (dog)

# WaveNet

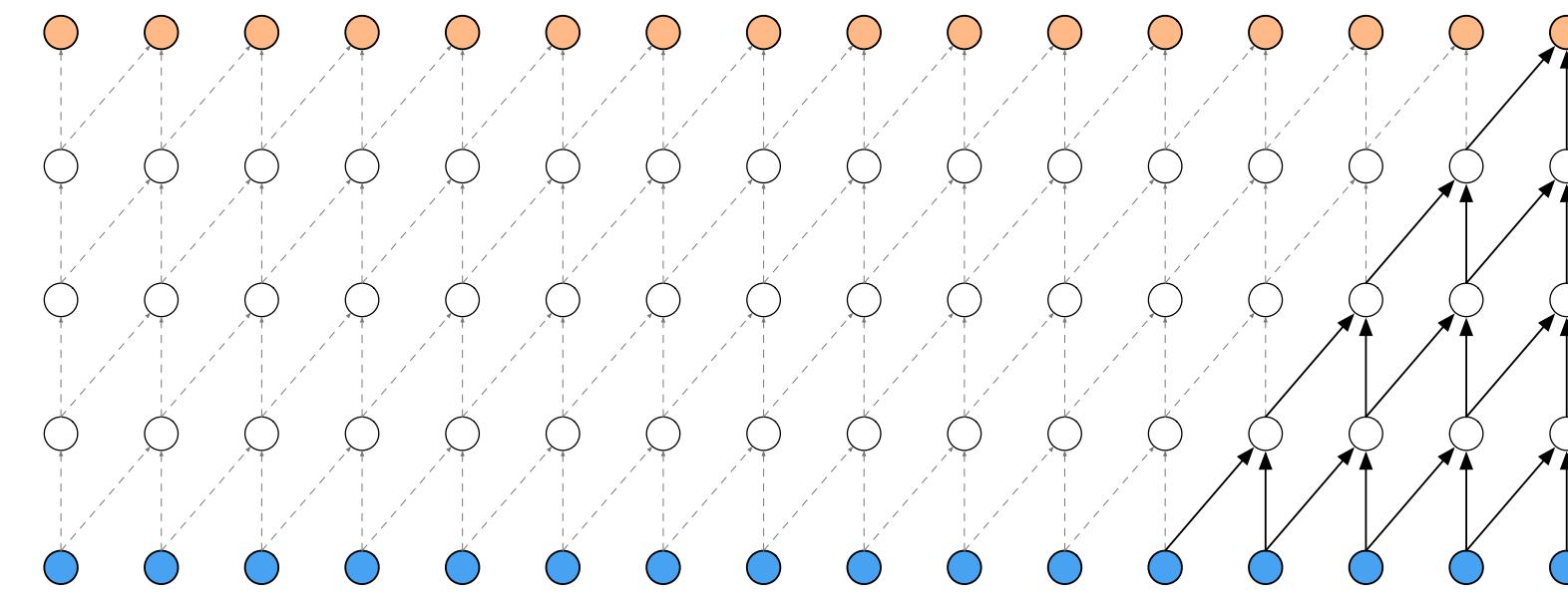


Audio: **much** larger dimensionality than images (at least 16,000 samples per **second**)

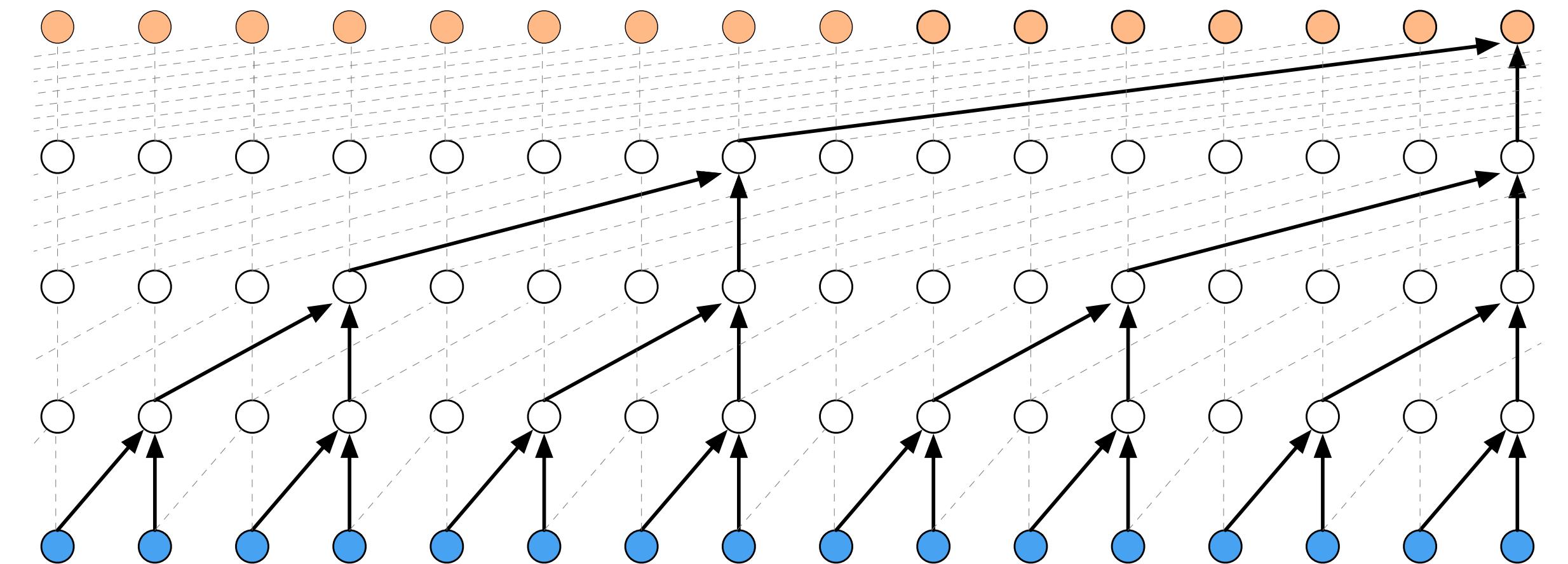
Idea: adapt PixelCNN to allow very large temporal dependencies

# WaveNet

Addressing large-scale temporal dependencies



Regular convolutions



Dilated convolutions

*Note: strided convolutions cannot be used because the output has to have the **same** dimensionality as the input.*

# WaveNet

Discrete conditional probabilities

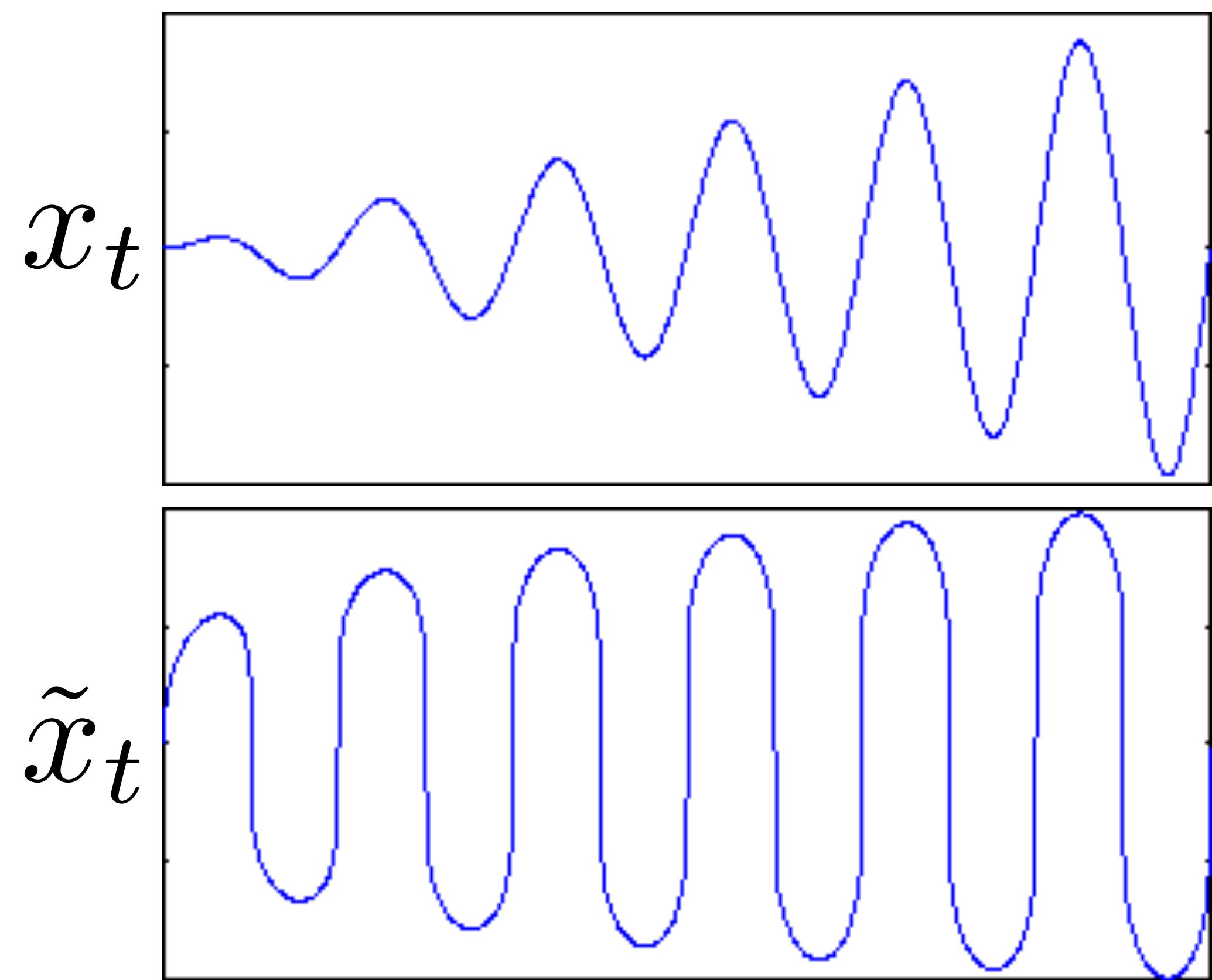
$\mu$ -law companding  
transformation

$$a_t \text{ (16-bit int)} \rightarrow x_t \in [-1, 1]$$

$$\tilde{x}_t = \text{sign}(x_t) \frac{\ln(1 + 255|x_t|)}{\ln 256}$$

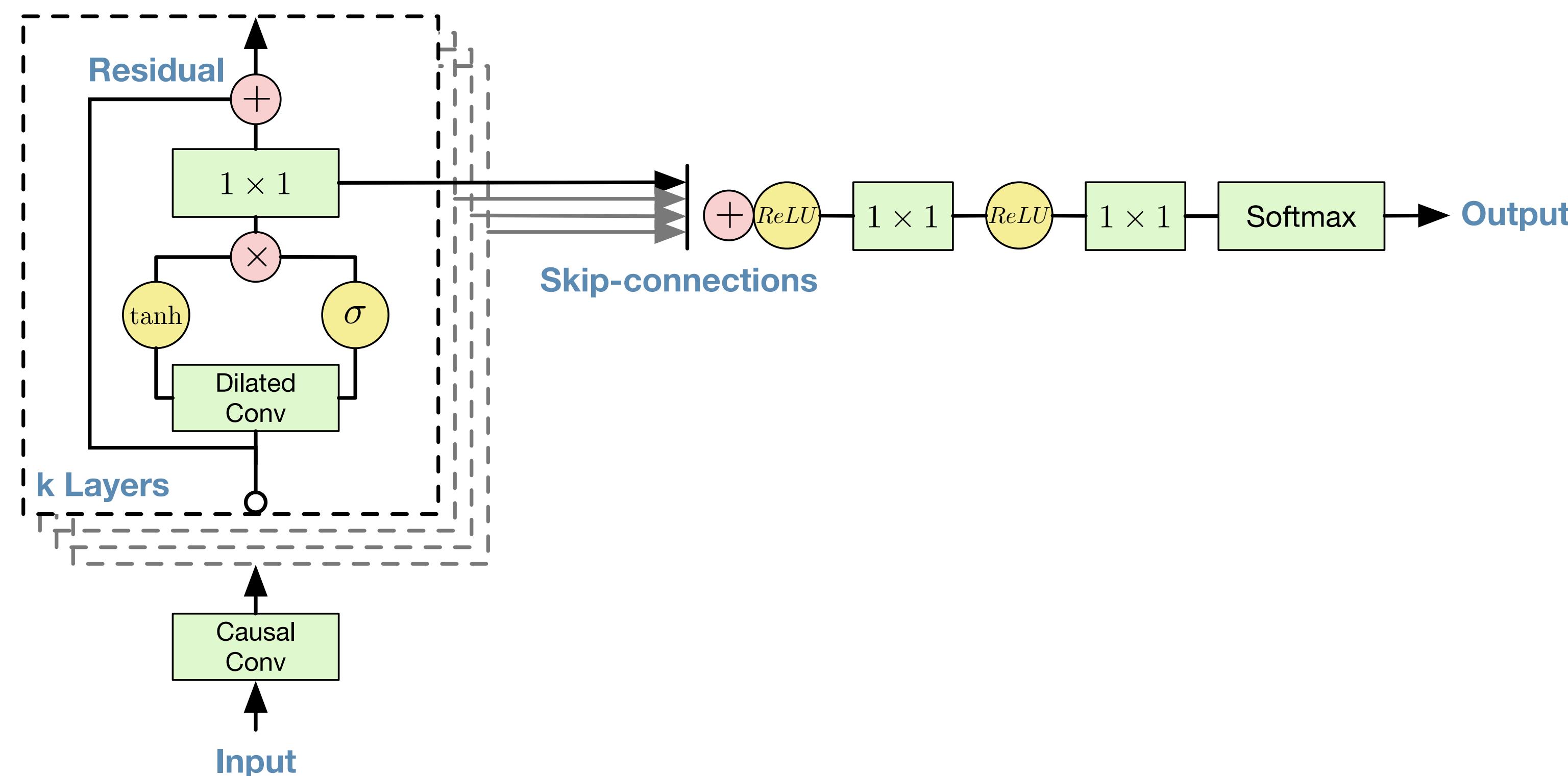
$$\tilde{x}_t \in [-1, 1] \rightarrow \tilde{a}_t \text{ (8-bit int)}$$

quantize back



# WaveNet

## Complete architecture



van den Oord, Aäron, et al. "Wavenet: A generative model for raw audio." CoRR abs/1609.03499 (2016).

# WaveNet

Conditional generation

$$\mathbf{z} = \tanh(W_{k,f} * \mathbf{x} + V_{k,f}^T \mathbf{h}) \odot \sigma(W_{k,g} * \mathbf{x} + V_{k,g}^T \mathbf{h})$$

Global conditioning (e.g., speaker ID)

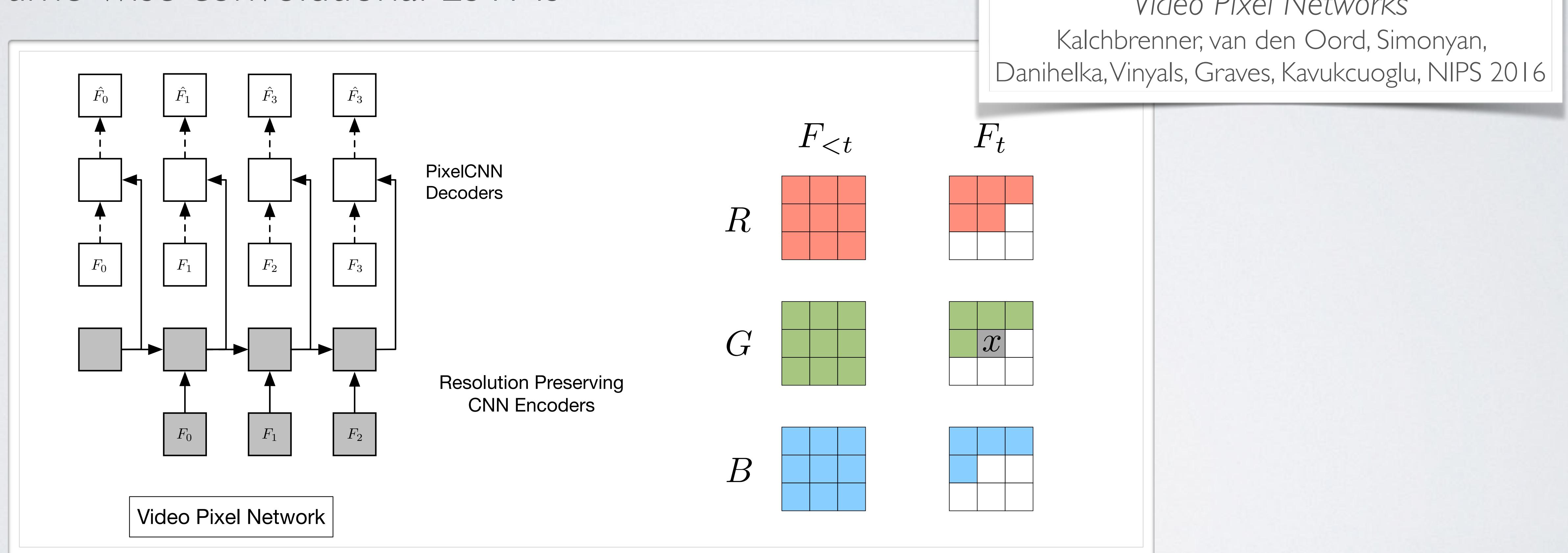
$$\mathbf{z} = \tanh(W_{k,f} * \mathbf{x} + V_{k,f} * \mathbf{h}) \odot \sigma(W_{k,g} * \mathbf{x} + V_{k,g} * \mathbf{h})$$

Local conditioning (e.g., text)

# AUTOREGRESSIVE VIDEO MODELS

## Topics: Video Pixel Network

- Connect Pixel CNN to frame-wise convolutional networks and time-wise convolutional LSTMs



# AUTOREGRESSIVE VIDEO MODELS

## **Topics:** Video Pixel Network

- Connect Pixel CNN to frame-wise convolutional networks and time-wise convolutional LSTMs
- Videos of robot manipulating
  - ▶ objects seen in the training set
  - ▶ new objects not seen in training set

*Video Pixel Networks*  
Kalchbrenner, van den Oord, Simonyan,  
Danhelka, Vinyals, Graves, Kavukcuoglu, NIPS 2016

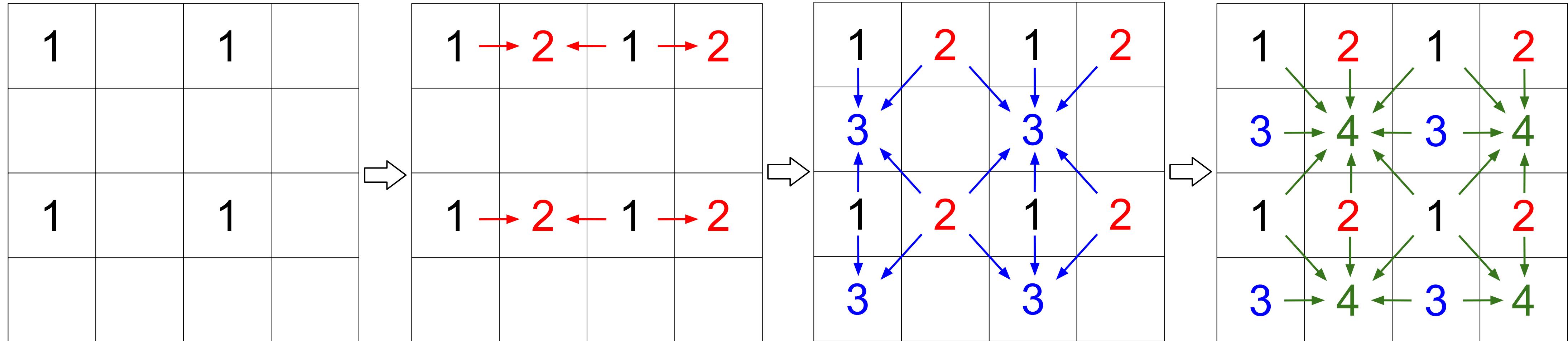
# Parallel Multiscale Autoregressive Density Estimation



Scott Reed, Aaron vanden Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Dan Belov, Nando de Freitas (2017)

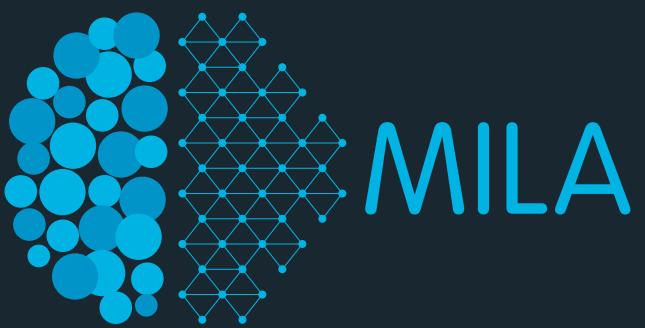
Can we speed up the generation time of PixelCNN?

- Yes, via multiscale generation:



*Figure 2.* Example pixel grouping and ordering for a  $4 \times 4$  image. The upper-left corners form group 1, the upper-right group 2, and so on. For clarity we only use arrows to indicate immediately-neighboring dependencies, but note that all pixels in preceding groups can be used to predict all pixels in a given group. For example all pixels in group 2 can be used to predict pixels in group 4. In our image experiments pixels in group 1 originate from a lower-resolution image. For video, they are generated given the previous frames.

# Parallel Multiscale Autoregressive Density Estimation

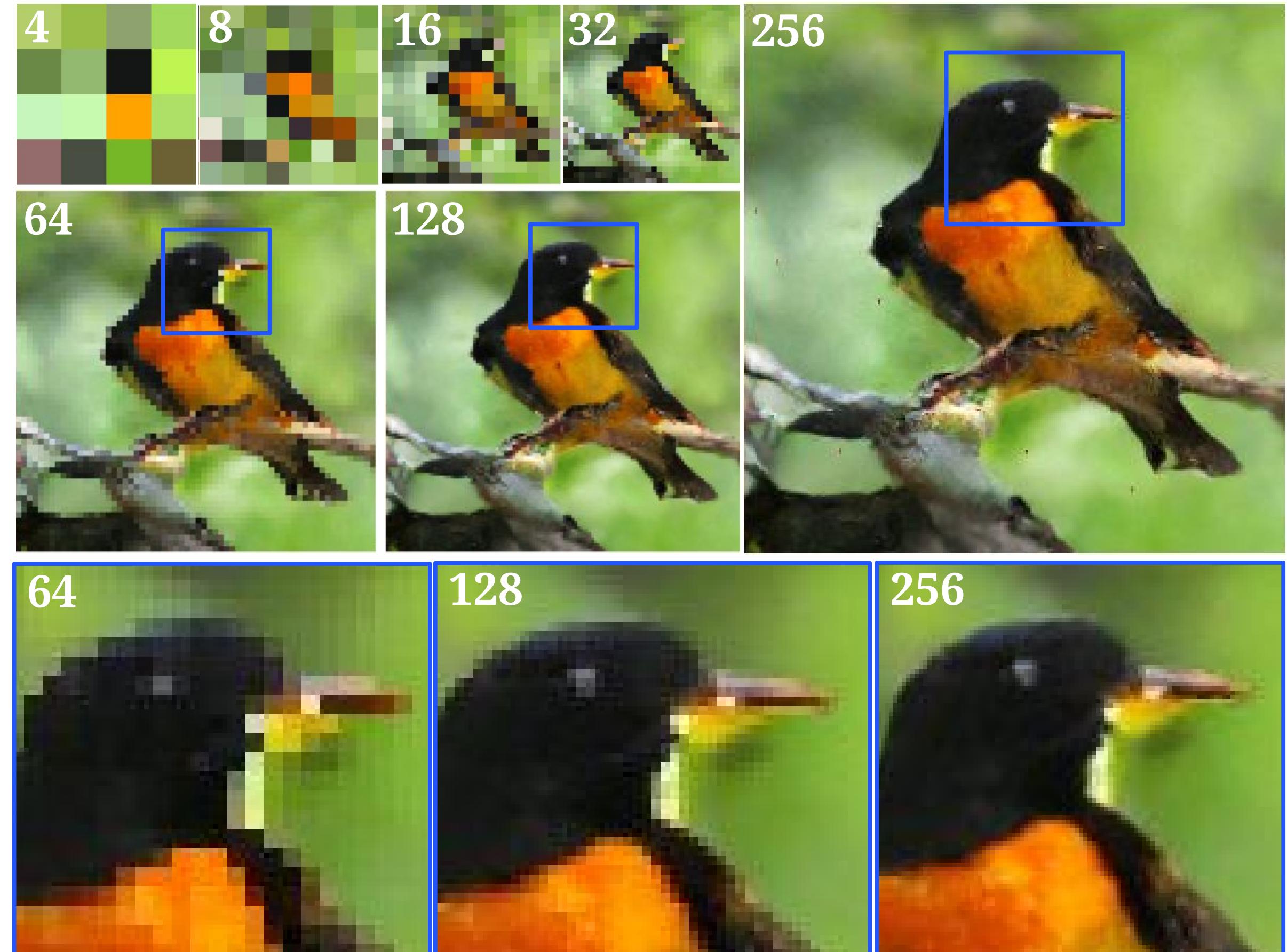


Scott Reed, Aaron vanden Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Dan Belov, Nando de Freitas (2017)

Can we speed up the generation time of PixelCNN?

- Yes, via multiscale generation.
- Also seems to help to provide better global structure

“A yellow bird with a black head, orange eyes and an orange bill.”



*Figure 1.* Samples from our model at resolutions from  $4 \times 4$  to  $256 \times 256$ , conditioned on text and bird part locations in the CUB data set. See Fig. 4 and the supplement for more examples.

# Parallel WaveNet:Fast High-Fidelity Speech Synthesis

(van den Oord et al., 2017)

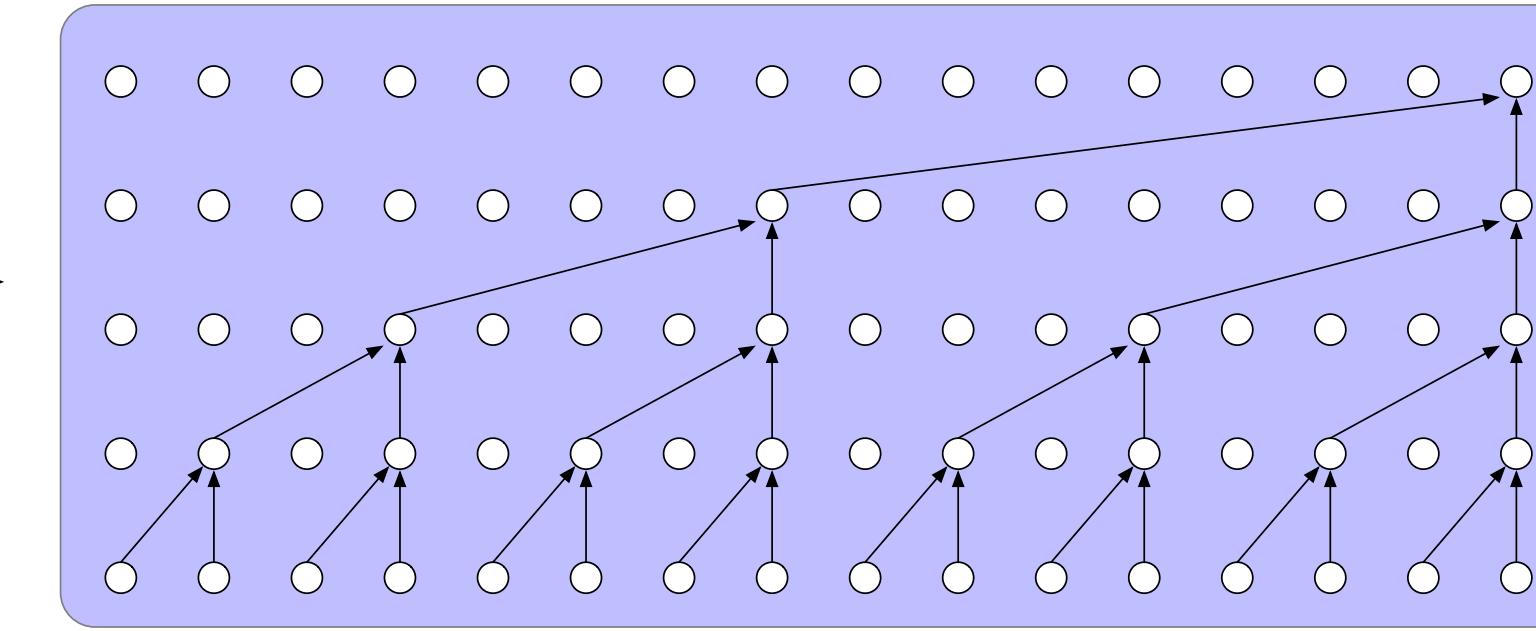


Can we speed up generation time of WaveNet?

- Yes, via distillation training with a teacher WaveNet.
- Used additional losses to improve performance:
  - **power loss**: match the power spectrum to real data (speech)
  - **perceptual loss**: distance in pre-trained classifier activation space.
  - **contrastive loss**: bring the output closer to similar data and farther from dissimilar data.

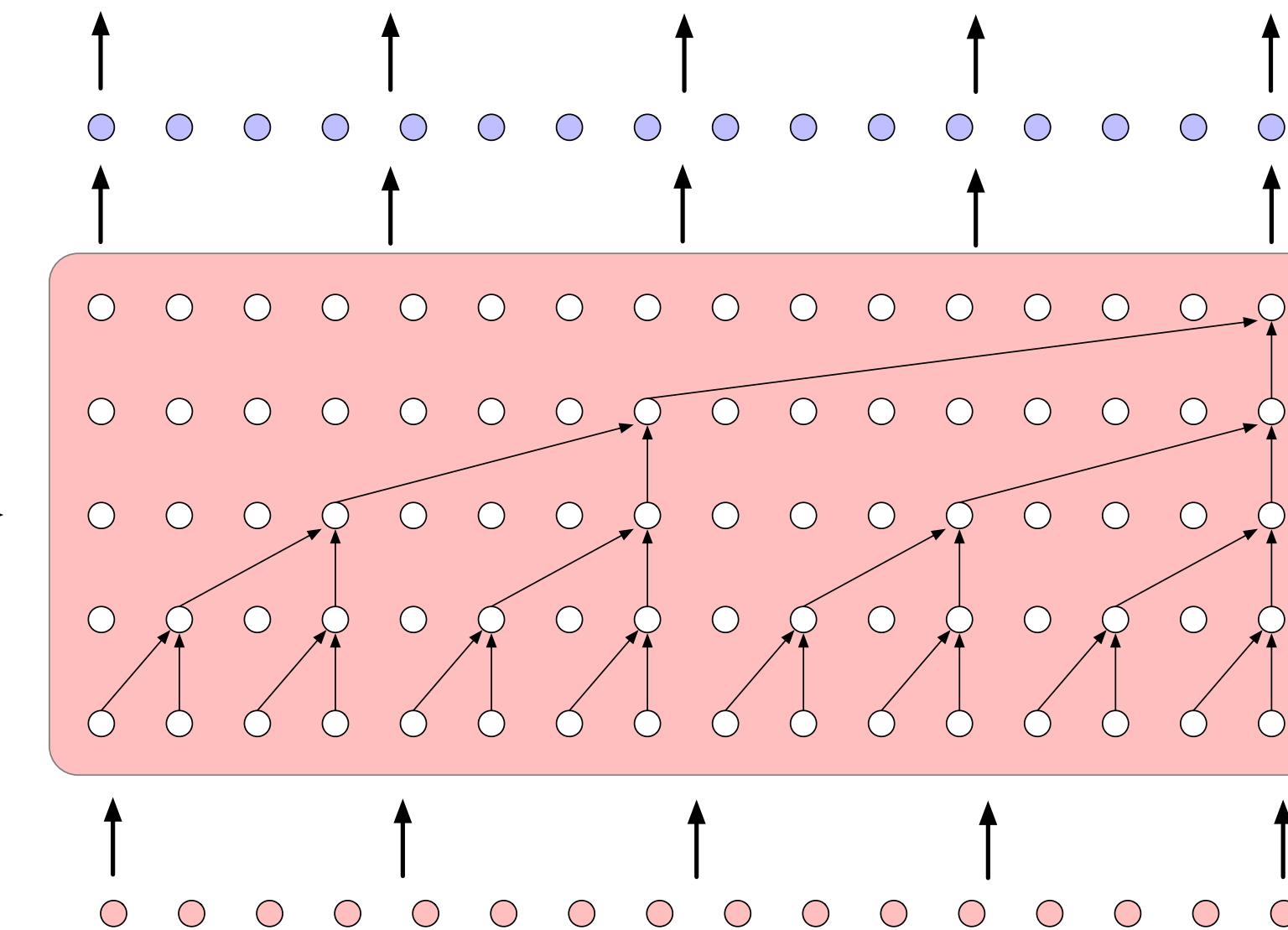
WaveNet Teacher

Linguistic features  $\dashrightarrow$



WaveNet Student

Linguistic features  $\dashrightarrow$



Teacher Output  
 $P(x_i|x_{<i})$

Generated Samples  
 $x_i = g(z_i|z_{<i})$

Student Output  
 $P(x_i|z_{<i})$

Input noise  
 $z_i$

Figure 2: **Overview of Probability Density Distillation.** A pre-trained WaveNet teacher is used to score the samples  $x$  output by the student. The student is trained to minimise the KL-divergence between its distribution and that of the teacher by maximising the log-likelihood of its samples under the teacher and maximising its own entropy at the same time.

# FAMILY OF GENERATIVE MODELS

- **Autoregressive generative models**

- ▶ choose an ordering of the dimensions in  $\mathbf{x}$
- ▶ define the conditionals in the product rule expression of  $p(\mathbf{x})$

$$p(\mathbf{x}) = \prod_{k=1}^D p(x_k | \mathbf{x}_{<k})$$

- ▶ examples: masked autoencoder distribution estimator (MADE), pixelCNN  
neural autoregressive distribution estimator (NADE), spatial LSTM, pixelRNN

- Properties

- ▶ pros:  $p(\mathbf{x})$  is tractable, so easy to train, easy to sample (though slower)
- ▶ cons: doesn't have a natural latent representation

