Institut
des algorithmes
d'apprentissage
de Montréal

MILA

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

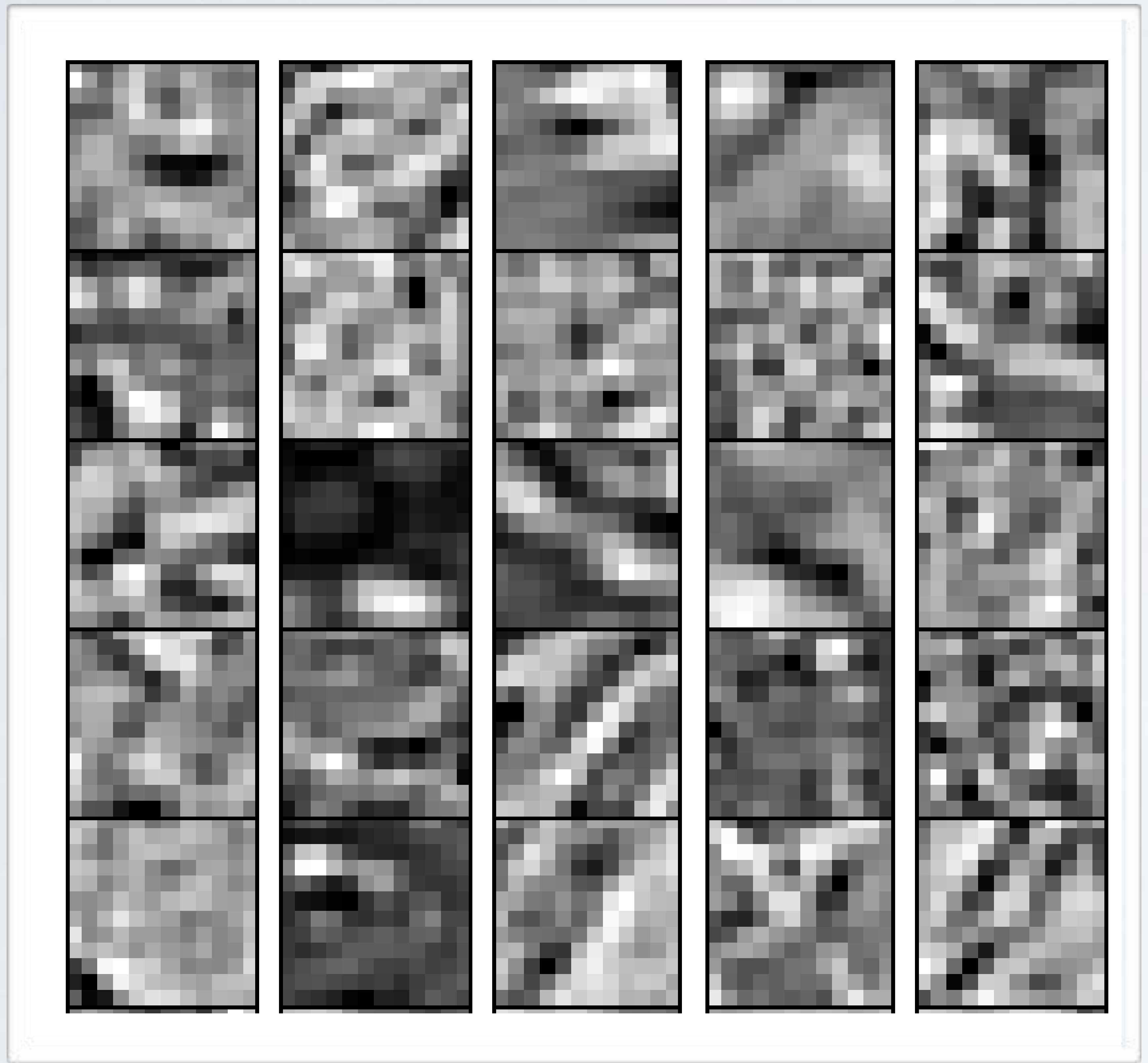
real image

Why is one a character and not the other?

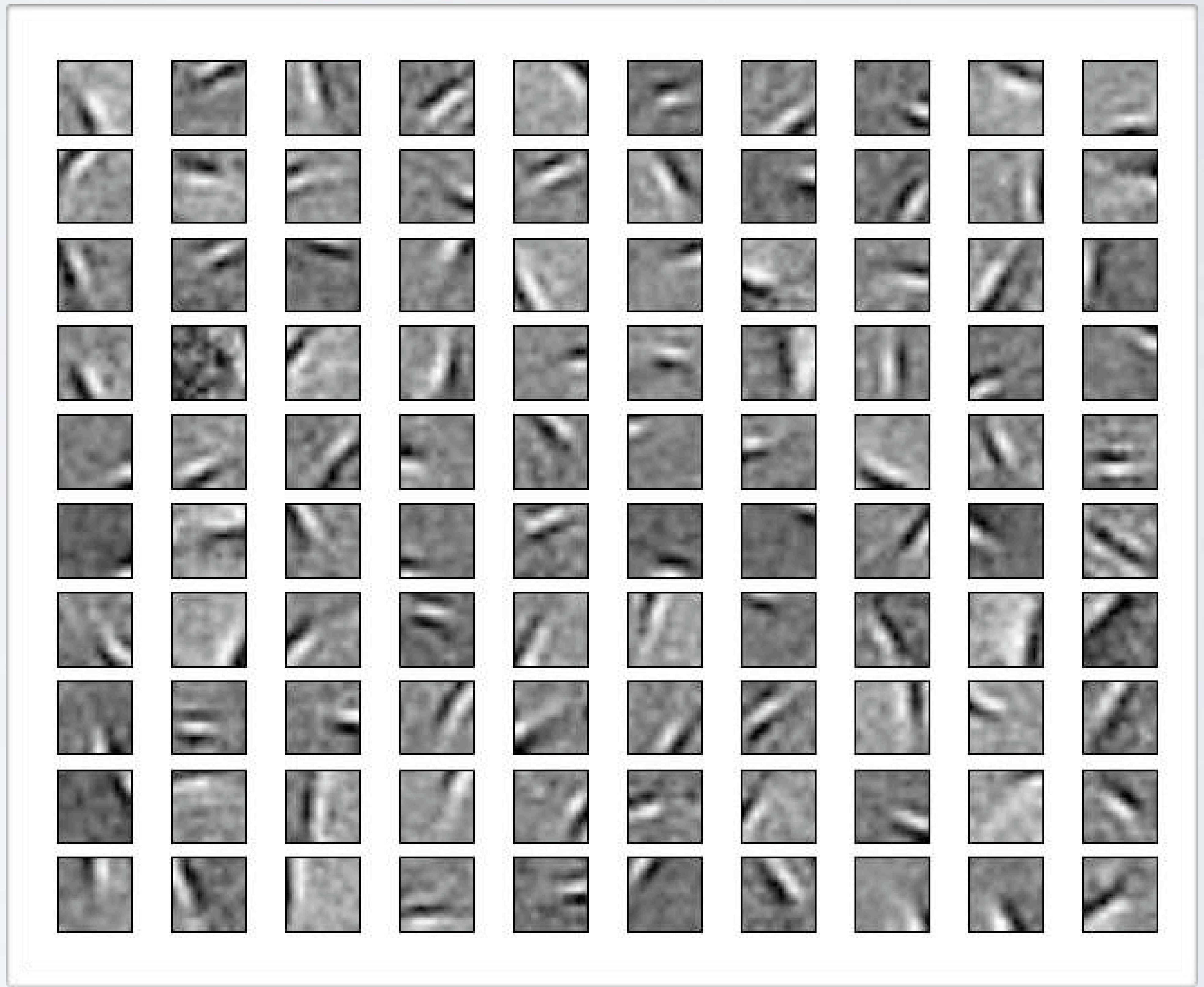
random image

WHY GENERATIVE MODELS

- Perhaps that's what the brain does?
  - sparse coding neurons vs. neurons in V1



Data

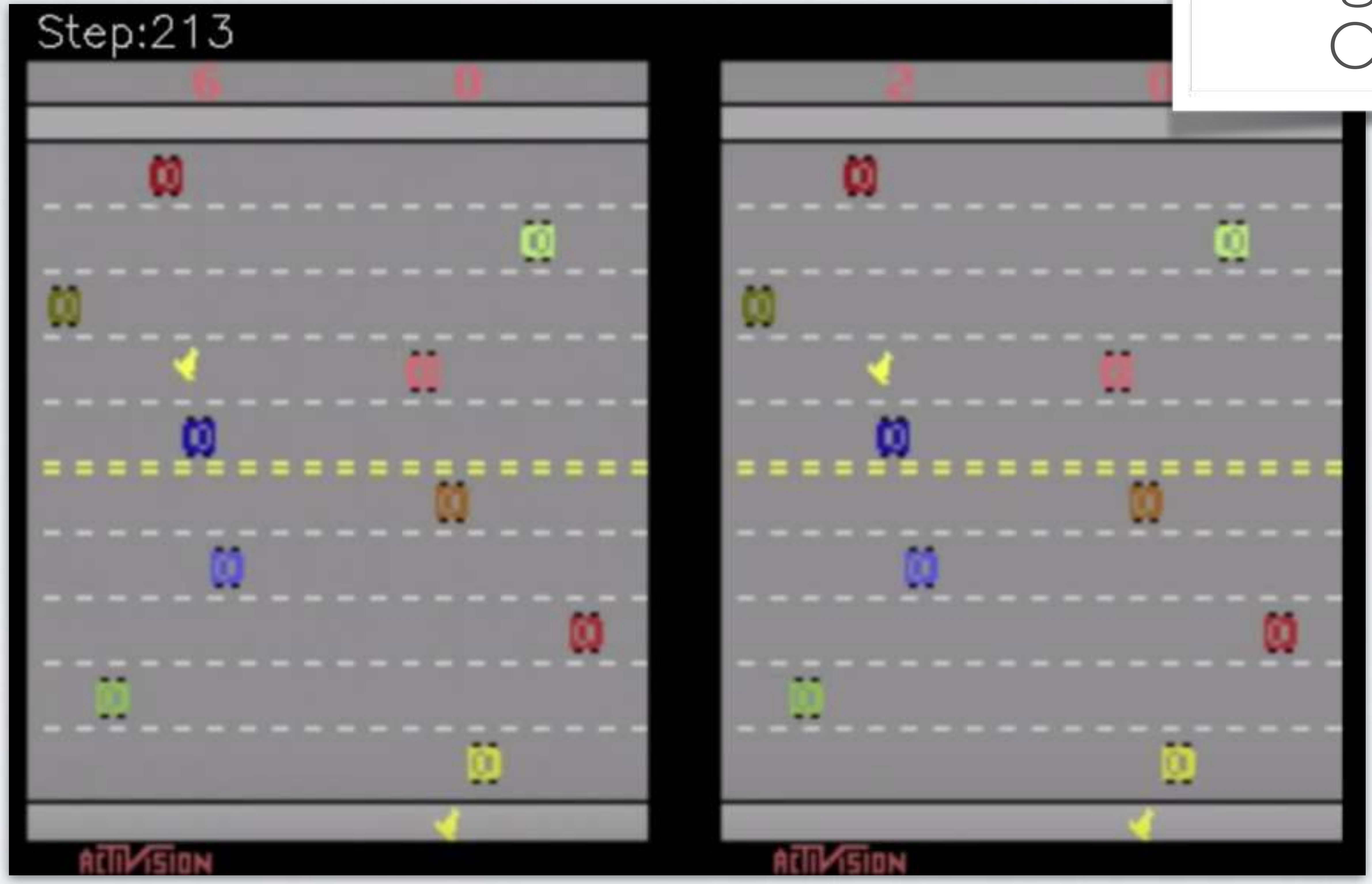


Learned representation

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

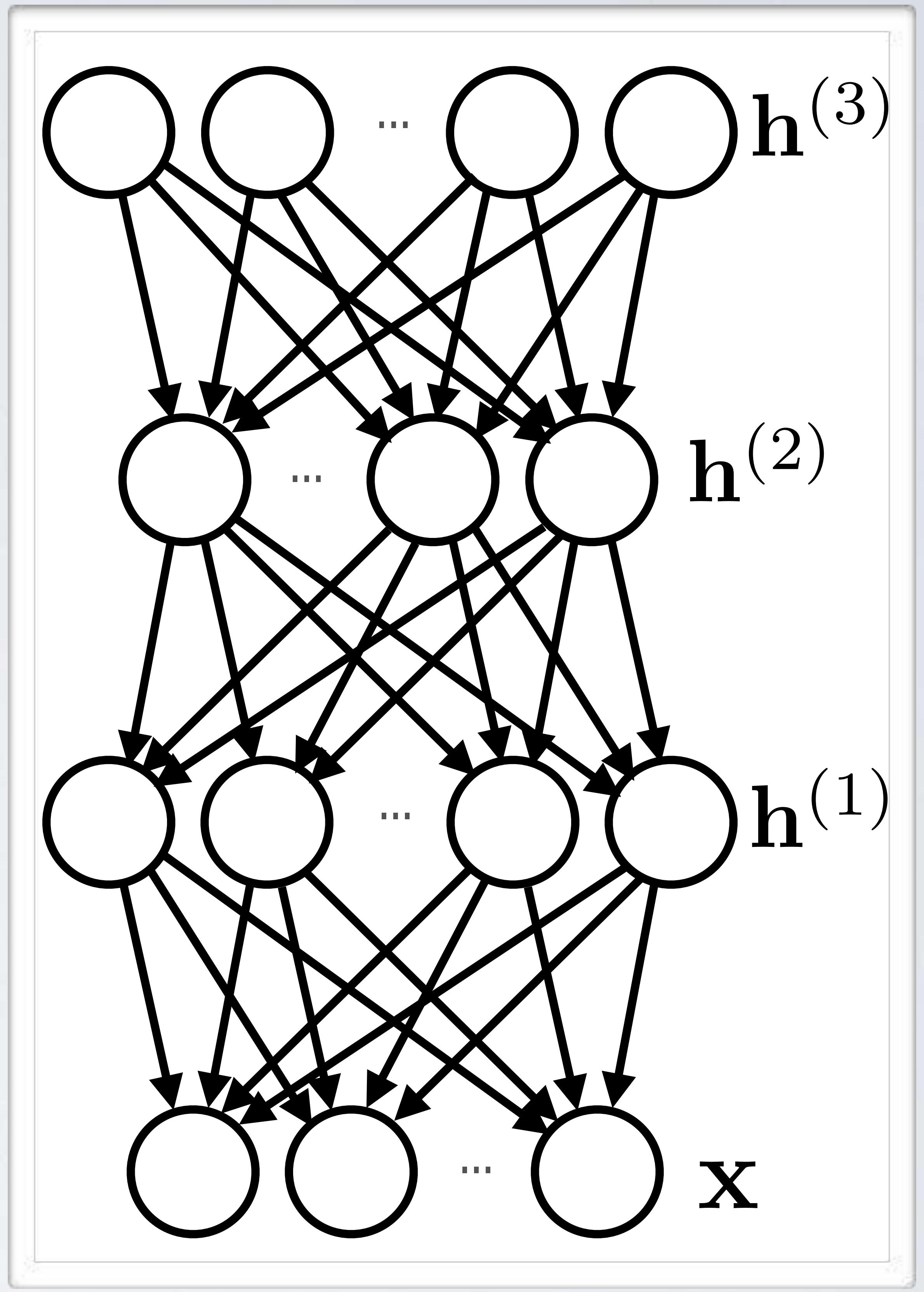
Amortised MAP Inference for Image Super-resolution
Sønderby, Caballero, Theis, Shi, Huszár. arXiv 2016

FAMILY OF GENERATIVE MODELS

- **Directed graphical models**
  - define prior over top-most latent representation
  - define conditionals from top latent representation to observation

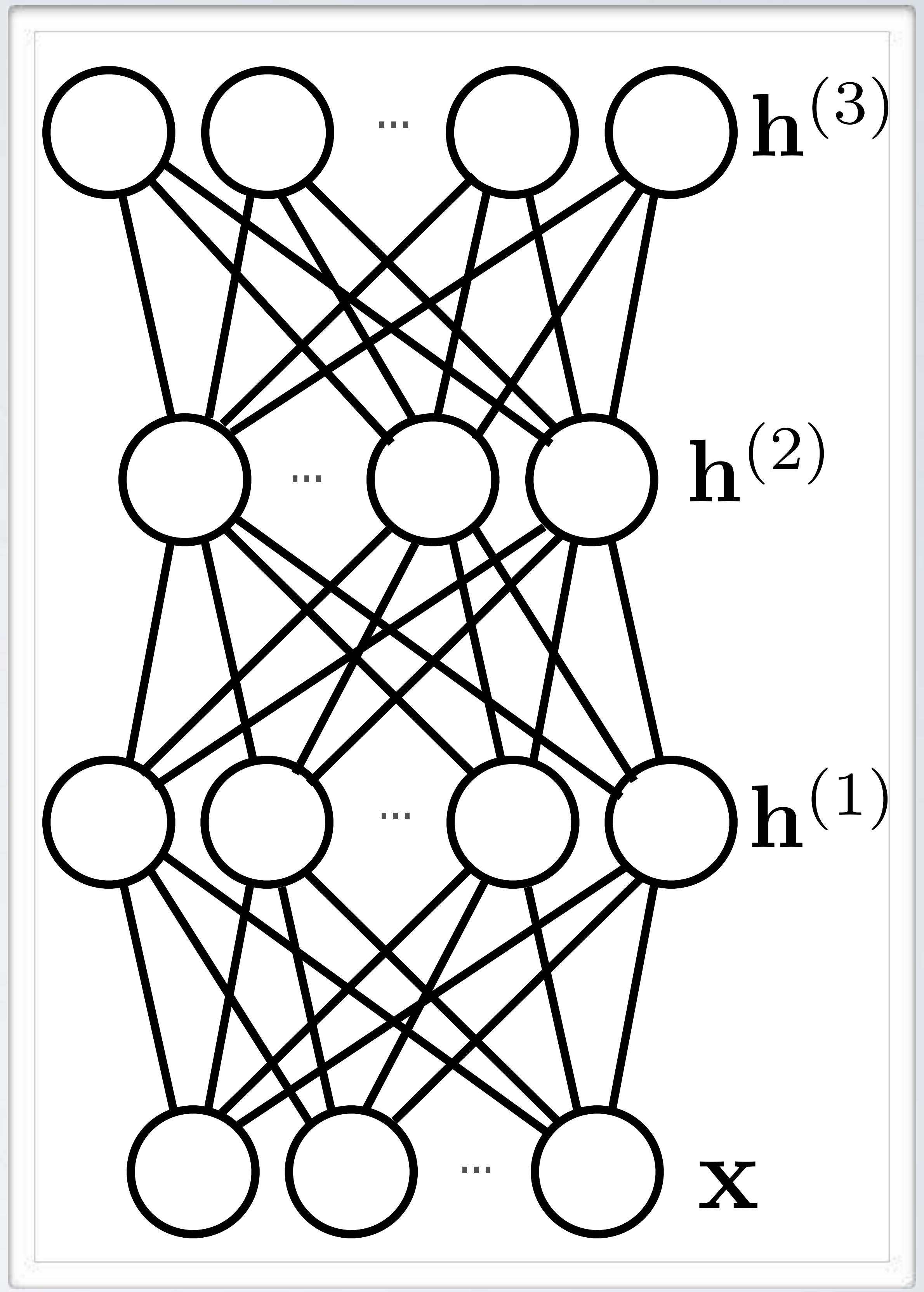
$$p(\mathbf{x}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = p(\mathbf{x}|\mathbf{h}^{(1)})p(\mathbf{h}^{(1)}|\mathbf{h}^{(2)})p(\mathbf{h}^{(2)}|\mathbf{h}^{(3)})p(\mathbf{h}^{(3)})$$

  - 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**
  - 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

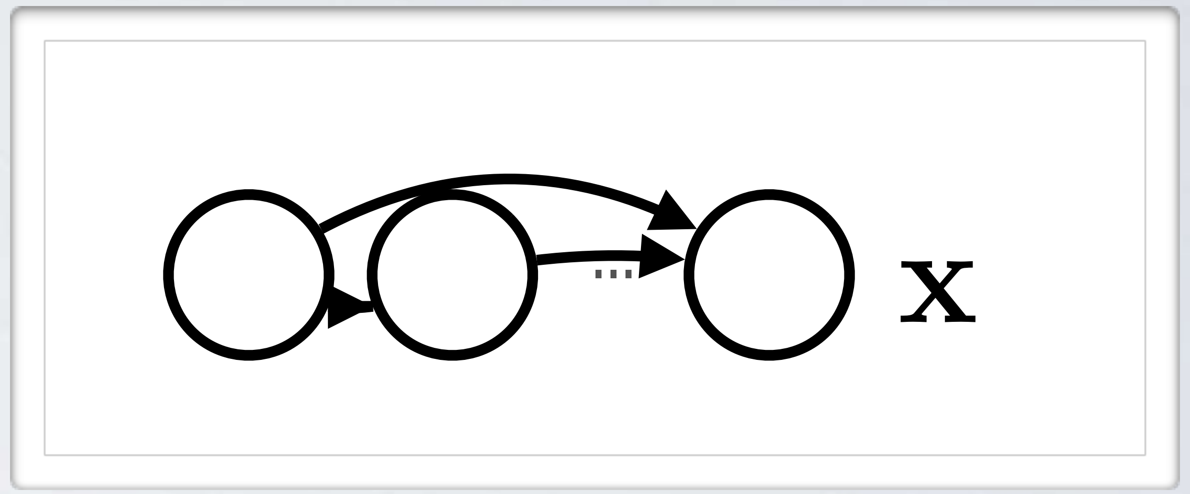
- Properties
  - *pros*: can compute $p(\mathbf{x})$ up to a multiplicative factor (true for RBMs not general BMs)
  - *cons*: hard to sample from (MCMC), $p(\mathbf{x})$ is intractable, so hard to train

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 $p(\mathbf{x})$

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

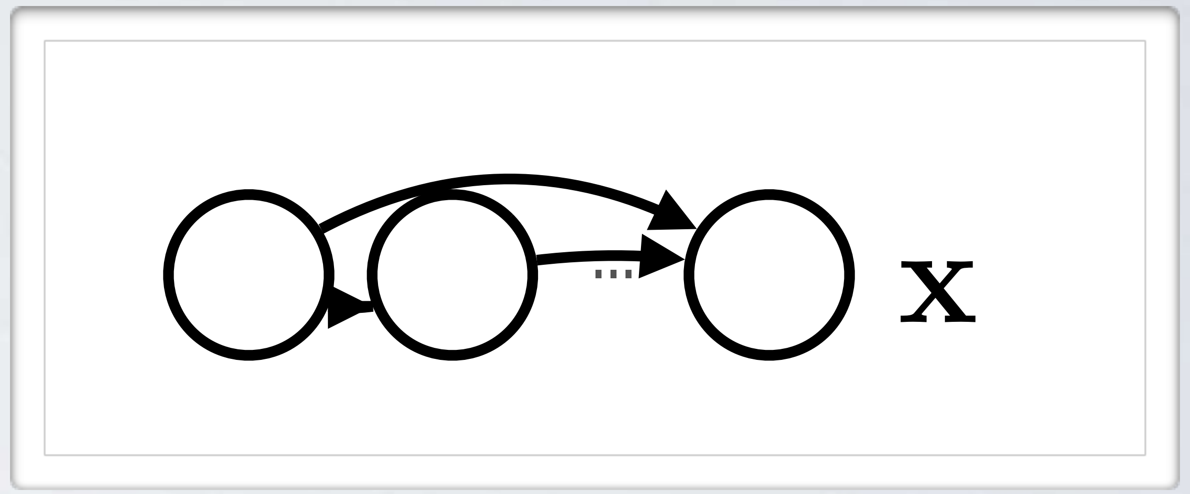
  - examples: masked autoencoder distribution estimator (MADE), pixelCNN
    neural autoregressive distribution estimator (NADE), spatial LSTM, pixelRNN



- 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

MILA

On the menu:

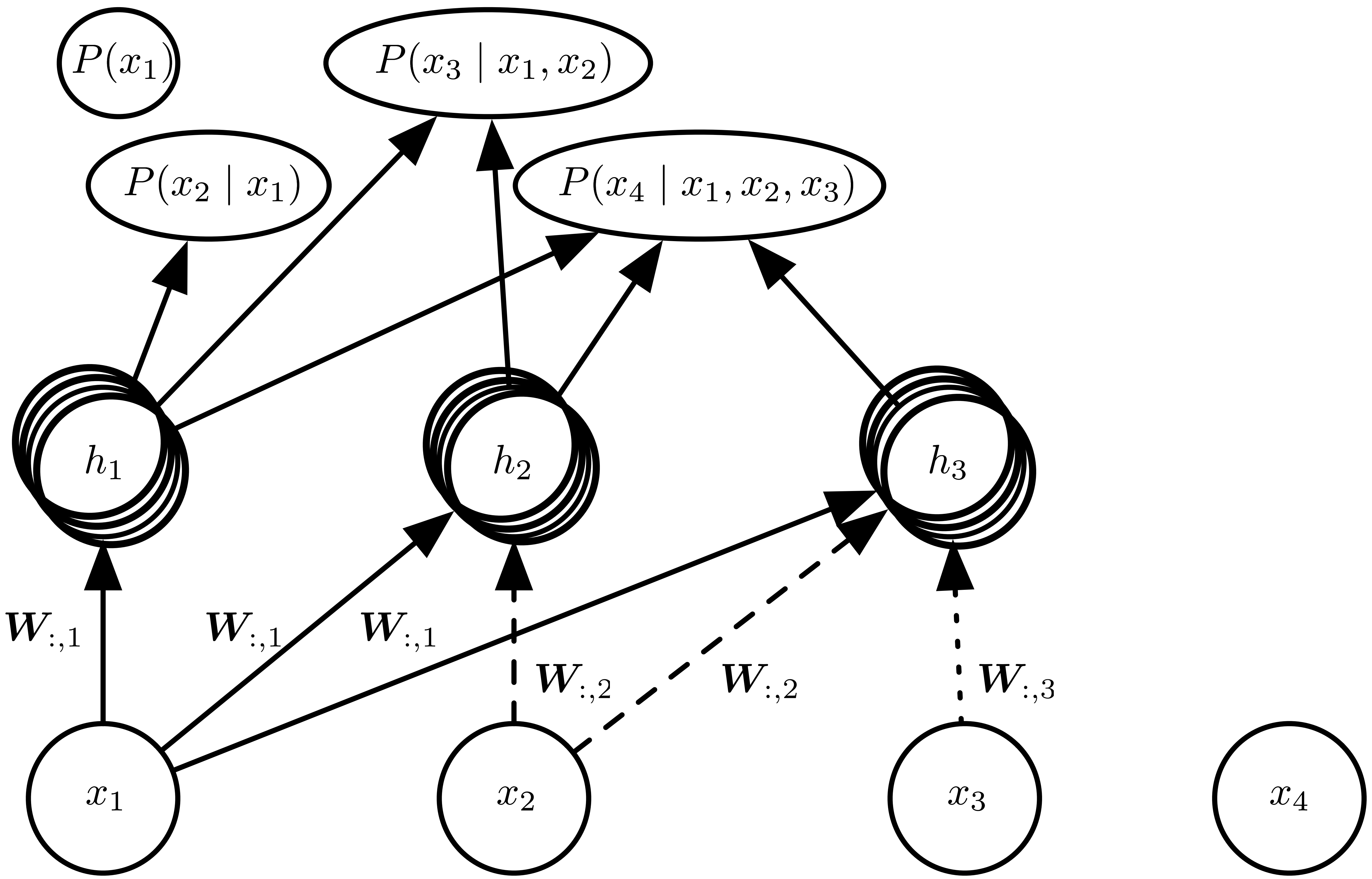
- NADE: Neural Autoregressive Density Estimator
- MADE: Masked Autoencoder for Density Estimator
- PixelCNN Autoregressive CNN
- PixelRNN: Autoregressive RNN for images
- WaveNet: Audio synthesis model, autoregressive dialated CNN
- PixelVAE: VAE with PixelCNN decoder
- Bonus models

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 $\boldsymbol{W'}_{j,k,i}$ from the i-th input $\boldsymbol{x}_i$ to the k-th element of the j-th group of hidden unit $h_k^{(j)}$ ($j \geq i$) are shared among the groups:

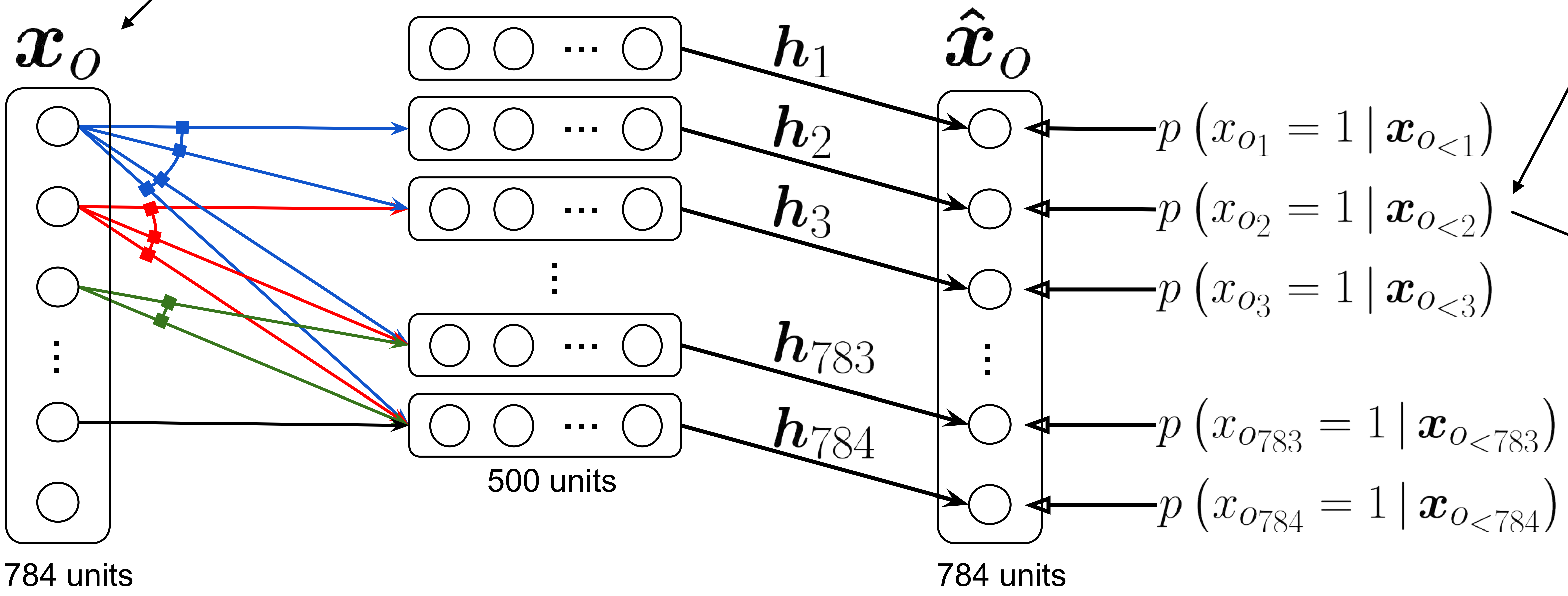
$$\boldsymbol{W'}_{j,k,i} = \boldsymbol{W}_{k,i}$$



Neural autoregressive distribution estimator (NADE)

some ordering of **x**

given previous variables in the ordering



NADE is for binary data

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

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

Neural autoregressive distribution estimator (NADE)

Negative
log-likelihood

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

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

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

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

NADE Training

NADE training is done using "**teacher forcing**"

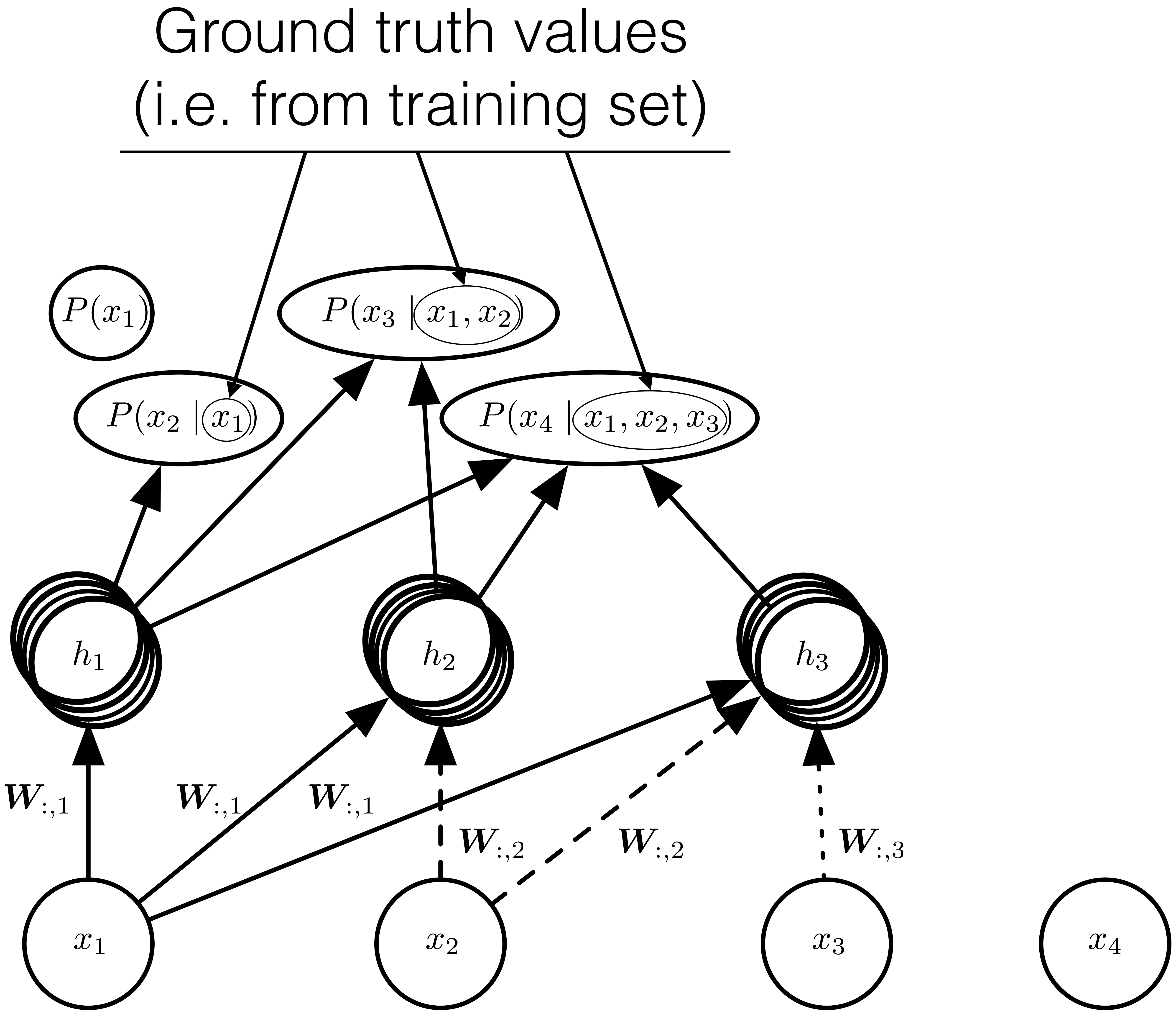
- Ground truth values of the pixels are used for conditioning when predicting subsequent values

Negative log-likelihood

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

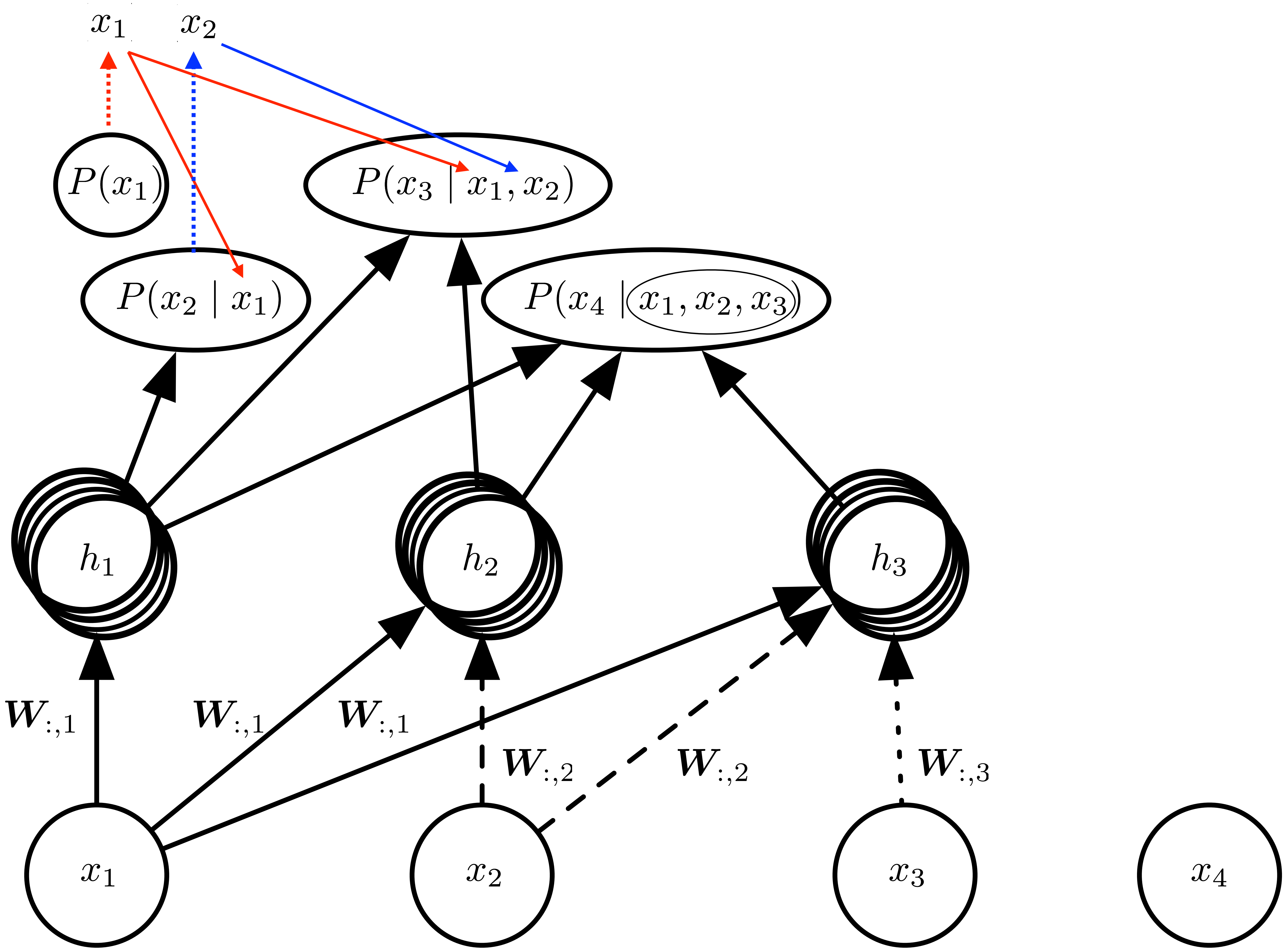
$$p(x_{o_d} = 1 \mid x_{o_{<d}}) = \sigma(V_{o_d,:}\mathbf{h}_d + b_{o_d})$$

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



NADE 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.

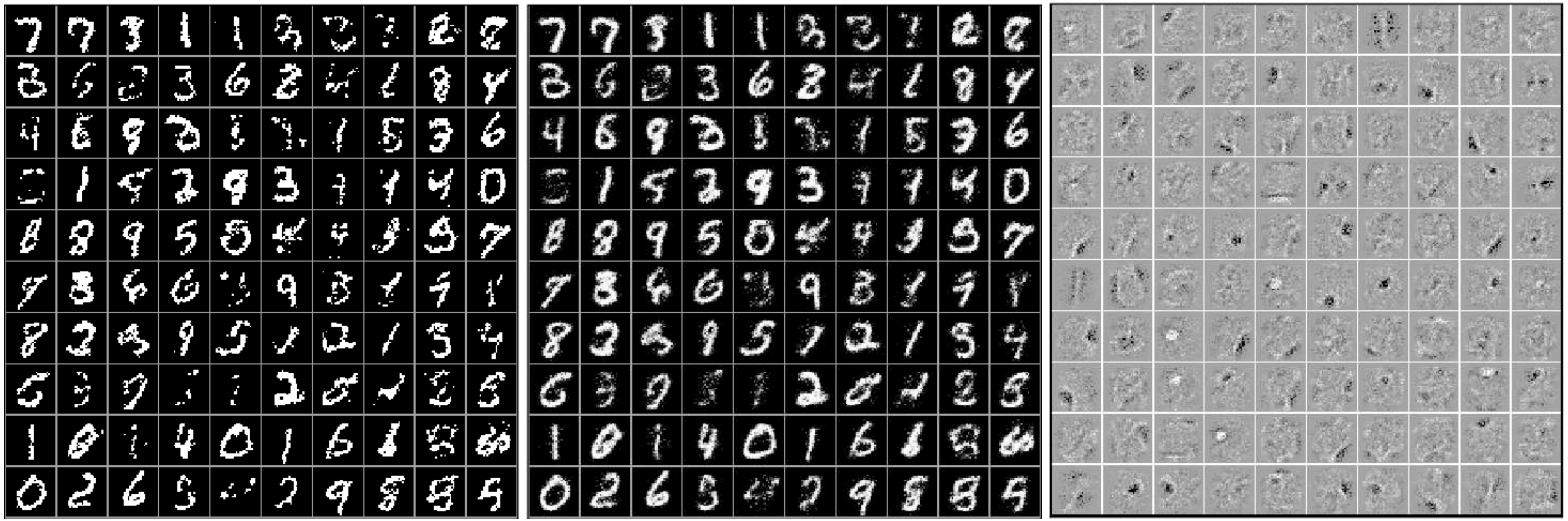


Figure 2: **(Left)**: samples from NADE trained on a binary version of MNIST. **(Middle)**: probabilities from which each pixel was sampled. **(Right)**: visualization of some of the rows of $\mathbf{W}$. This figure is better seen on a computer screen.

Neural autoregressive distribution estimator (NADE)

Extensions

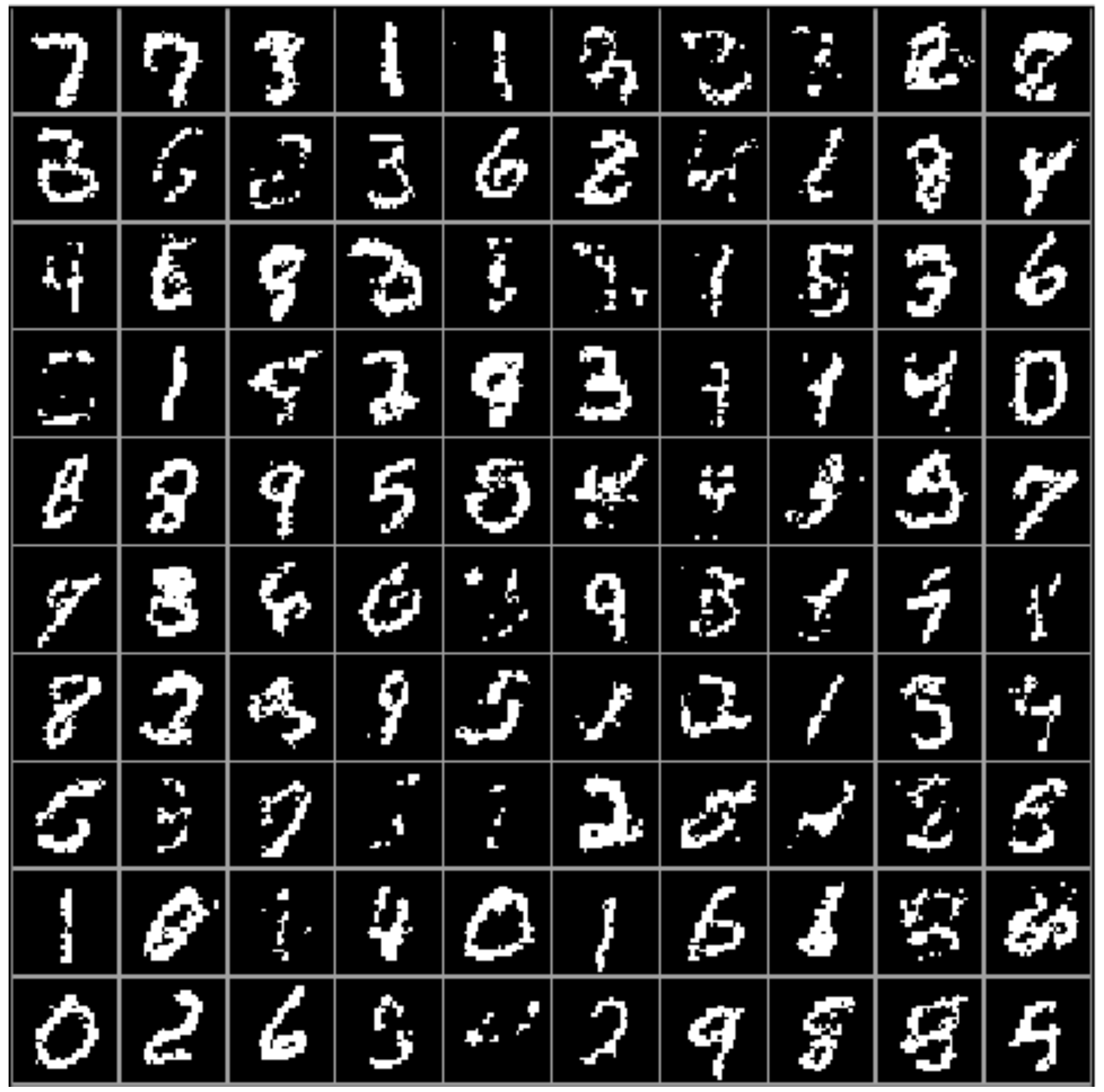
- Real-valued NADE (RNADE): conditionals are modelled by a mixture of gaussians.

- 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)

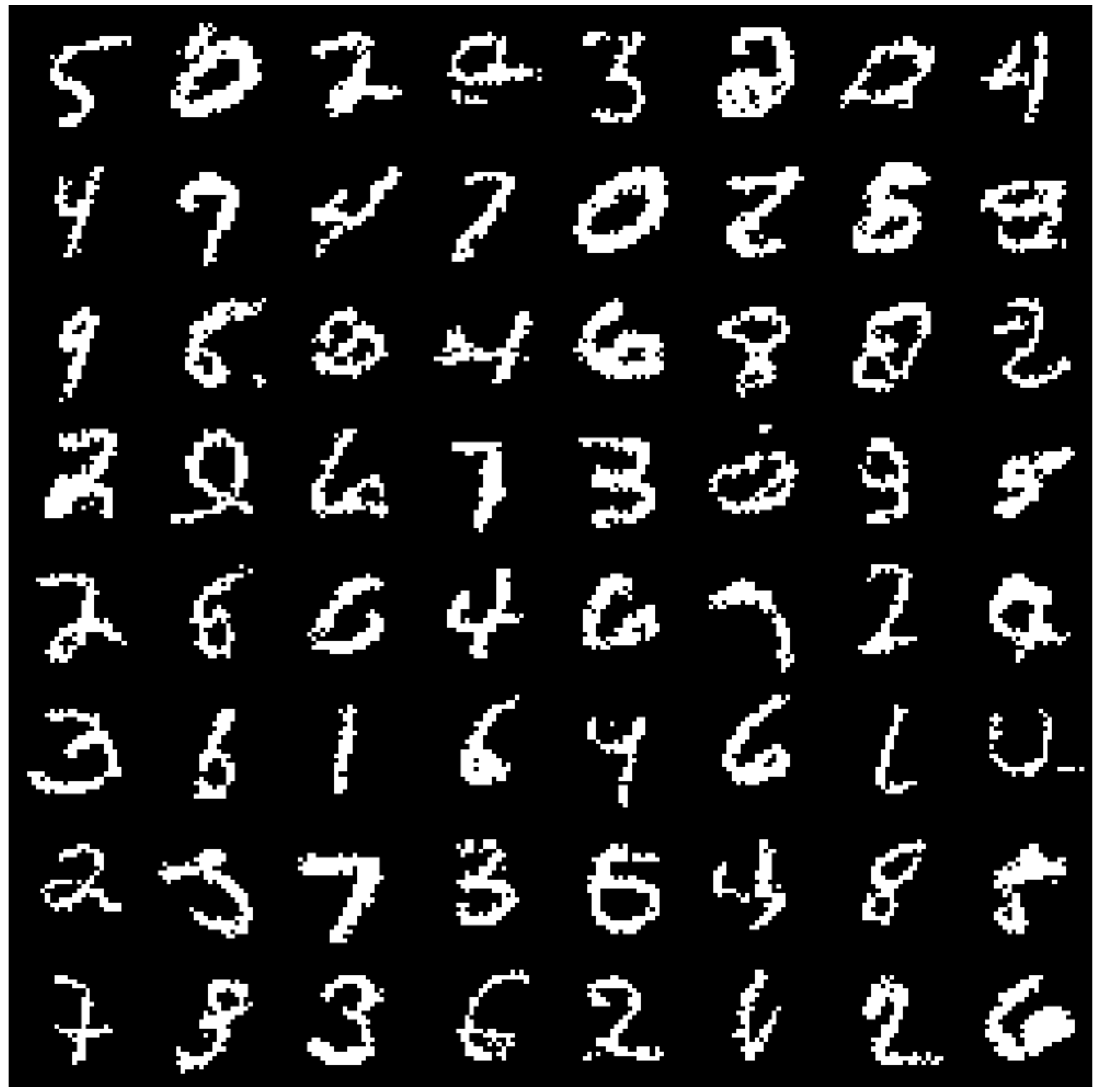
Uria, Benigno, et al. "Neural Autoregressive Distribution Estimation." Journal of Machine Learning Research 17.205 (2016): 1-37.

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 (Germain, Gregor, Murray and Larochelle, ICML 2015)

MILA

- **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 $x_d, \ldots, x_D$, *again relative to some ordering.*

  ➡ I.e. For each of these paths, at least one connection in the weight matrix must be 0.

MADE (Germain, Gregor, Murray and Larochelle, ICML 2015)

- **Question**: How do you construct an autoregressive autoencoder?

- Convenient way of zeroing connections is to elementwise-multiply each matrix by a binary mask matrix $\boldsymbol{M}$, whose entries that are set to 0 correspond to the connections we wish to remove.

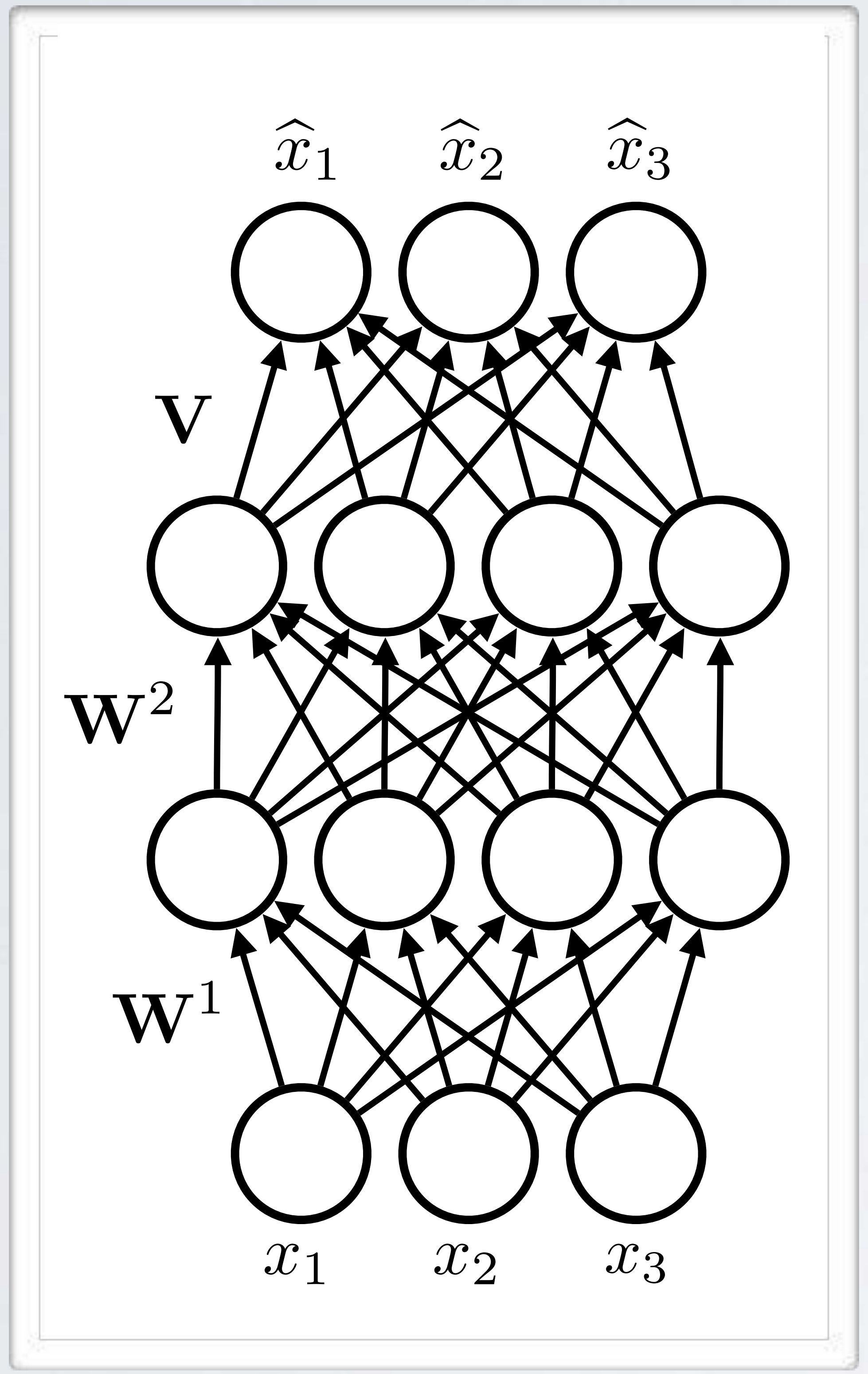
- For a single hidden layer autoencoder:

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

MASKED AUTOENCODER DISTRIBUTION ESTIMATION

Topics: MADE (Germain et al. 2015)

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

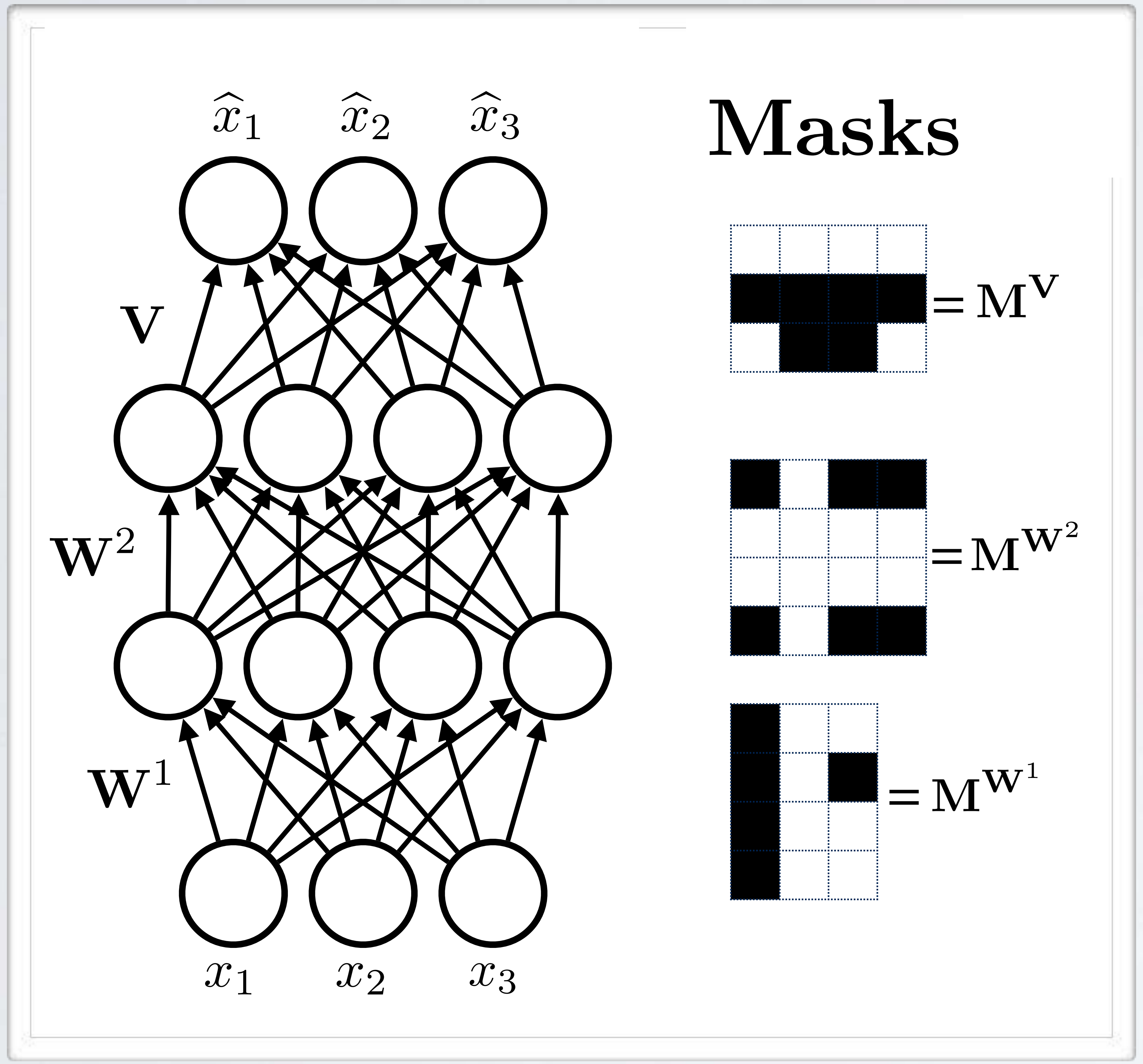


- Generalization of the work from Bengio and Bengio (2000)

MASKED AUTOENCODER DISTRIBUTION ESTIMATION

Topics: MADE (Germain et al. 2015)

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

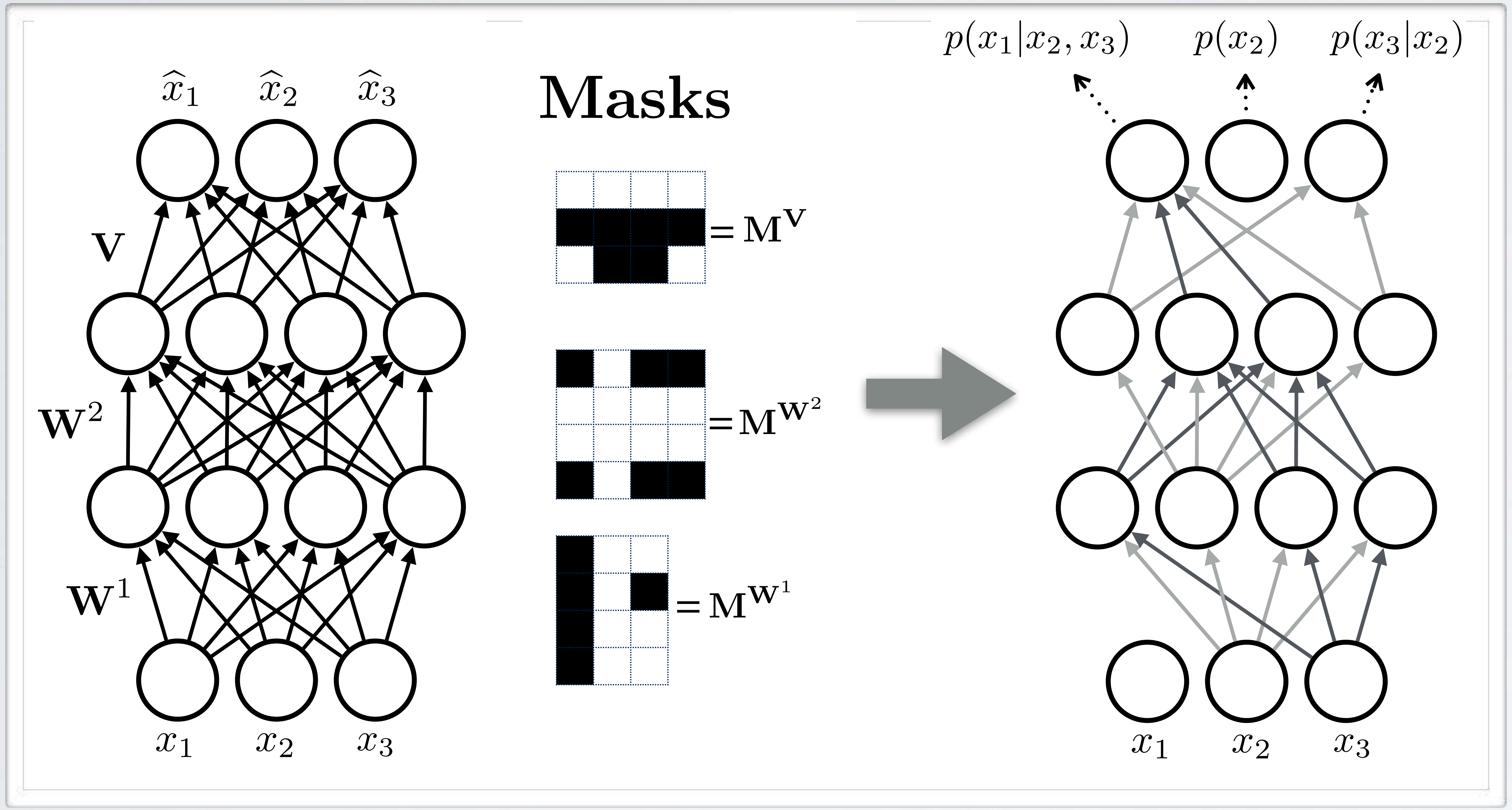


- Generalization of the work from Bengio and Bengio (2000)

MASKED AUTOENCODER DISTRIBUTION ESTIMATION

Topics: MADE (Germain et al. 2015)

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

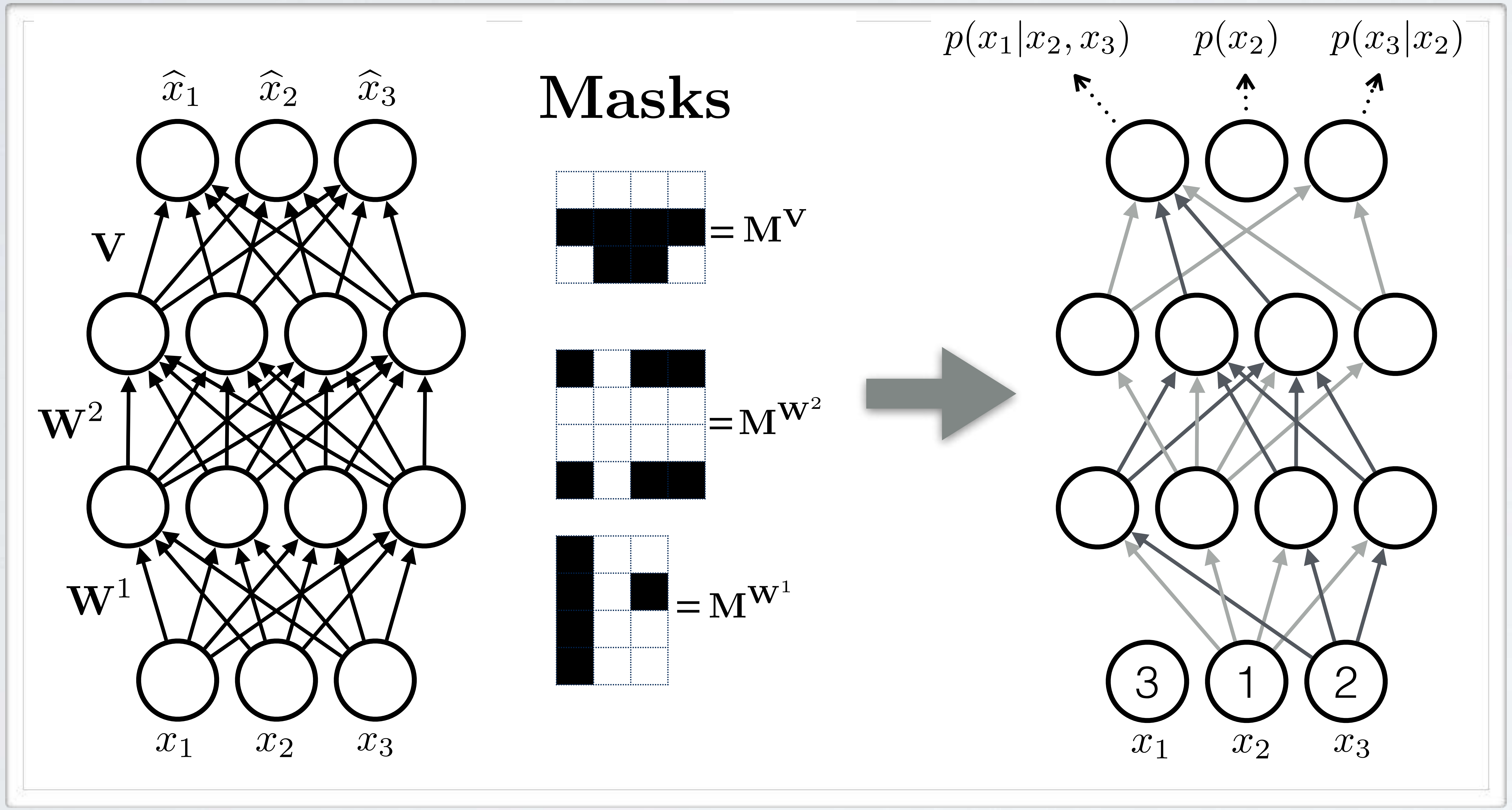


- Generalization of the work from Bengio and Bengio (2000)

MASKED AUTOENCODER DISTRIBUTION ESTIMATION

Topics: MADE (Germain et al. 2015)

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

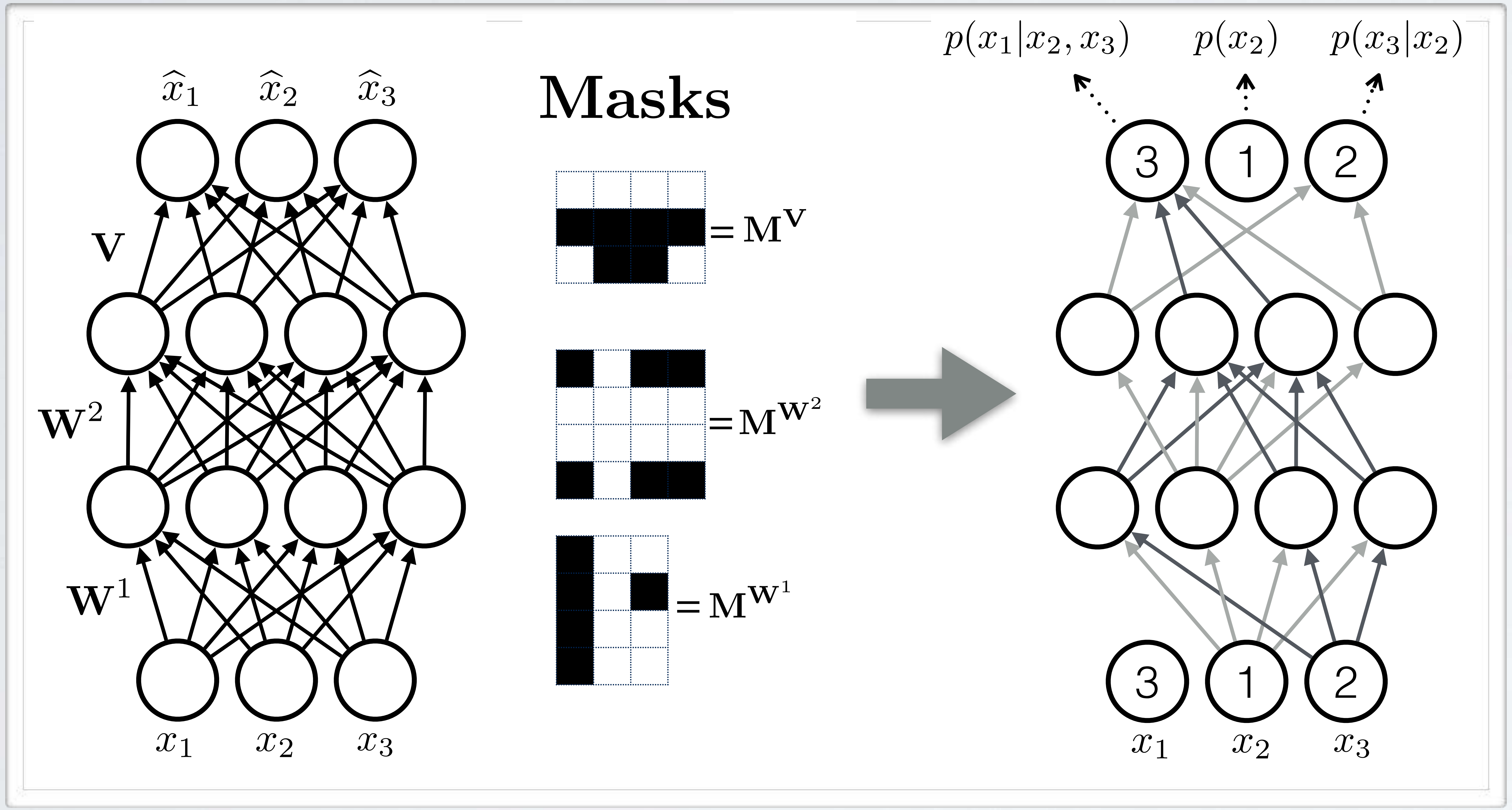


- Generalization of the work from Bengio and Bengio (2000)

MASKED AUTOENCODER DISTRIBUTION ESTIMATION

Topics: MADE (Germain et al. 2015)

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

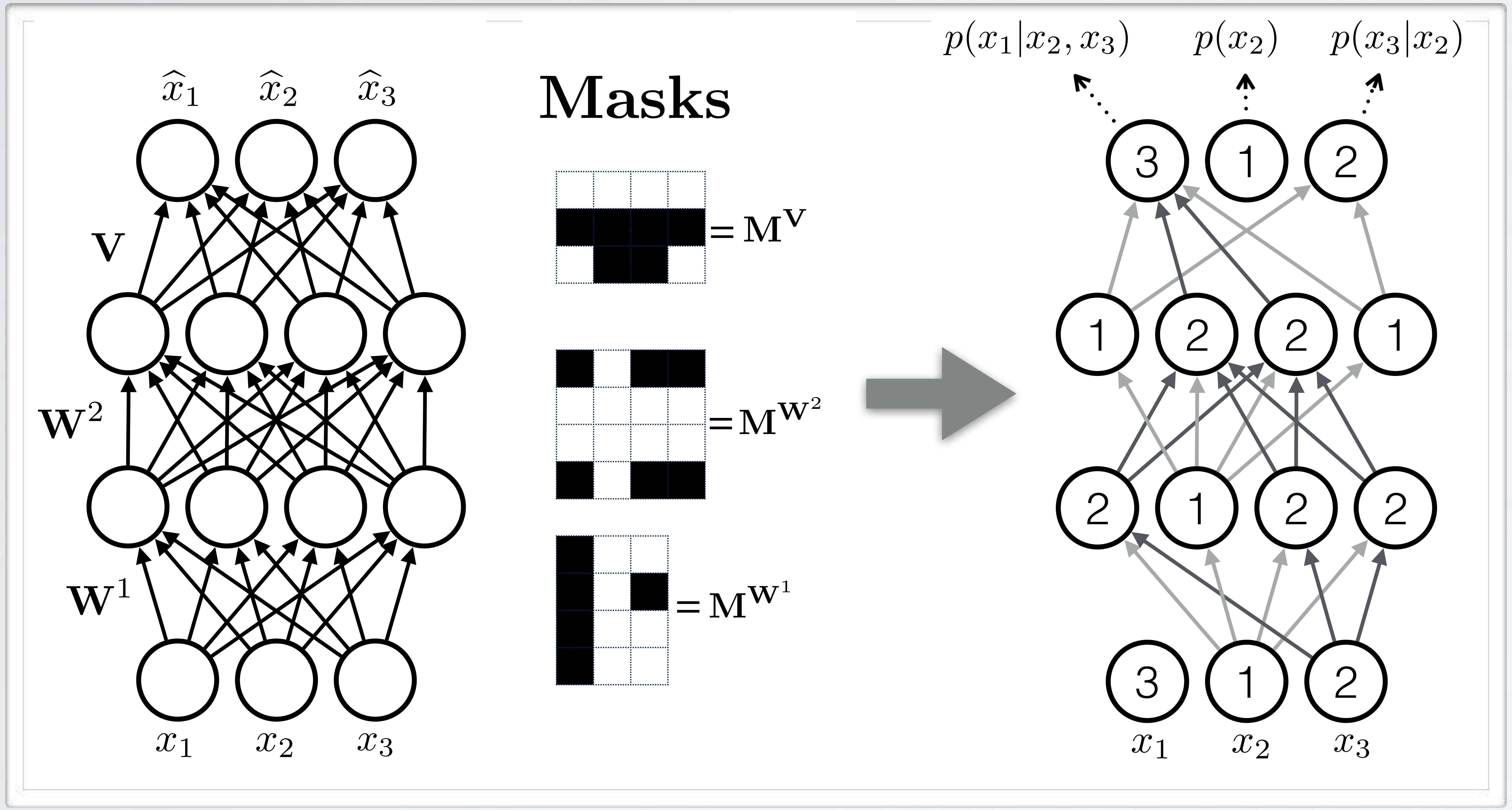


- Generalization of the work from Bengio and Bengio (2000)

MASKED AUTOENCODER DISTRIBUTION ESTIMATION

Topics: MADE (Germain et al. 2015)

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



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

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

- 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

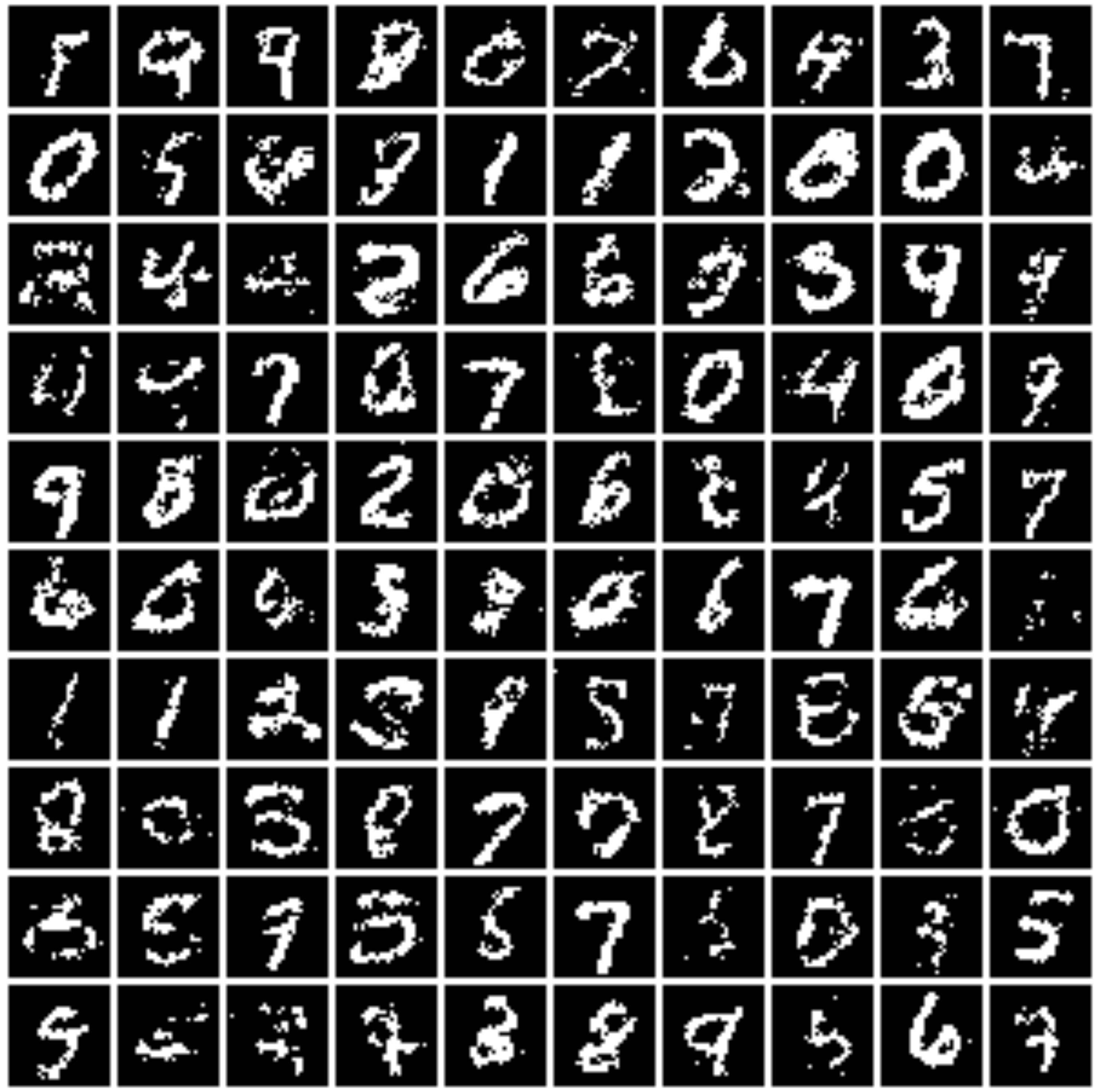
$$\hat{\mathbf{x}} = \text{decode}(\text{encode}(\mathbf{x}))$$

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

NLL criterion for a binary **x**

Germain, Mathieu, et al. "MADE: Masked Autoencoder for Distribution Estimation." ICML. 2015.

Masked Autoencoder for Distribution Estimation (MADE)

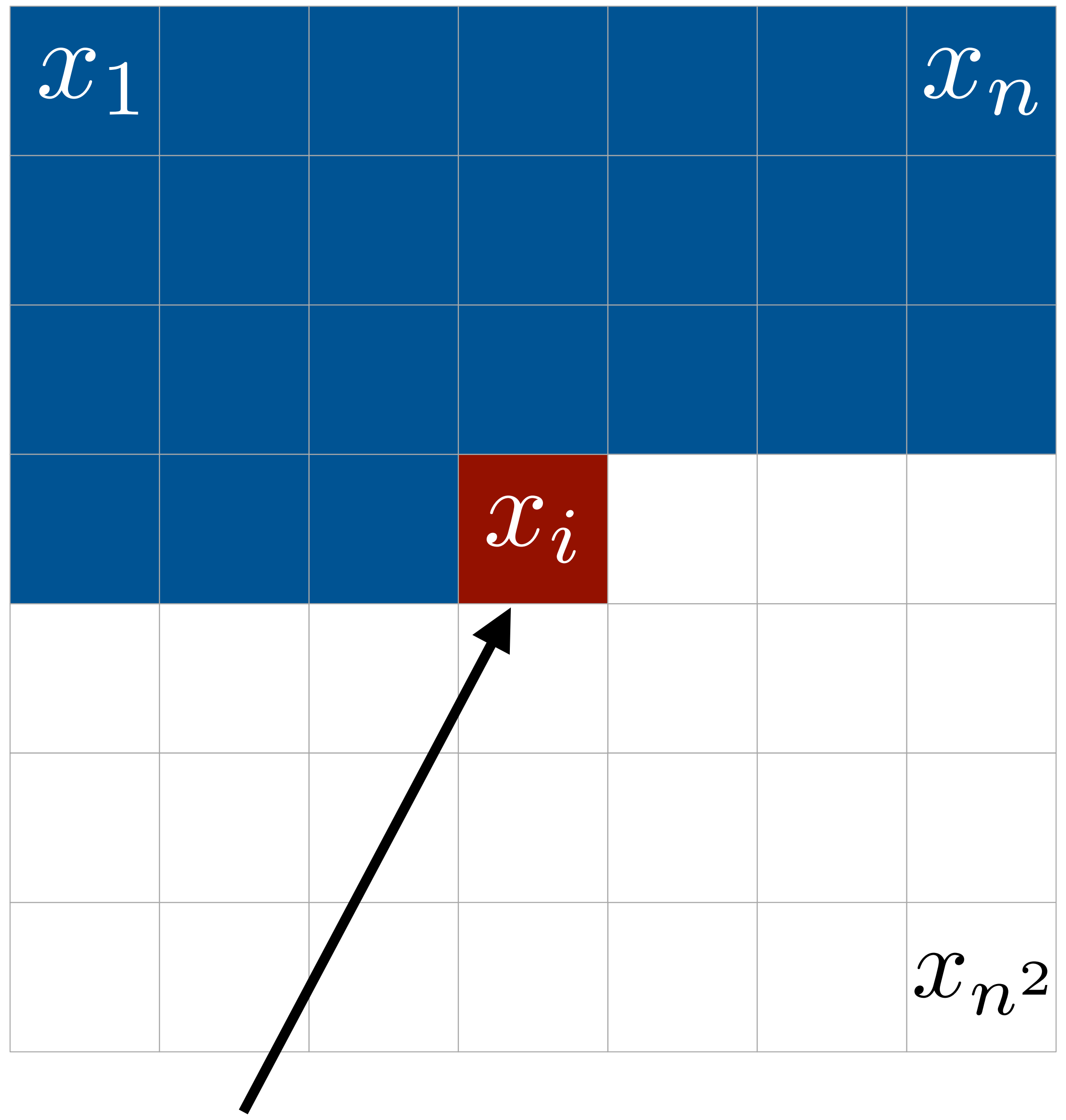


Binarized MNIST samples

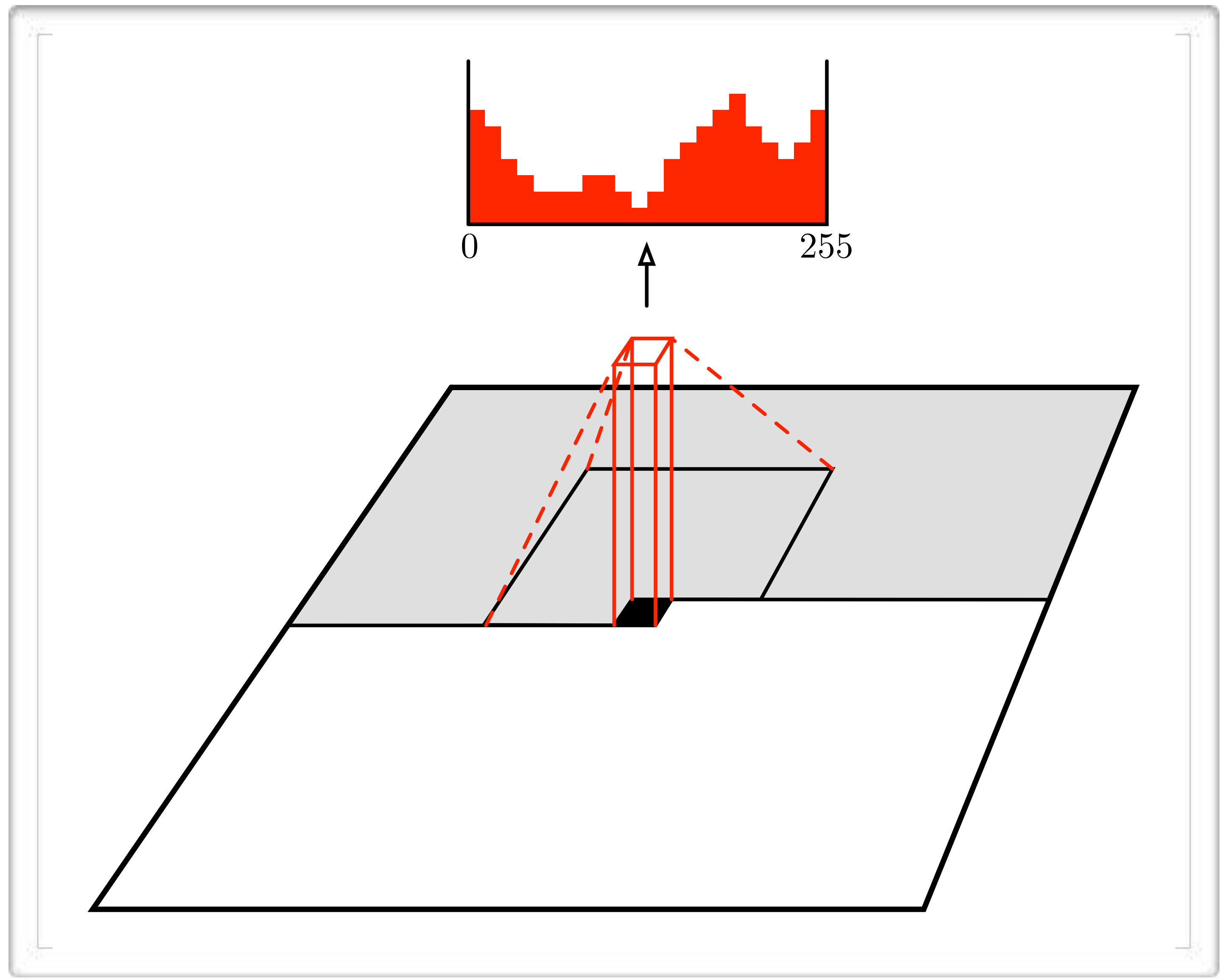
Germain, Mathieu, et al. "MADE: Masked Autoencoder for Distribution Estimation." ICML. 2015.

PixelCNN

Idea: use masked convolutions to enforce the autoregressive relationship



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

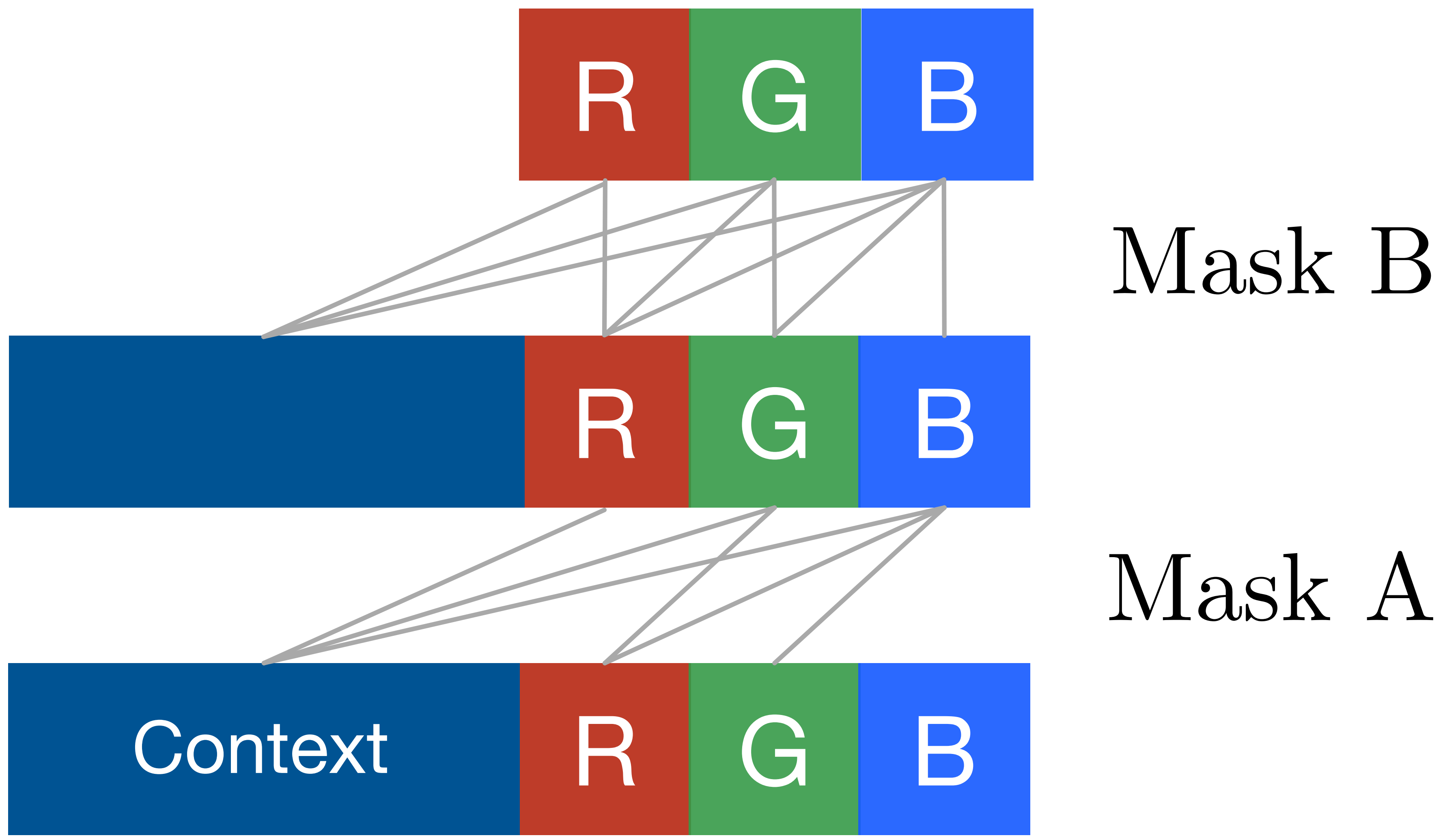


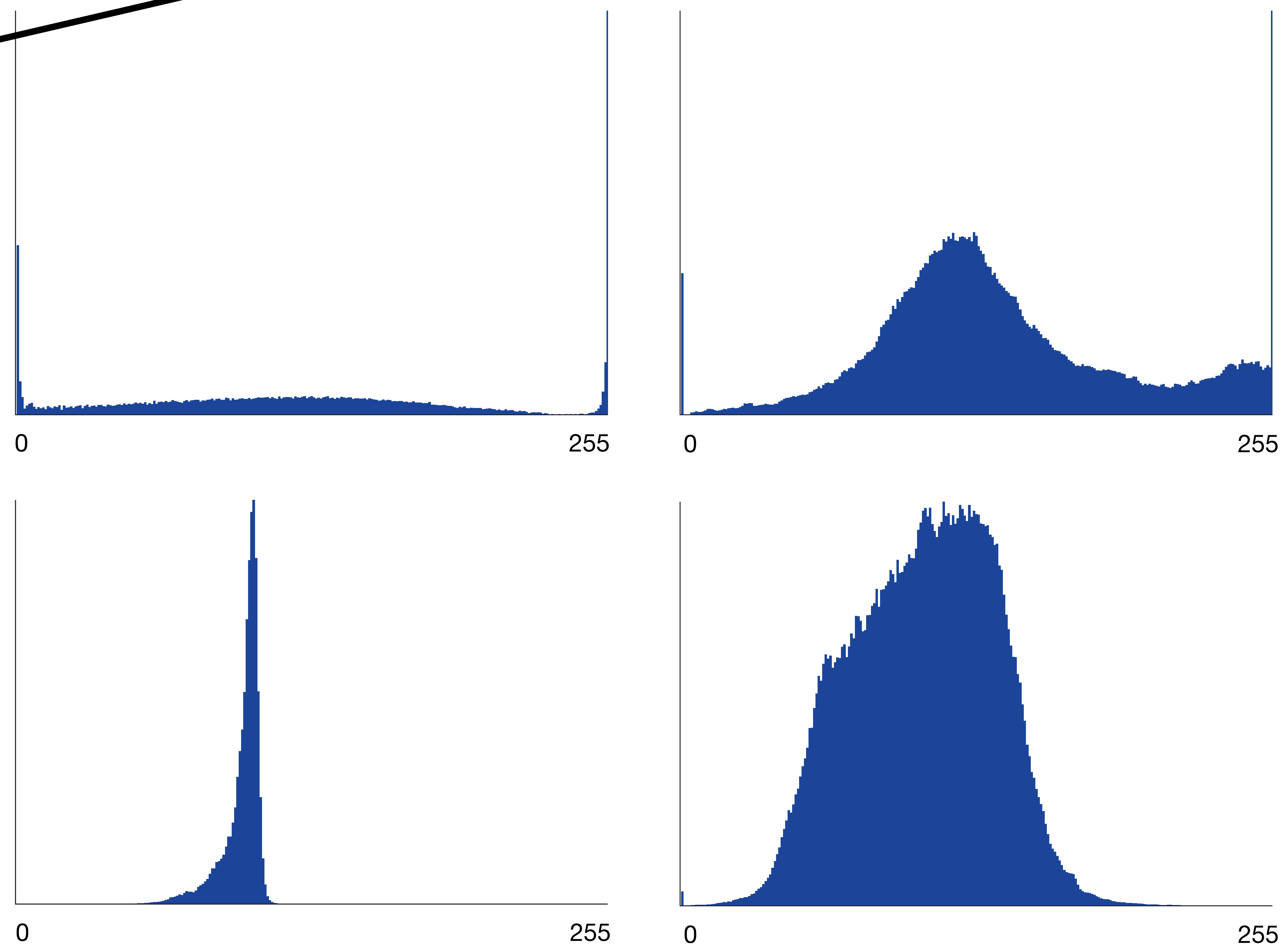
Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." arXiv preprint arXiv:1601.06759 (2016).

PixelCNN

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

autoregressive over color channels



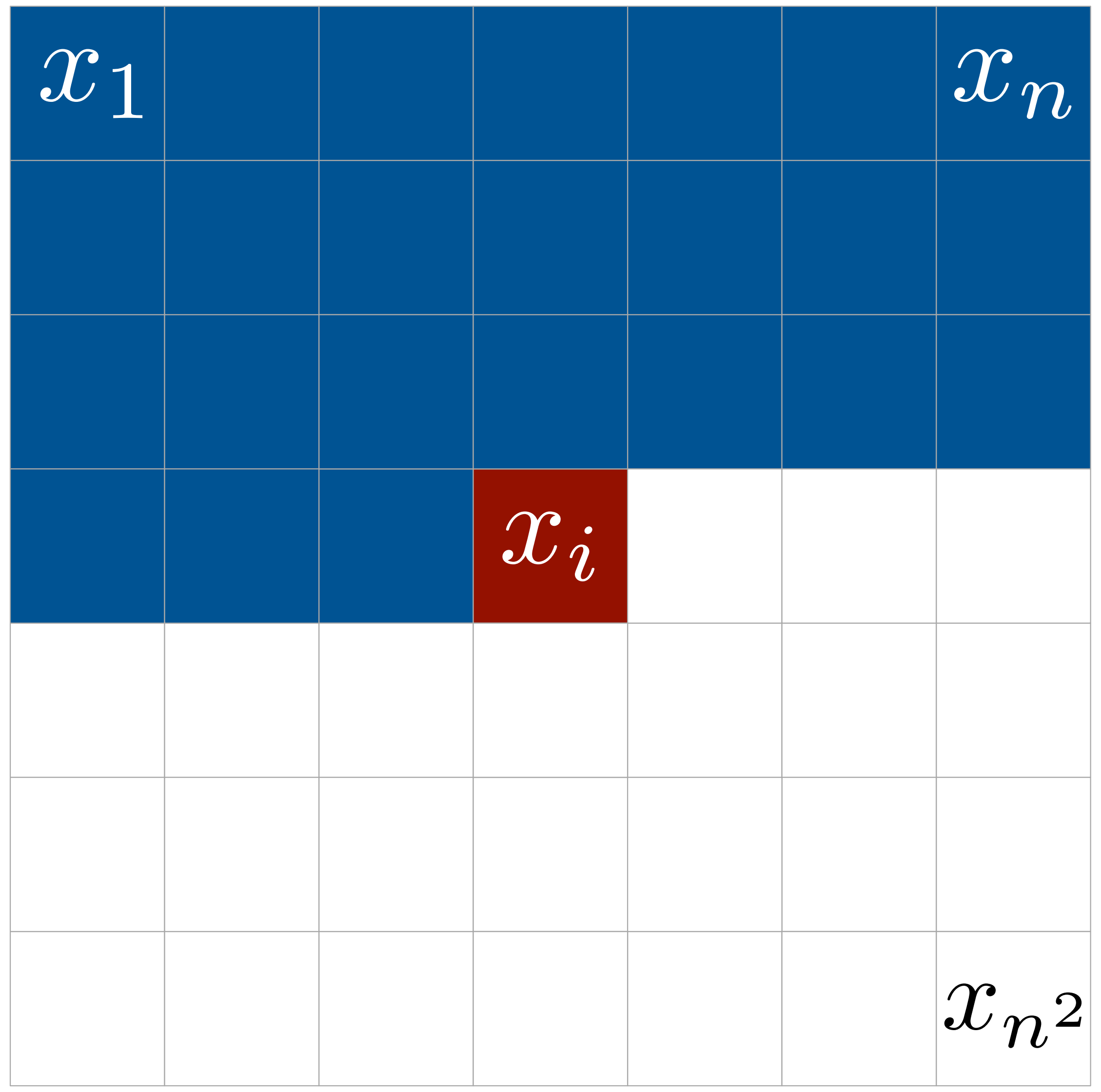


8-bits pixel values (multinoulli distribution)

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

PixelCNN

How can convolutions make this raster scan faster?



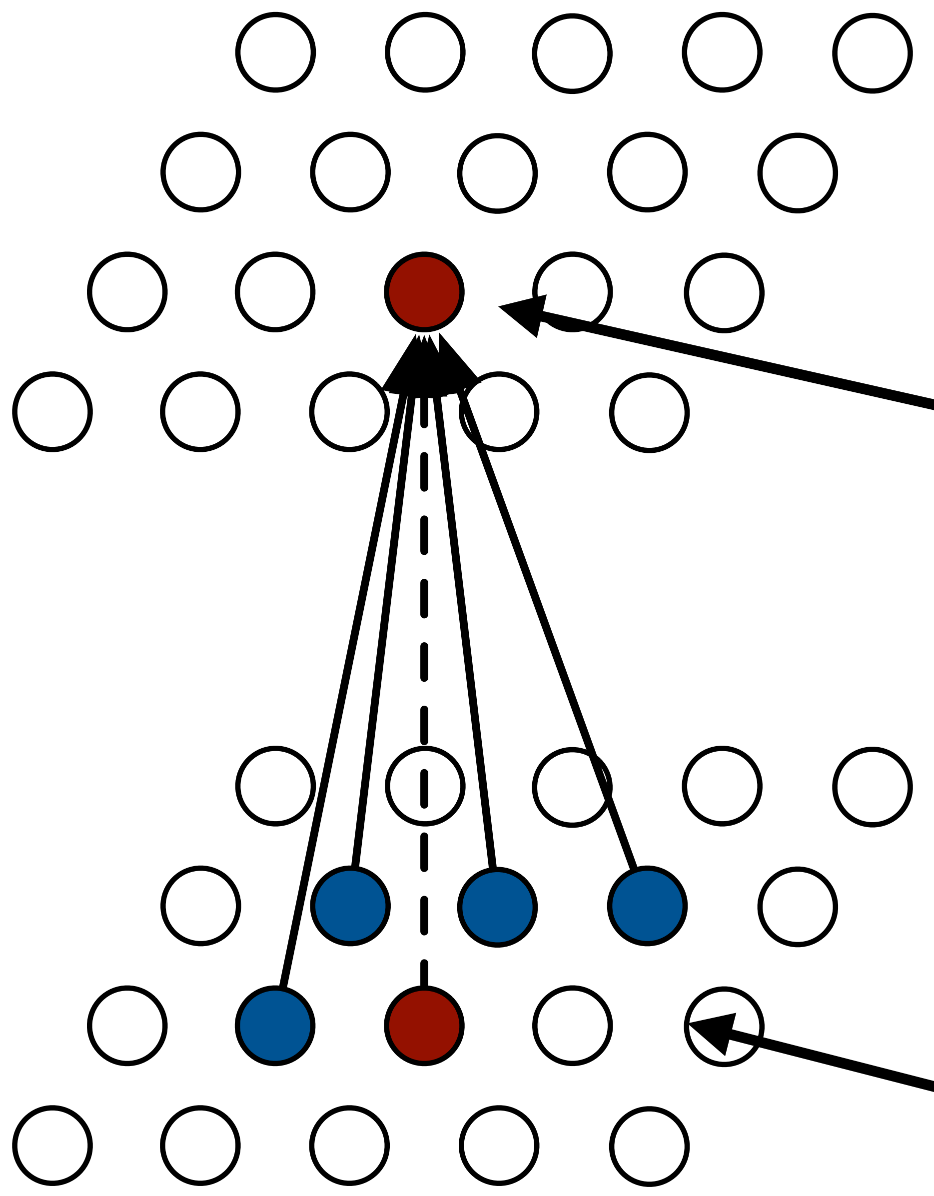
Use a stack of masked convolutions

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

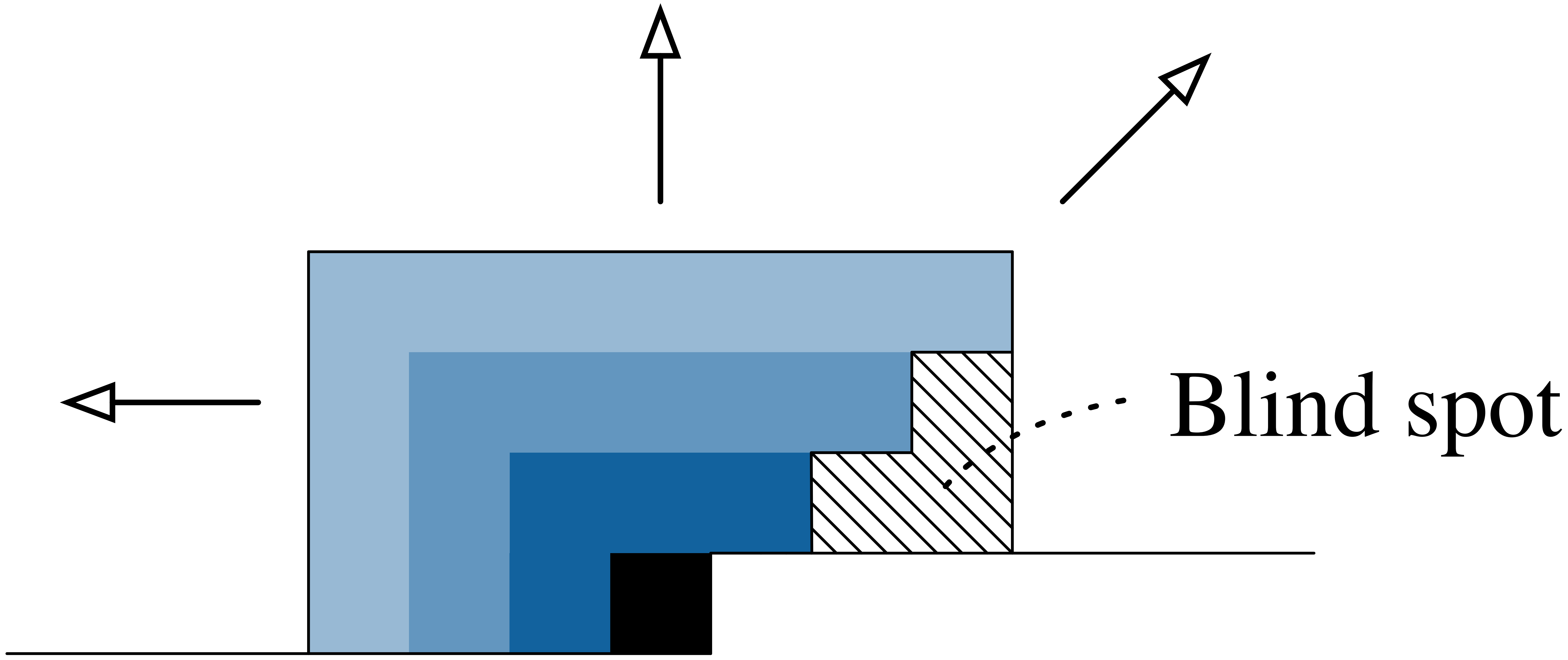
masked convolution

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

Improving PixelCNN

There is a problem with this form of masked convolution.

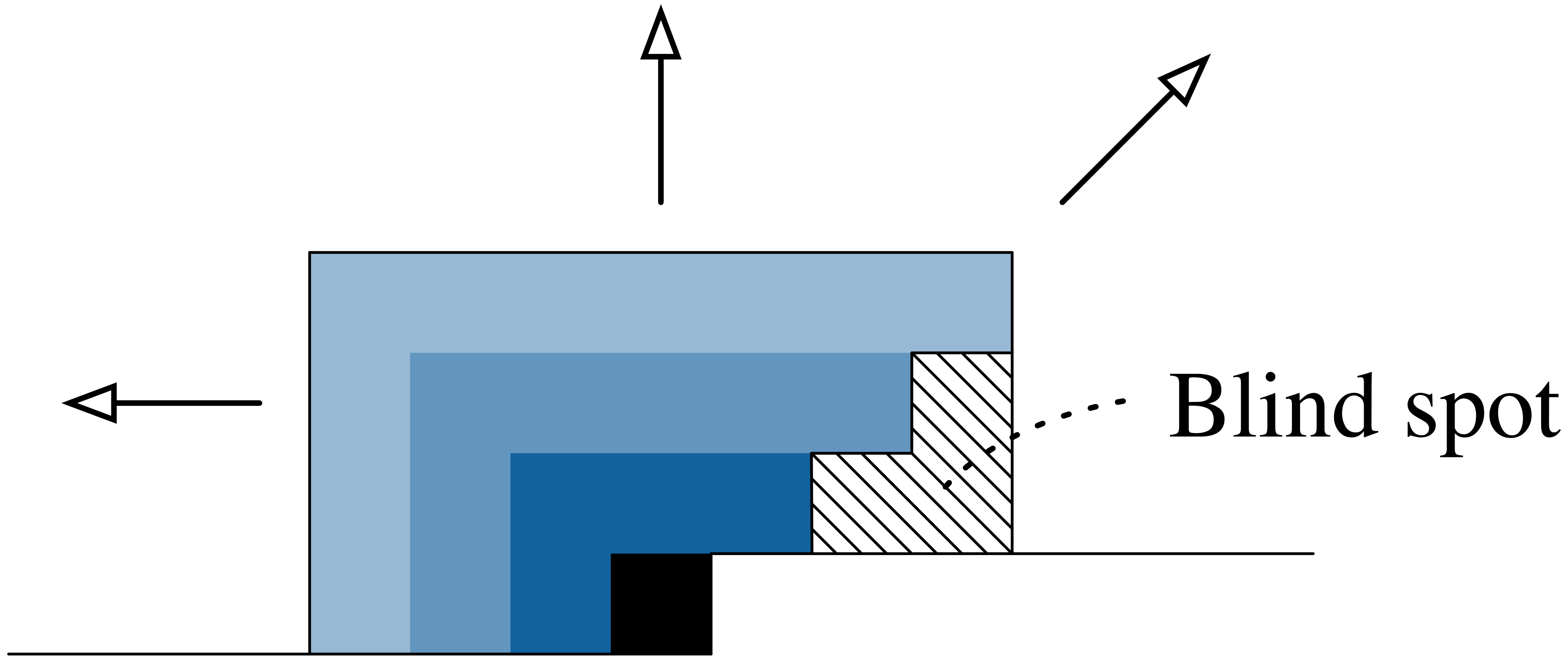
1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0



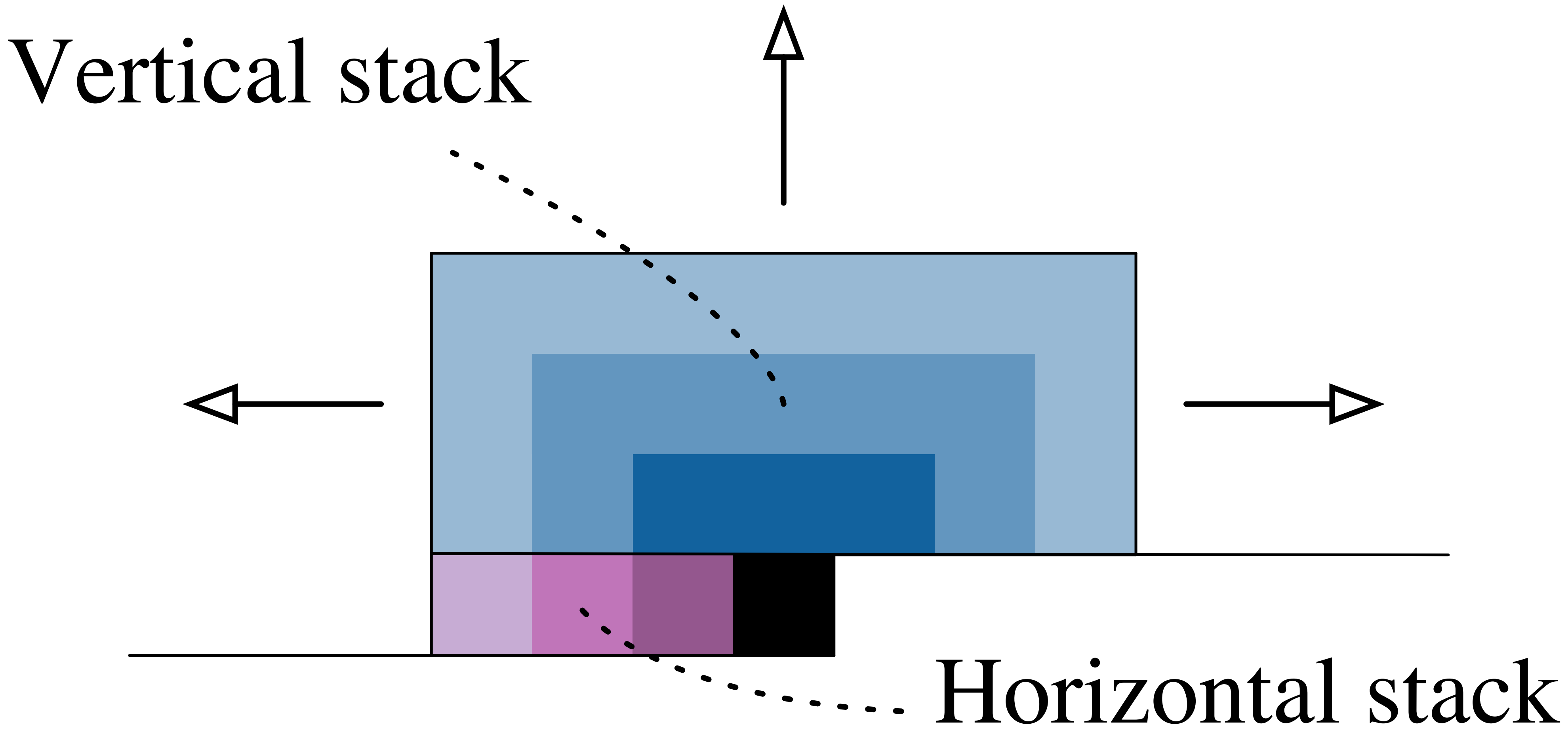
Stacking layers of masked convolution creates a blindspot

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

Improving PixelCNN I



Stacking layers of masked convolution creates a blindspot



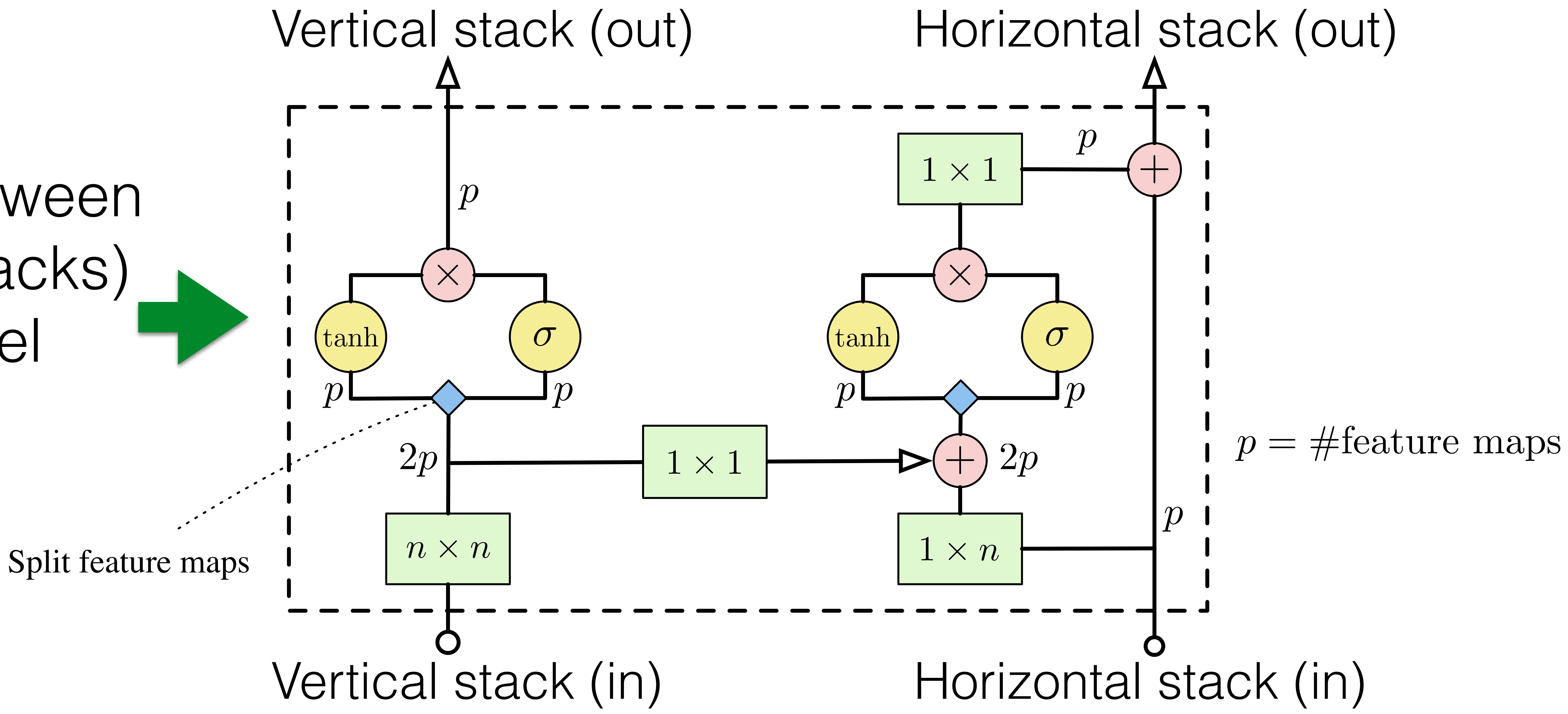
Solution: use two stacks of convolution, a vertical stack and a horizontal stack

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

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



van den Oord, Aaron, et al. "Conditional image generation with PixelCNN decoders." NIPS 2016.

EXPERIMENTAL RESULTS

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

Topics: CIFAR-10

- 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)
Gated PixelCNN:	3.03 (2.90)

EXPERIMENTAL RESULTS

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

Topics: CIFAR-10

- Samples from a class-conditional PixelCNN

Coral Reef

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

Sorrel horse

EXPERIMENTAL RESULTS

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

Topics: CIFAR-10

- Samples from a class-conditional PixelCNN

Sandbar

EXPERIMENTAL RESULTS

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

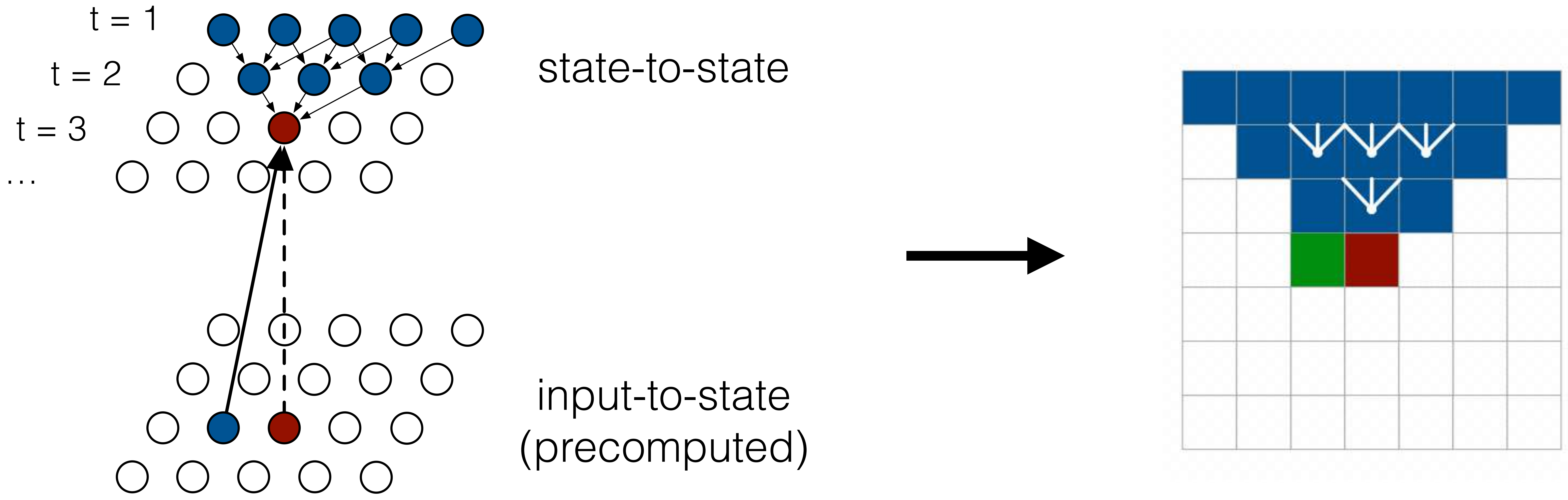
Topics: CIFAR-10

- Samples from a class-conditional PixelCNN

Lhasa Apso (dog)

PixelRNN

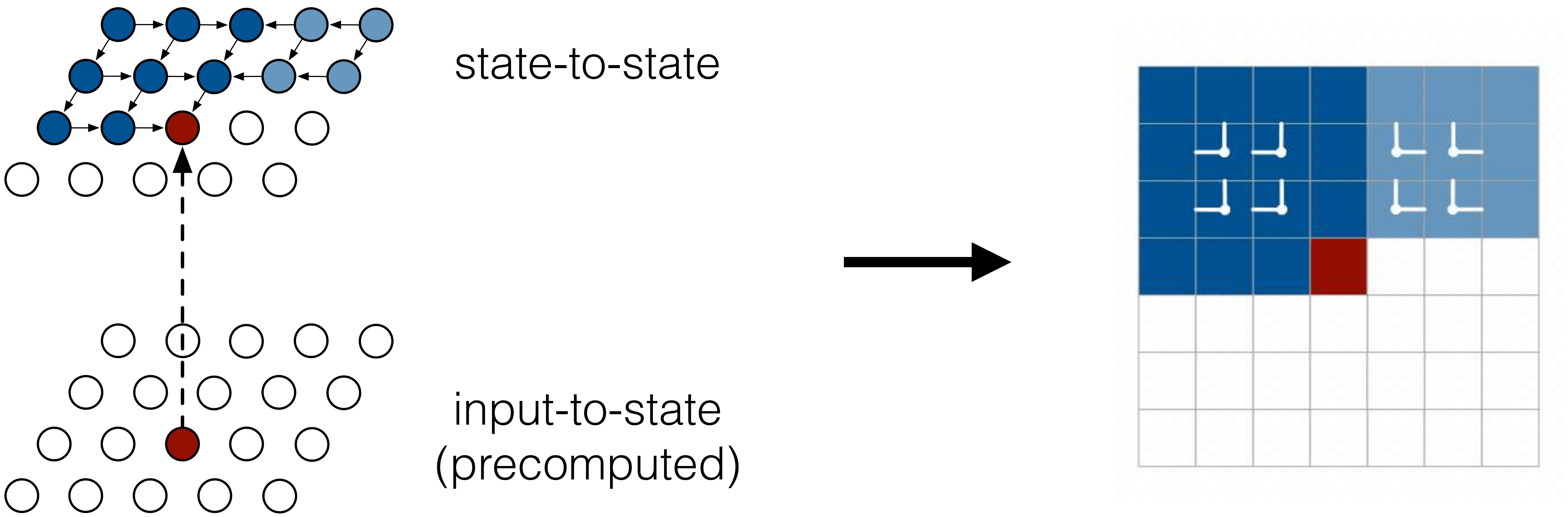
Row LSTM



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

PixelRNN

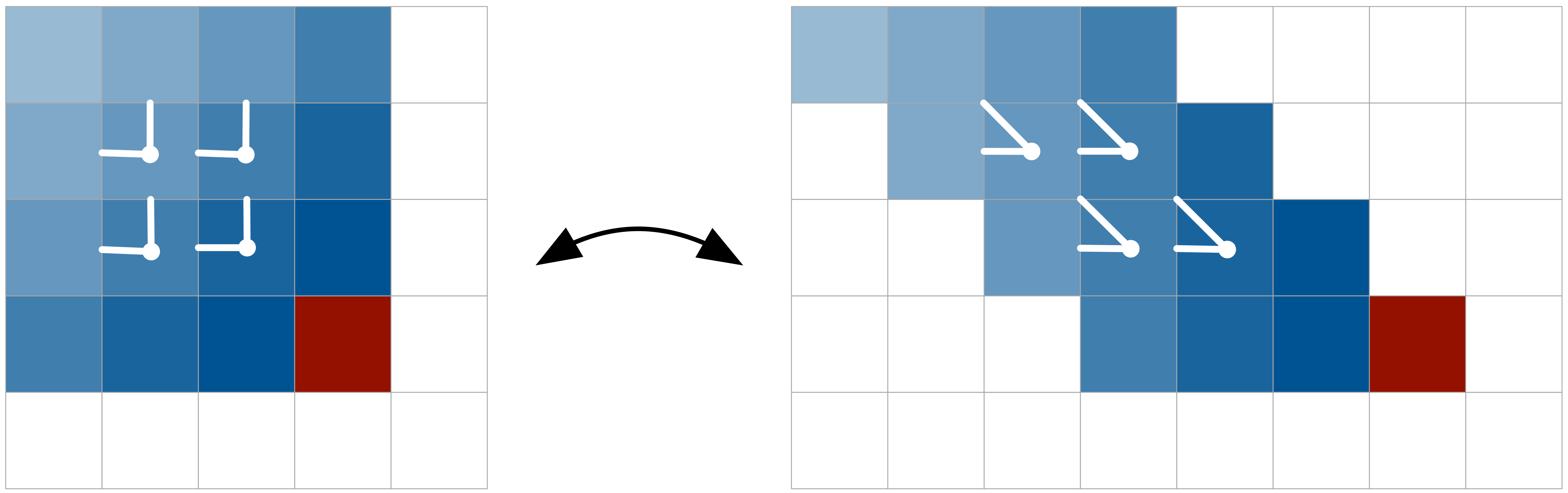
Diagonal BiLSTM



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

PixelRNN

Diagonal BiLSTM



- PixelRNN training can be sped up but is less parallelizable than PixelCNN.
- PixelCNN seems more promising (more work building on it)

In the Diagonal BiLSTM, to allow for parallelization along the diagonals, the input map is skewed by offsetting each row by one position with respect to the previous row. When the spatial layer is computed left to right and column by column, the output map is shifted back into the original size. The convolution uses a kernel of size 2×1.

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

PixelRNN

Downsampled ImageNet samples

Model	NLL Test
DBM 2hl [1]:	≈ 84.62
DBN 2hl [2]:	≈ 84.55
NADE [3]:	88.33
EoNADE 2hl (128 orderings) [3]:	85.10
EoNADE-5 2hl (128 orderings) [4]:	84.68
DLGM [5]:	≈ 86.60
DLGM 8 leapfrog steps [6]:	≈ 85.51
DARN 1hl [7]:	≈ 84.13
MADE 2hl (32 masks) [8]:	86.64
DRAW [9]:	≤ 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**

Table 4. Test set performance of different models on MNIST in *nats* (negative log-likelihood). Prior results taken from [1] (Salakhutdinov & Hinton, 2009), [2] (Murray & Salakhutdinov, 2009), [3] (Uria et al., 2014), [4] (Raiko et al., 2014), [5] (Rezende et al., 2014), [6] (Salimans et al., 2015), [7] (Gregor et al., 2014), [8] (Germain et al., 2015), [9] (Gregor et al., 2015).

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

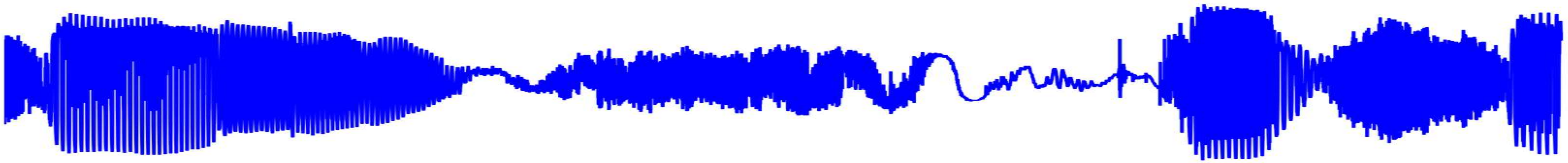
PixelRNN

Theano implementation: https://github.com/igul222/pixel_rnn

TensorFlow implementation: https://github.com/carpedm20/pixel-rnn-tensorflow

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

WaveNet



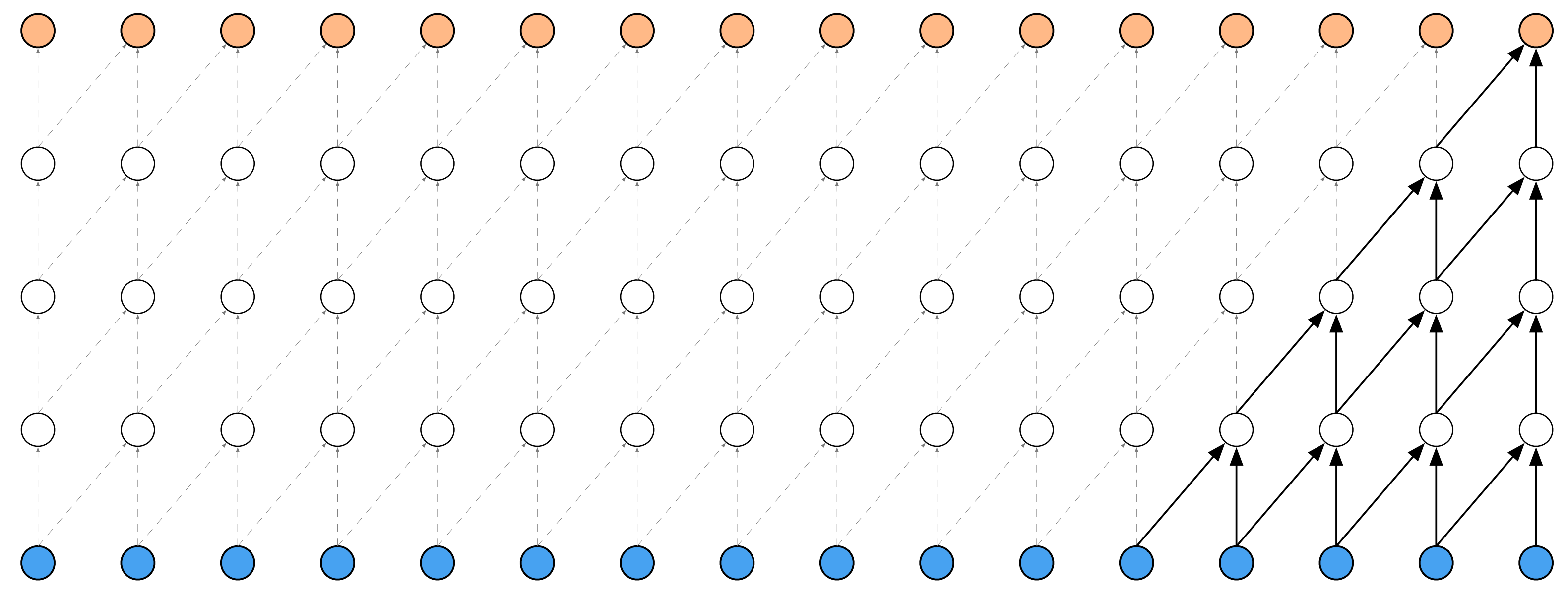
Audio: **much** larger dimensionality than images (at least 16,000 samples per **second**)

Idea: adapt PixelCNN to allow very large temporal dependencies

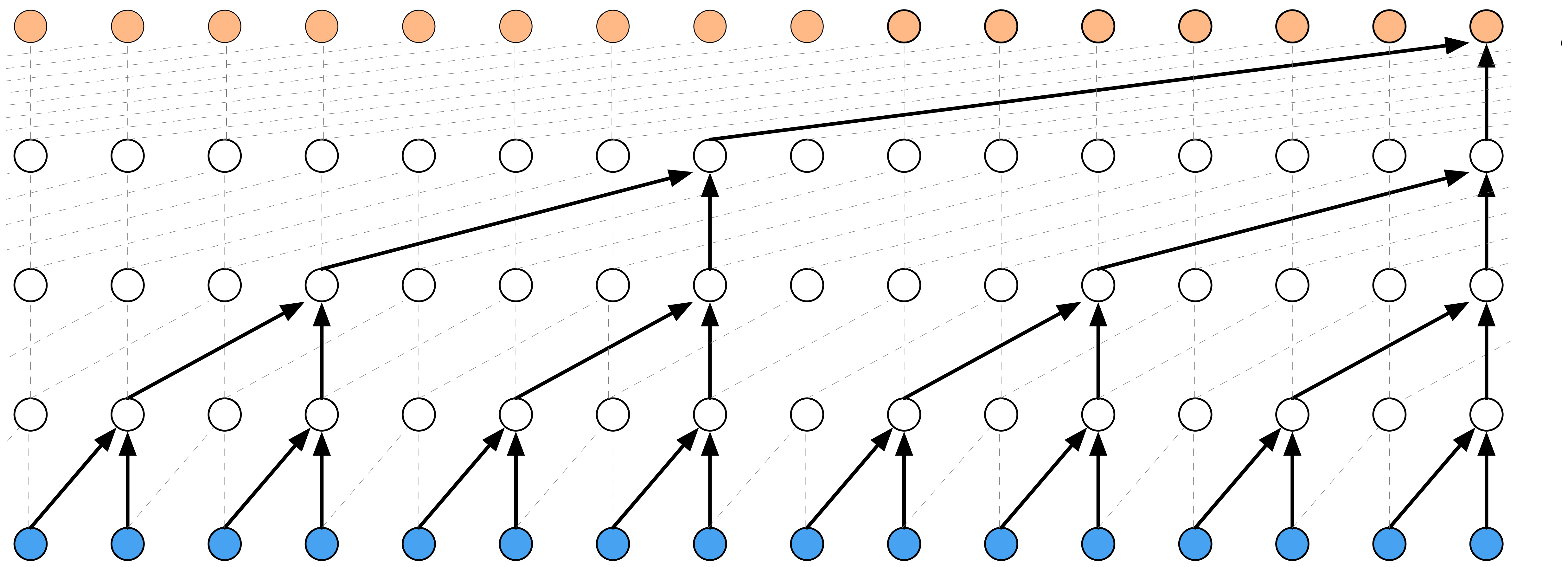
van den Oord, Aäron, et al. "Wavenet: A generative model for raw audio." CoRR abs/1609.03499 (2016).

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.*

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

WaveNet

Discrete conditional probabilities

μ-law companding transformation

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

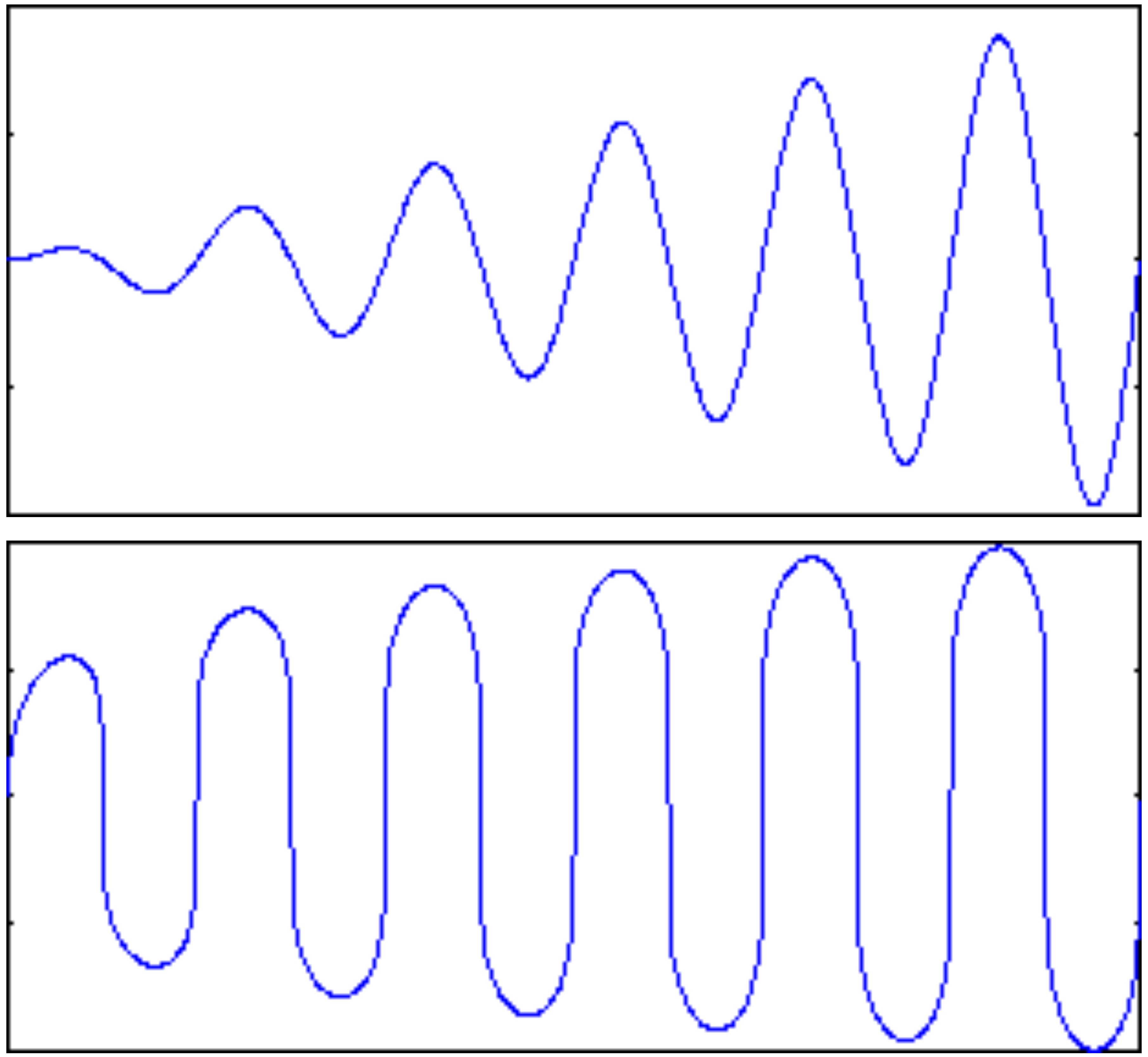
$$\tilde{x}_t = \operatorname{sign}(x_t) \frac{\ln(1 + 255|x_t|)}{\ln 256}$$

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

quantize back

x_t

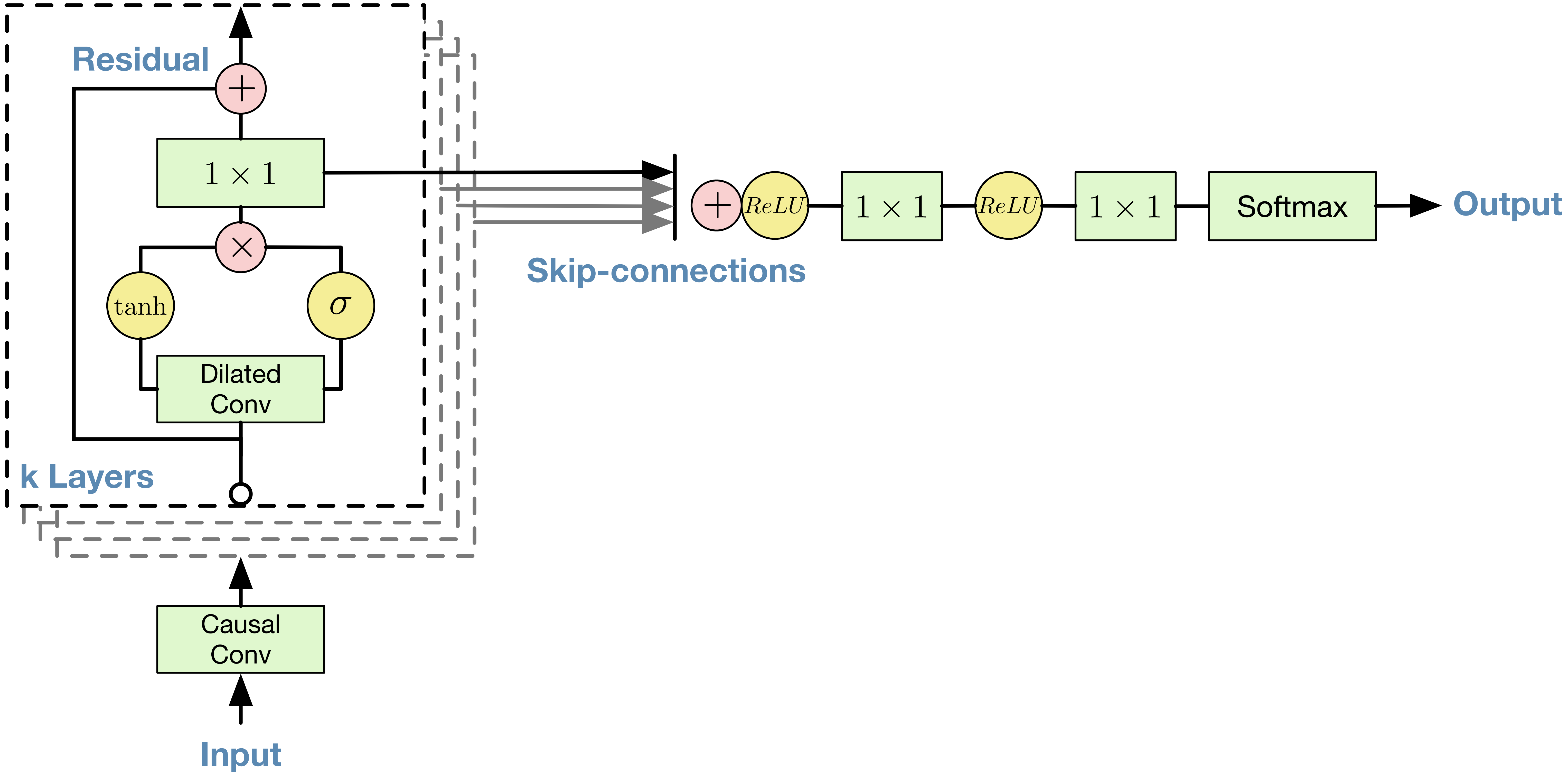
$\tilde{x}_t$



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

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)

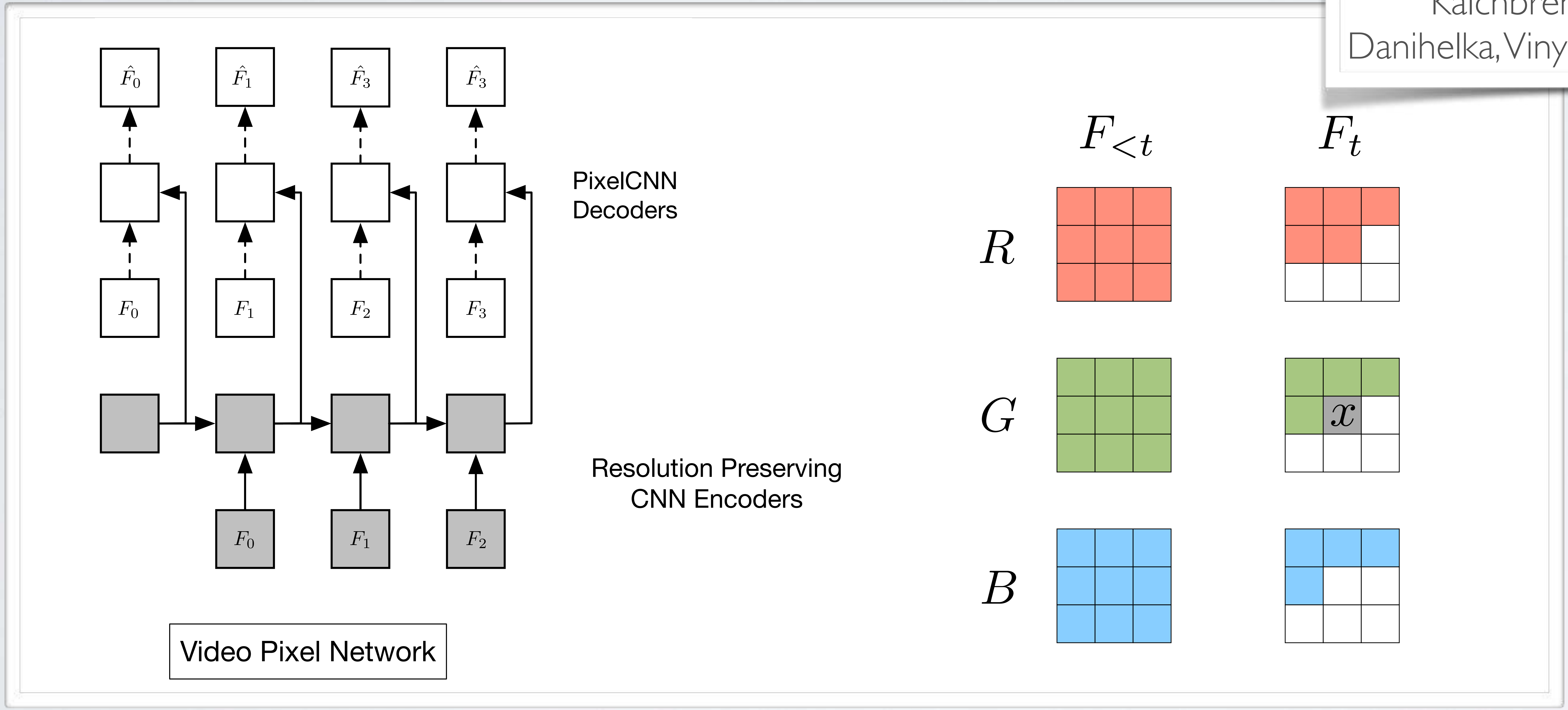
van den Oord, Aäron, et al. "Wavenet: A generative model for raw audio." CoRR abs/1609.03499 (2016).

AUTOREGRESSIVE VIDEO MODELS

Topics: Video Pixel Network

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

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



AUTOREGRESSIVE VIDEO MODELS

Topics: Video Pixel Network

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

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

- Videos of robot manipulating
  - objects seen in the training set
  - new objects not seen in training set

Parallel Multiscale Autoregressive Density Estimation

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

MILA

Can we speed up the generation time of PixelCNN?

- Yes, via multiscale generation:

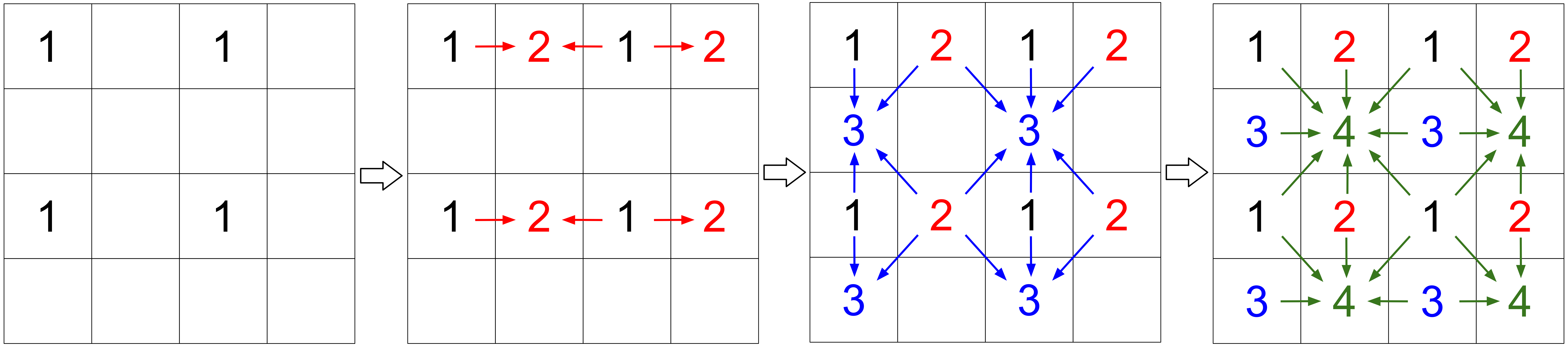


Figure 2. Example pixel grouping and ordering for a 4×4 image. The upper-left corners form group 1, the upper-right group 2, and so on. For clarity we only use arrows to indicate immediately-neighboring dependencies, but note that all pixels in preceding groups can be used to predict all pixels in a 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 Go´mez 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.”

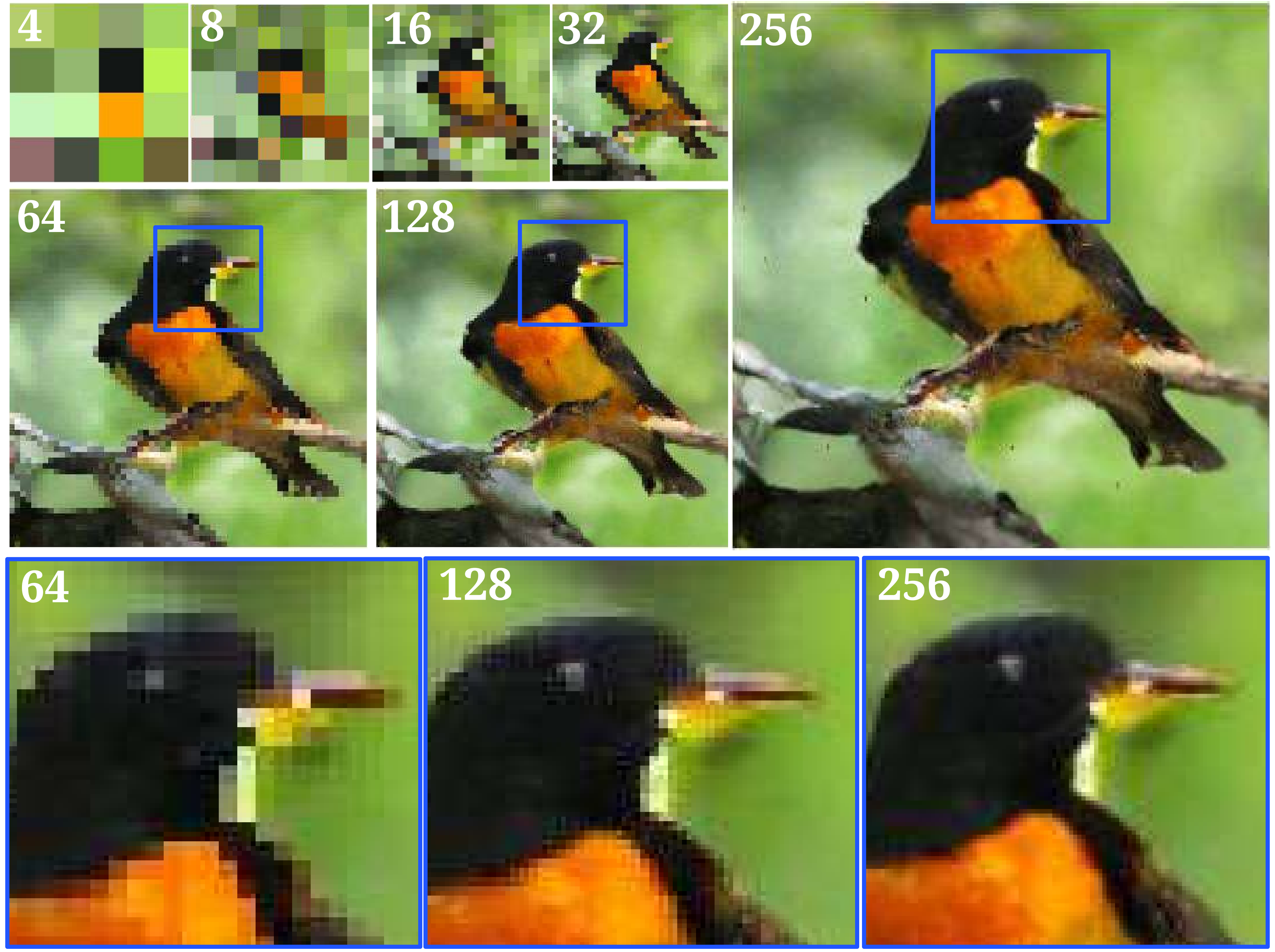


Figure 1. Samples from our model at resolutions from 4×4 to 256×256, conditioned on text and bird part locations in the CUB data set. See Fig. 4 and the supplement for more examples.

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.

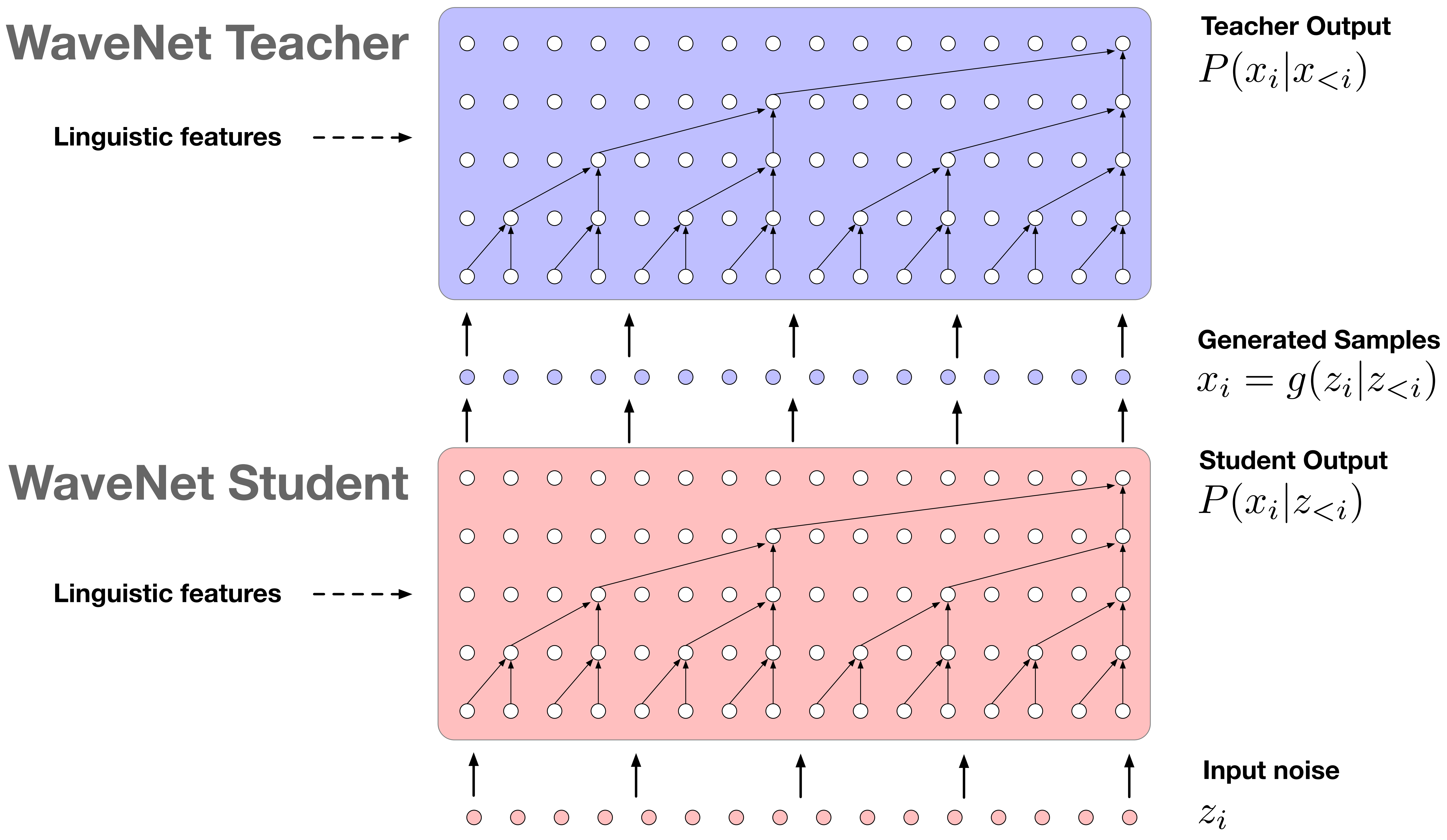


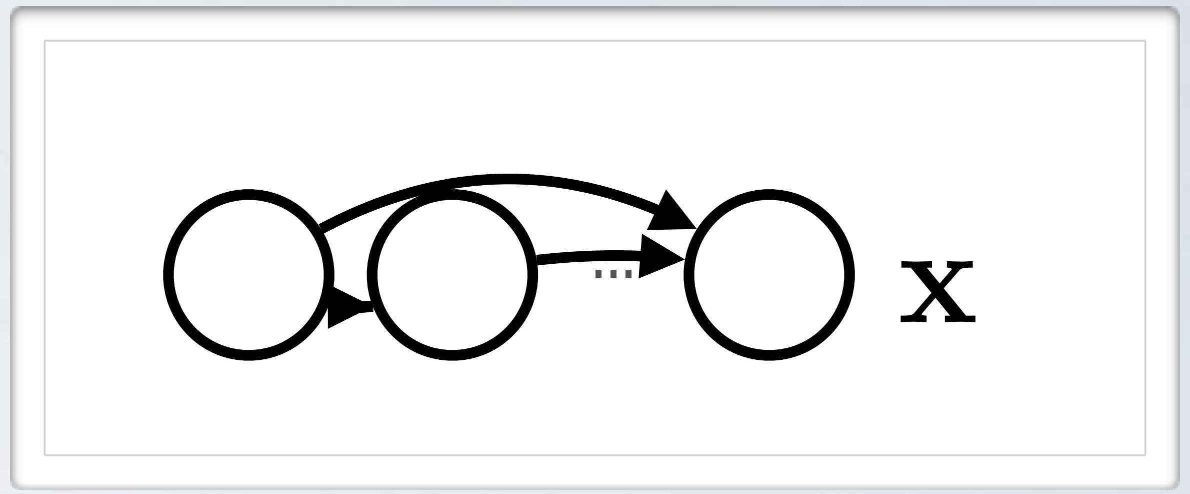
Figure 2: **Overview of Probability Density Distillation**. A pre-trained WaveNet teacher is used to score the samples $\boldsymbol{x}$ output by the student. The student is trained to minimise the KL-divergence between its distribution and that of the teacher by maximising the log-likelihood of its 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



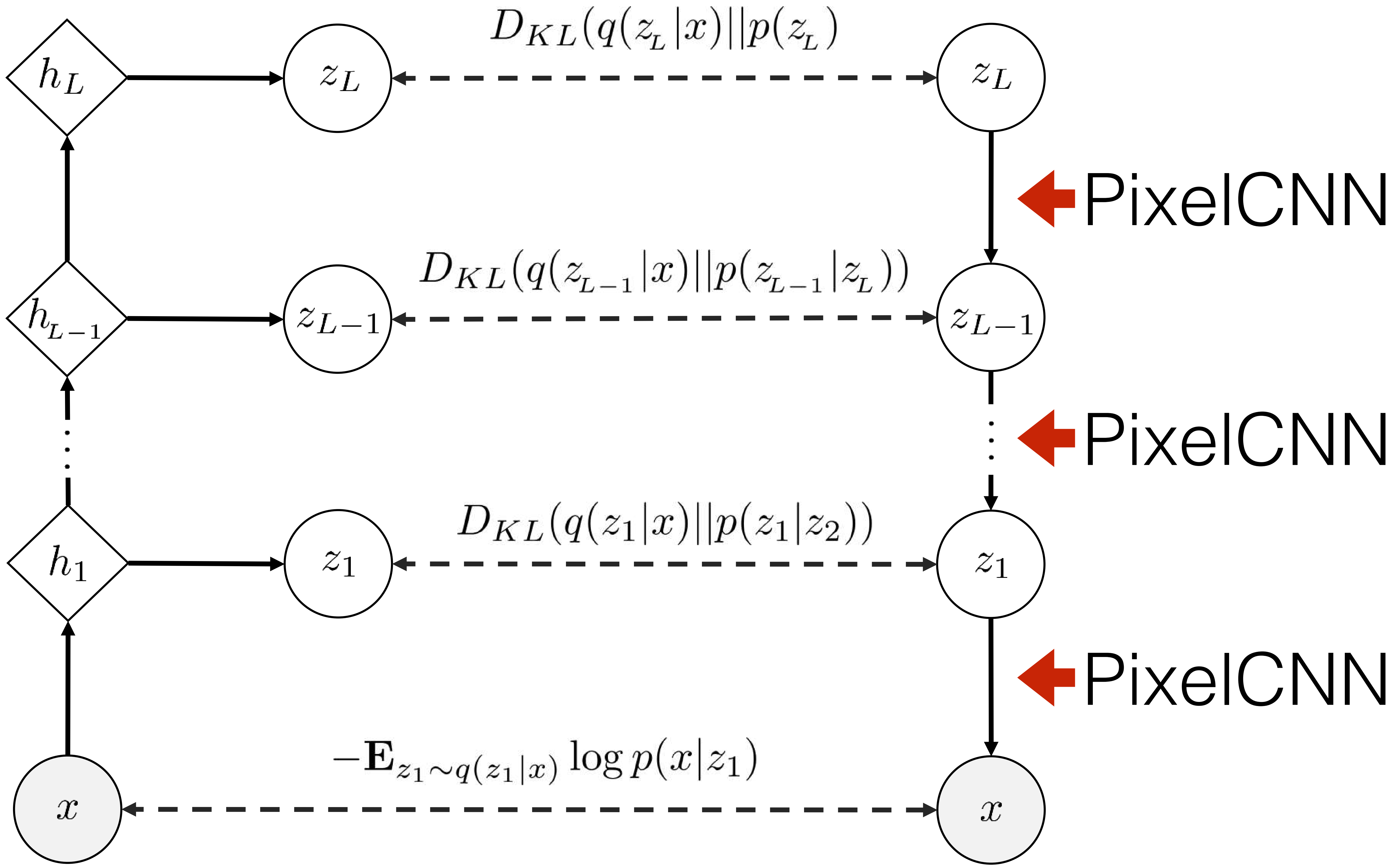
- Properties
  - *pros*: $p(\mathbf{x})$ is tractable, so easy to train, easy to sample (though slower)
  - *cons*: doesn't have a natural latent representation

PixelVAE

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed Adrien Ali Taiga, Francesco Visin, David Vazquez, Aaron Courville. ICLR 2017

MILA

- Uses a PixelCNN in the VAE decoder to help avoid the blurring caused by the standard VAE assumption of independent pixels.



PixelVAE samples

MILA

64x64 LSUN bedroom scenes

64x64 ImageNet

PixelVAE

varying only the top-level latent variables

varying only the bottom-level latent variables

varying only the pixel-level noise