

Lecture 12: Deep Sequential Models

Efstratios Gavves

Lecture overview

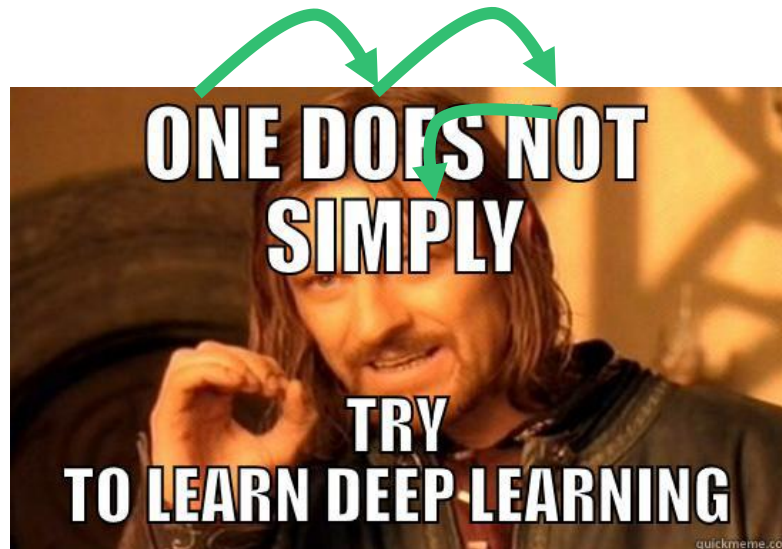
- Autoregressive Models
- PixelCNN, PixelCNN++, PixelRNN
- WaveNet
- Time-Aligned DenseNets

Autoregressive Models

- Let's assume we have signal modelled by an input random variable x
 - Can be an image, video, text, music, temperature measurements
- Is there an order in all these signals?

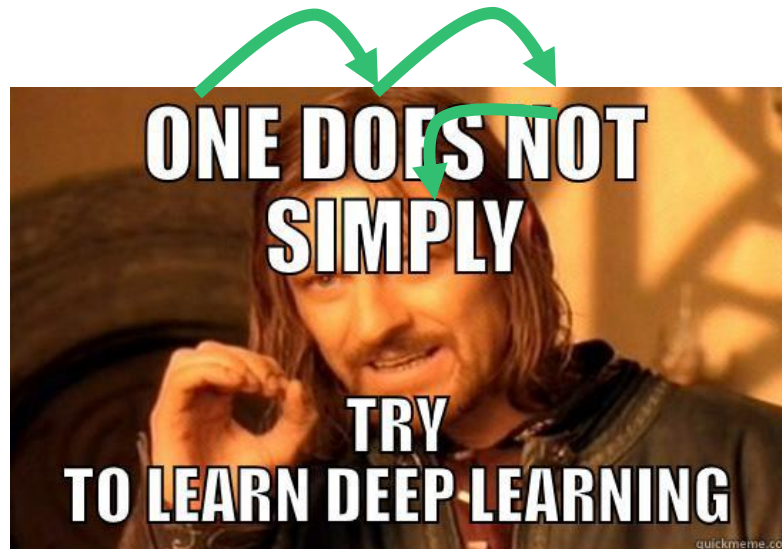
Autoregressive Models

- Let's assume we have signal modelled by an input random variable x
 - Can be an image, video, text, music, temperature measurements
- Is there an order in all these signals?



Autoregressive Models

- Let's assume we have signal modelled by an input random variable x
 - Can be an image, video, text, music, temperature measurements
- Is there an order in all these signals? Other signals and orders?



Autoregressive Models

- Let's assume we have signal modelled by an input random variable x
 - Can be an image, video, text, music, temperature measurements
- Is there an order in all these signals?



Autoregressive Models

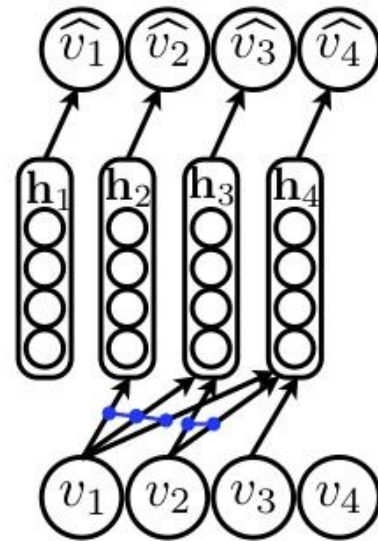
- If x is sequential, there is an order: $x = [x_1, \dots, x_k]$
 - E.g., the order of words in a sentence
- If x is *not* sequential, we can create an artificial order $x = [x_{r(1)}, \dots, x_{r(k)}]$
 - E.g., the order with which pixels make (generate) an image
- Then, the marginal likelihood is a product of conditionals

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

- Different from Recurrent Neural Networks
 - (a) no parameter sharing
 - (b) chains are not infinite in length

Autoregressive Models

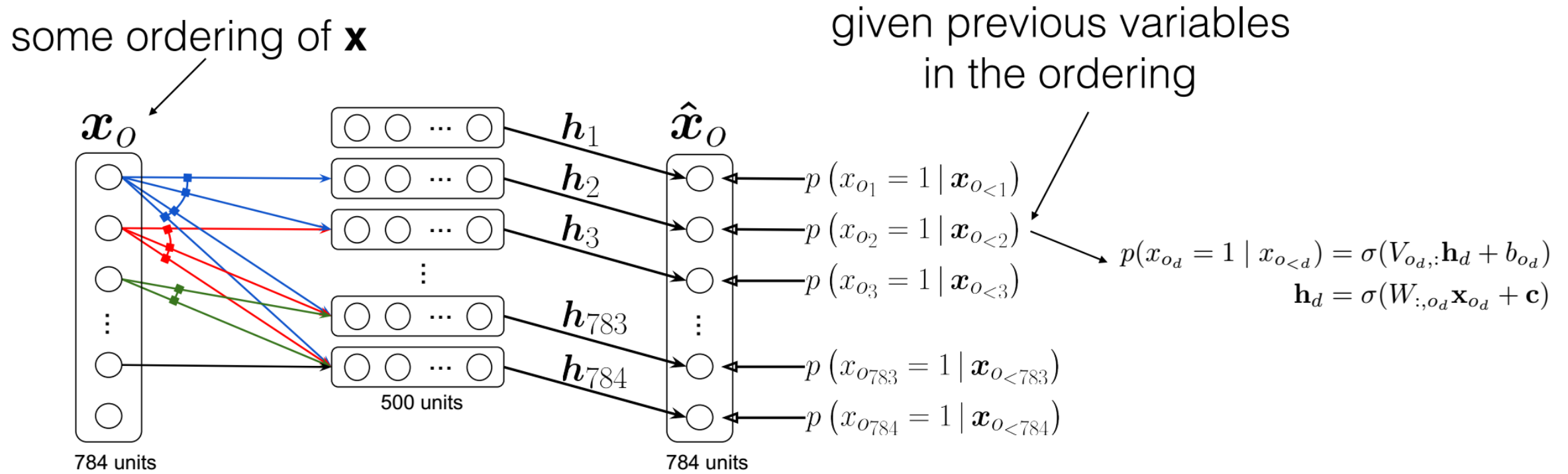
- Pros: because of the product decomposition, $p(x)$ is tractable
- Cons: because the $p(x)$ is sequential, training is slower
 - To generate every new word/frame/pixel, the previous words/frames/pixels in the order must be generated first → no parallelism



The where and now of Deep Autoregressive Models

- Sequential data is a natural fit
 - Language modelling, time series, etc
 - For non-sequential data they are ok, although arguably artificial
- **Question:** How to model the conditionals $p(x_k | x_{j < k})$
 - Logistic regression (Frey et al., 1996)
 - Neural networks (Bengio and Bengio, 2000)
- Modern deep autoregressors
 - NADE, MADE, PixelCNN, PixelCNN++, PixelRNN, WaveNet

NADE



Neural Autoregressive Distribution Estimation, Larochelle and Murray, AISTATS 2011

NADE

- Minimizing negative log-likelihood as usual

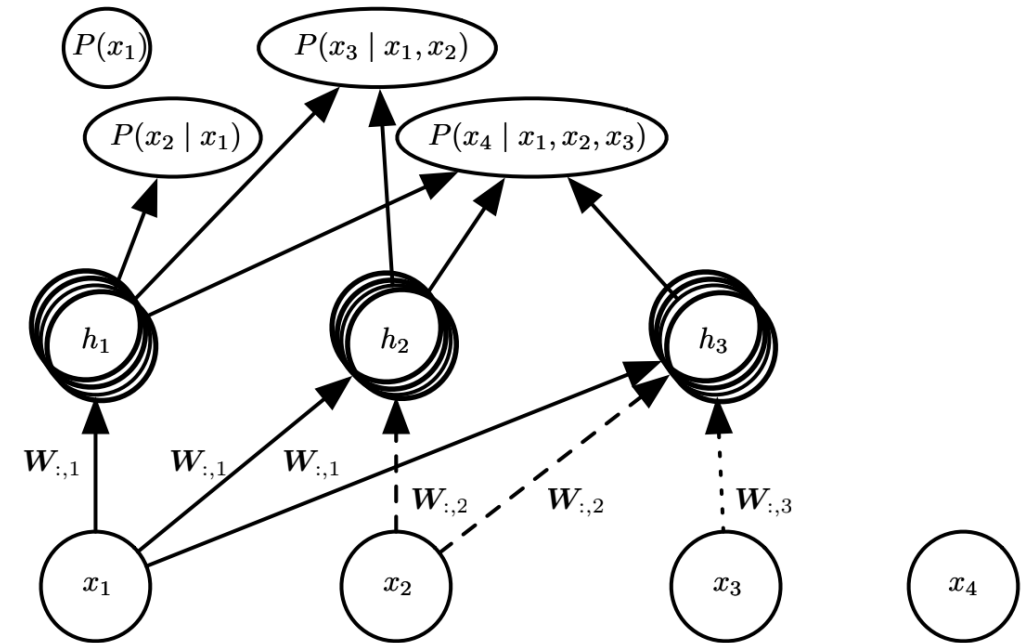
$$\mathcal{L} = -\log p(x) = -\sum_{k=1}^D p(x_k | x_{<k})$$

- Then, we model the conditional as

$$p(x_d | x_{<d}) = \sigma(V_{d,:} \cdot h_d + b_d)$$

where the latent variable h_d is defined as

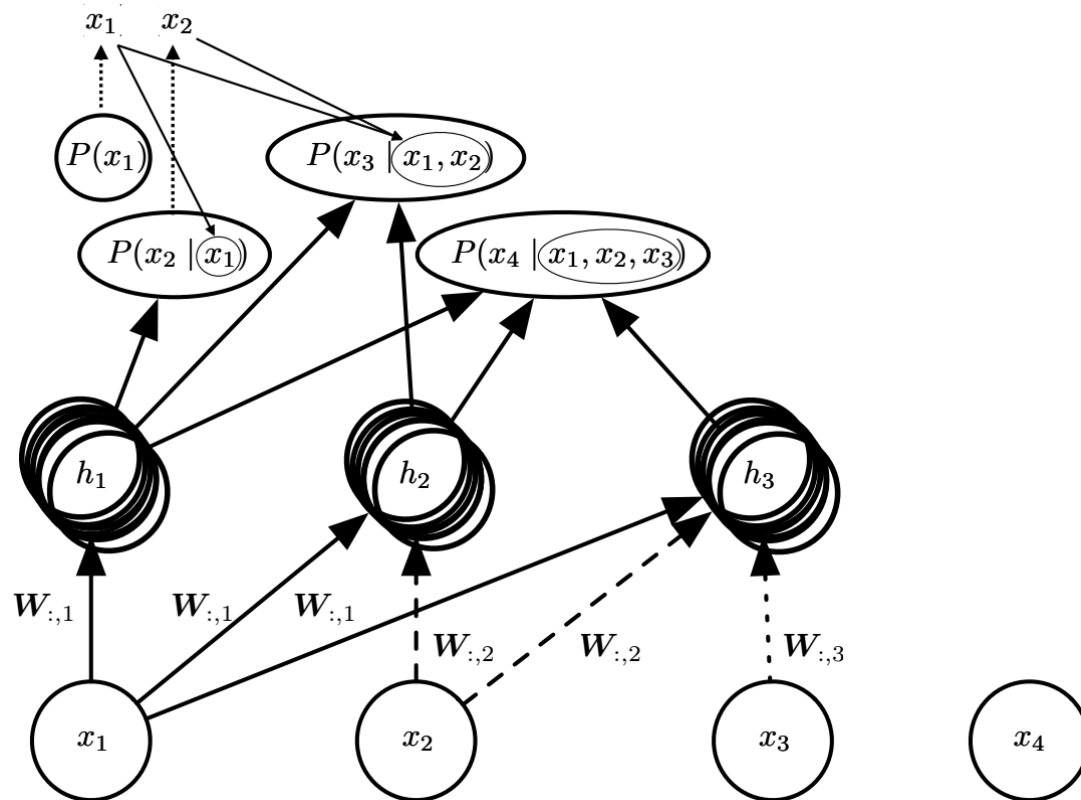
$$h_d = \sigma(W_{:,d} \cdot x_d + c)$$



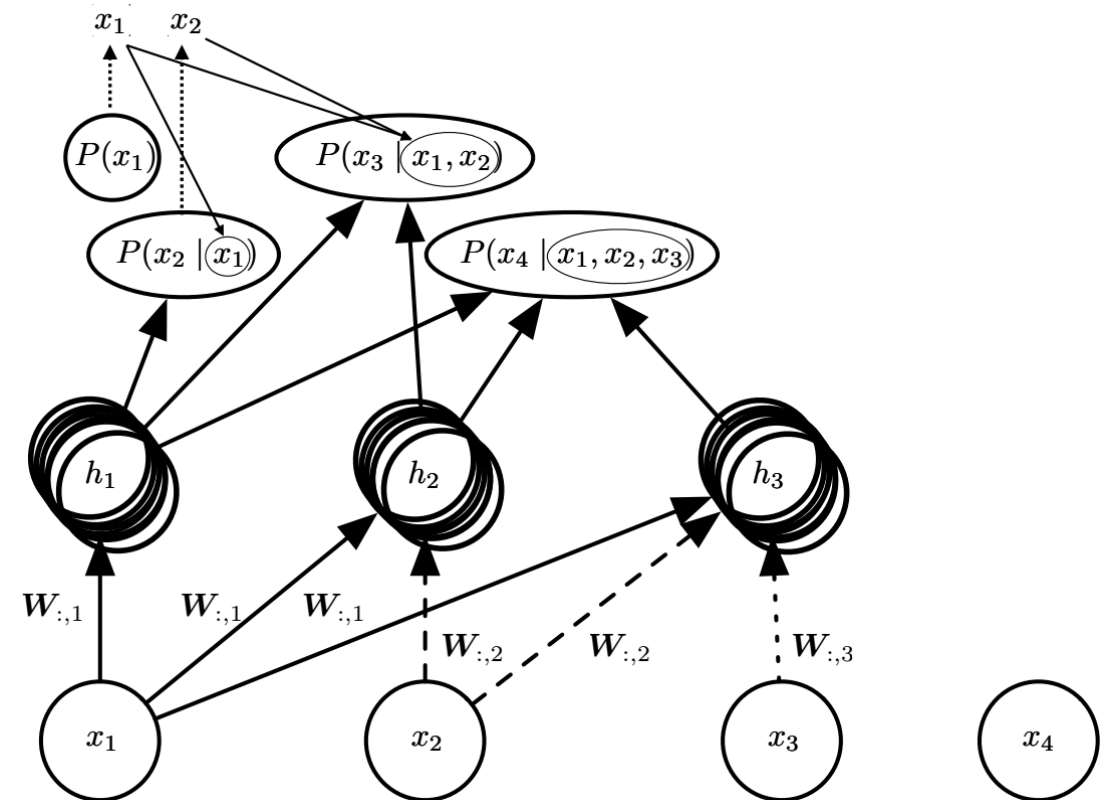
NADE: Training & Testing

- “Teacher forcing” training

Training: Use ground truth values (e.g. of pixels)



Testing: Use predicted values in previous order



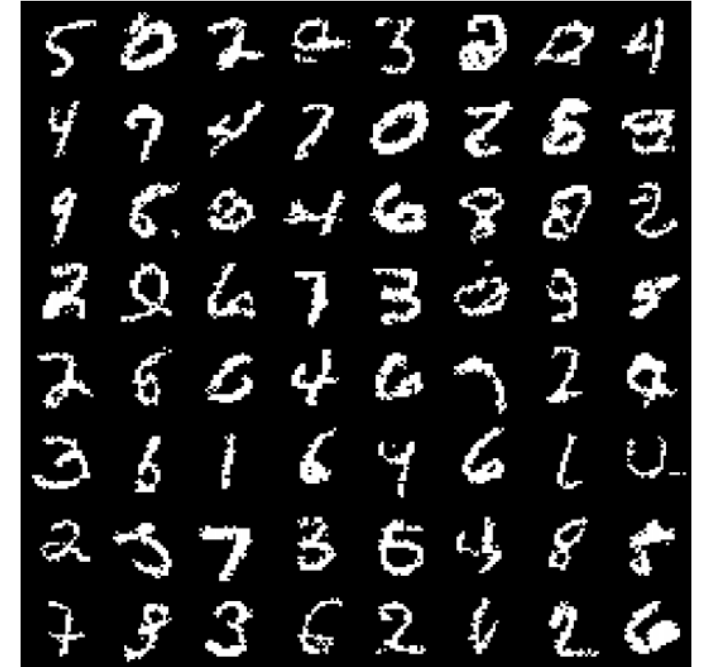
NADE Visualizations



Binarized MNIST
samples (NADE)



Binarized MNIST
samples (DeepNADE)



Binarized MNIST
samples (ConvNADE)

- **Question:** How could we construct an autoregressive autoencoder?
- **To rephrase:** How to modify an autoencoder such that each output x_k depends only on the previous outputs $x_{<k}$ (autoregressive property)?
 - Namely, the present k -th output \tilde{x}_k must not depend on a computational path from future inputs x_{k+1}, \dots, x_D
 - Autoregressive: $p(x|\theta) = \prod_{k=1}^D p(x_k|x_{j<k}, \theta)$
 - Autoencoder: $p(\tilde{x}|x, \theta) = \prod_{k=1}^D p(\tilde{x}_k|x_k, \theta)$

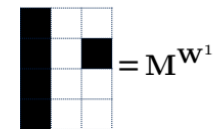
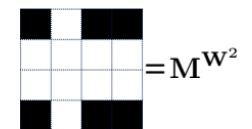
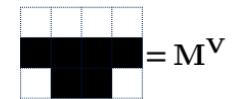
Masked Autoencoder for Distribution Estimation, Germain, Mathieu et al., ICML 2015

MADE

- **Question:** How could we construct an autoregressive autoencoder?
- **To rephrase:** How to modify an autoencoder such that each output x_k depends only on the previous outputs $x_{<k}$ (autoregressive property)?
 - Namely, the present k -th output \tilde{x}_k must not depend on a computational path from future inputs x_{k+1}, \dots, x_D
 - Autoregressive: $p(x|\theta) = \prod_{k=1}^D p(x_k | x_{j < k}, \theta)$
 - Autoencoder: $p(\tilde{x}|x, \theta) = \prod_{k=1}^D p(\tilde{x}_k | x_k, \theta)$
- Answer: Masked convolutions!

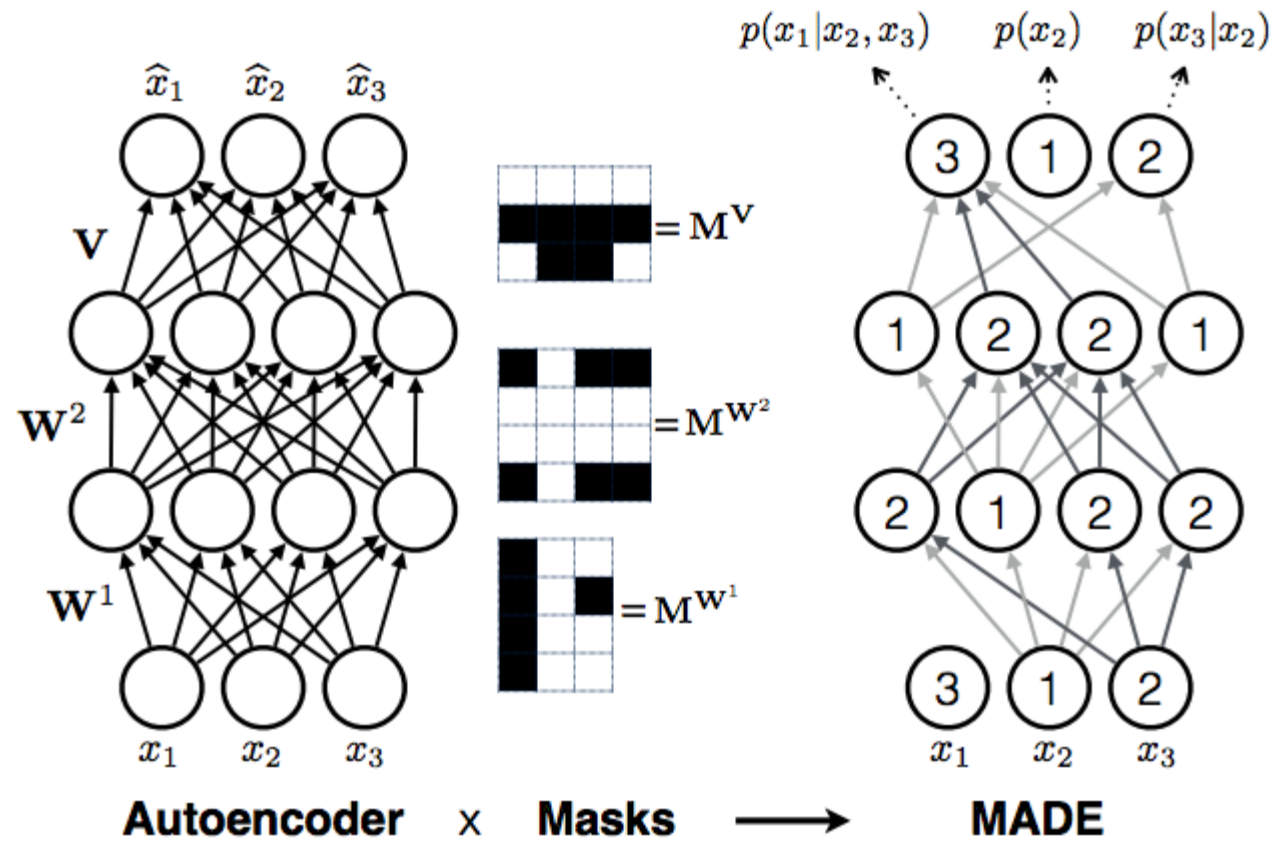
$$h(x) = g(b + (W \odot M^W) \cdot x)$$
$$\tilde{x} = \sigma(c + (V \odot M^V) \cdot h(x))$$

Masks



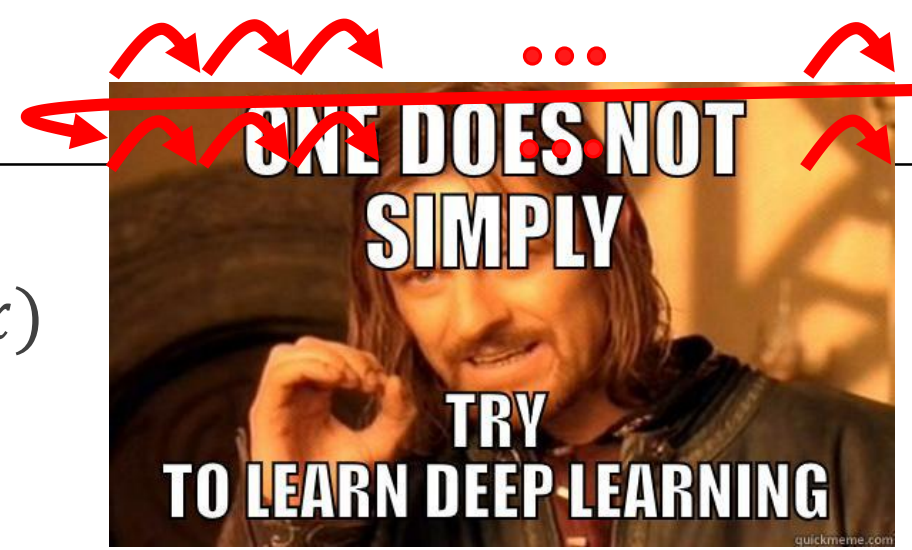
Masked Autoencoder for Distribution Estimation, Germain, Mathieu et al., ICML 2015

MADE



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

PixelRNN



- Unsupervised learning: learn how to model $p(x)$
- Decompose the marginal

$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \dots, x_{i-1})$$

- Assume row-wise pixel by pixel generation and sequential colors $R \rightarrow G \rightarrow B$
 - Each color conditioned on all colors from previous pixels and specific colors in the same pixel

$$p(x_{i,R} | x_{<i}) \cdot p(x_{i,G} | x_{<i}, x_{i,R}) \cdot p(x_{i,B} | x_{<i}, x_{i,R}, x_{i,G})$$

- Final output is 256-way softmax

Pixel Recurrent Neural Networks, van den Oord, Kalchbrenner and Kavukcuoglu, arXiv 2016

PixelRNN

- How to model the conditionals?

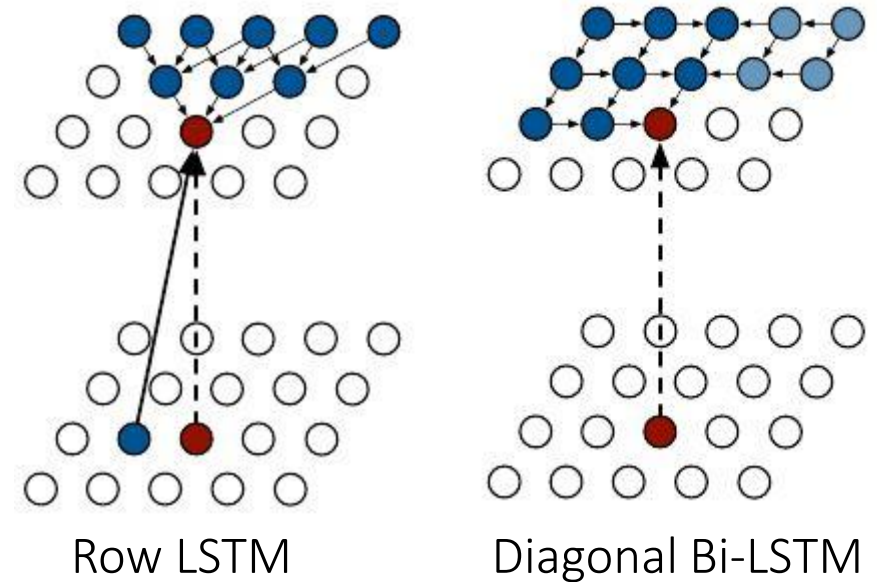
$$p(x_{i,R} | x_{<i}), p(x_{i,G} | x_{<i}, x_{i,R}), p(x_{i,B} | x_{<i}, x_{i,R}, x_{i,G})$$

- LSTM variants

- 12 layers

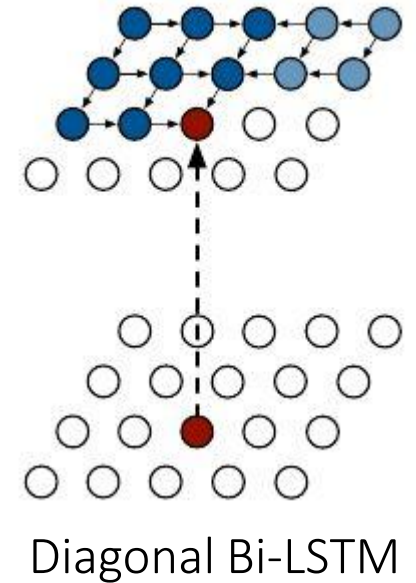
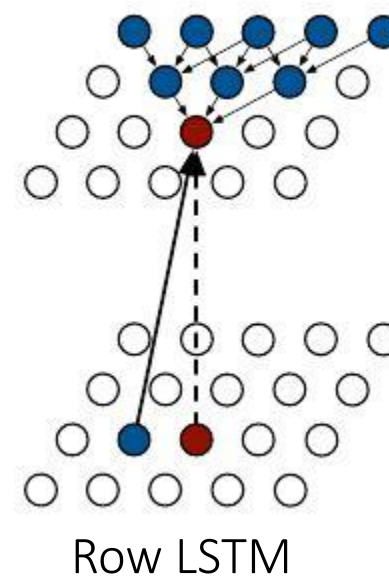
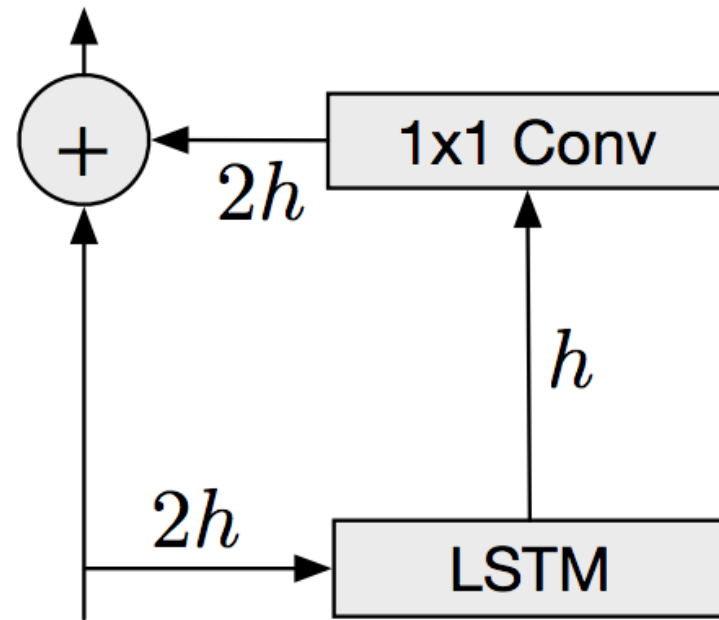
- Row LSTM

- Diagonal Bi-LSTM



PixelRNN

- Residual connections also to speed up convergence
- Pros: good modelling of $p(x) \rightarrow$ nice image generation
- Cons: slow training, slow generation



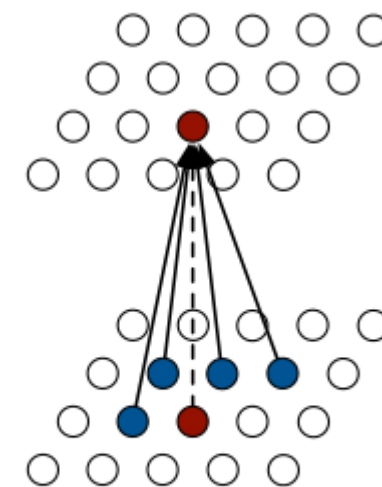
PixelRNN - Generations



Figure 1. Image completions sampled from a PixelRNN.

PixelCNN

- Unfortunately, PixelRNN is too slow
- Solution: replace recurrent connections with convolutions
 - Multiple convolutional layers to preserve spatial resolution
- Training is much faster because all true pixels are known in advance, so we can parallelize
 - Generation still sequential (pixels must be generated) → still slow



PixelCNN

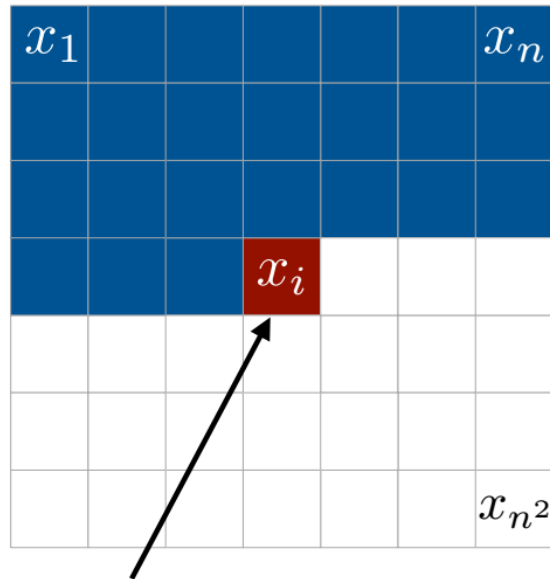
Stack of masked convolutions

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

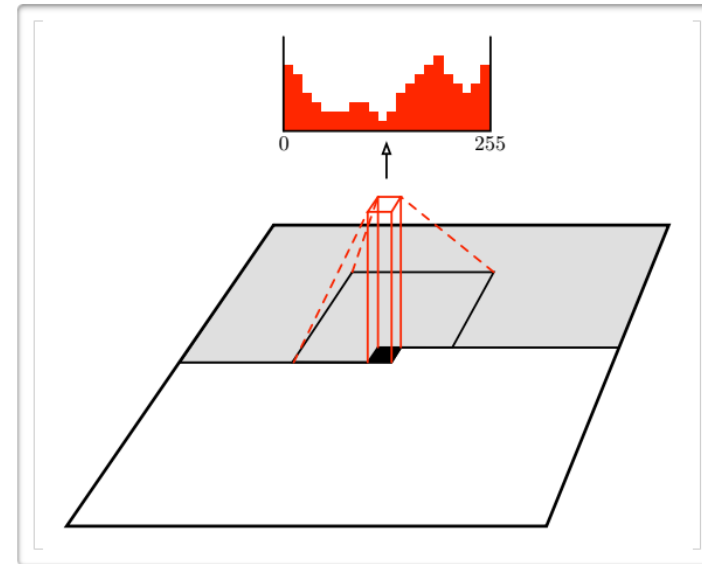
Pixel Recurrent Neural Networks, van den Oord, Kalchbrenner and Kavukcuoglu, arXiv 2016

PixelCNN

- Use masked convolutions again to enforce autoregressive relationships



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

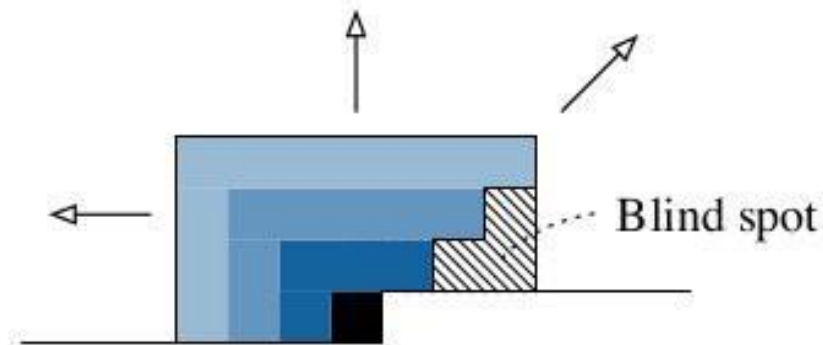


PixelCNN – Pros and Cons

- Cons: Performance is worse than PixelRNN
 - Why?
- **New problem:** the cascaded convolutions create a “blind spot”

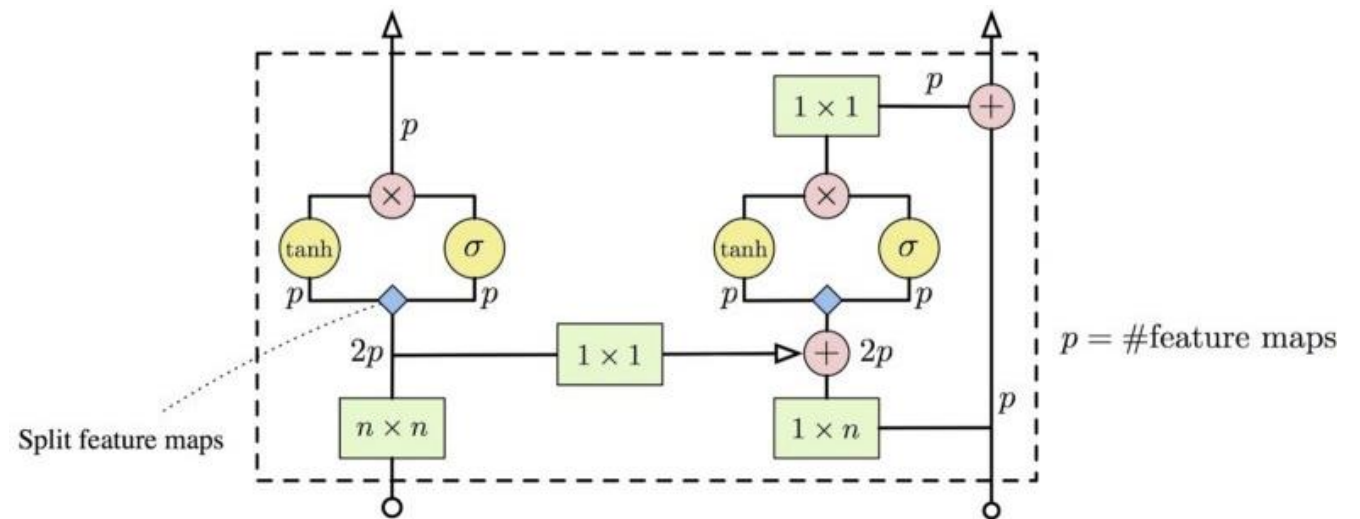
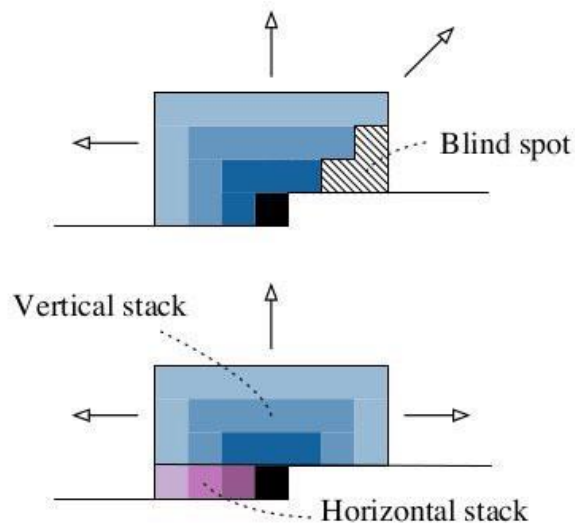
Blind spot

- Because of
 - (a) the limited receptive field of convolutions and
 - (b) computing all features at once (not sequentially)
 - cascading convolutions makes current pixel not depend on all previous
 - blind spot



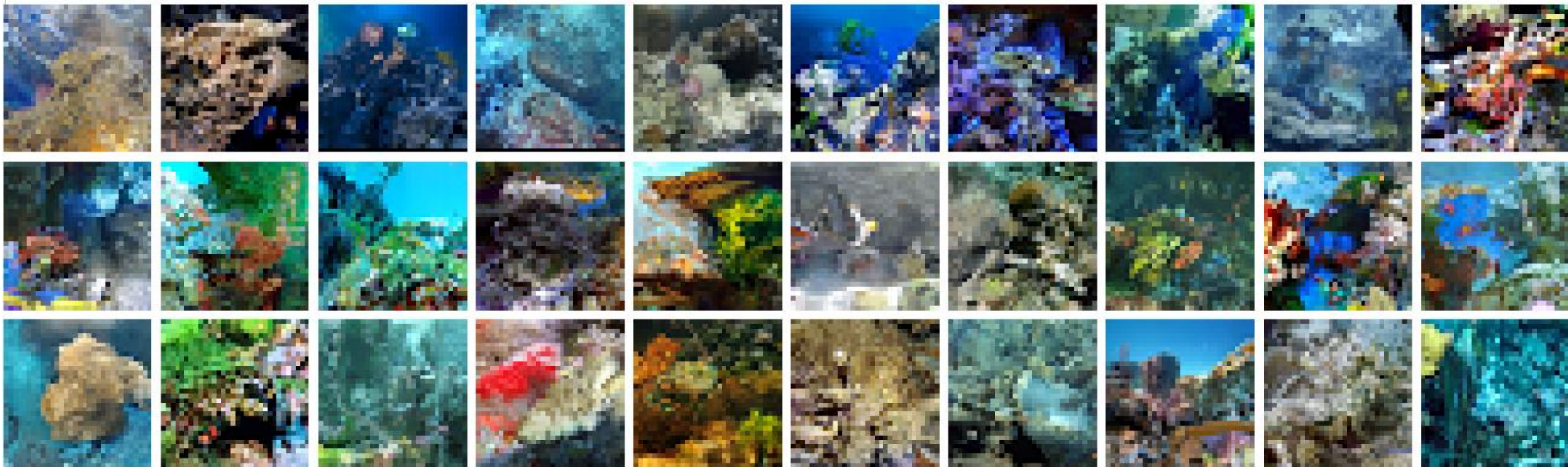
Fixing the blind spot: Gated PixelCNN

- Use two layers of convolutions stacks
 - Horizontal stack: conditions on current row and takes as input the previous layer output and the vertical stack
 - Vertical stack: conditions on all rows above current pixels
- Also replace ReLU with a $\tanh(W * x) \cdot \sigma(U * x)$



PixelCNN - Generations

- Coral reef



PixelCNN - Generation

○ Sorrel horse



PixelCNN - Generation

○ Sandbar



PixelCNN - Generation

○ Lhasa Apso

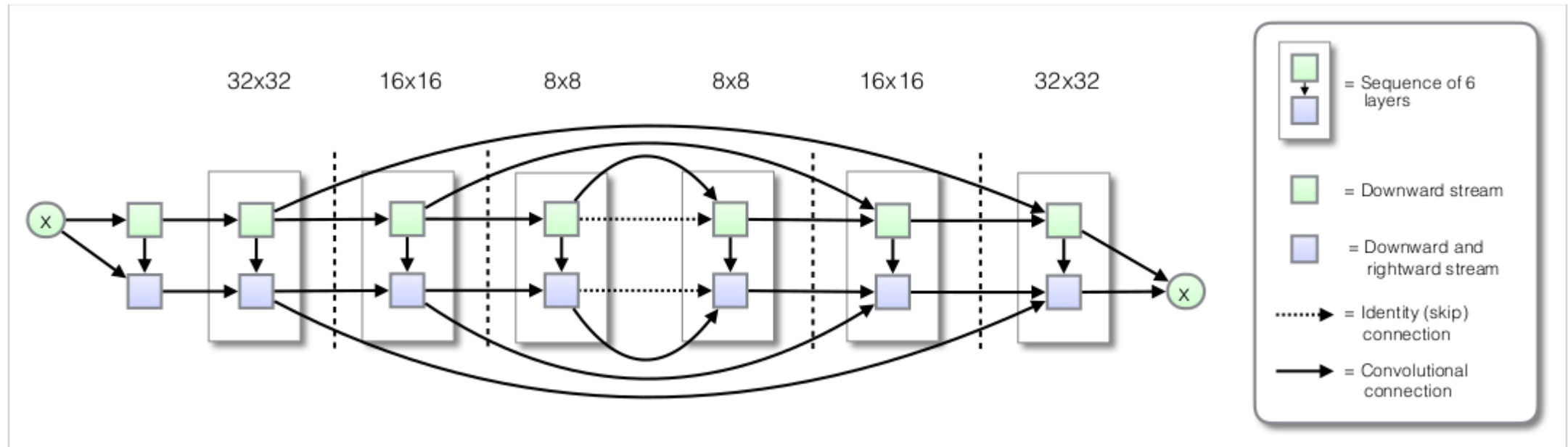


PixelCNN++

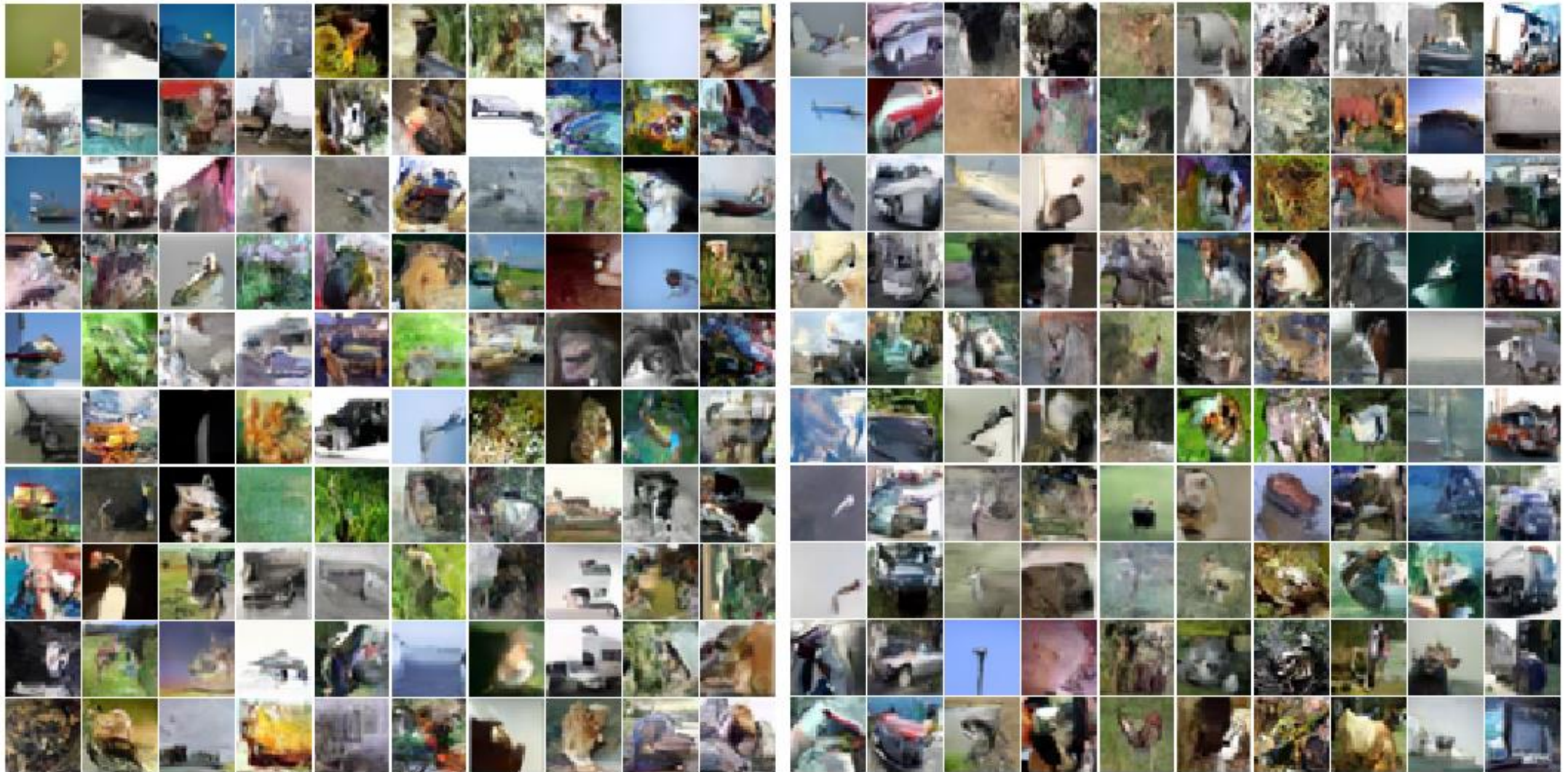
- Improving the PixelCNN model
- Replace the softmax output with a discretized logistic mixture likelihood
 - Softmax is too memory consuming and gives sparse gradients
 - Instead, assume logistic distribution of intensity and round off to 8-bits
- Condition on whole pixels, not pixel colors
- Downsample with stride-2 convs to compute long-range dependencies
- Use shortcut connections
- Dropout
 - PixelCNN is too powerful a framework → can overfit easily

PixelCNN++: Improving the PixelCNN with Discretized Logistic, Salimans, Karpathy, Chen, Kingma, ICLR 2017

PixelCNN++

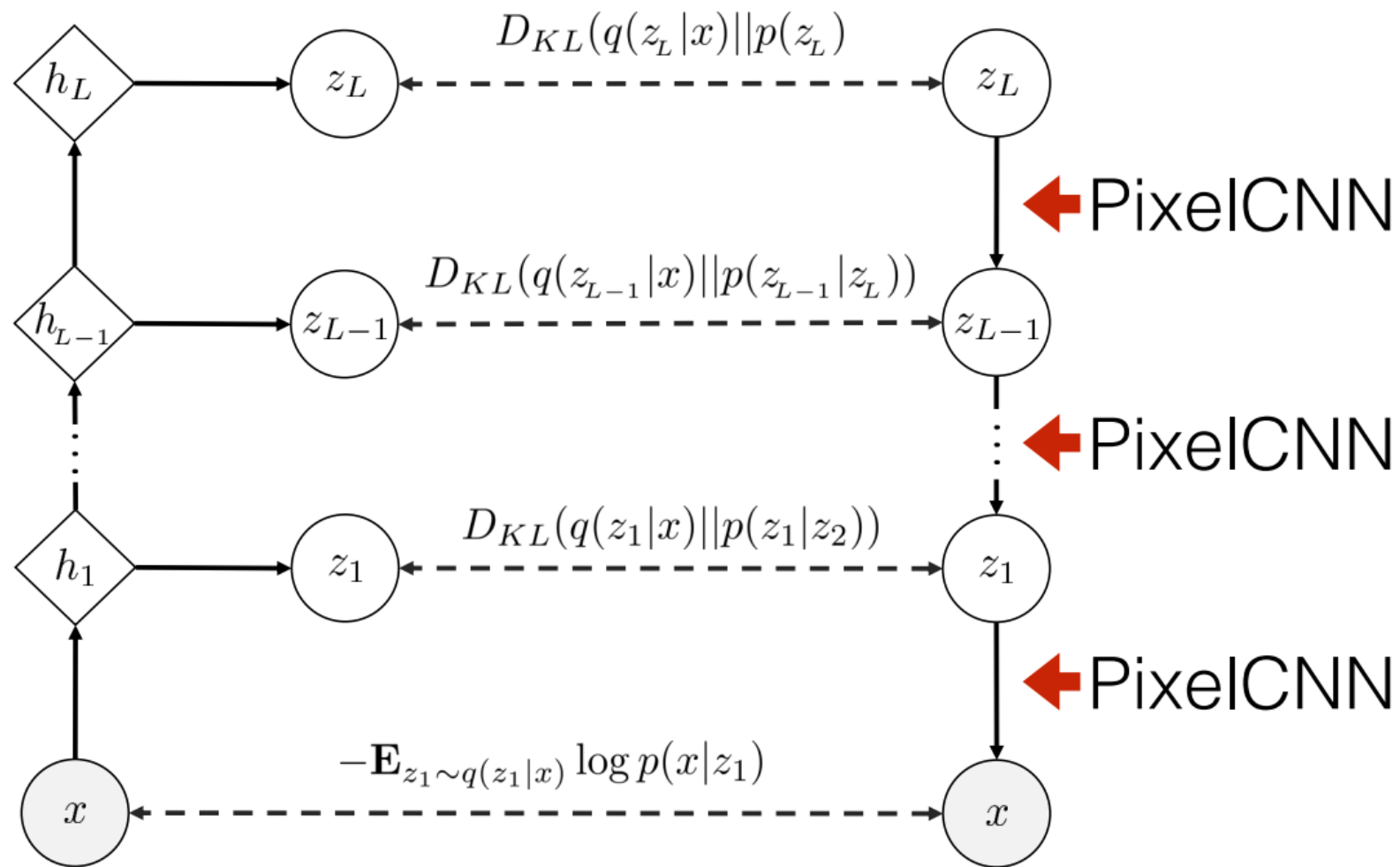


PixelCNN++ - Generations



- A standard VAE with a PixelCNN generator/decoder
- Be careful. Often the generator is so powerful, that the encoder/inference network is ignored ← Whatever the latent code z there will be a nice image generated

PixelVAE: A Latent Variable Model for Natural Images, Gulrajani et al., ICLR 2017



PixelVAE - Generations



64x64 LSUN Bedrooms



64x64 ImageNet

PixelVAE - Generations

Varying top latents



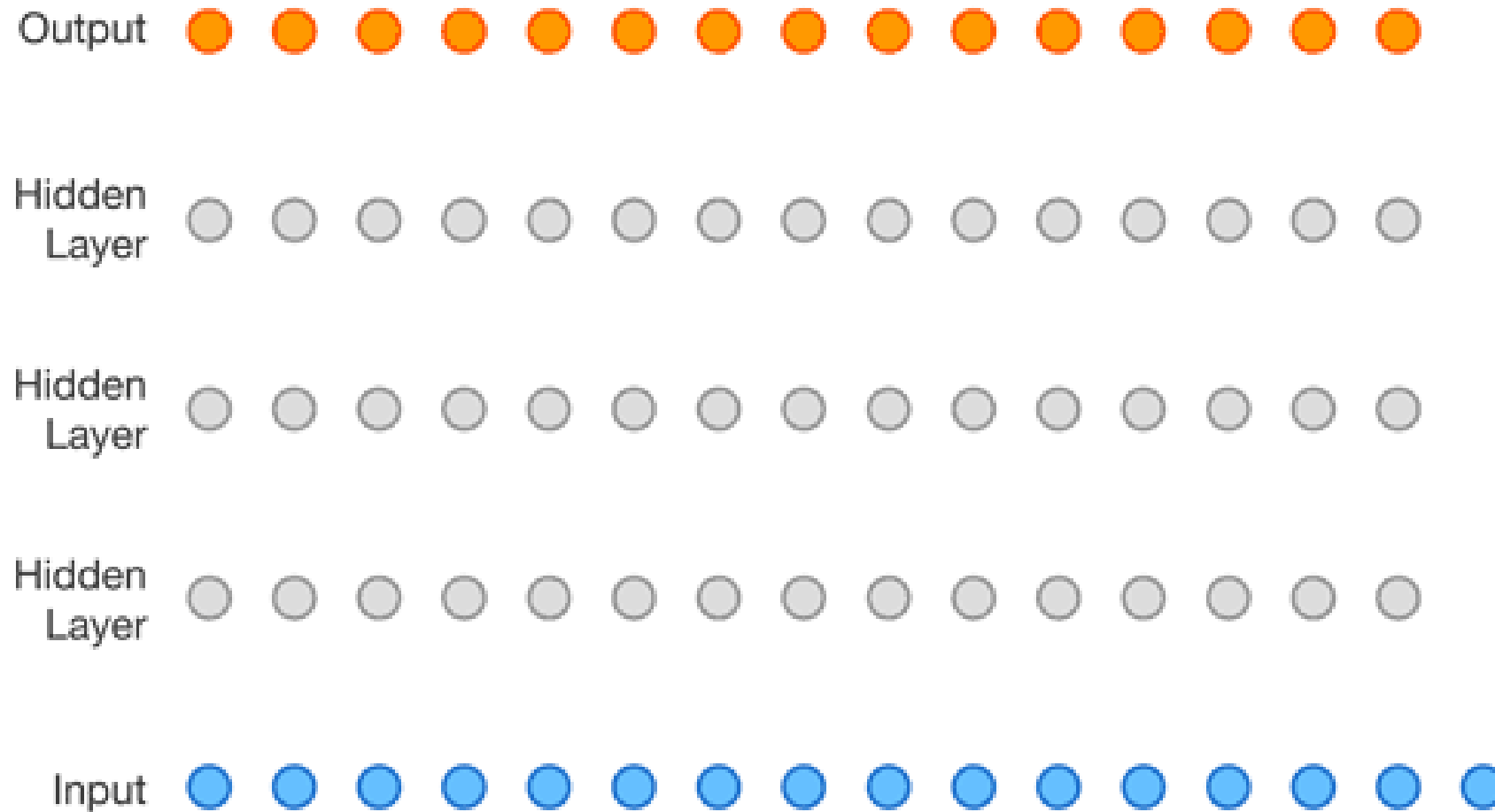
Varying bottom latents



Varying pixel-level noise



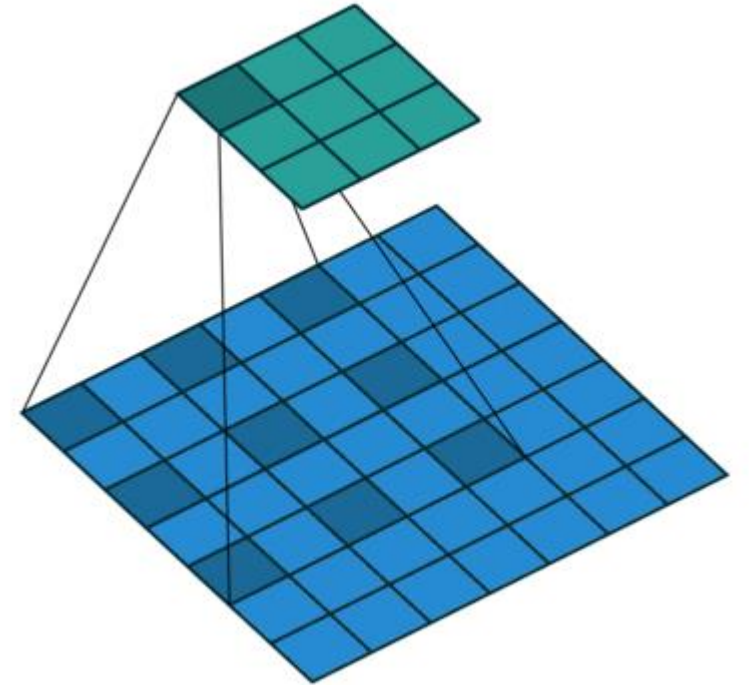
WaveNet



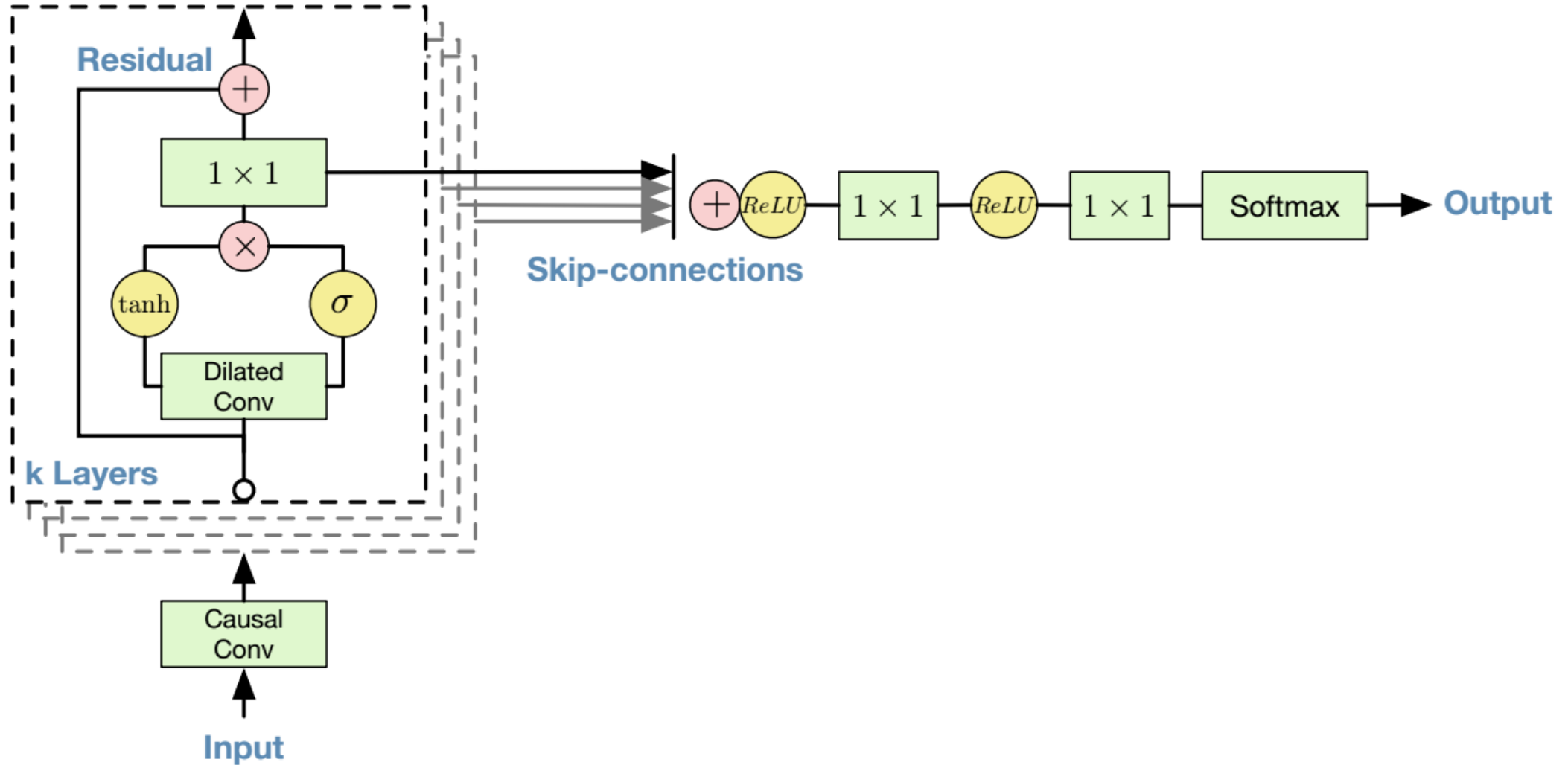
WaveNet: A Generative Model for Raw Audio, van den Oord et al., arXiv 2017

WaveNet

- Inspired by PixelRNN and PixelCNN
- Fully convolutional neural network
- Use dilated convolutions
- [Samples](#)

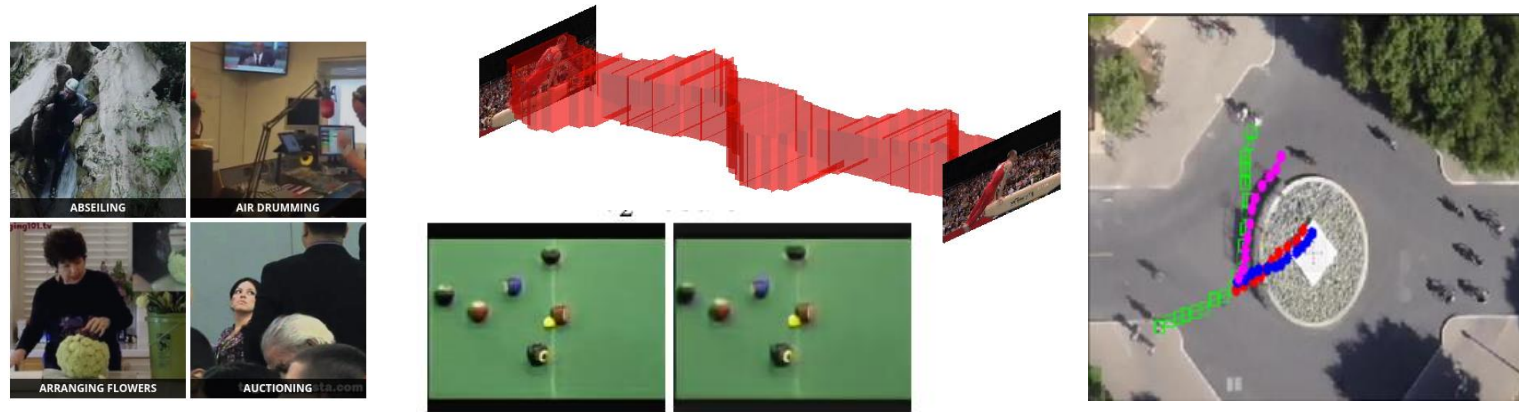


WaveNet architecture



VideoTime: Time-Aligned DenseNets

- Video Time: Properties, Encoders and Evaluation
 - A. Ghodrati, E. Gavves, C. Snoek, BMVC 2018
- How to model image sequences optimally?



Video Time: Properties, Encoders and Evaluation, Ghodrati, Gavves, Snoek, BMVC 2018

Properties of video

A standard VAE with a PixelCNN



Forward

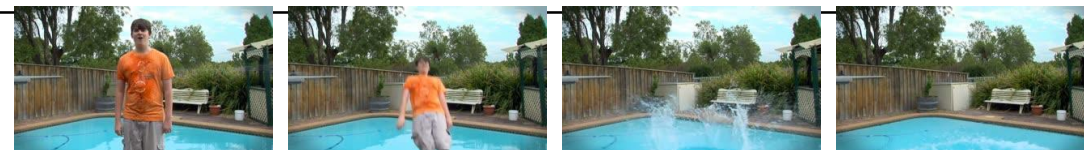


Backward

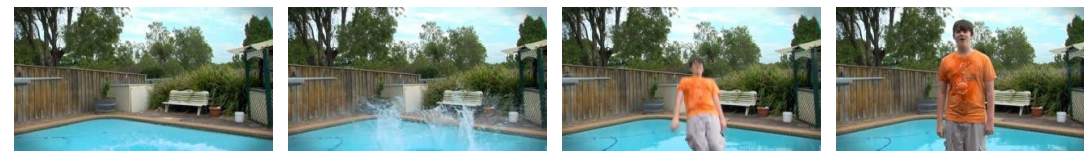
Video properties

- Temporal asymmetry

Natural order (+)



Reverse order (-)

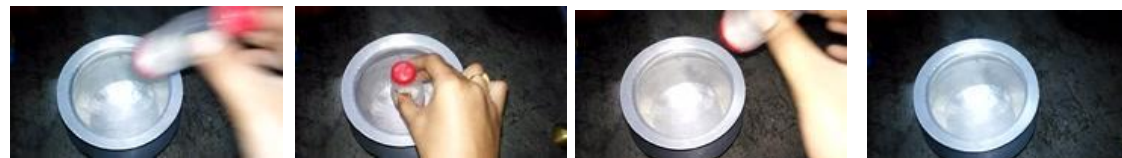


- Temporal continuity



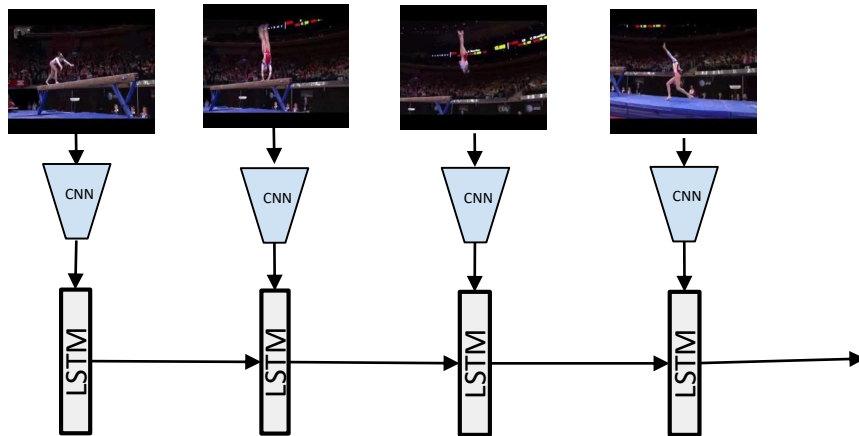
- Temporal causality

“Pretending to put something into something”

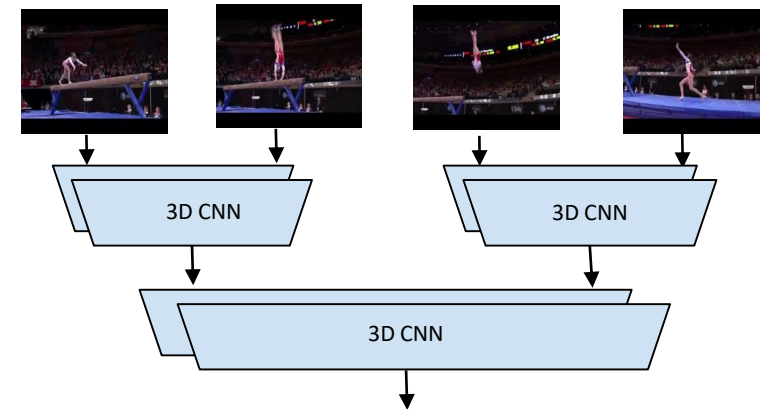


Current solutions

LSTMs learn transitions between subsequent states

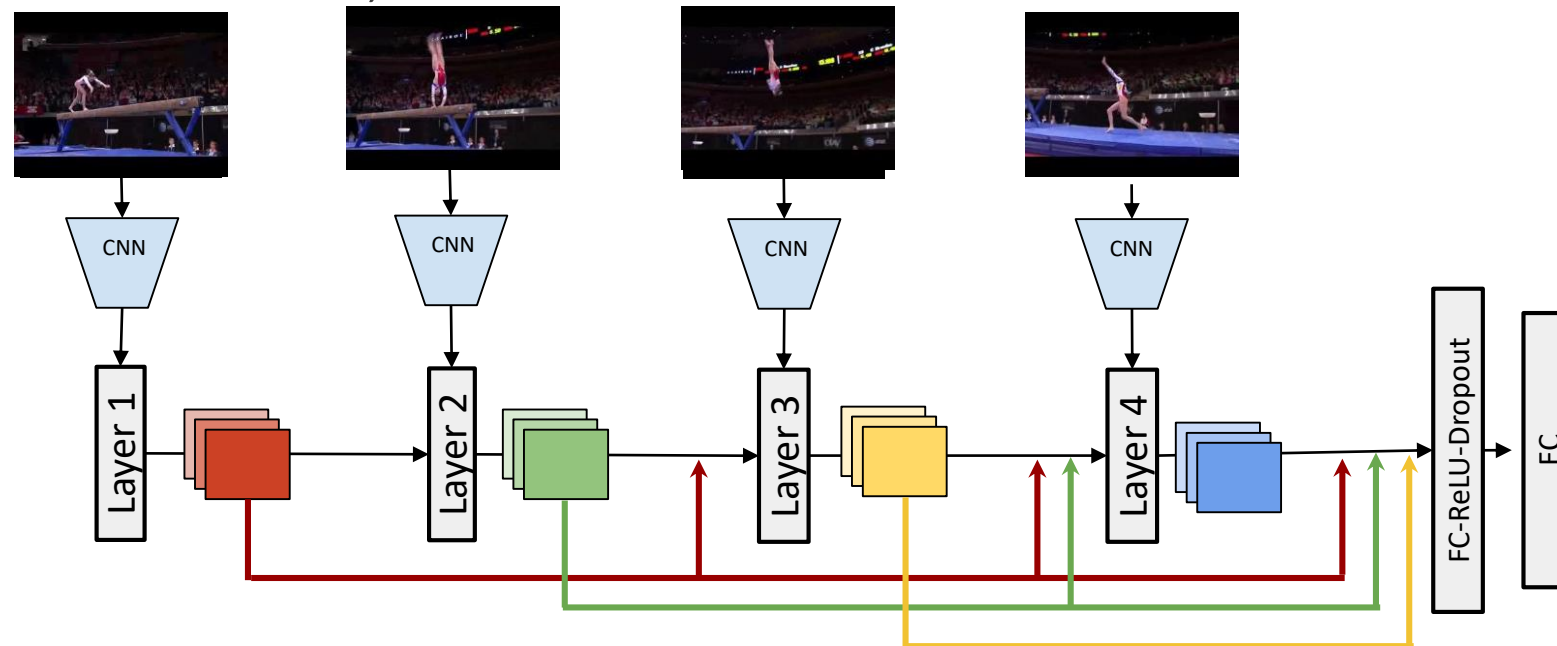


3D convolutions learn spatiotemporal patterns within a video

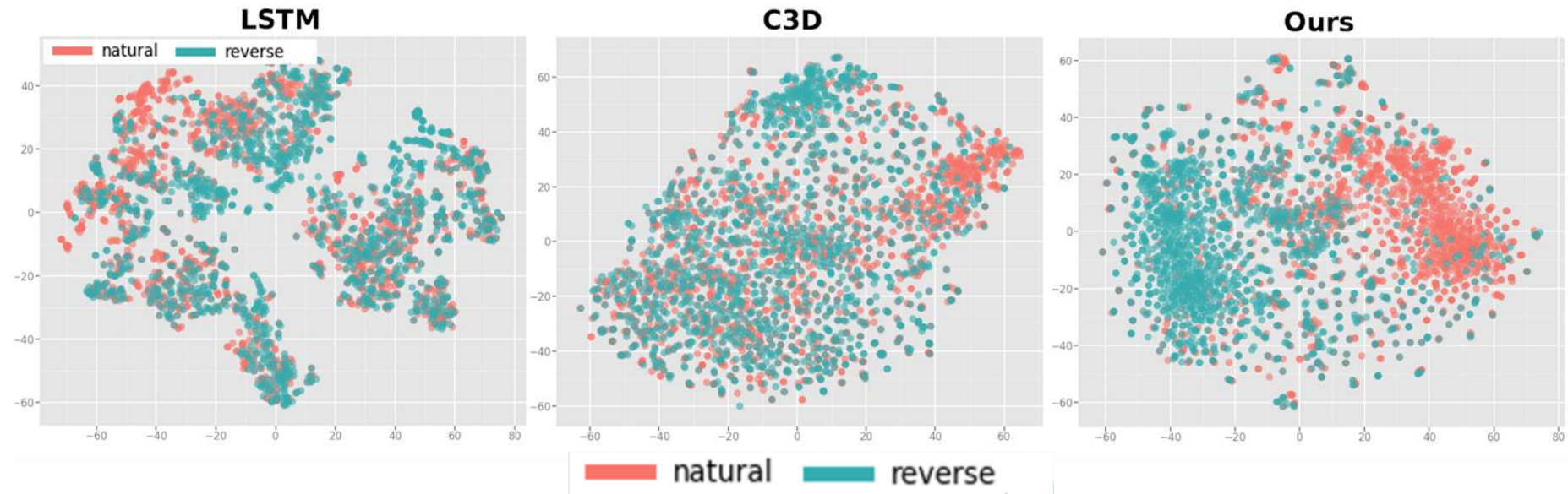


Time-Aligned DenseNets

- Autoregressive-like: each new frame directly depends on all previous frames
 - They are not generative
- No parameter sharing → like ConvNets, unlike RNNs
- Sequential → like RNNs, unlike ConvNets



Better latent space



Summary

- Autoregressive Models
- PixelCNN, PixelCNN++, PixelRNN
- WaveNet
- Time-Aligned DenseNets