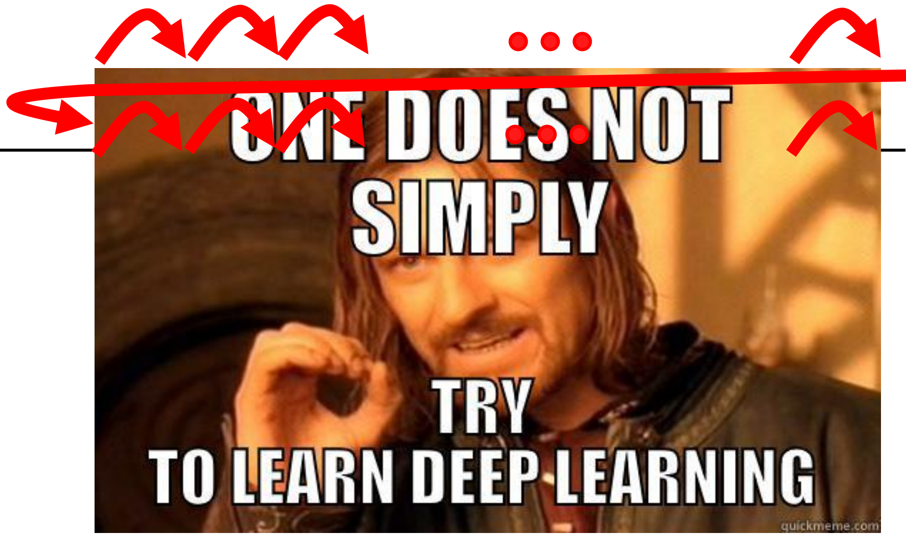
Modern VAE models

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→G→B
  - Each color conditioned on all colors from previous pixels and specific colors in th 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

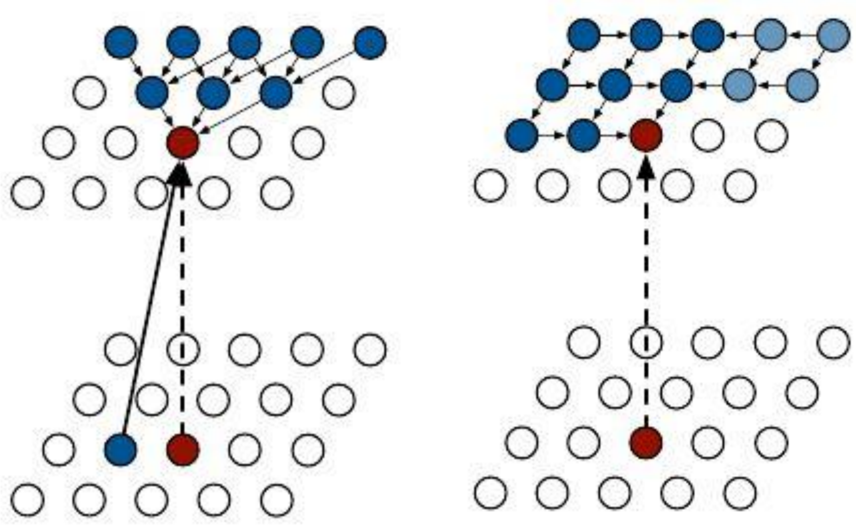
PixelRNN

- 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

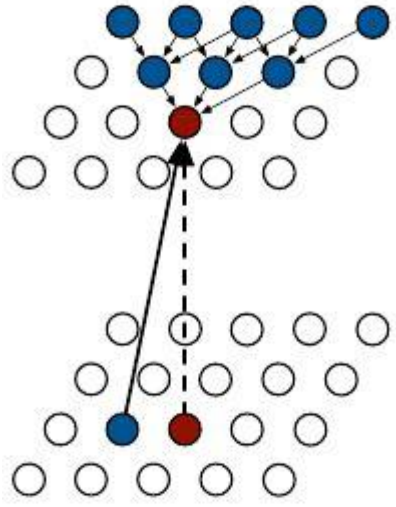
- Row LSTM
- Diagonal Bi-LSTM



Row LSTM   Diagonal Bi-LSTM

PixelRNN - RowLSTM

- Hidden state (i, j) =
  Hidden state (i-1, j-1) +
  Hidden state (i-1, j) +
  Hidden state (i-1, j+1) +
  p(i, j)
- By recursion the hidden state captures a fairly triangular region

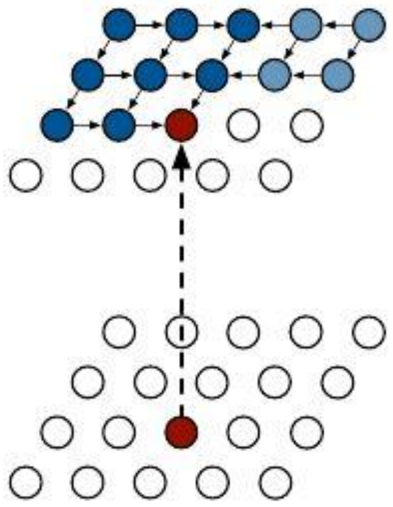


Row
LSTM

PixelRNN – Diagonal BiLSTM

- How to capture the whole previous context
- Pixel (i, j) =
  Pixel (i, j-1) +
  Pixel (i-1, j)

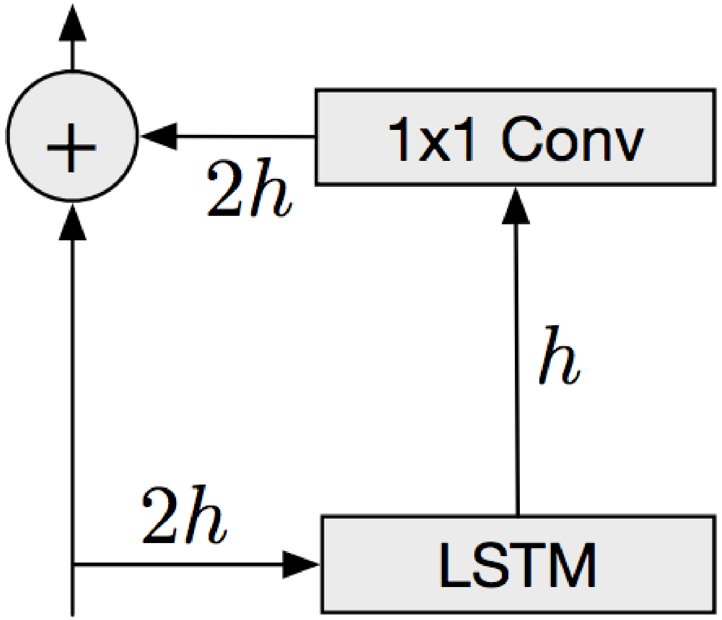
- Processing goes on diagonally
- Receptive layer encompasses entire region



Diagonal Bi-
LSTM

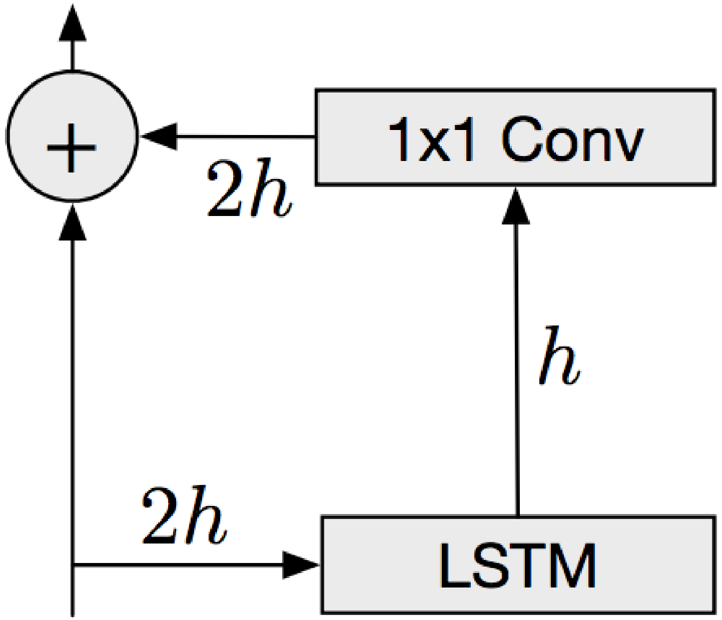
PixelRNN – Residual connections

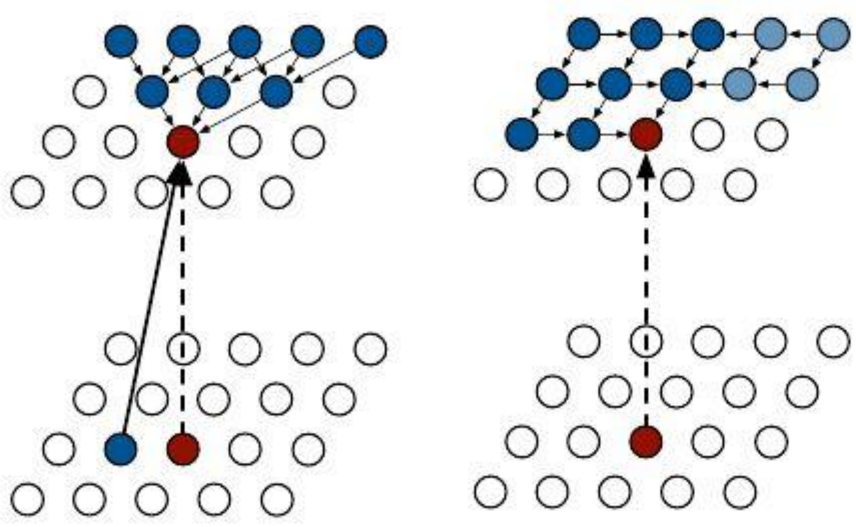
- Propagate signal faster
- Speed up convergence



PixelRNN – Pros & Cons

- Pros: good modelling of $p(x)$ → nice image generation
- Half pro: Residual connections speeds up convergence
- Cons: still slow training, slow generation





Row LSTM

Diagonal Bi-LSTM

PixelRNN - Generations

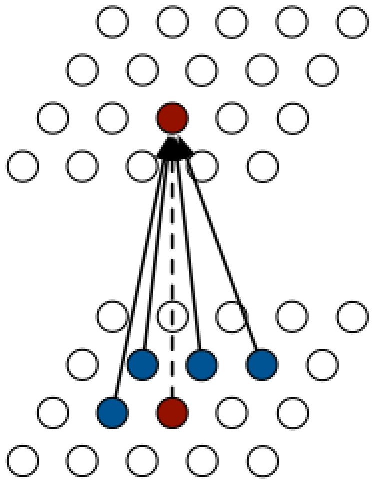
Figure 1. Image completions sampled from a PixelRNN.

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

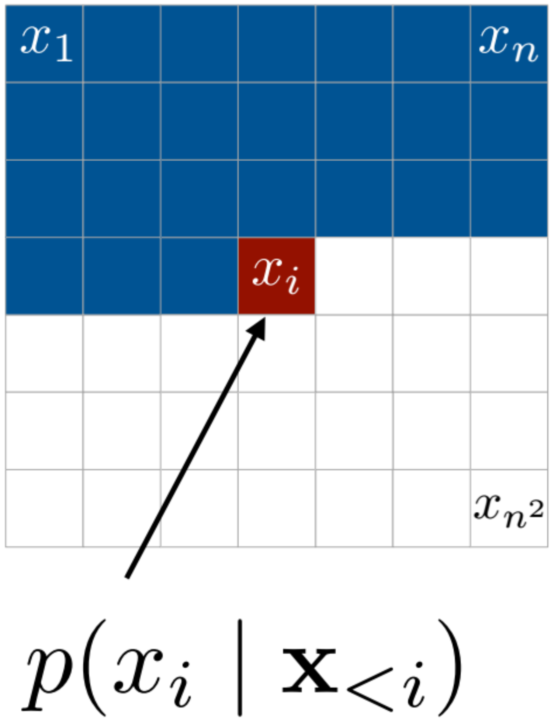
Stack of masked convolutions

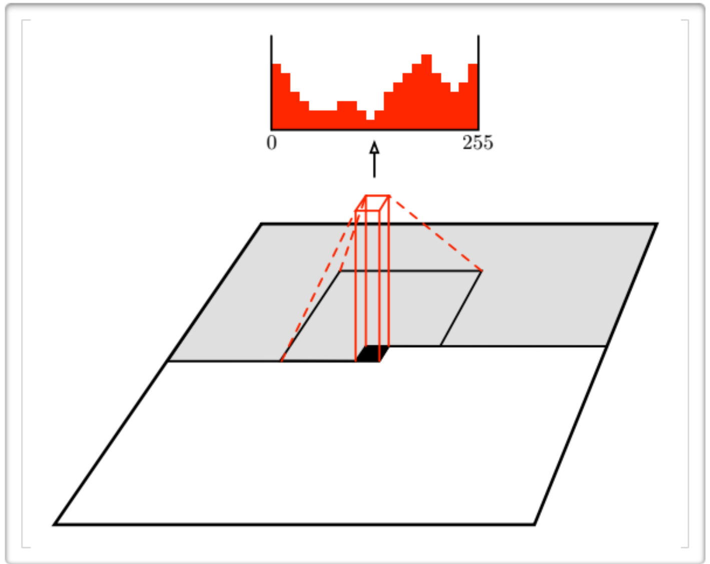
1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

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

PixelCNN

- Use masked convolutions again to enforce autoregressive relationships





PixelCNN – Pros and Cons

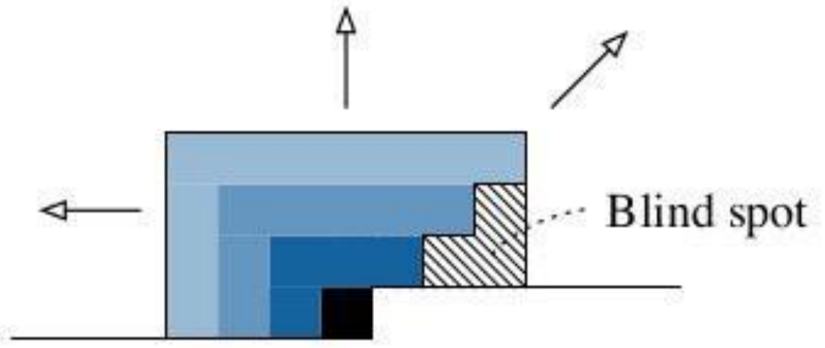
- Cons: Performance is worse than PixelRNN
- Why?

PixelCNN – Pros and Cons

- Cons: Performance is worse than PixelRNN
- Why?
- Not all past context is taken into account
- **New problem:** the cascaded convolutions create a **“blind spot”**

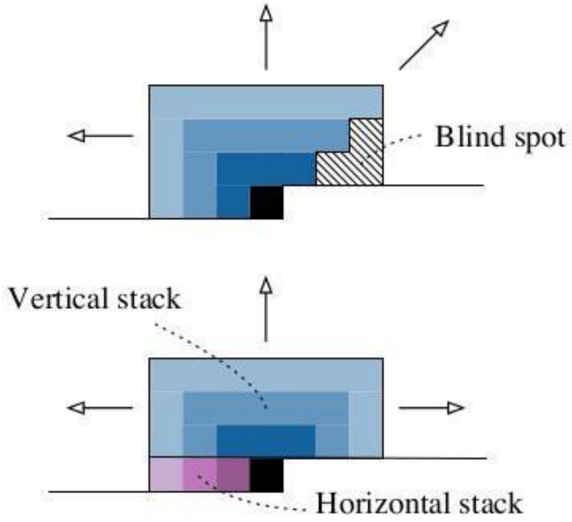
Blind spot

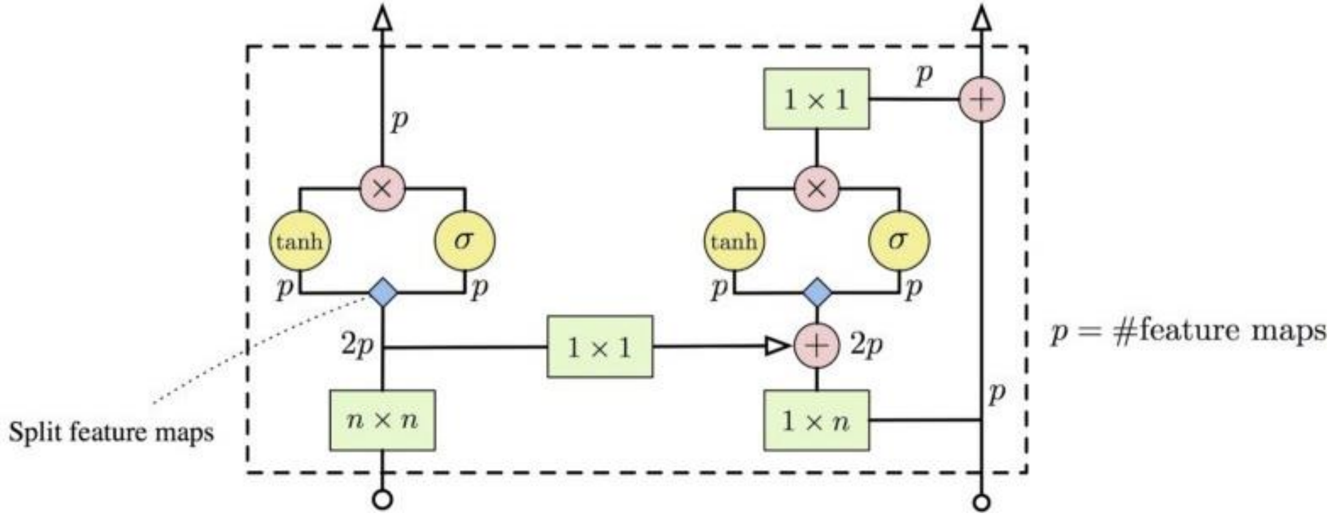
- Because of
  (a) the limited receptive field of convolutions and
  (b) computing all features at once (not sequentially)
  → cascading convolutions makes current pixel not depend on **<u>all</u>** previous
  → blind spot



Fixing the blind spot: Gated PixelCNN

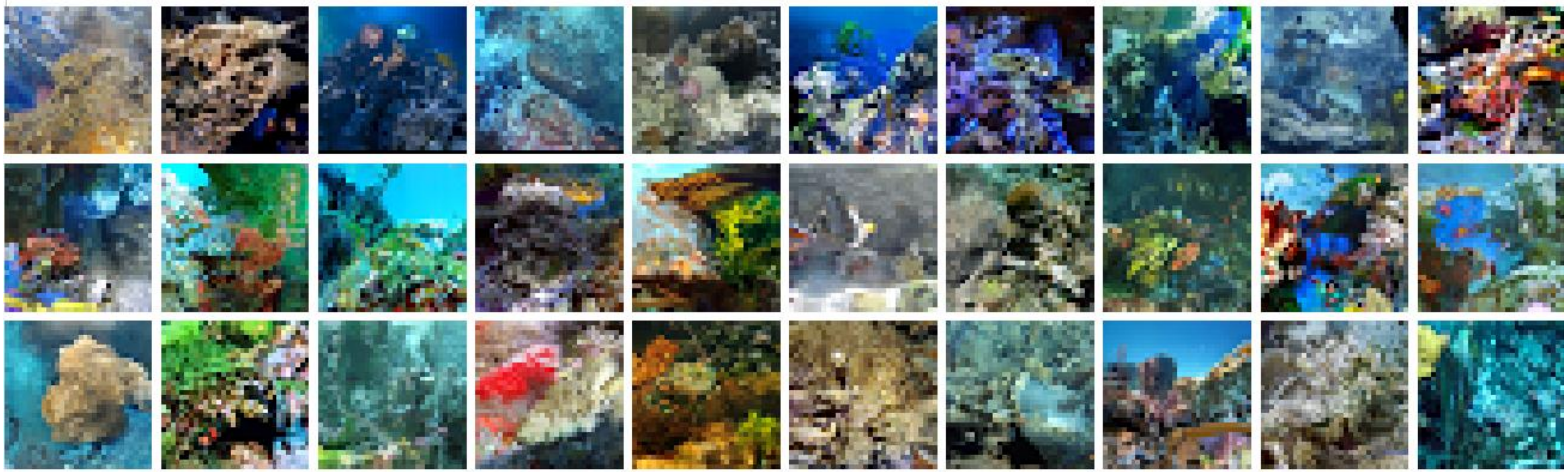
- Use two layers of convolutions stacks
  - Horizontal stack: conditions on current row and takes as input the previous layer output and the vertical stack
  - Vertical stack: conditions on all rows above current pixels
- Also replace ReLU with a $\tanh(W * x) \cdot \sigma(U * x)$





PixelCNN - Generations

- Coral reef



PixelCNN - Generation

- Sorrel horse

PixelCNN - Generation

- Sandbar

PixelCNN - Generation

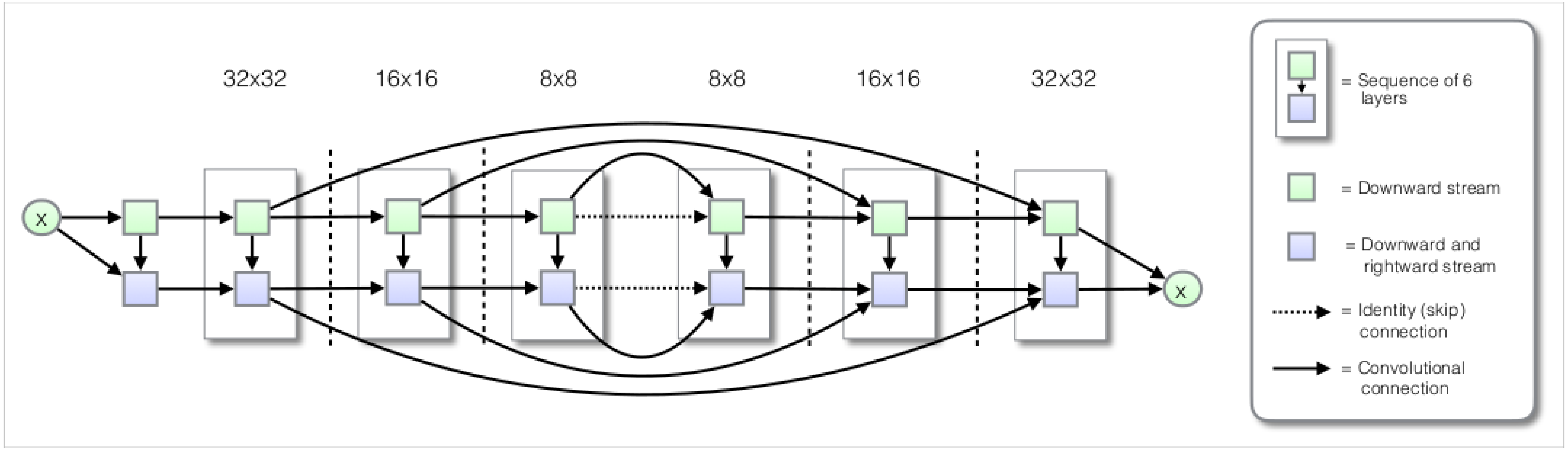
- Lhasa Apso

PixelCNN++

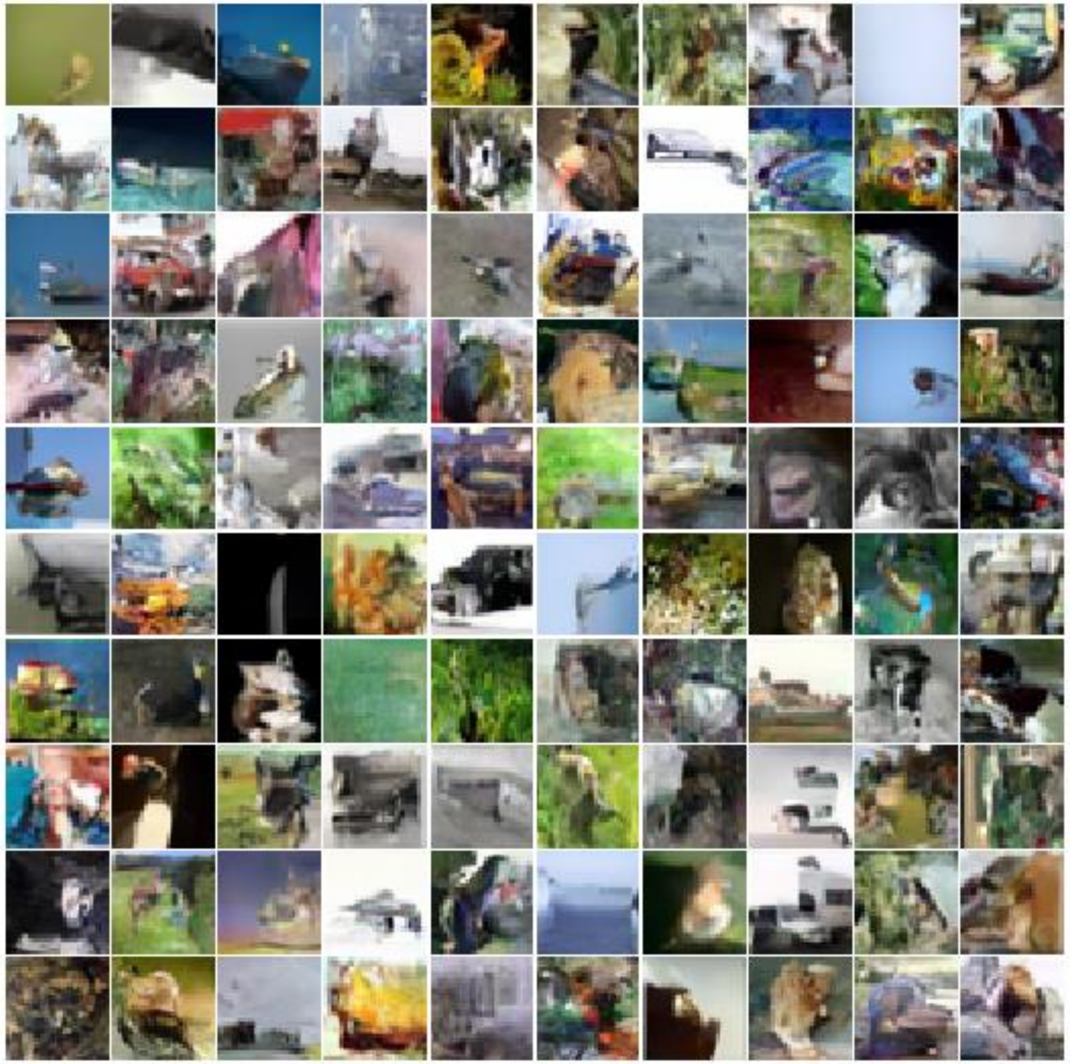
- Improving the PixelCNN model
- Replace the softmax output with a discretized logistic mixture lihelihood
  - Softmax is too memory consuming and gives sparse gradients
  - Instead, assume logistic distribution of intensity and round off to 8-bits
- Condition on whole pixels, not pixel colors
- Downsample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

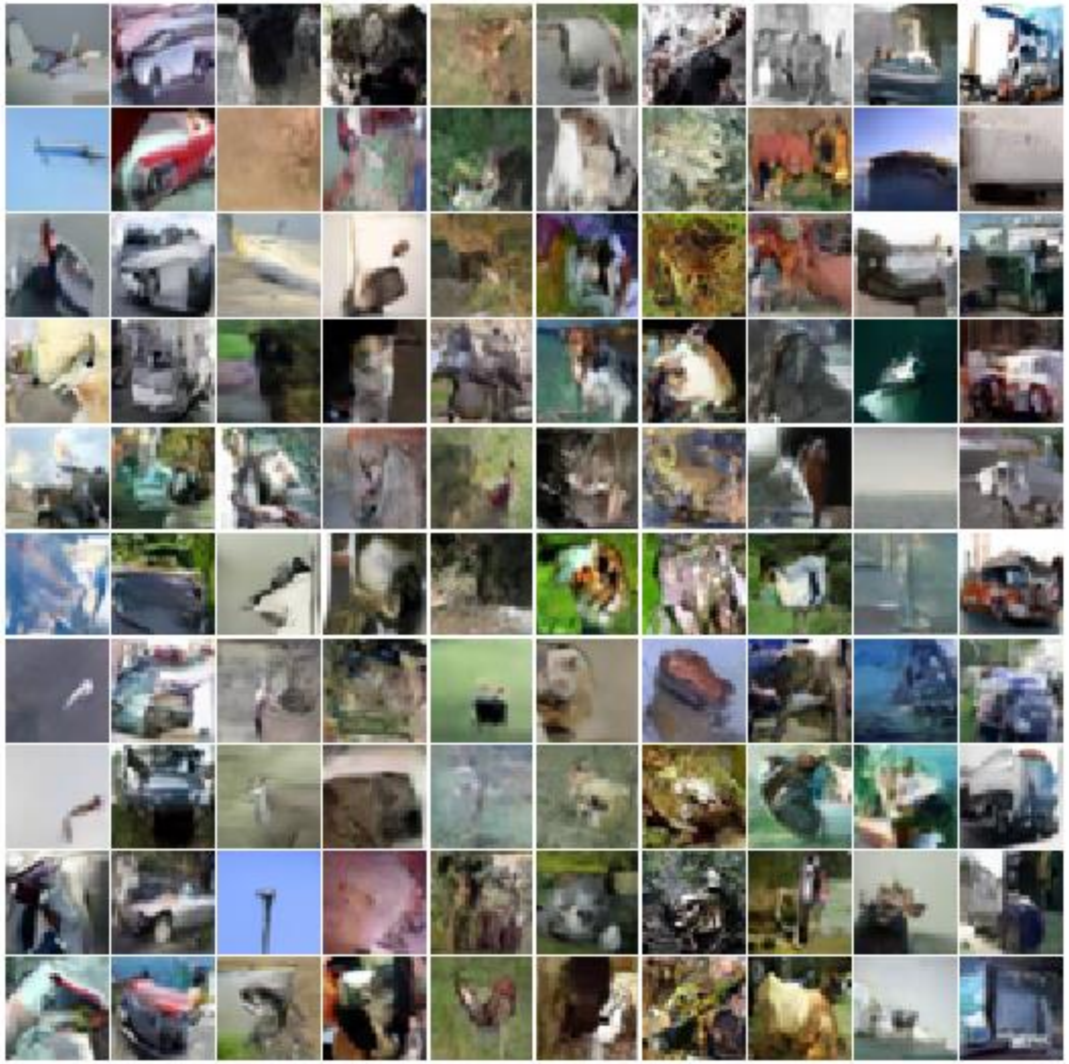
PixelCNN++: Improving the PixelCNN with Discretized Logistic, Salimans, Karpathy, Chen, Kingma, ICLR 2017

PixelCNN++


32x32
16x16
8x8
8x8
16x16
32x32
x
x
= Sequence of 6 layers
= Downward stream
= Downward and rightward stream
= Identity (skip) connection
= Convolutional connection

PixelCNN++ - Generations





Advantages/Disadvantages

- SoTA density estimation

- Quite slow because of autoregressive nature
  - They must sample sequentially

- They do not have a latest space

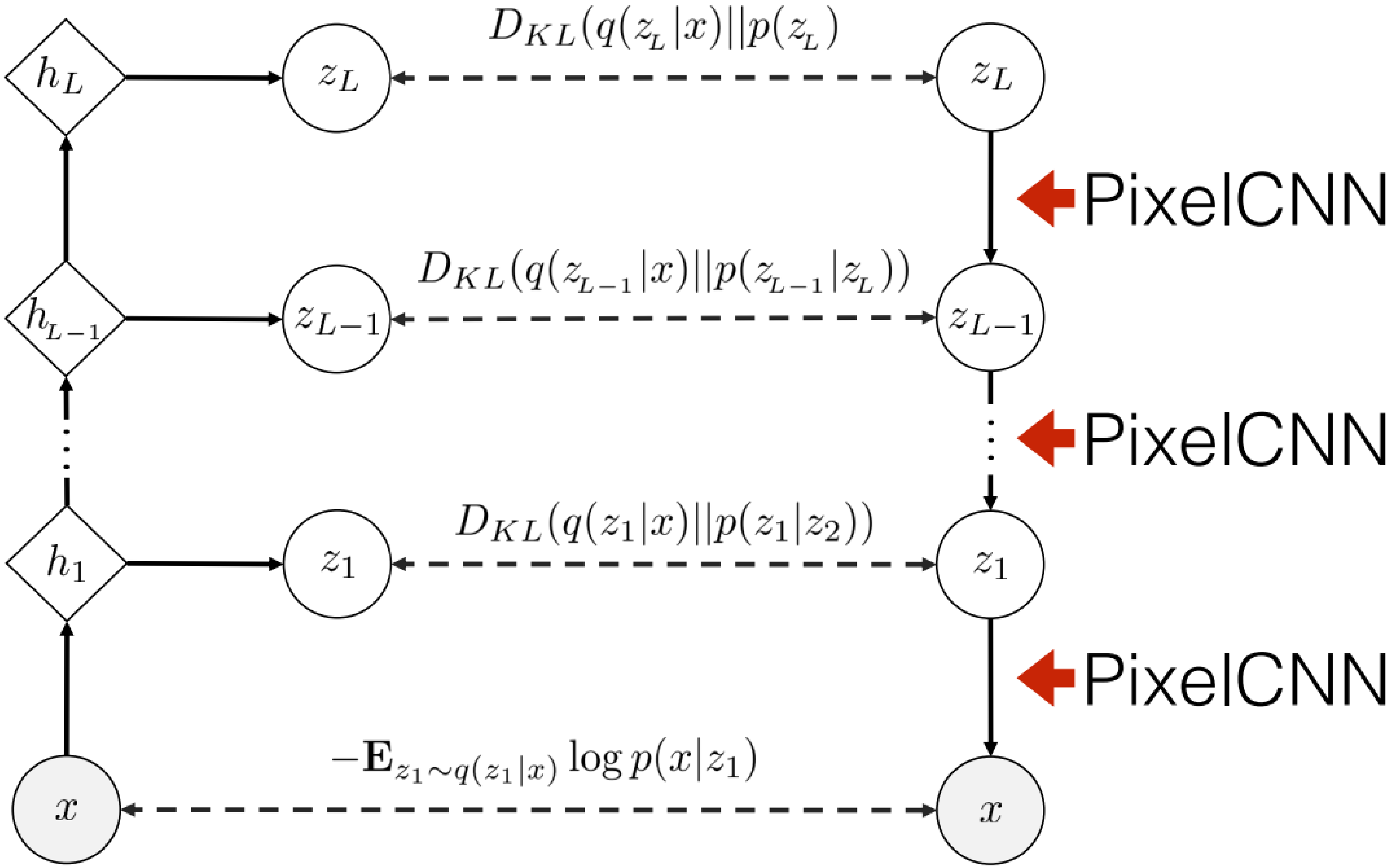
PixelVAE

- 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



$D_{KL}(q(z_L|x)||p(z_L))$
$D_{KL}(q(z_{L-1}|x)||p(z_{L-1}|z_L))$
$D_{KL}(q(z_1|x)||p(z_1|z_2))$
$-\mathbf{E}_{z_1 \sim q(z_1|x)} \log p(x|z_1)$
h_L
h_{L-1}
h_1
z_L
z_{L-1}
z_1
x
PixelCNN
PixelCNN
PixelCNN

PixelVAE - Generations

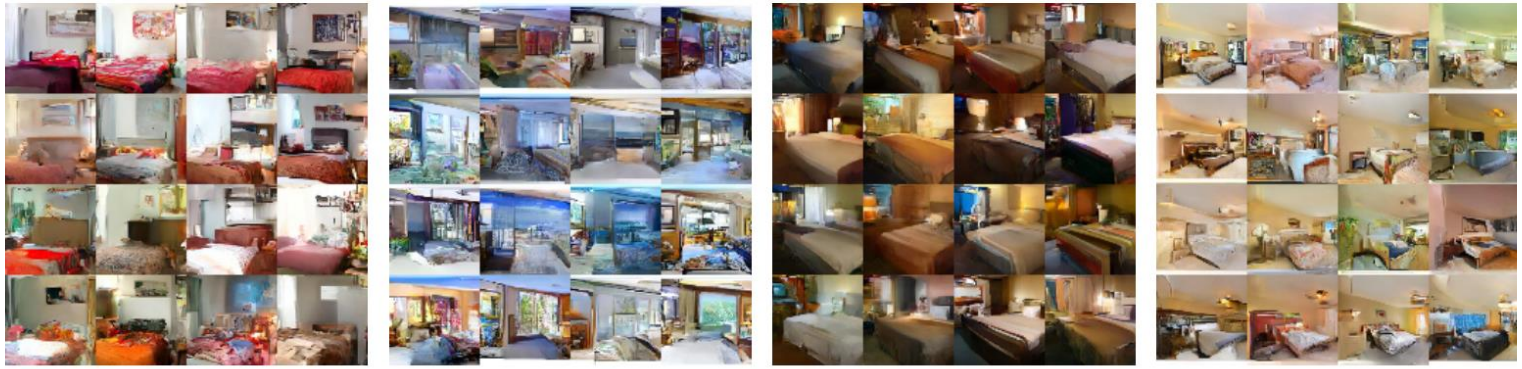
64x64 LSUN
Bedrooms

64x64 ImageNet

PixelVAE - Generations

Varying top latents

Varying bottom latents



Varying pixel-level noise