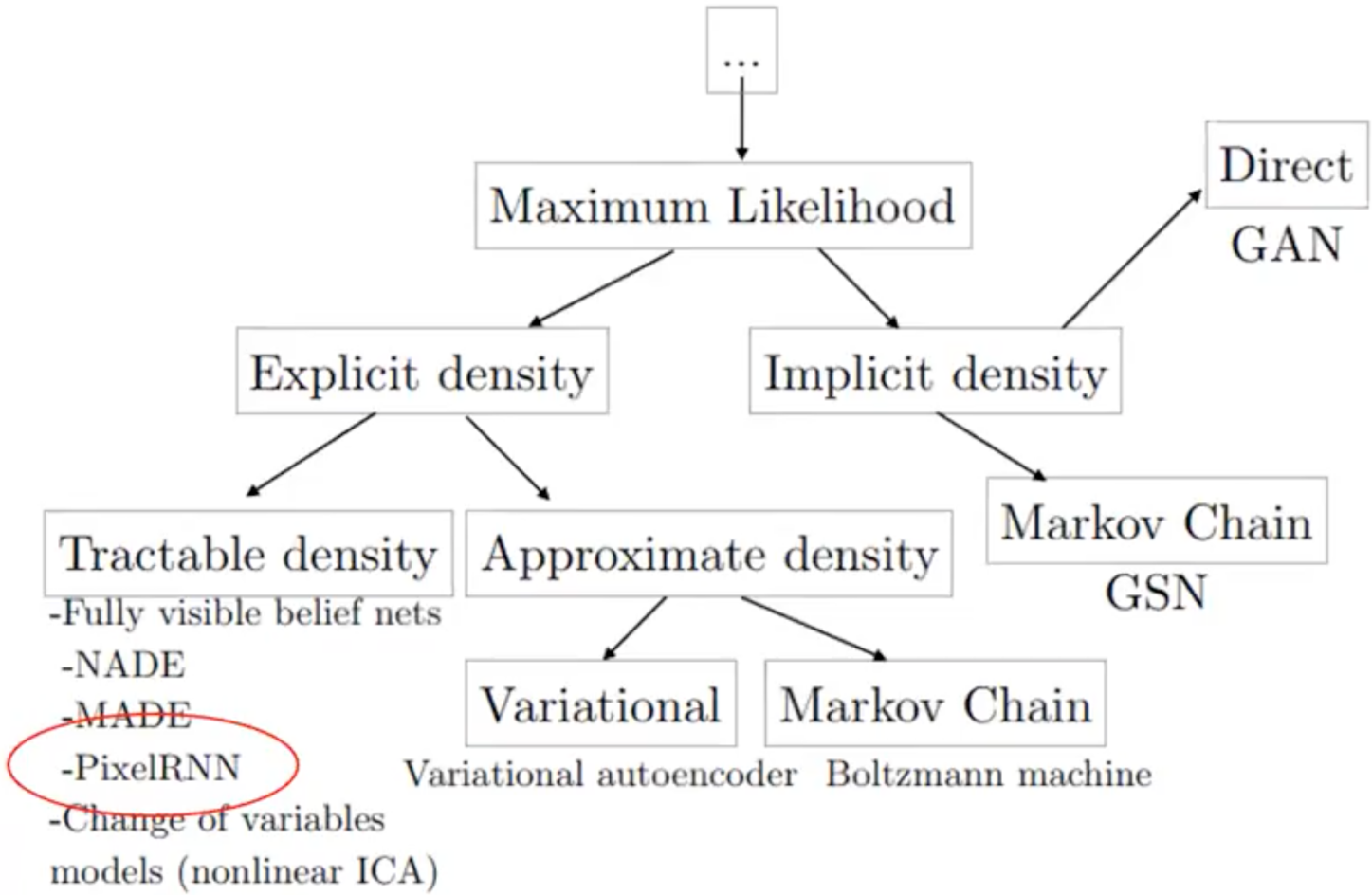
Paper overview:
Pixel Recurrent Neural Networks

Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu
Google DeepMind 19 Aug 2016

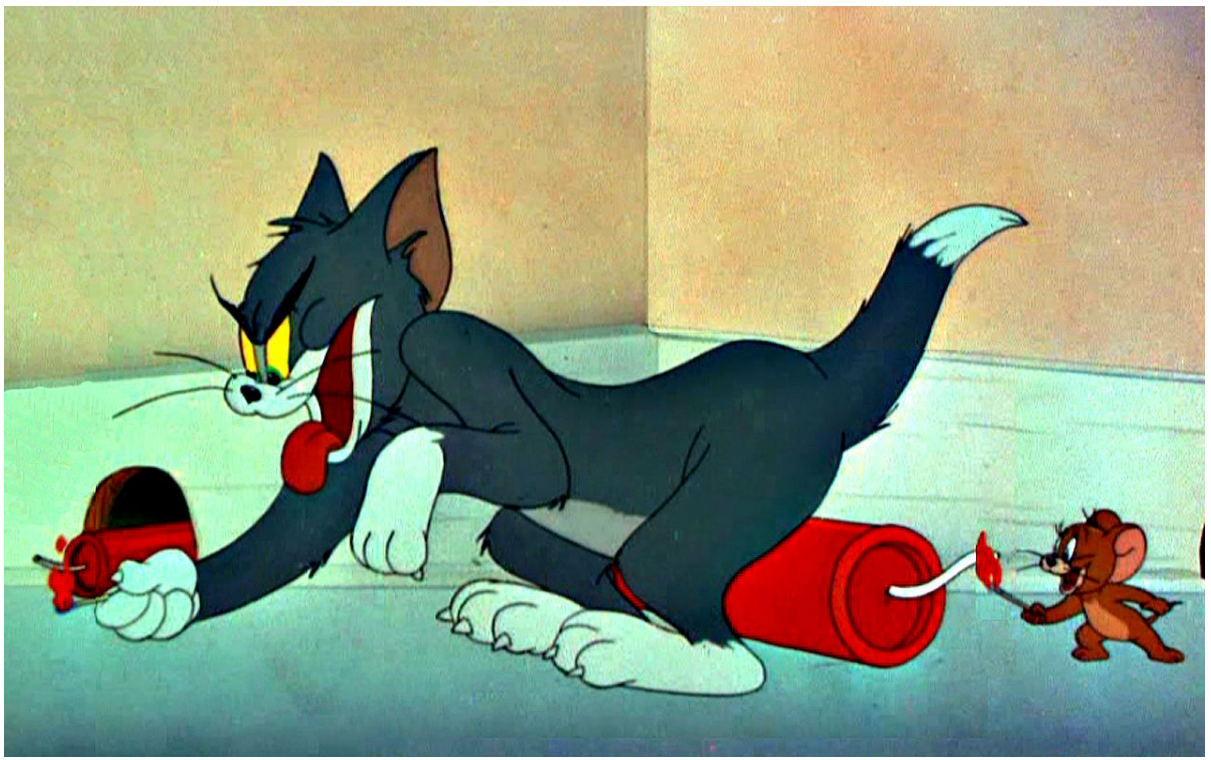
Musatkina Daria
CS HSE

Generative model!



Intuition

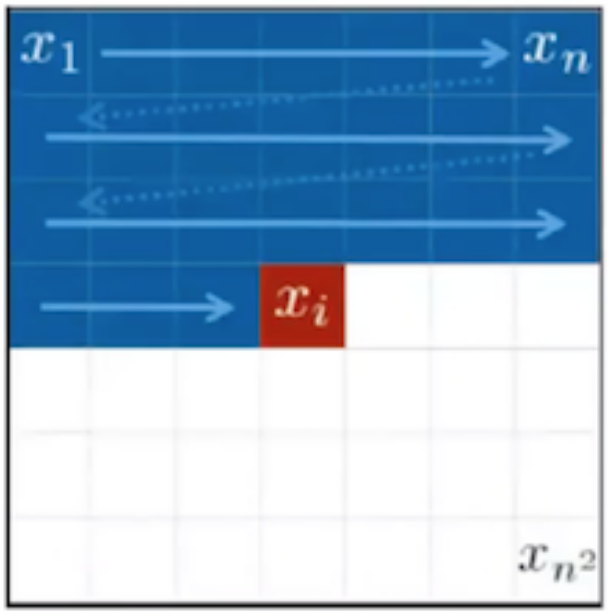
How to include statistical dependencies over hundreds of pixels?



Intuition

$$p(\mathbf{x}) = p(x_1, x_2, ..., x_{n^2})$$

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

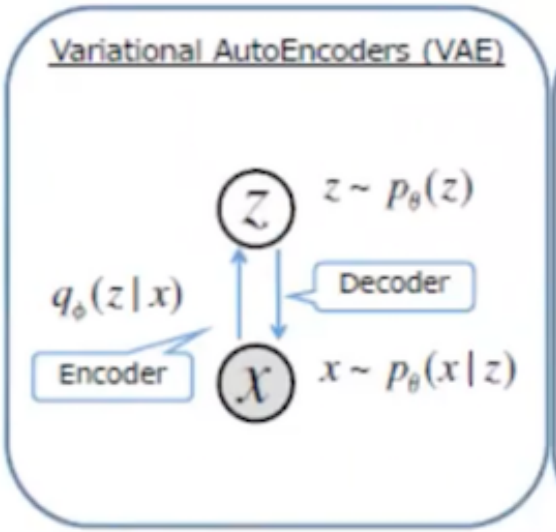


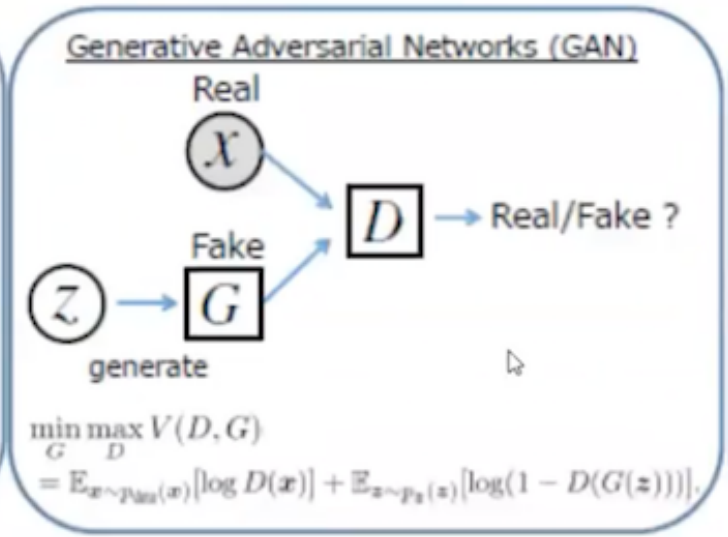
A sequential problem!

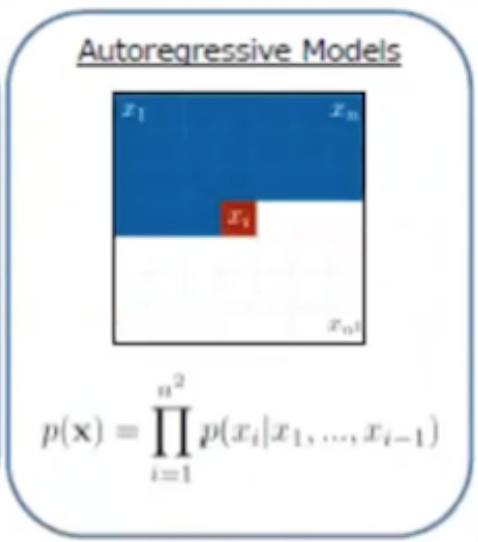
There are also three color channels for each pixel x_i, so

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

Dominating Image Generation Models





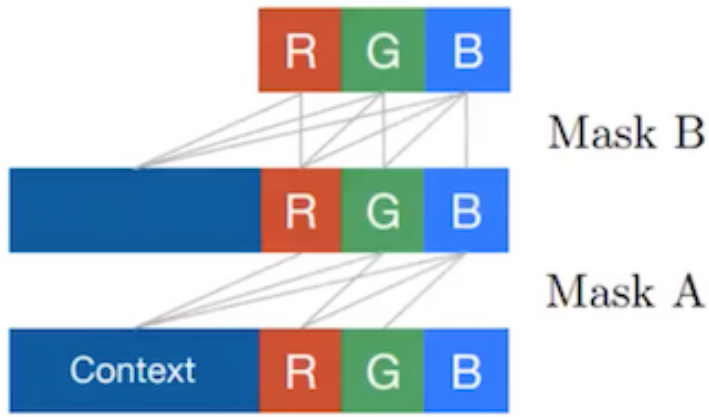


	VAE	GAN	Autoregressive Models
Pros.	- Efficient inference with approximate latent variables.	- generate sharp image. - no need for any Markov chain or approx networks during sampling.	- very simple and stable training process - currently gives the best log likelihood. - tractable likelihood
Cons.	- generated samples tend to be blurry.	- difficult to optimize due to unstable training dynamics.	- relatively inefficient during sampling

Mask

Channel Masks

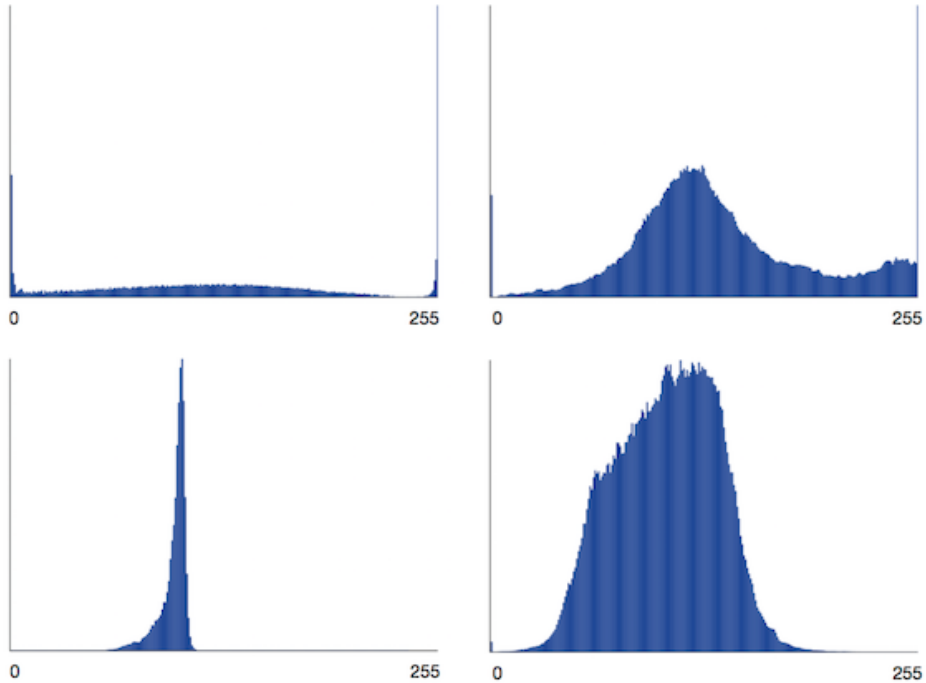
- Sequential order: R $\rightarrow$ G $\rightarrow$ B
- Used in input-to-state convolutions
- Two types of masks:
  - Mask A used in the first layer; channels are **not** connected to themselves.
  - Mask B used in all other subsequent layers; channels are connected to themselves.



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

Continuous distribution?

Treat pixels as discrete variables: do classification in every channel(256 classes indicating pixel values 0-255) with final softmax layer.

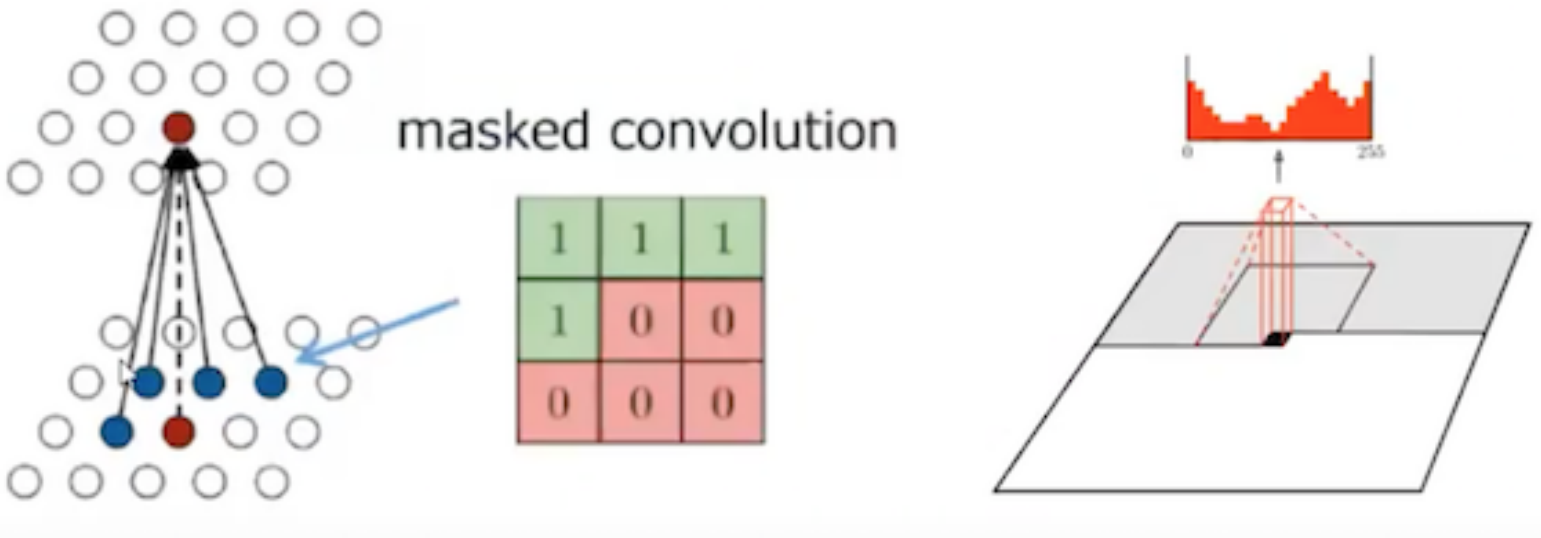


Models

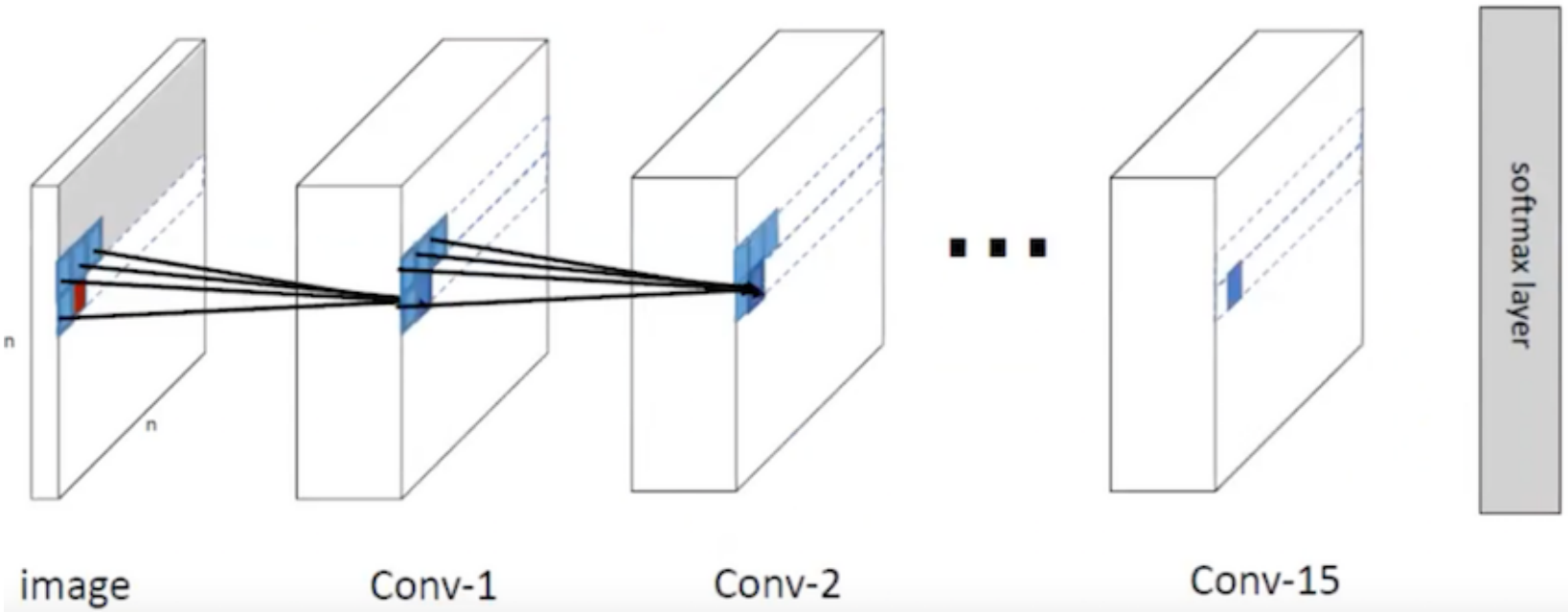
- PixelCNN
- PixelRNN
  - RowLSTM
  - Diagonal BiLSTM
  - Multiscale PixelRNN

PixelCNN

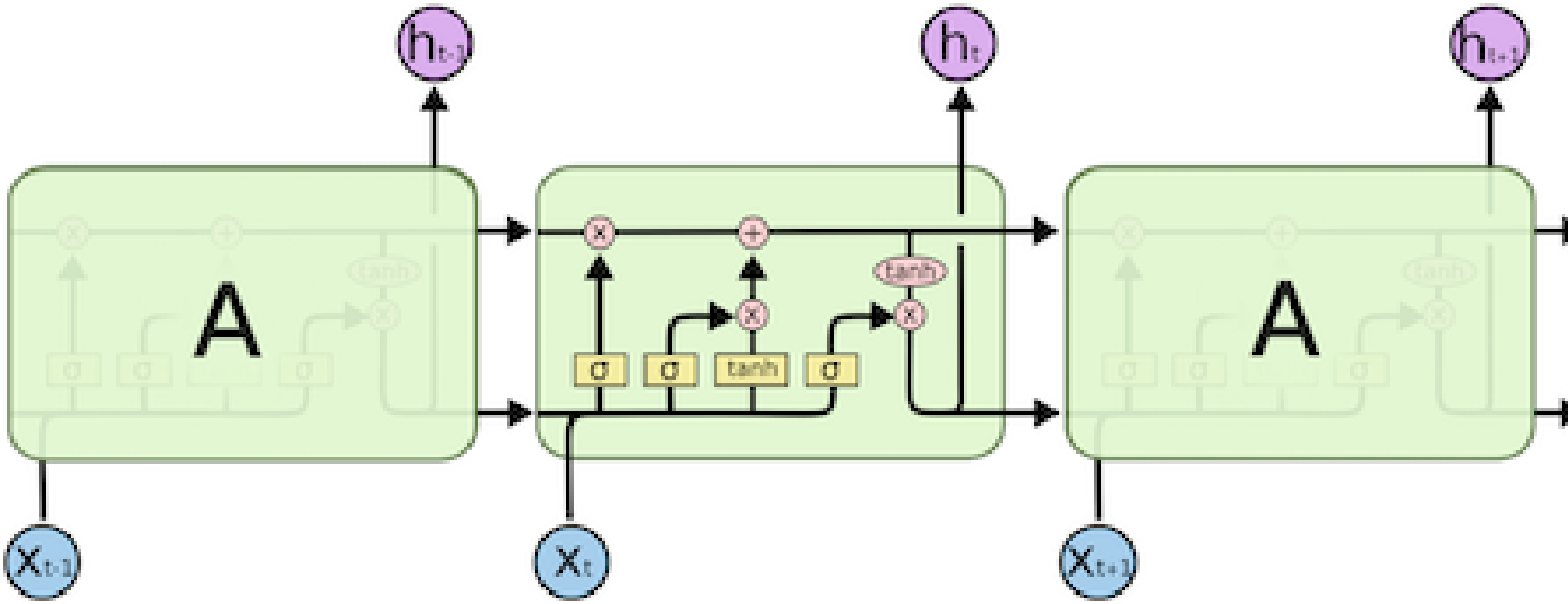
- 2D convolution on previous layer
- Apply mask so a pixel does not see future pixels



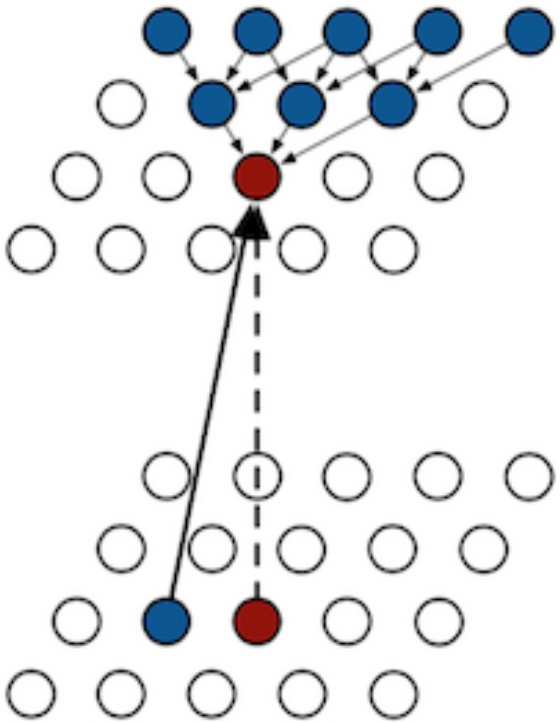
PixelCNN



LSTM reminder



RowLSTM



Row LSTM

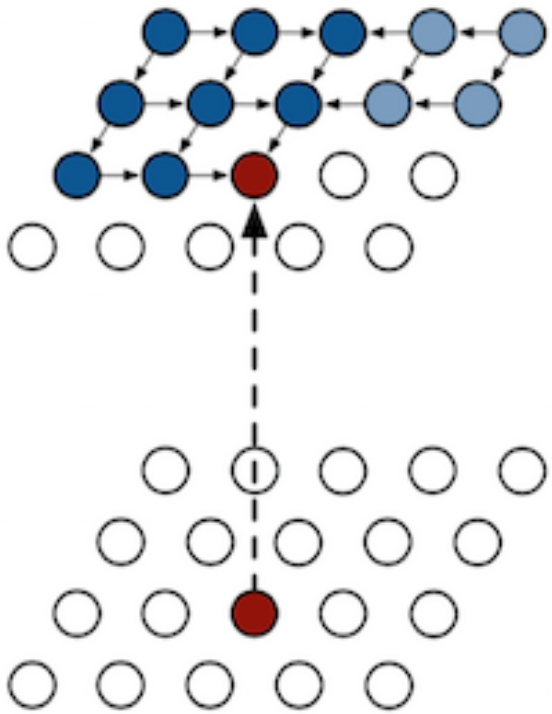
$$[\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] = \sigma(\mathbf{K}^{ss} \circledast \mathbf{h}_{i-1} + \mathbf{K}^{is} \circledast \mathbf{x}_i)$$
$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i$$
$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$$

$\mathbf{x}_i$ of size $h \times n \times 1$ is a row i of the input map.
The input-to-state component is precomputed for the entire 2D input map.

RowLSTM

PixelCNN	Row LSTM	Diagonal BiLSTM
7×7 conv mask A		
Multiple residual blocks: (see fig 5)		
Conv 3×3 mask B	Row LSTM i-s: 3×1 mask B s-s: 3×1 no mask	Diagonal BiLSTM i-s: 1×1 mask B s-s: 1×2 no mask
ReLU followed by 1×1 conv, mask B (2 layers)		
256-way Softmax for each RGB color (Natural images) or Sigmoid (MNIST)		

Diagonal BiLSTM

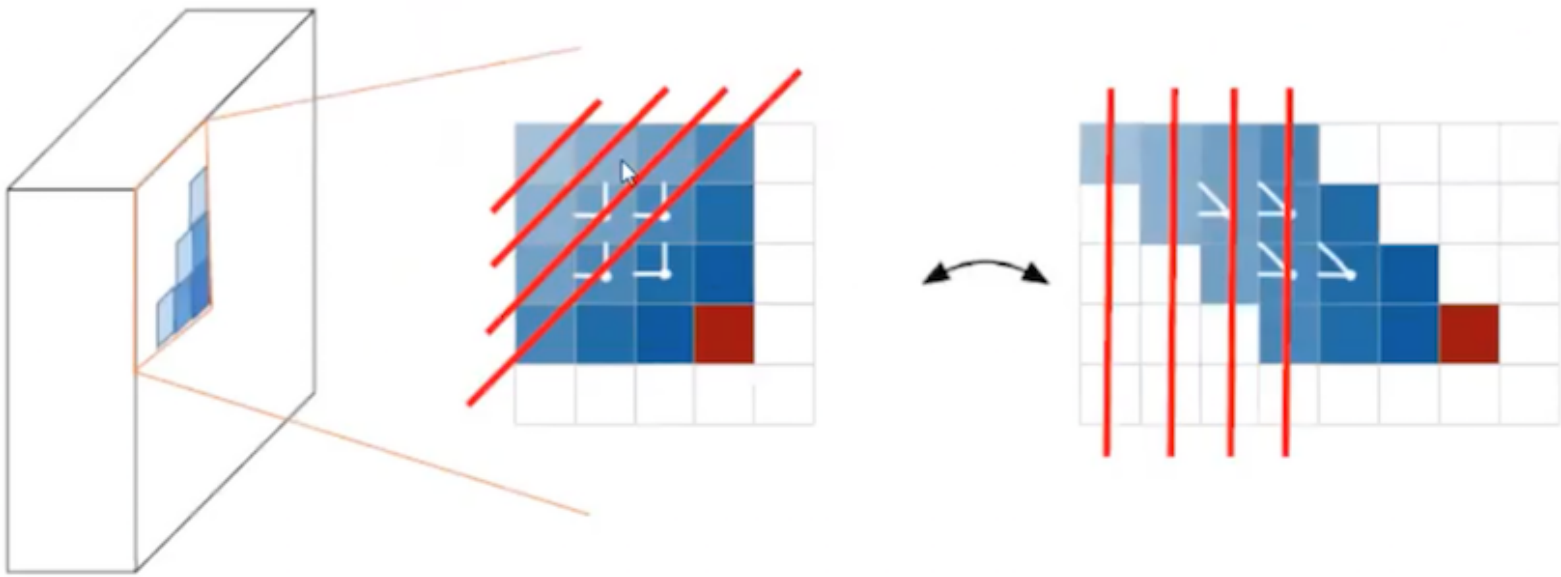


Diagonal BiLSTM

$$[\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] = \sigma(\mathbf{K}^{ss} \circledast \mathbf{h}_{i-1} + \mathbf{K}^{is} \circledast \mathbf{x}_i)$$
$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i$$
$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$$

Diagonal BiLSTM

To optimize, we skew the feature maps so it can be parallelized (input is a column of skewed map).

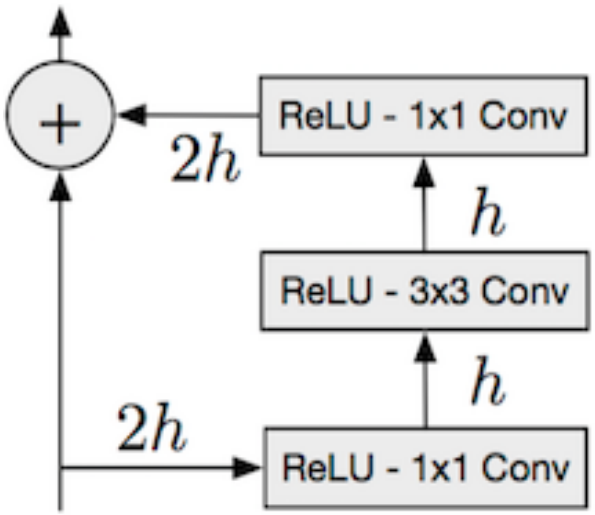


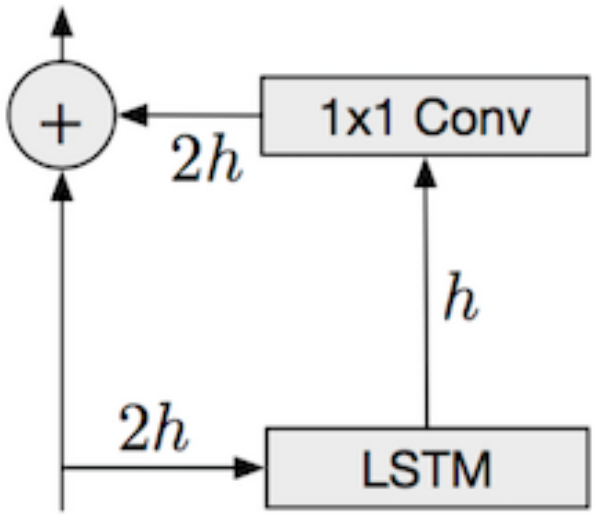
Brief Diagonal BiLSTM

- Bidirectional: 2LSTMs (from top left and top right)
- The input is a column of skewed map
- Capture entire context

PixelCNN	Row LSTM	Diagonal BiLSTM
7×7 conv mask A		
Multiple residual blocks: (see fig 5)		
Conv 3×3 mask B	Row LSTM i-s: 3×1 mask B s-s: 3×1 no mask	Diagonal BiLSTM i-s: 1×1 mask B s-s: 1×2 no mask
ReLU followed by 1×1 conv, mask B (2 layers)		
256-way Softmax for each RGB color (Natural images) or Sigmoid (MNIST)		

Residual Blocks





Residual Blocks

	No skip	Skip
No residual:	3.22	3.09
Residual:	3.07	3.06

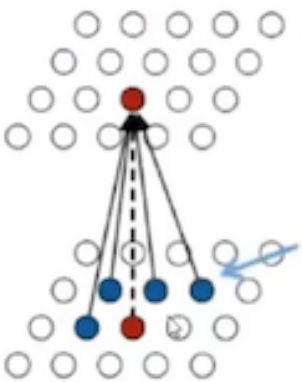
Table 2. Effect of residual and skip connections in the Row LSTM network evaluated on the Cifar-10 validation set in bits/dim.

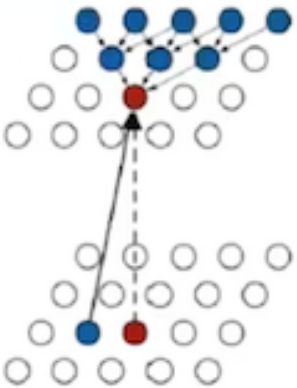
# layers:	1	2	3	6	9	12
NLL:	3.30	3.20	3.17	3.09	3.08	3.06

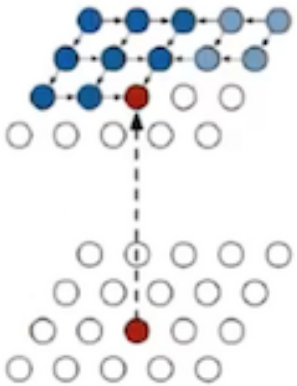
Table 3. Effect of the number of layers on the negative log likelihood evaluated on the CIFAR-10 validation set (bits/dim).

Comparision

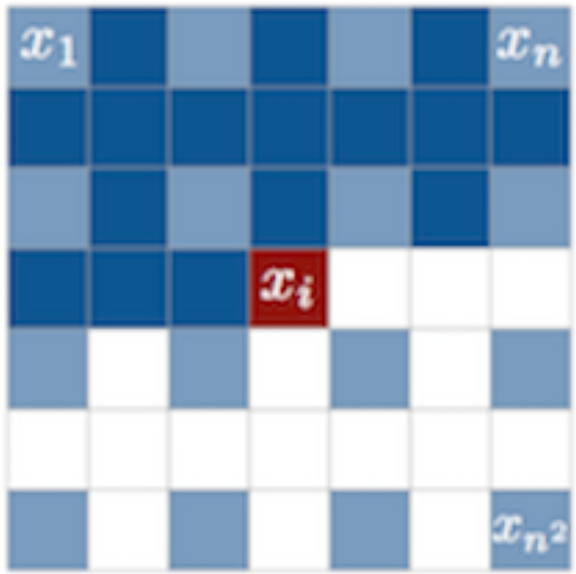
PixelCNN	PixelRNN – Row LSTM	PixelRNN – Diagonal BiLSTM
Full dependency field	Triangular receptive field	Full dependency field
Fastest	Slow	Slowest
Worst log-likelihood	-	Best log-likelihood







Multiscale model



Multi-scale context

Outcomes

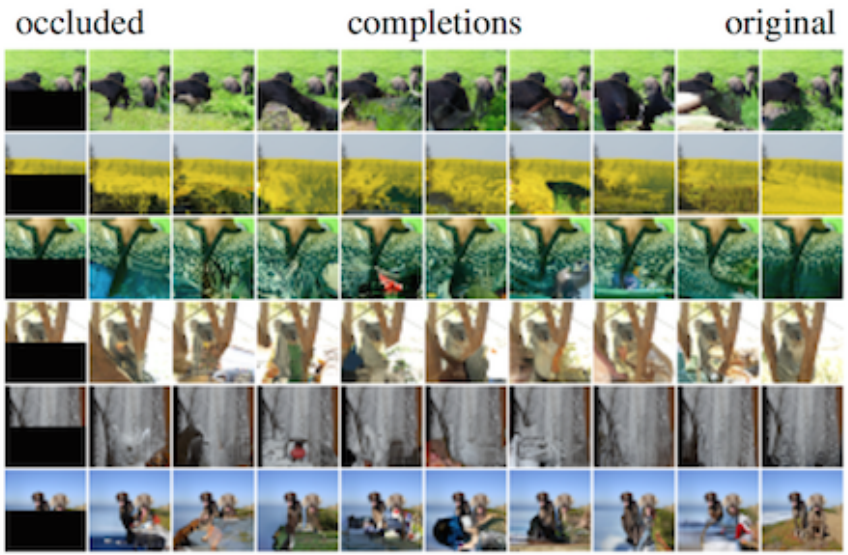
Perfomance on MNIST

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**

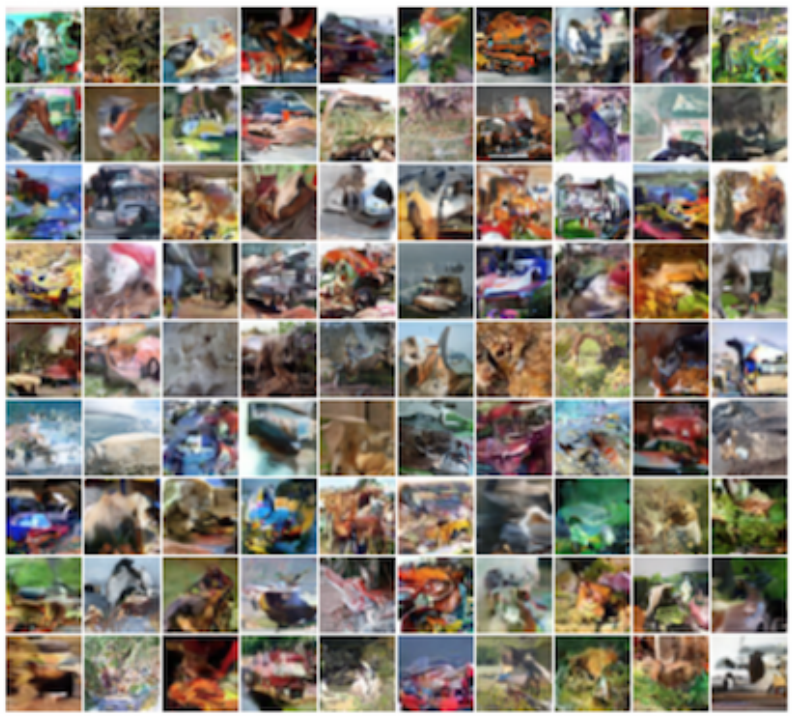
Perfomance on CIFAR-10

Model	**NLL Test (Train)**
Uniform Distribution:	8.00
Multivariate Gaussian:	4.70
NICE [1]:	4.48
Deep Diffusion [2]:	4.20
Deep GMMs [3]:	4.00
RIDE [4]:	3.47
PixelCNN:	3.14 (3.08)
Row LSTM:	3.07 (3.00)
Diagonal BiLSTM:	**3.00** (2.93)

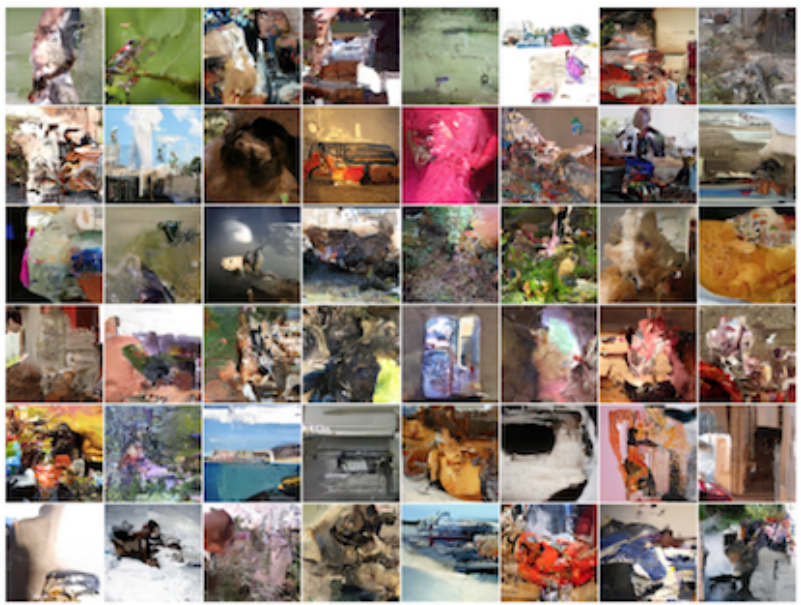
Good examples of occluded images



Psychedelic examples of occluded images on CIFAR(left) and ImageNet(right)



Psychedelic examples of occluded images by normal model(left) and multiscale model(right)



GRAN examples

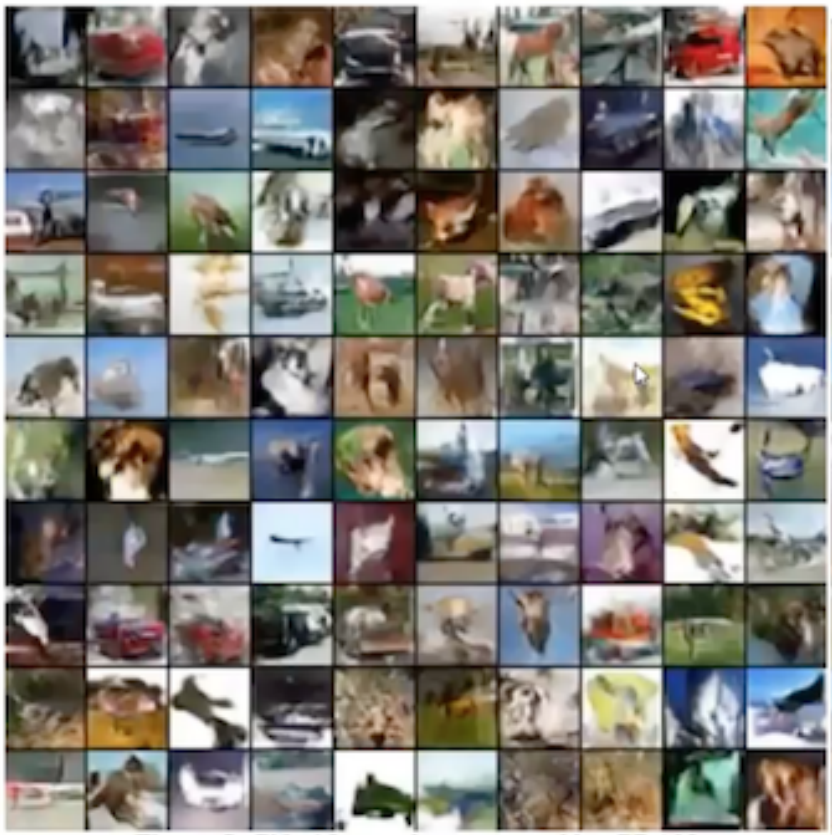


Figure 5. Cifar10 samples generated by GRAN

Figure 6. LSUN samples generated by GRAN

Summary

- Deep neural network that sequentally predicts the pixels in an image along the two spatial dimensions.
- Suggests three novel architectures to achieve this goal: PixelRNN (RowLSTM, Diagonal BiLSTM), PixelCNN (All Convolutional Net).
- Combined with efficient convolution preprocessing, both PixelCNN and PixelRNN use the "product of conditionals" approach to great effect.
- The state of the art (log-likelihood) on the MNIST and CIFAR-10 datasets and realistic generated samples.

Reference

Paper:
Pixel Recurrent Neural Networks
Aron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu
Google DeepMind 19 Aug 2016