PR12와 함께 이해하는

Pixel Recurrent Neural Network

Jaejun Yoo

Ph.D. Candidate @KAIST

PR12

16TH July, 2017

PR12와 함께 이해하는

Pixel Recurrent Neural Network

Jaejun Yoo
Ph.D. Candidate @KAIST
PR12
16TH July, 2017

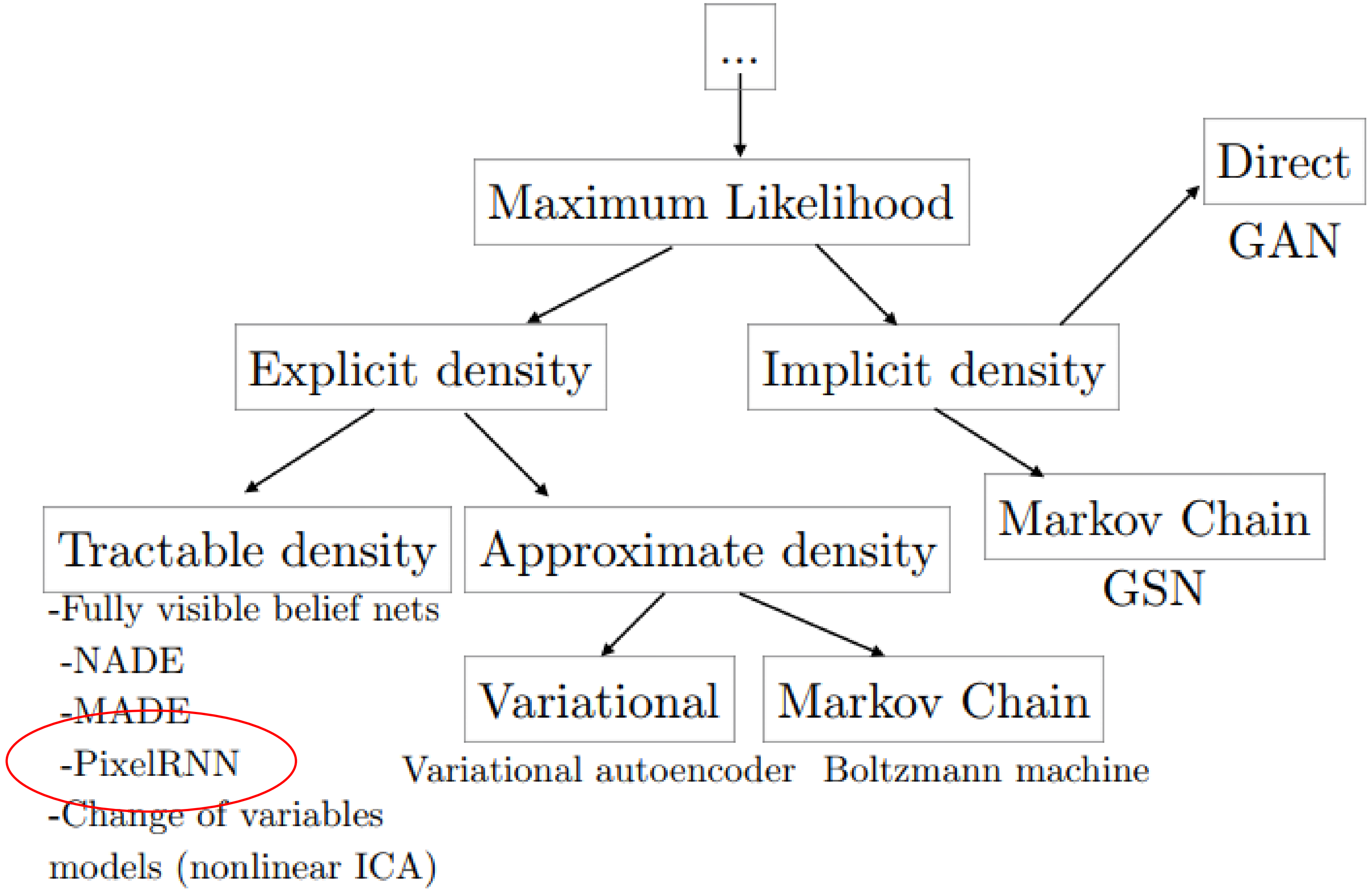
PR12와 함께 이해하는

Pixel Recurrent Neural Network

Jaejun Yoo
Ph.D. Candidate @KAIST
PR12

16TH July, 2017

GENERATIVE MODEL!



NIPS 2016 Tutorial: GANs

Intuition ...(A customary CAT slide!)

How to include statistical dependencies over hundreds of pixels?

Image Source: http://wallpapers.ae/wp-content/uploads/2016/01/Tom-And-Jerry-Funny-Wallpaper.jpg

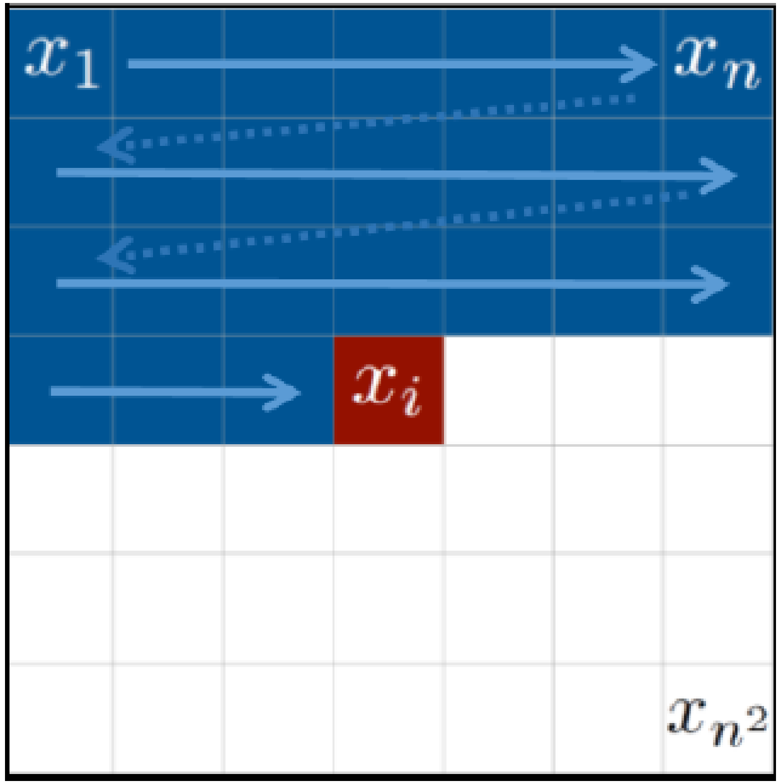
Intuition

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

Bayes Theorem:

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

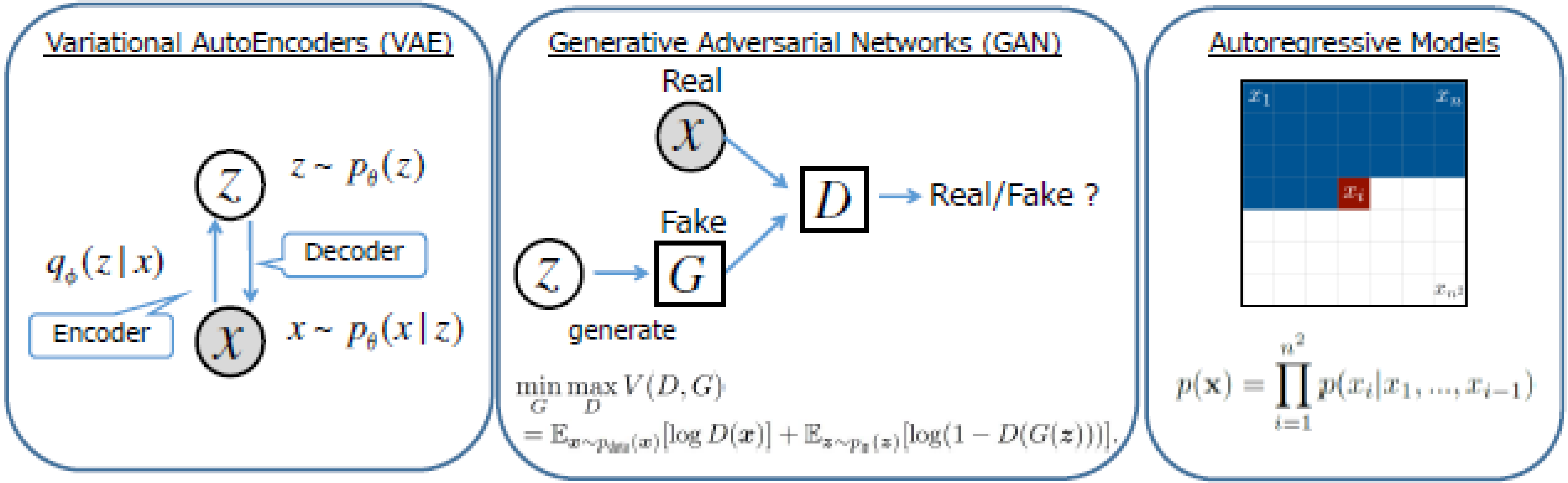
A sequential model!



Pixel recurrent neural networks, ICML 2016

Image Generation Models

-Three image generation approaches are dominating the field:



	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

(cf. https://openai.com/blog/generative-models/)

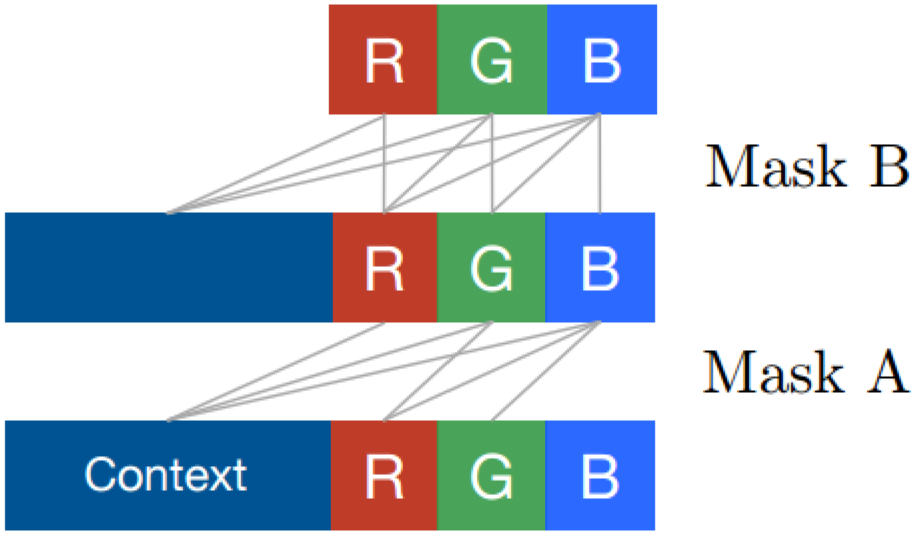
MASK

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

MASK

Channel Masks

- Sequential order: R -> G -> B
- Used in input-to-state convolutions
- Two types of masks:



- Channels are connected to themselves
- Used in all other subsequent layers

- Channels are **not** connected to themselves
- Only used in the first layer

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

Receptive Fields

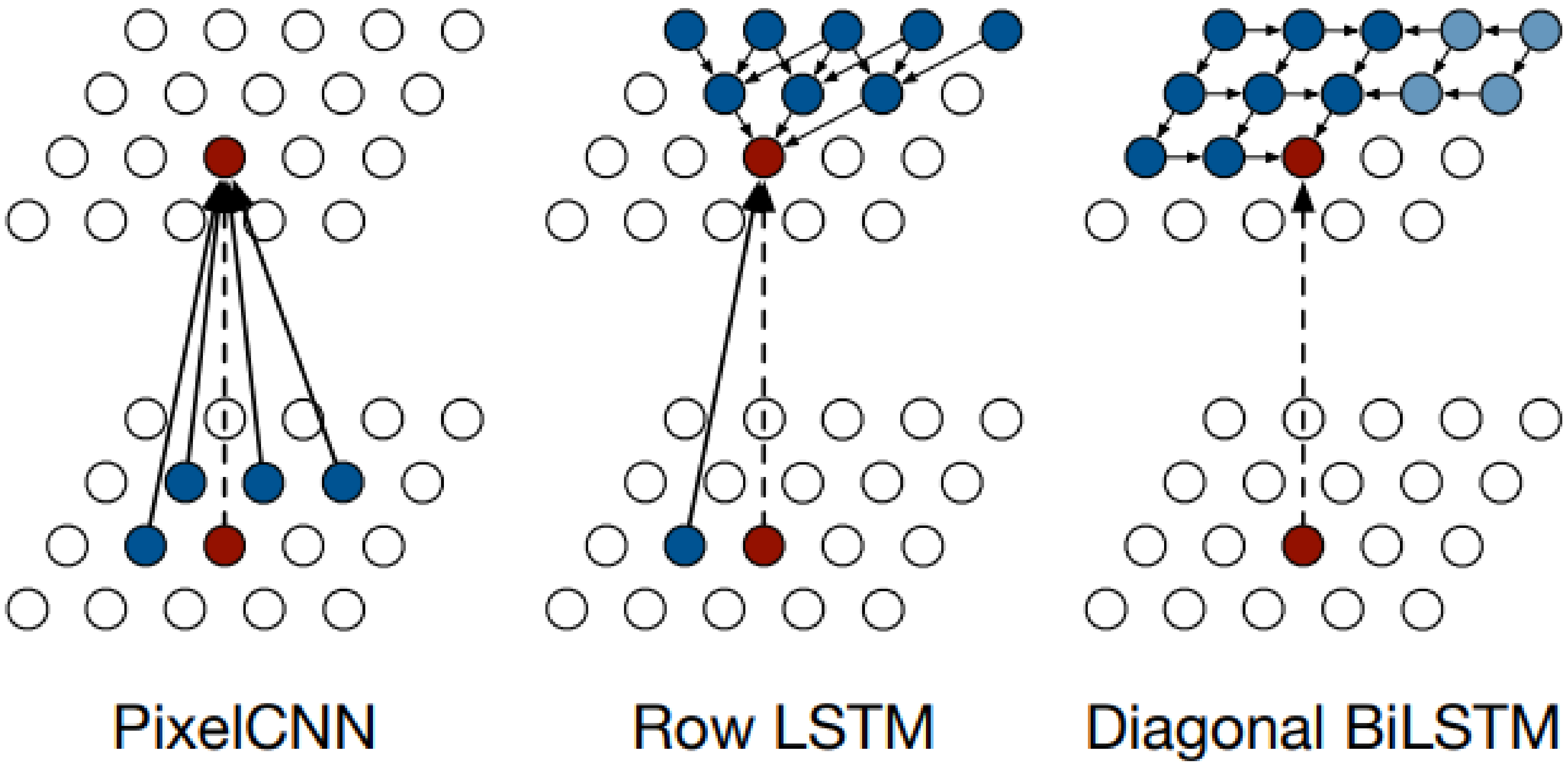
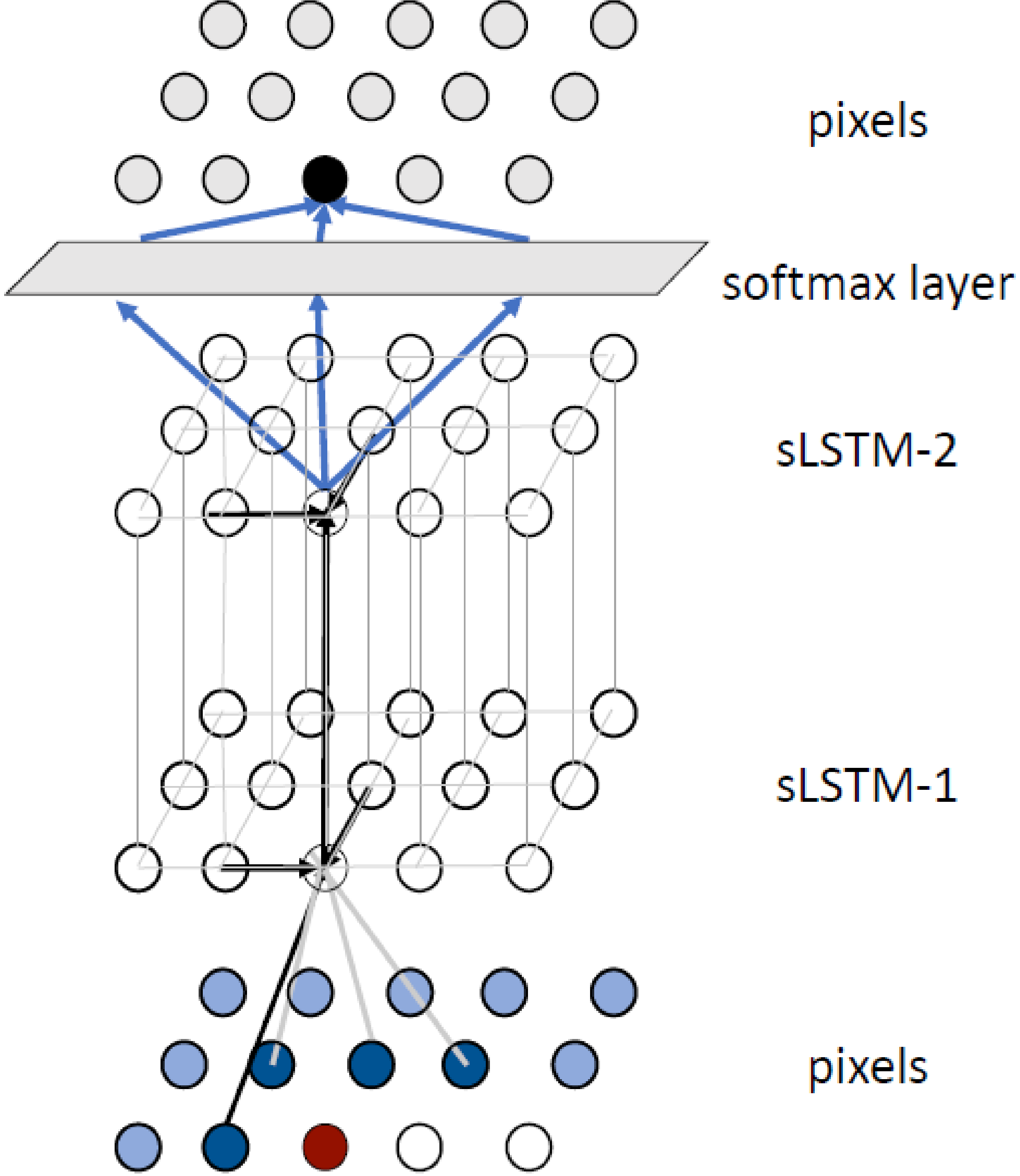


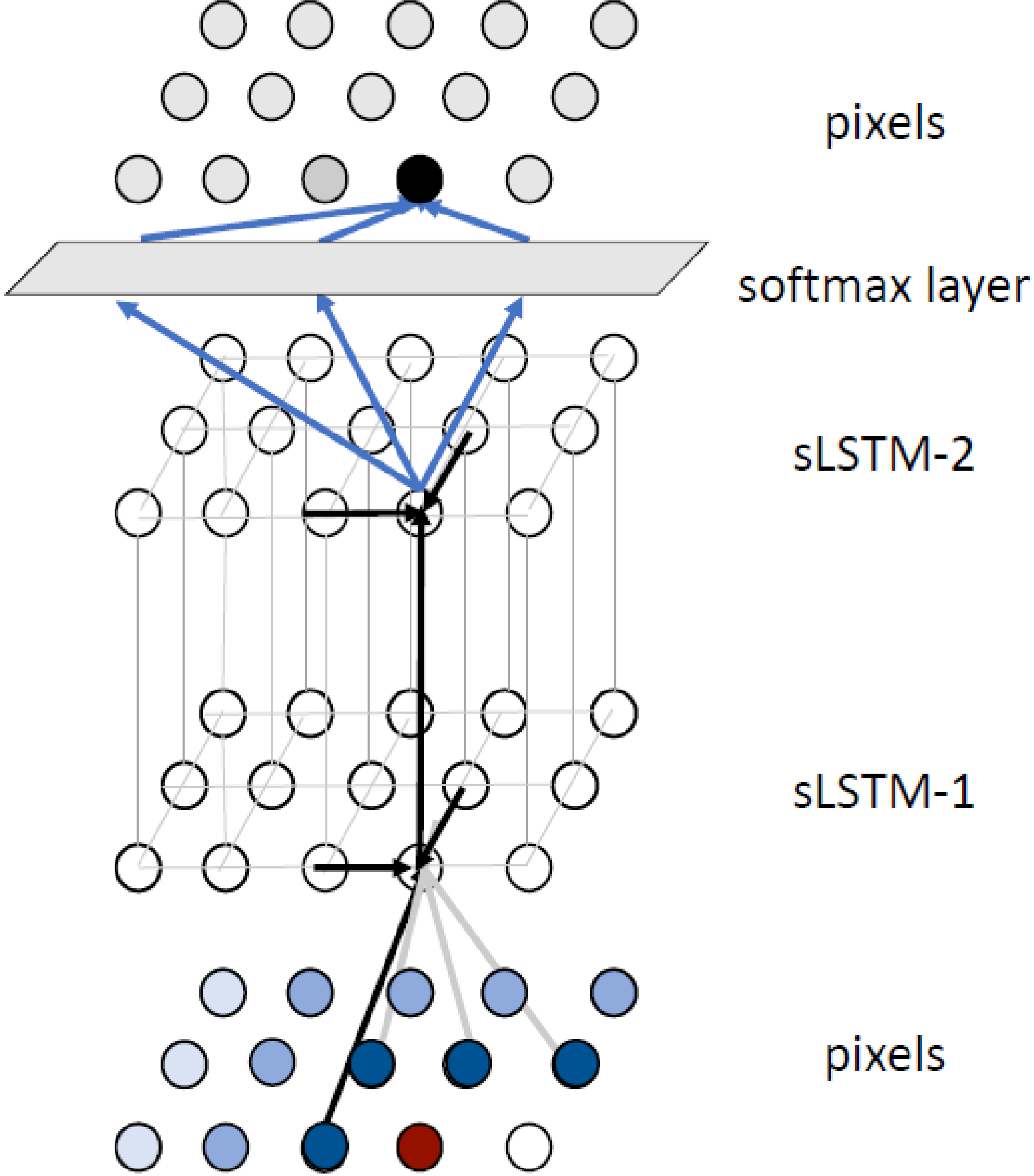
Figure 4. Visualization of the input-to-state and state-to-state mappings for the three proposed architectures.

Spatial LSTM



Adapted from: Generative image modeling using spatial LSTM. Theis & Bethge, 2015

Spatial LSTM



Adapted from: Generative image modeling using spatial LSTM. Theis & Bethge, 2015

Architecture

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

Architecture

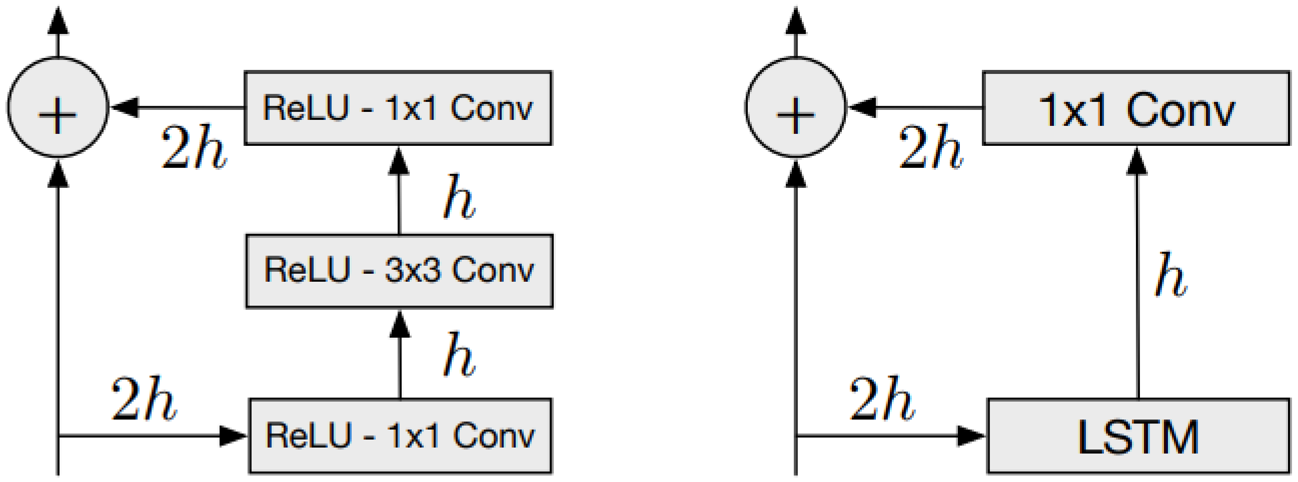
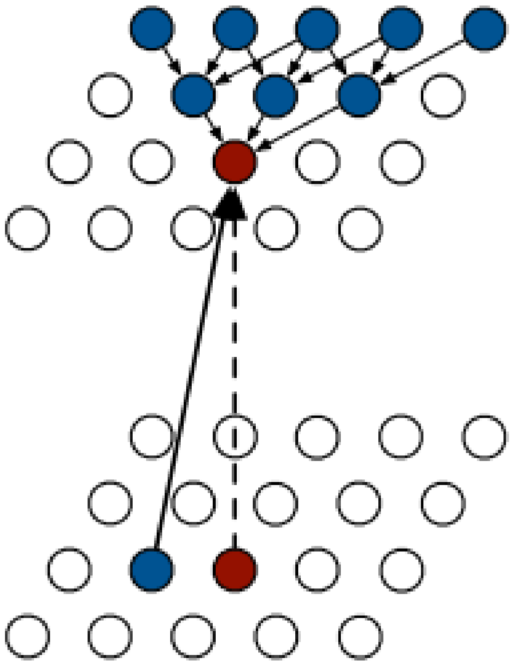


Figure 5. Residual blocks for a PixelCNN (left) and PixelRNNs.

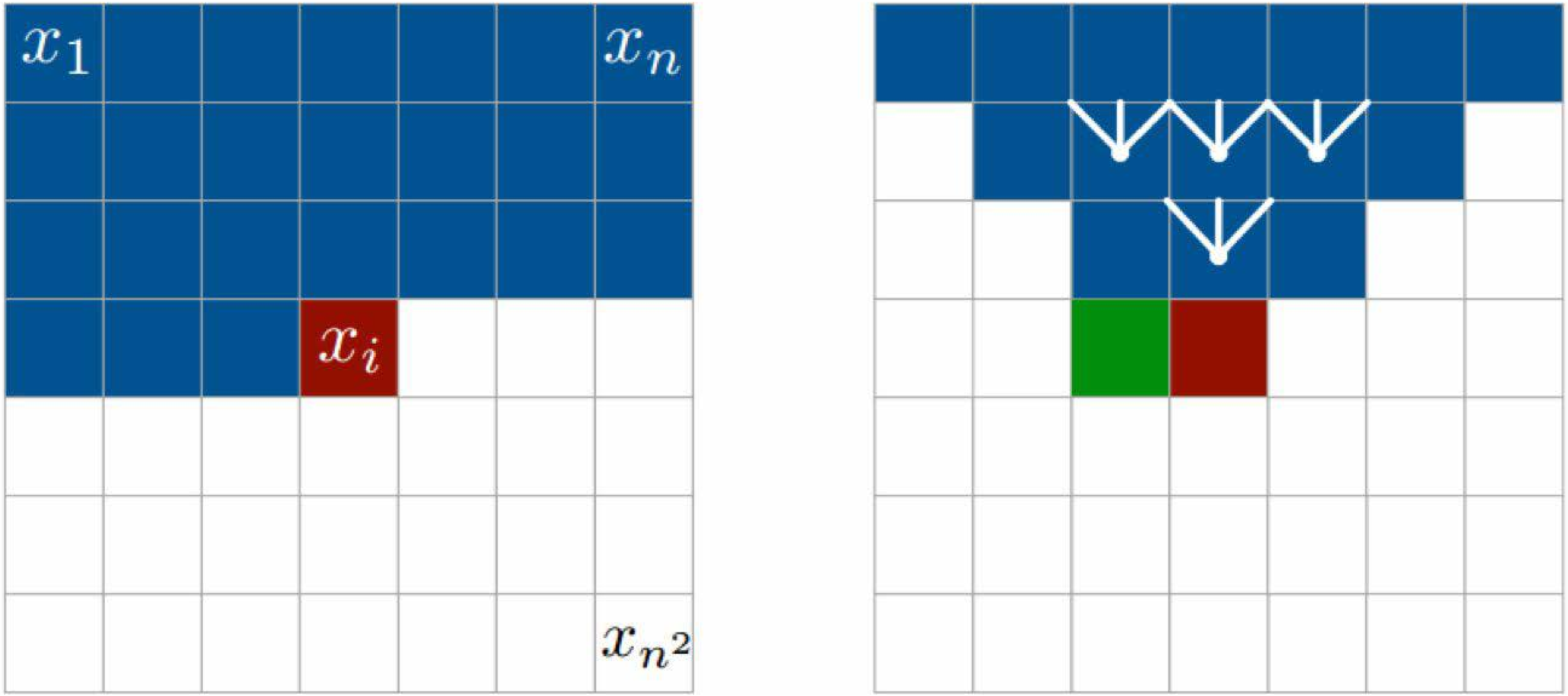
Row LSTM



Row LSTM

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

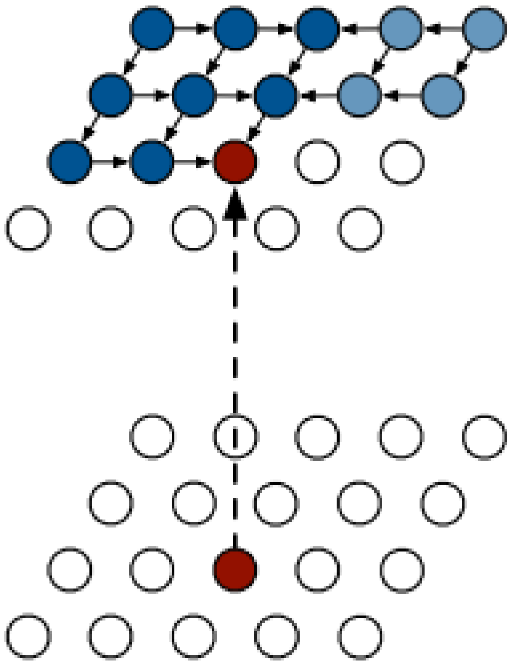
Row LSTM : Receptive Field



Architecture

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 LSTM

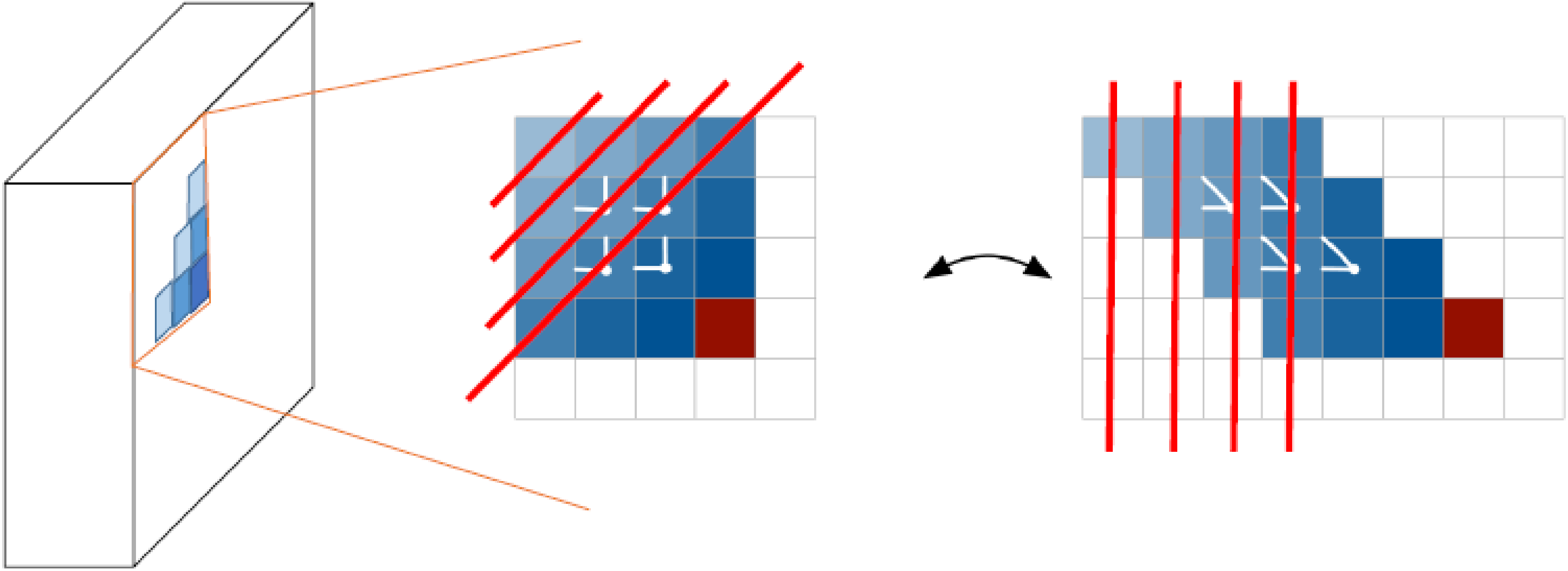


Diagonal BiLSTM

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

Diagonal LSTM

- To optimize, we skew the feature maps so it can be parallelized

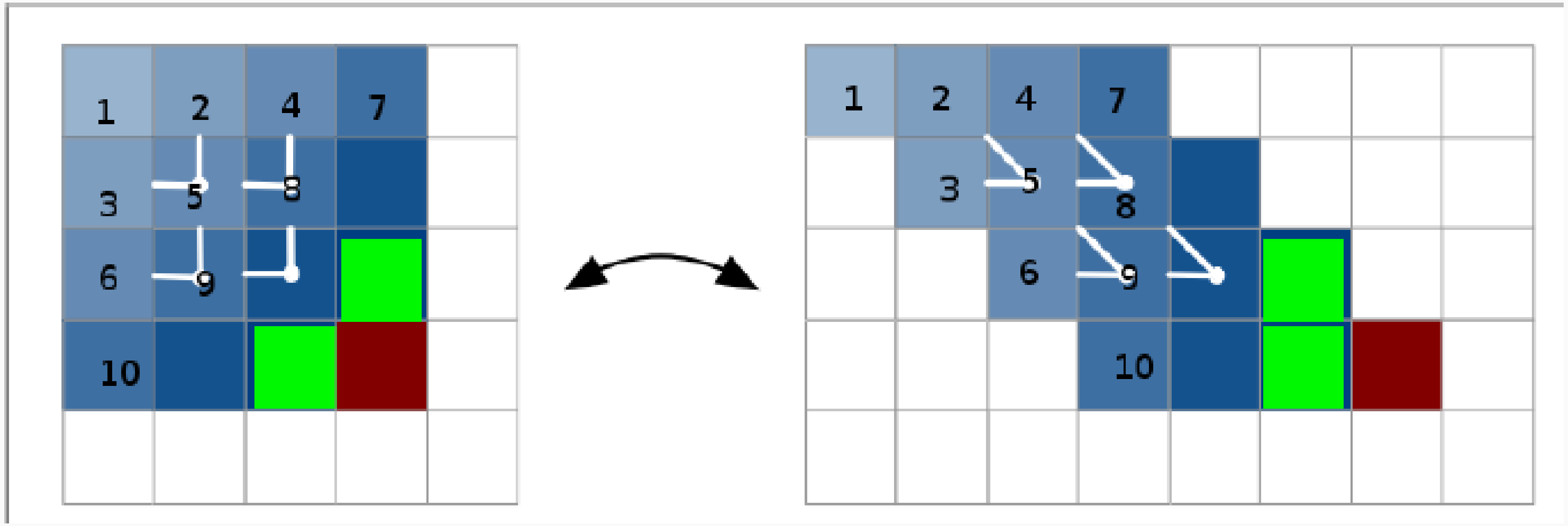


[Pixel recurrent neural networks](), ICML 2016

Diagonal LSTM

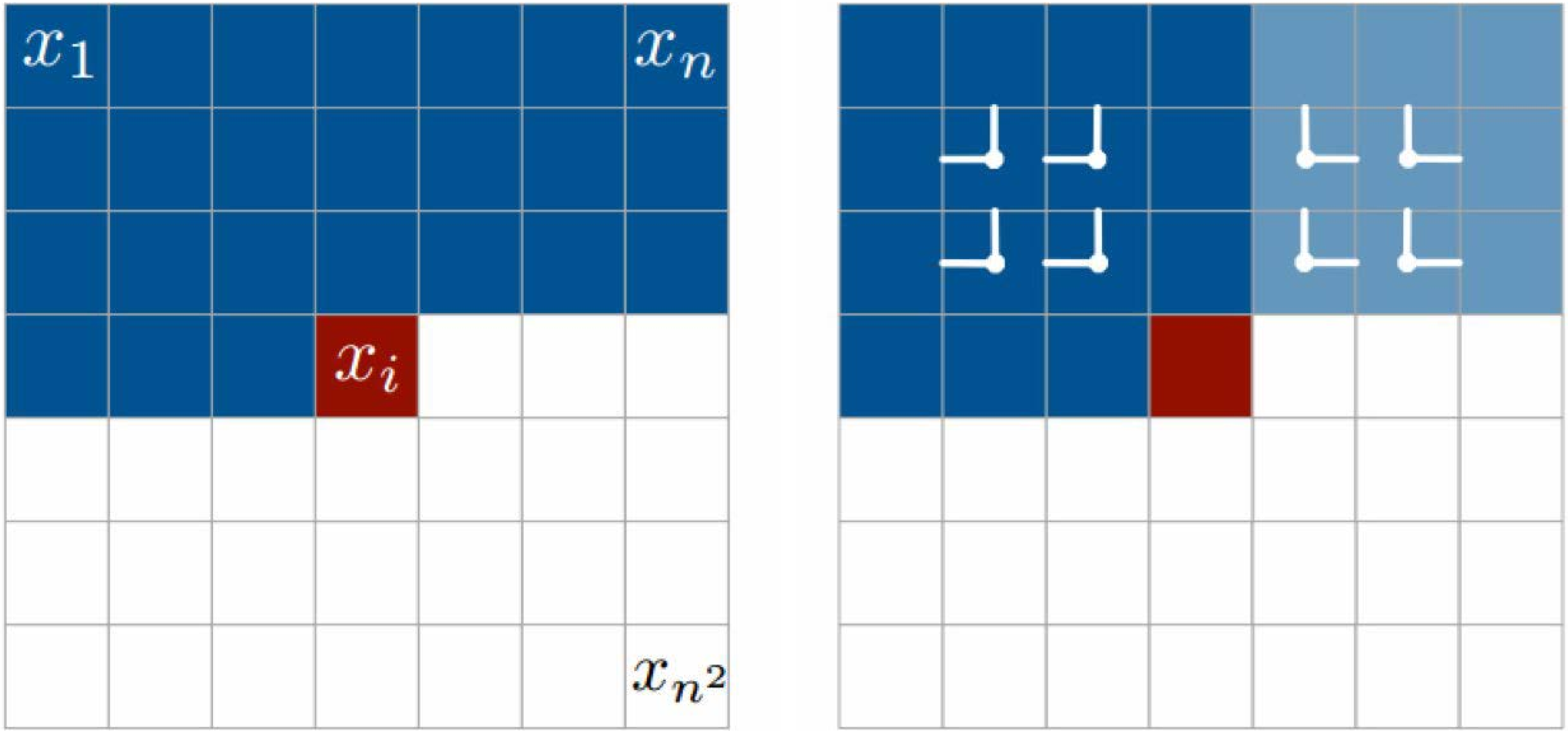
- To c

- Parallelized by skew operation
- $n \times n \longleftrightarrow n \times (2n-1)$
- Convolutional kernel is 2 x 1



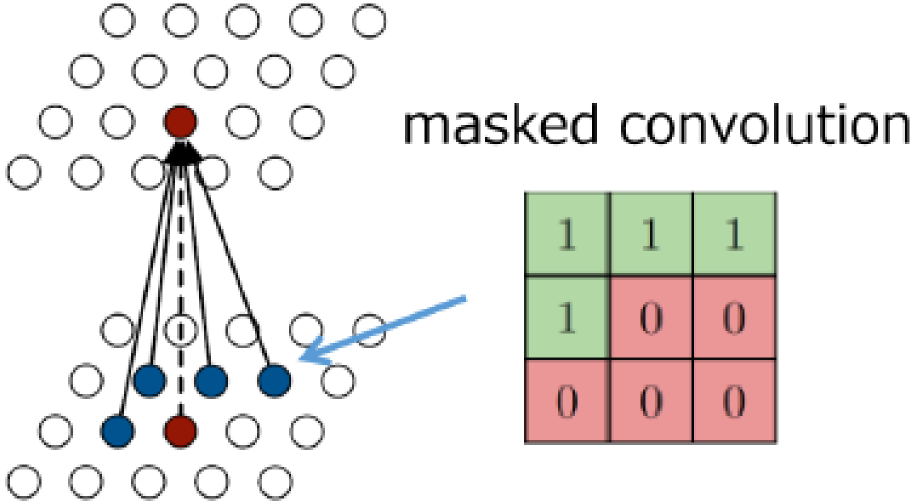
[Pixel recurrent neural networks](), ICML 2016

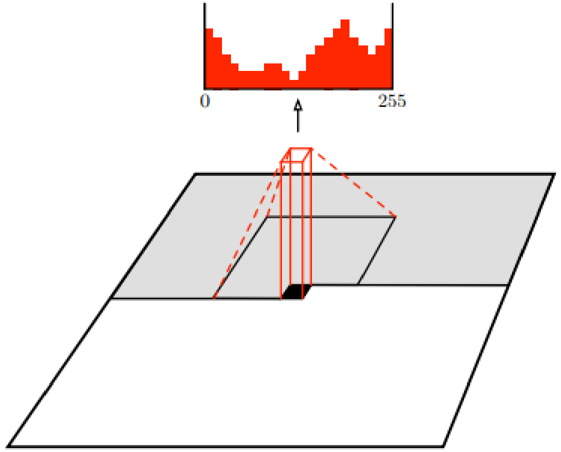
Diagonal BiLSTM : Receptive Field



PixelCNN

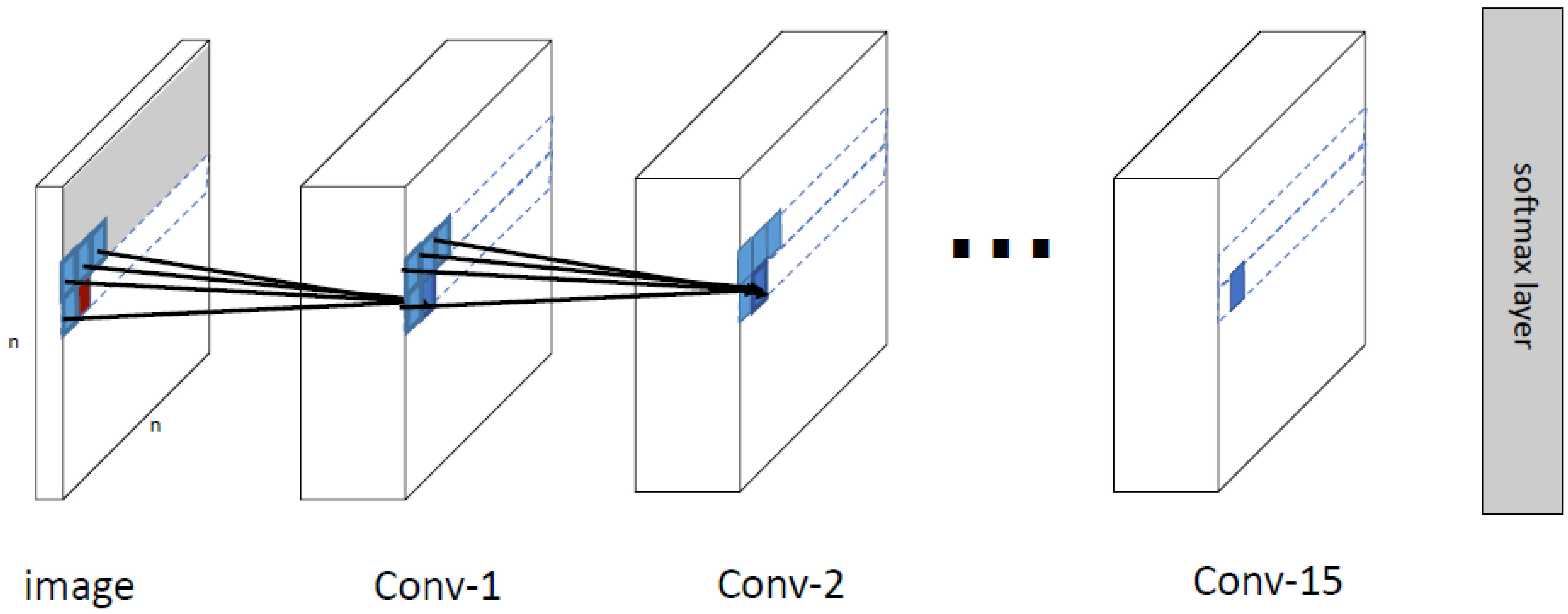
- 2D convolution on previous layer
- Apply masks so a pixel does not see future pixels (in sequential order)





Pixel recurrent neural networks, ICML 2016

PixelCNN



Comparison

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

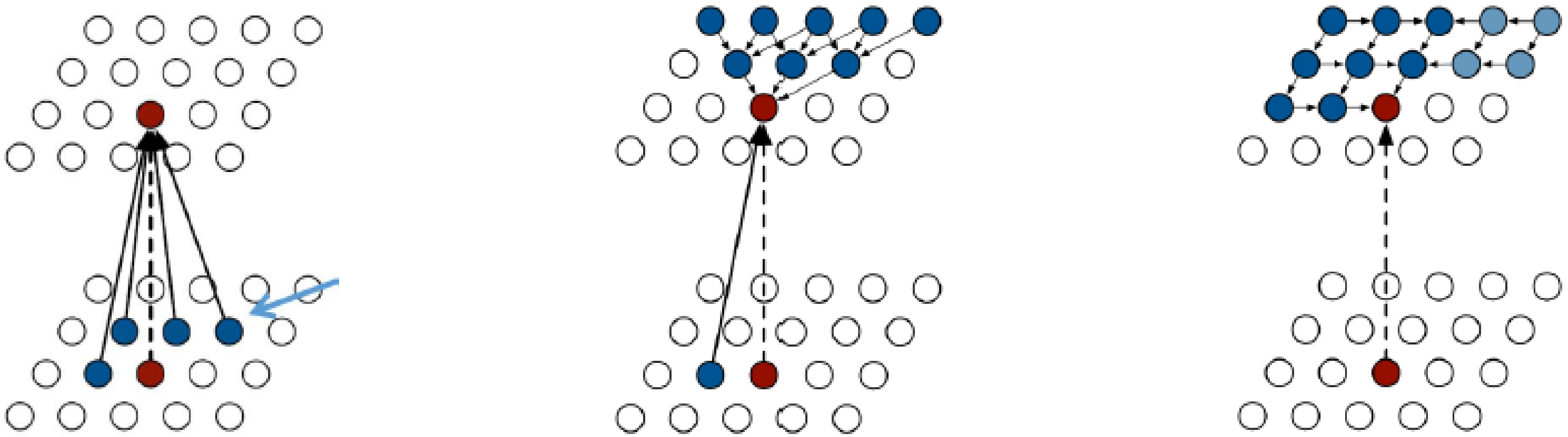


Figure from: [Oord et al.]

EXPERIMENT AND RESULTS

EXPERIMENT AND RESULTS

- Dataset: MNIST, CIFAR-10, and ImageNet
- Method: log-likelihood

Details about Soft Max

- Treat pixels as discrete variables:
  - To estimate a pixel value, do classification in every channel (256 classes indicating pixel values 0-255)
  - Implemented with a final softmax layer

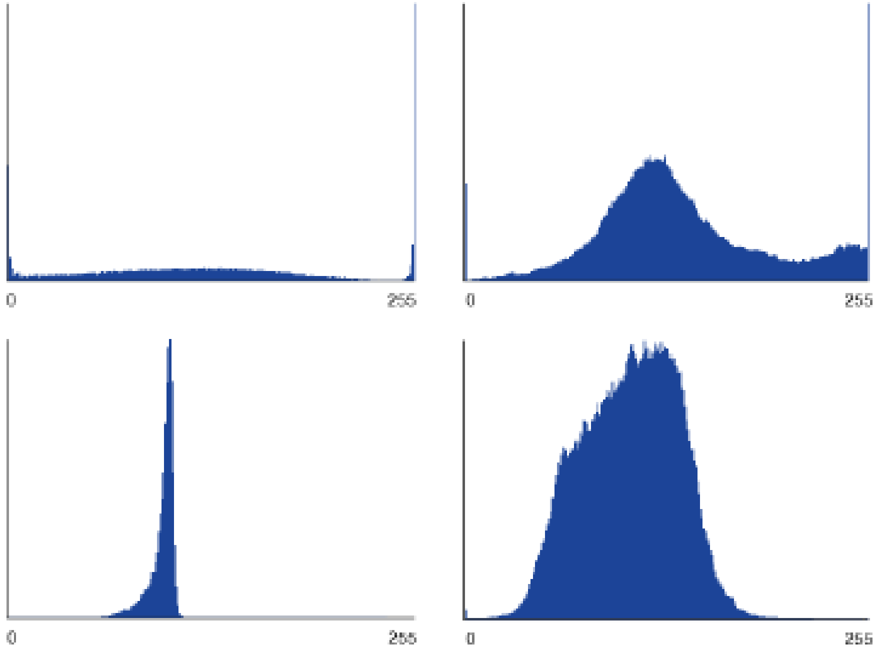


Figure: Example softmax outputs in the final layer, representing probability distribution over 256 classes.

Figure from: [Oord et al.](#)

Prerequisite: KL Divergence

WIKI: Information theory

Kullback–Leibler divergence (information gain) [edit]

The *Kullback–Leibler divergence* (or *information divergence*, *information gain*, or *relative entropy*) is a way of comparing two distributions: a "true" probability distribution *p(X)*, and an arbitrary probability distribution *q(X)*. If we compress data in a manner that assumes *q(X)* is the distribution underlying some data, when, in reality, *p(X)* is the correct distribution, the Kullback–Leibler divergence is the number of average additional bits per datum necessary for compression. It is thus defined

$$D_{\mathrm{KL}}(p(X)\|q(X)) = \sum_{x\in X} -p(x)\log q(x) \; - \; \sum_{x\in X} -p(x)\log p(x) = \sum_{x\in X} p(x)\log\frac{p(x)}{q(x)}.$$

Prerequisite: MLE & KL Divergence

Let $P(x_i|\theta)$ be the distribution for generating each data point x_i. We can define the model parameter distribution and the "empirical data distribution" as:

$$P_D(x) = \sum_{i=1}^{N} \frac{1}{N}\delta(x - x_i), \quad P_\theta(x) = P(x|\theta)$$

where δ is the Dirac delta function. We can verify this is a valid distribution by summing over all $x : \sum_{j=1}^{N} P_D(x_j) = 1$. By using this empirical data distribution, we wish to calculate the follwoing KL distribution:

$$\begin{aligned} KL[P_D(x)||P_\theta(x)] &= \int P_D(x)\log\frac{P_D(x)}{P_\theta(x)}dx \\ &= \int P_D(x)\log P_D(x)dx - \int P_D(x)\log P_\theta(x)dx \\ &= -H[P_D(x)] - \int P_D(x)\log P_\theta(x)dx \\ &\propto -\mathbb{E}_{P_D(x)}[\log P_\theta(x)] \end{aligned}$$

$$\begin{aligned} \mathbb{E}_{P_D(x)}[\log P_\theta(x)] &= \sum_x P_D(x)\log P(x|\theta) \\ &= \sum_x \left[\frac{1}{N}\sum_{i=1}^{N}\delta(x - x_i)\right]\log P(x|\theta) \\ &= \frac{1}{N}\sum_{i=1}^{N}\log P(x_i|\theta) \end{aligned}$$

EXPERIMENT AND RESULTS

	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).

EXPERIMENT AND RESULTS

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).

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)

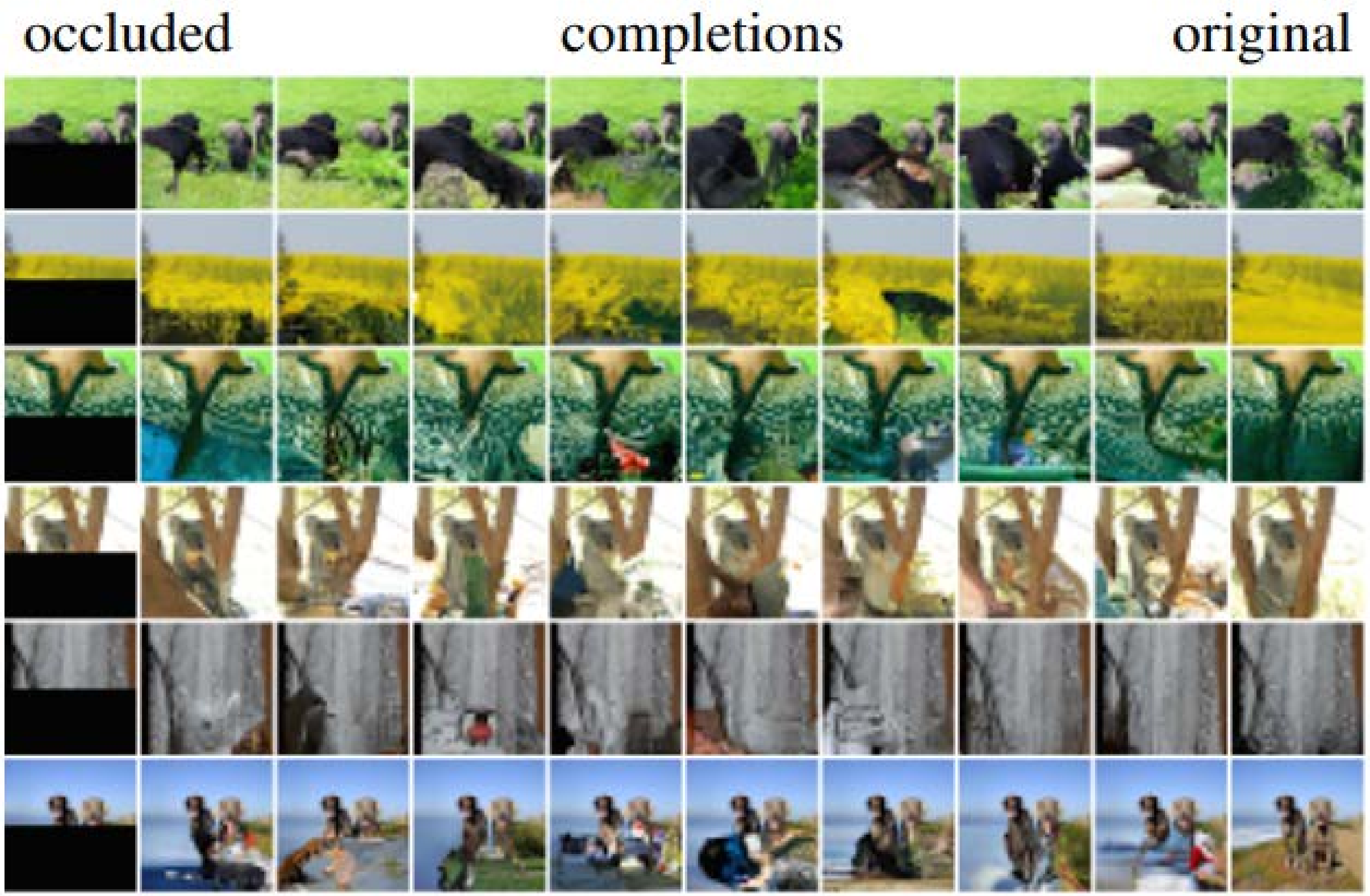
Table 5. Test set performance of different models on CIFAR-10 in *bits/dim*. For our models we give training performance in brackets. [1] (Dinh et al., 2014), [2] (Sohl-Dickstein et al., 2015), [3] (van den Oord & Schrauwen, 2014a), [4] personal communication (Theis & Bethge, 2015).

"Lower is better"

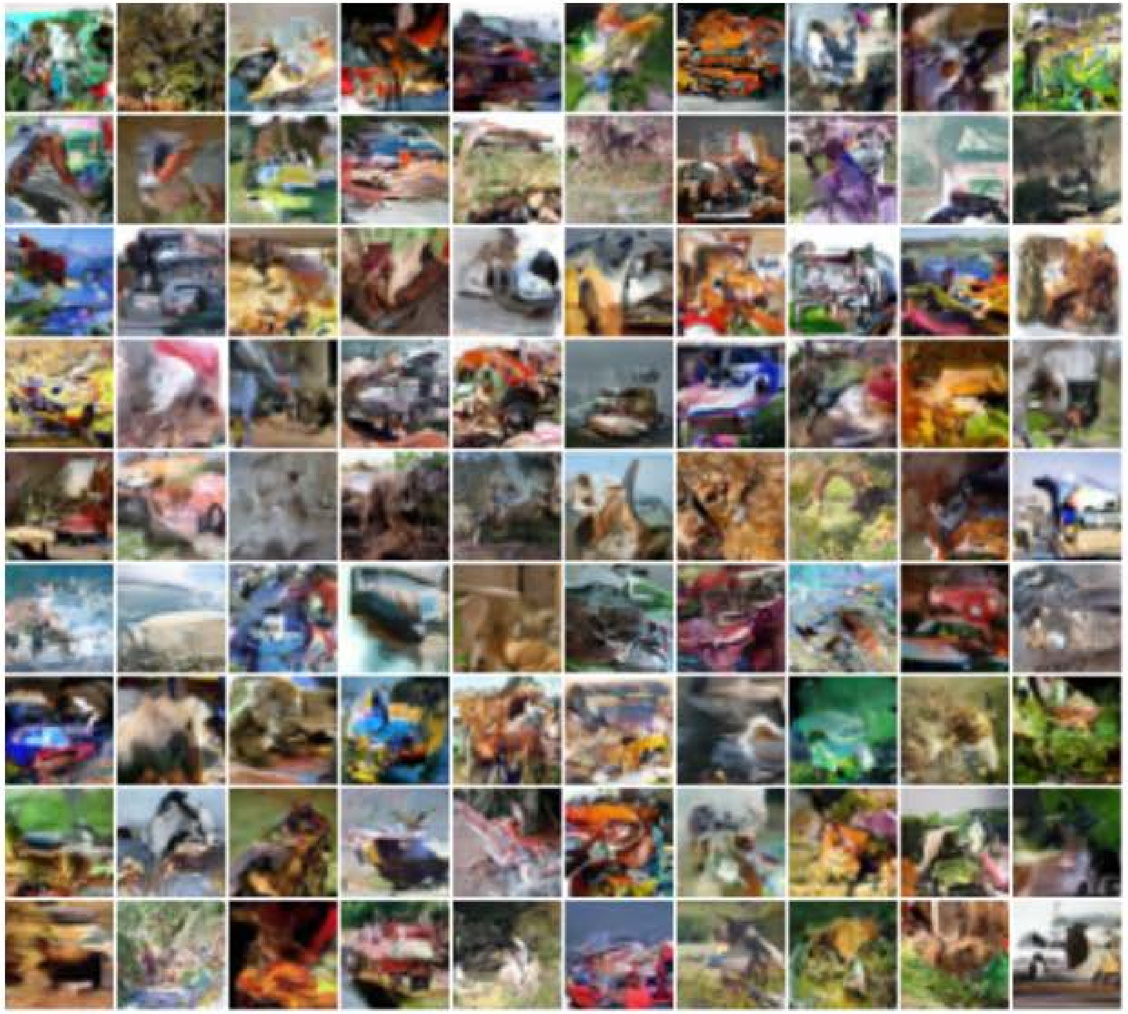
EXPERIMENT AND RESULTS

Figure 1. Image completions sampled from a PixelRNN.

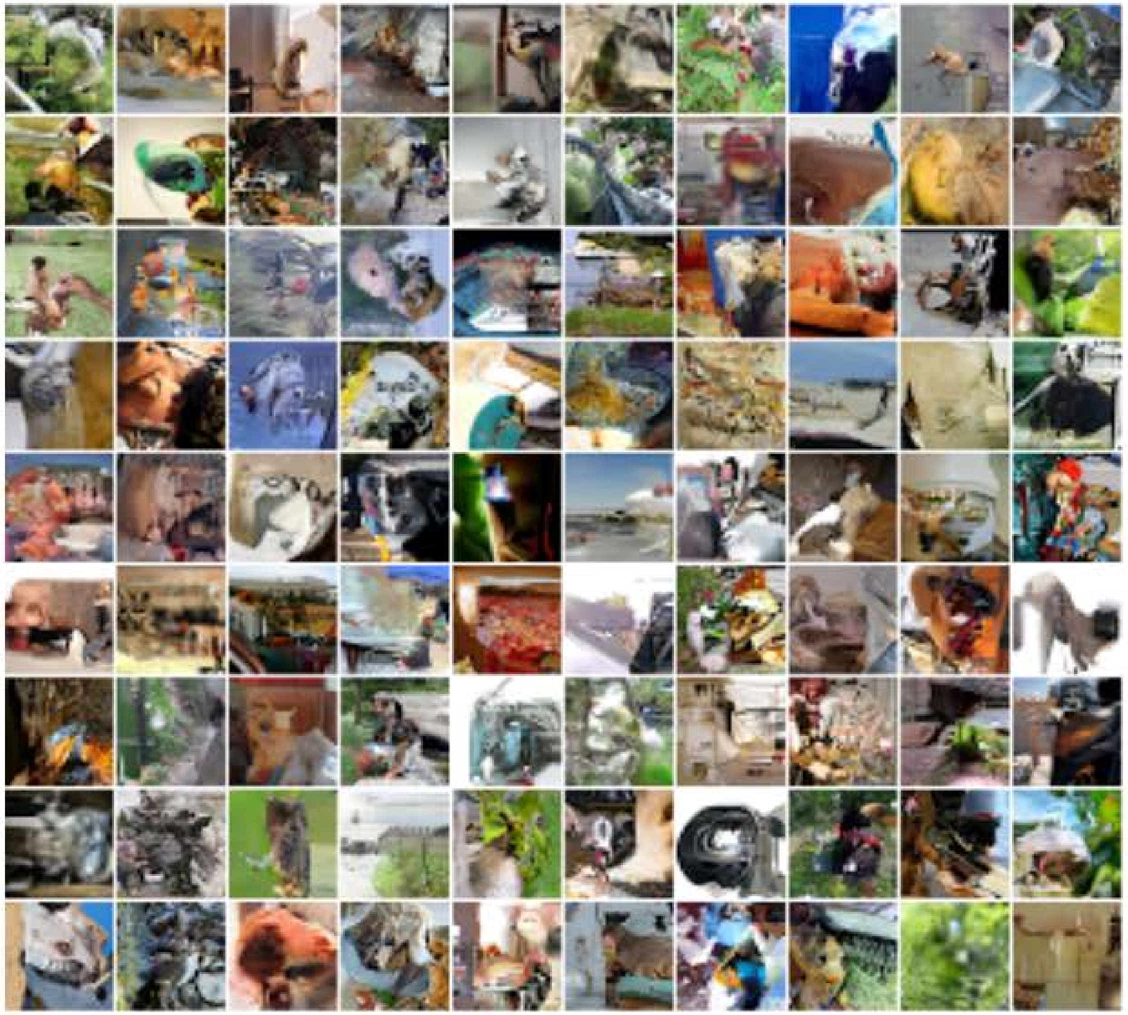
EXPERIMENT AND RESULTS



EXPERIMENT AND RESULTS



CIFAR-10 (32x32)



ImageNet (32x32)

SUMMARY

- Deep neural network that sequentially predicts the pixels in an image along the two spatial dimensions
- Suggests three novel architectures to achieve this goal
  - PixelRNN (Row LSTM, Diagonal BiLSTM), PixelCNN (All Convolutional Net)
- Combined with efficient convolution preprocessing, both PixelCNN and PixelRNN use this “product of conditionals” approach to great effect.
- The model provides a tractable P(x) with the best log-likelihood scores in its family.
- Image generation of good quality and diversity

Reference

- Slides: Hugo Larochelle, Google Brain: Autoregressive Generative Models with Deep Learning
- Slides: http://slazebni.cs.illinois.edu/spring17/lec13_advanced.pdf
- Slides: https://www.slideshare.net/neouyghur/pixel-recurrent-neural-networks-73970786
- Code: https://github.com/igul222/pixel_rnn/blob/master/pixel_rnn.py
- Related Paper "Generating images with recurrent adversarial networks"
- Related Paper "WaveNet"

GRAN

Figure 5. Cifar10 samples generated by GRAN

Figure 6. LSUN samples generated by GRAN