Pixel Recurrent Neural Networks

Logan Lebanoff
2/22/17

Outline

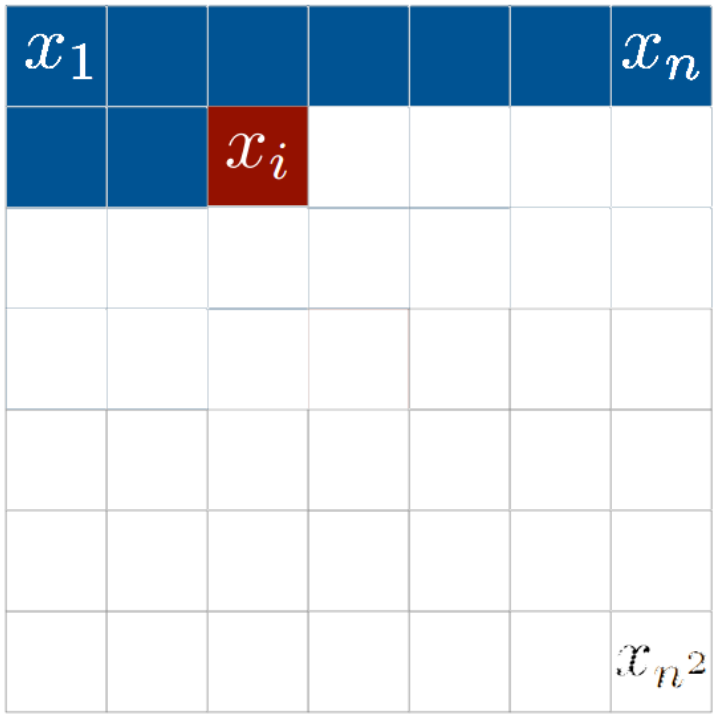
- Comparison to other Generative models
- Image Generation Process
- 4 Models
  - PixelCNN
  - Row LSTM
  - Diagonal BiLSTM
  - Multi-Scale PixelRNN
- Results

Image Generation Models

-Three image generation approaches are dominating the field:

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

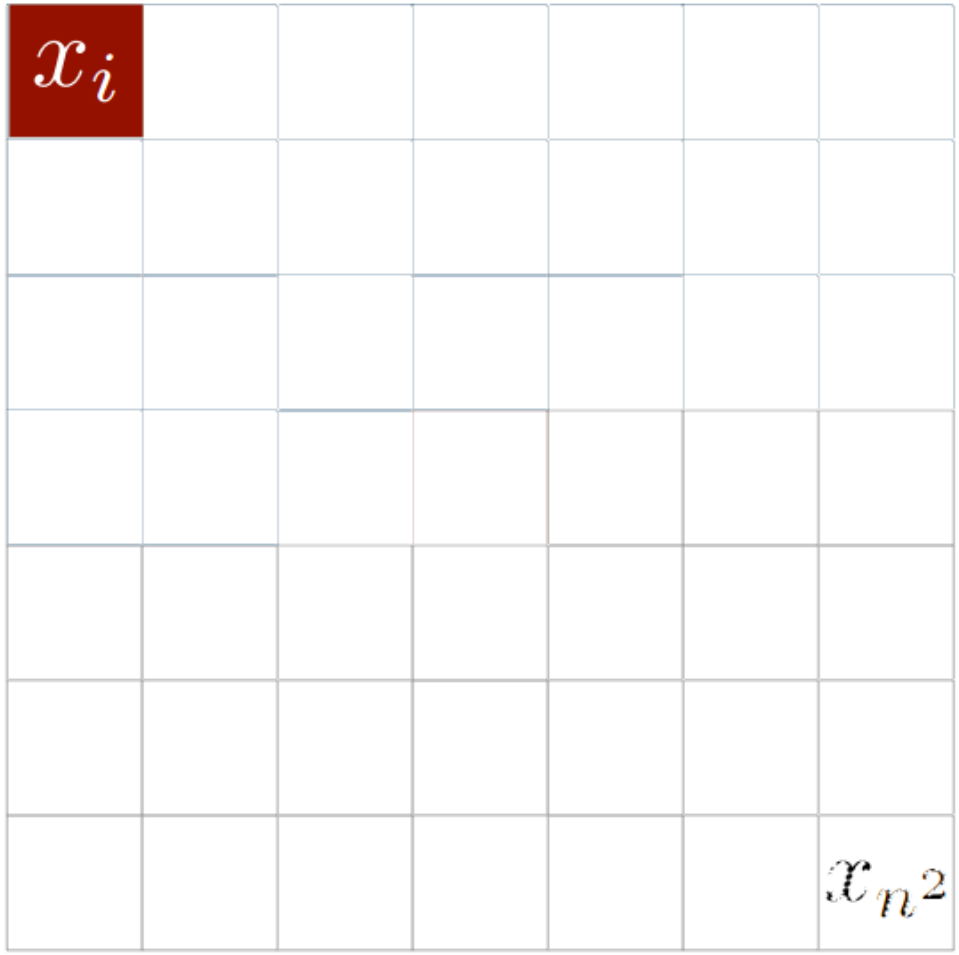
PixelRNN - Model

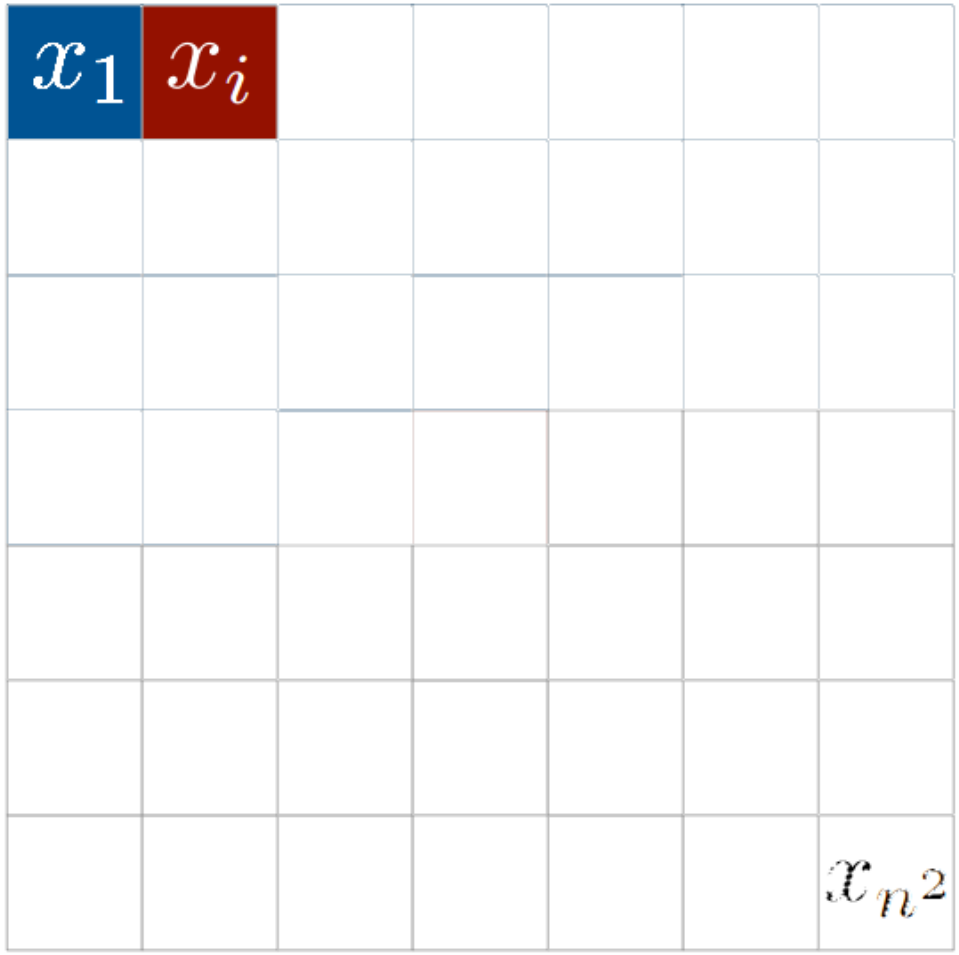


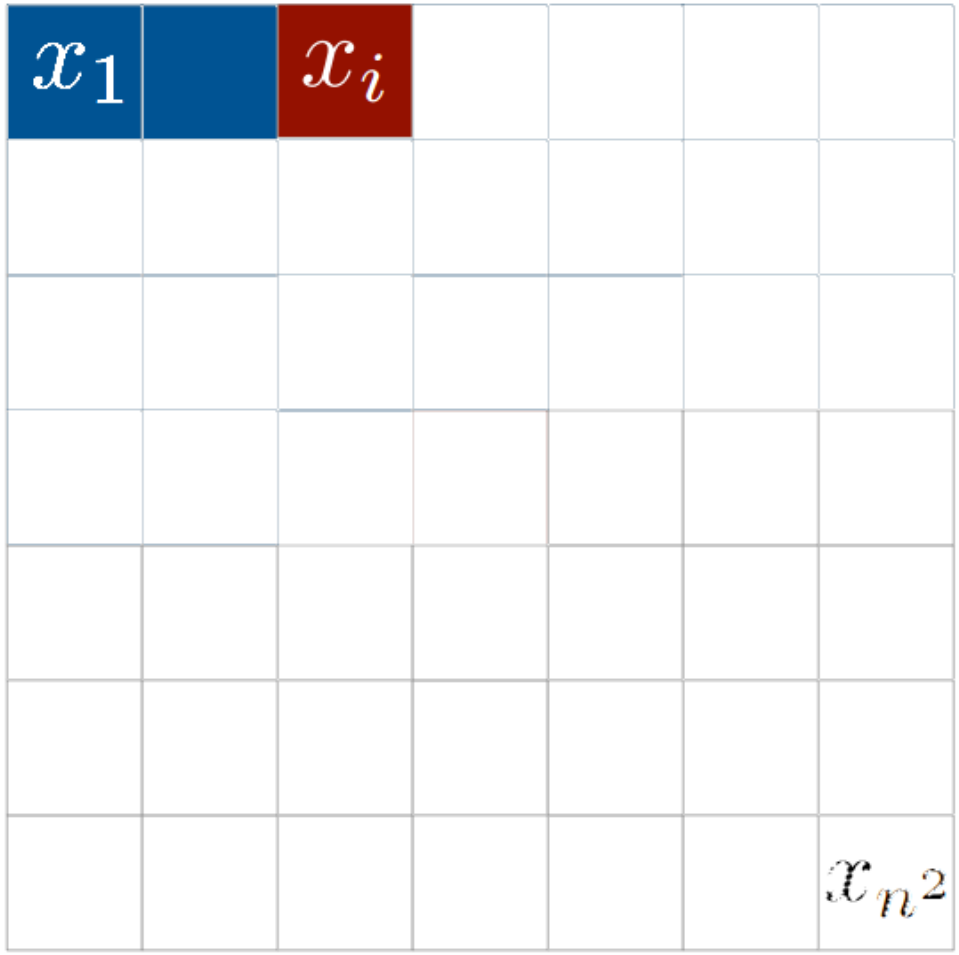
- Sequence of pixels
- Pixel is dependent on previous pixels

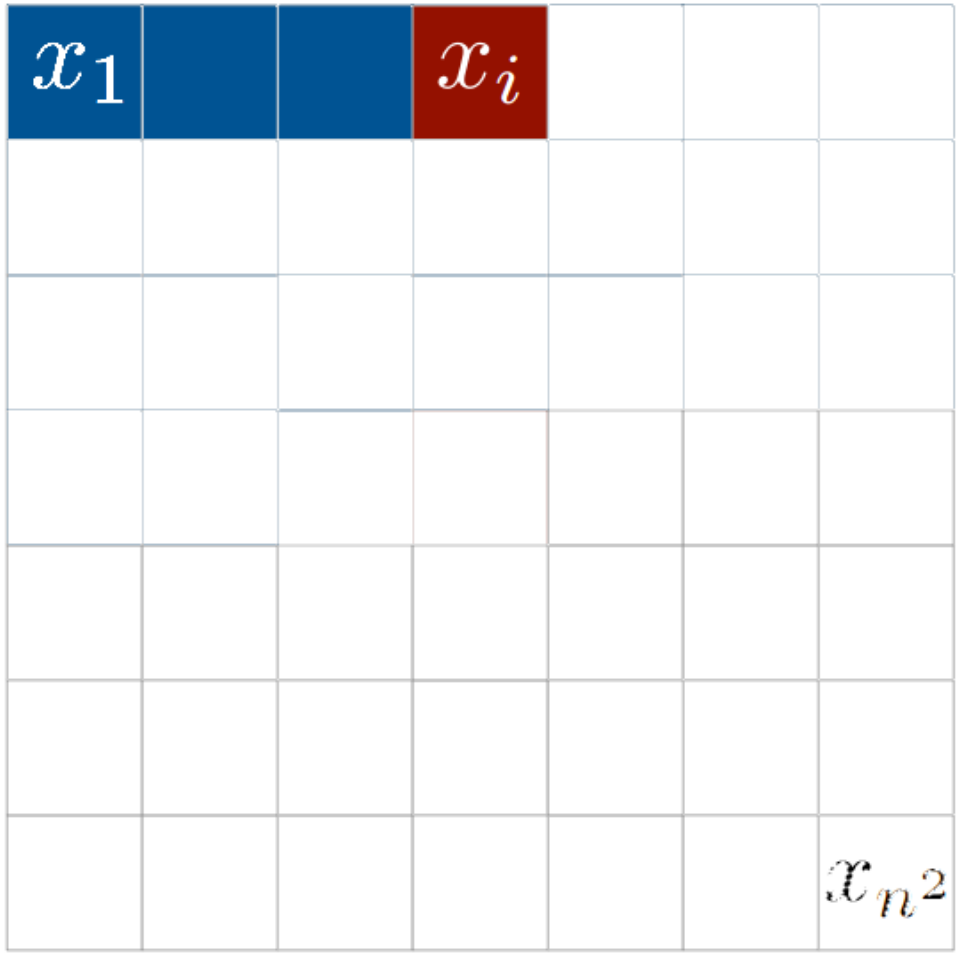
$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

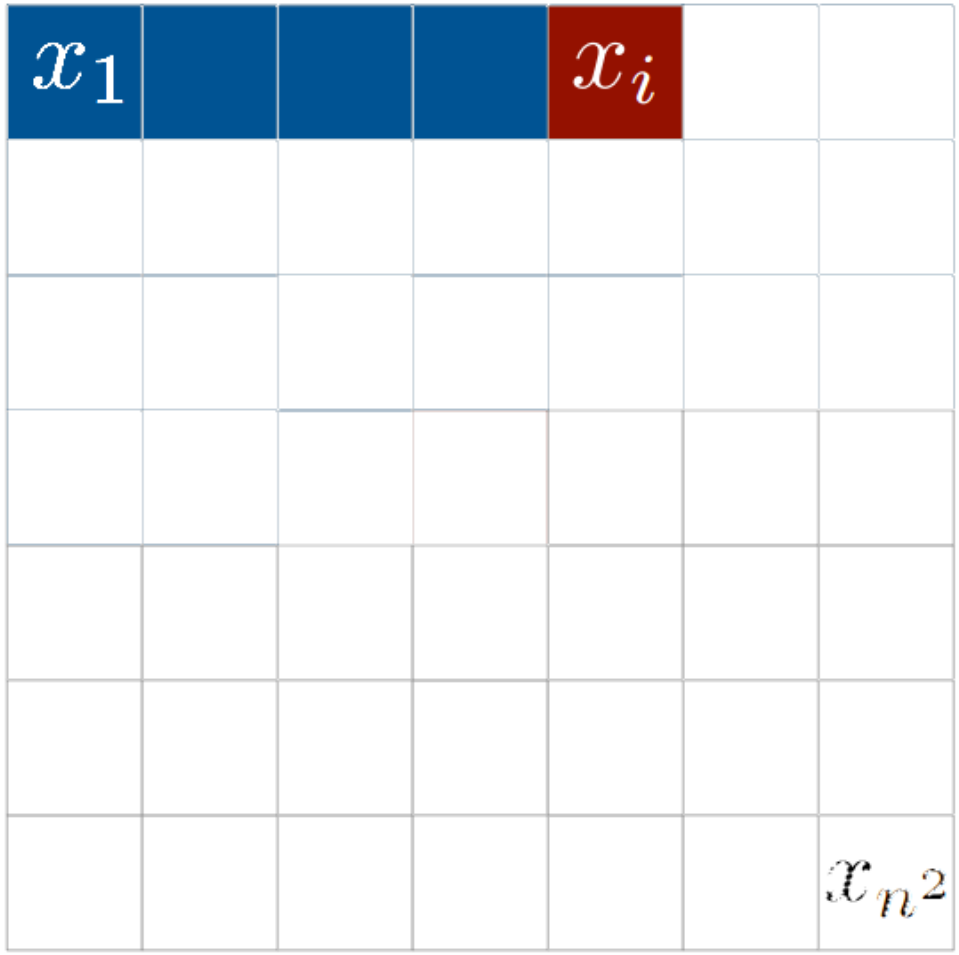
Image Generation

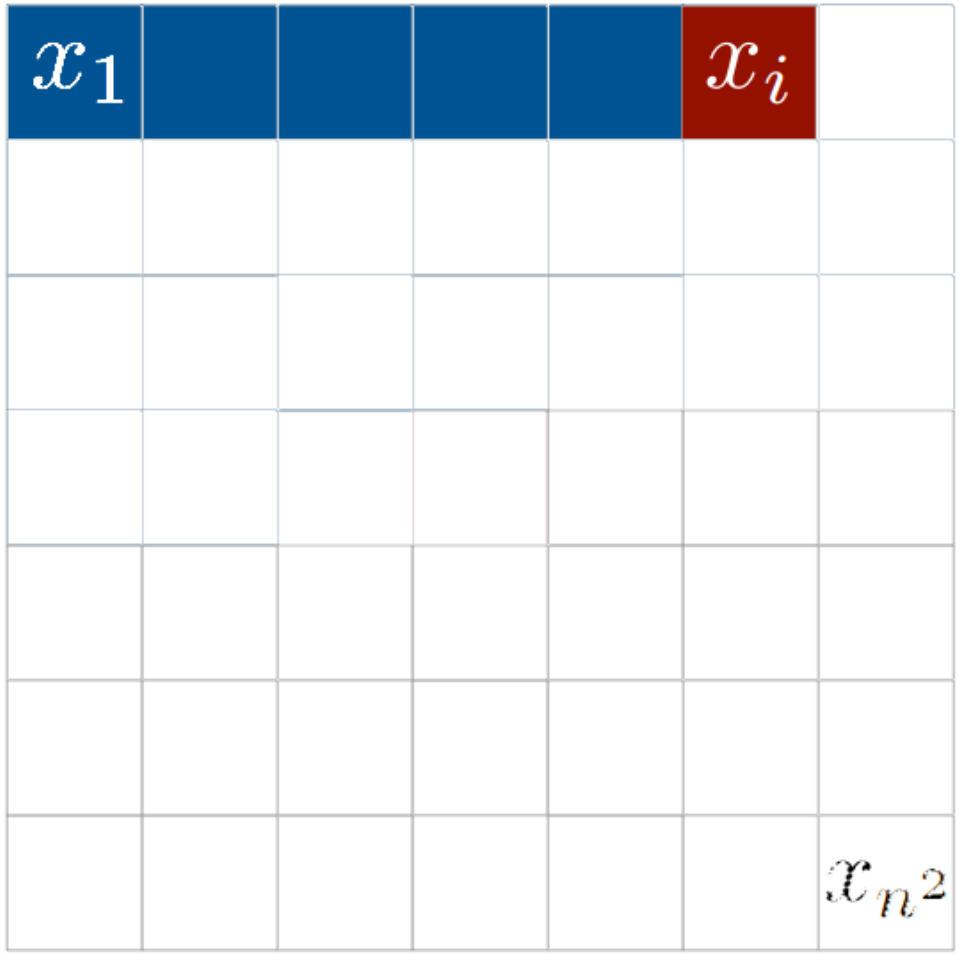

x_i
x_{n^2}

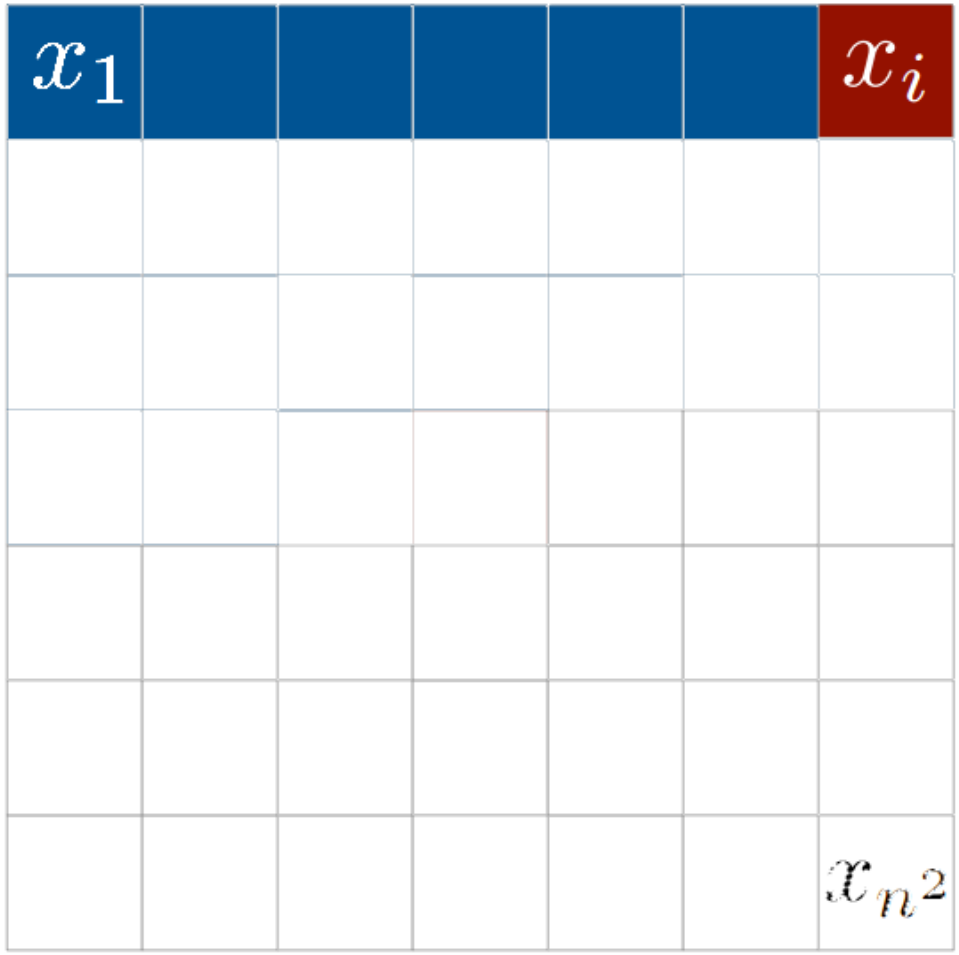

x_1
x_i
x_{n^2}

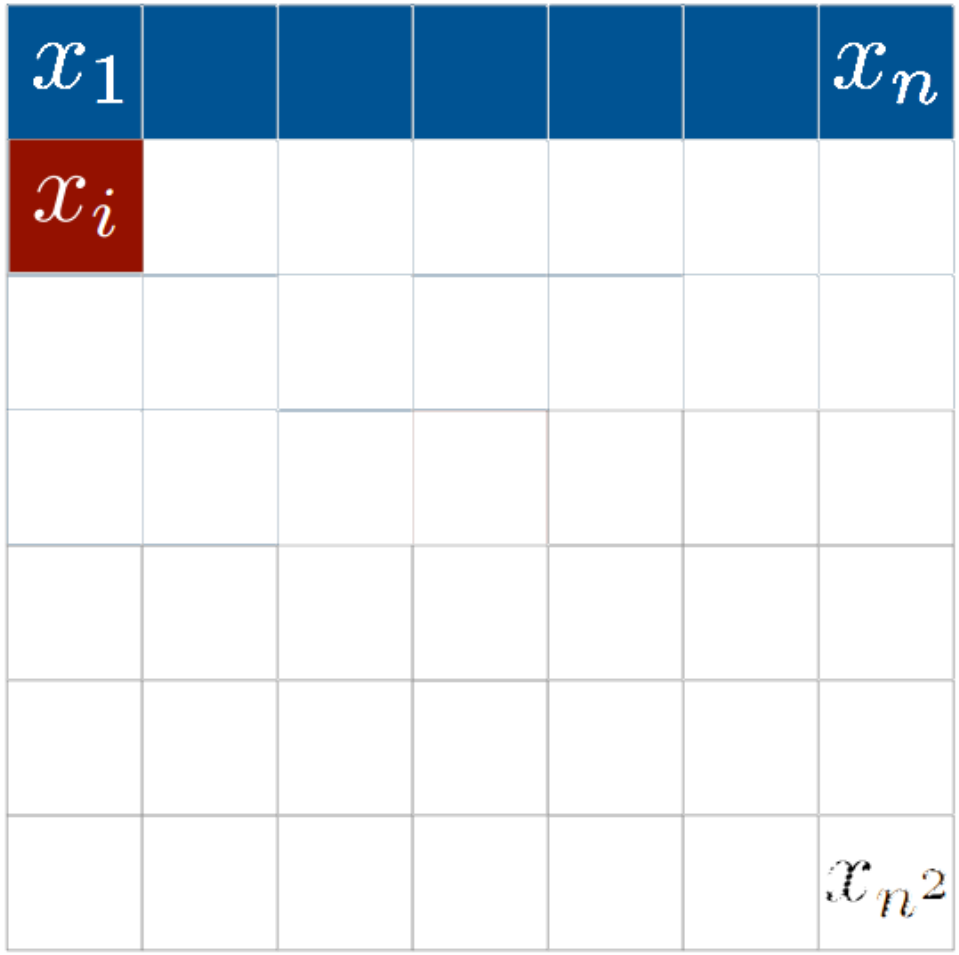

x_1
x_i
x_{n^2}

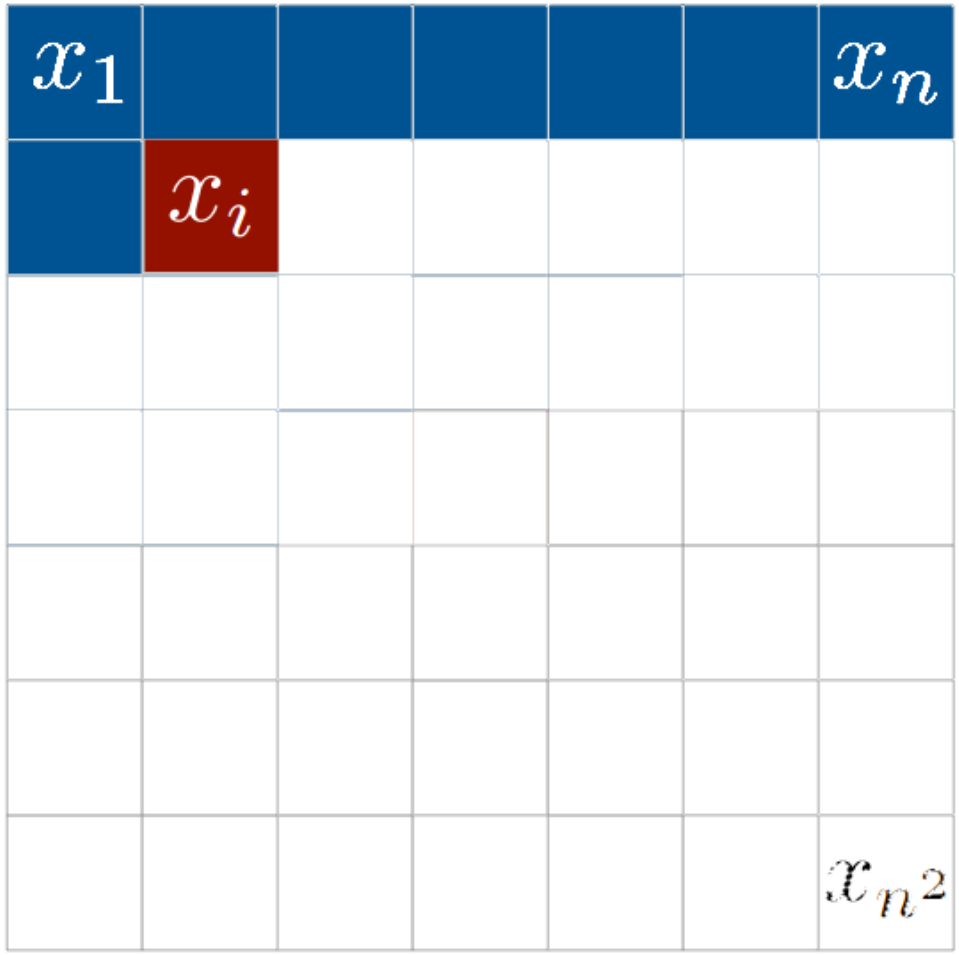

x_1
x_i
x_{n^2}

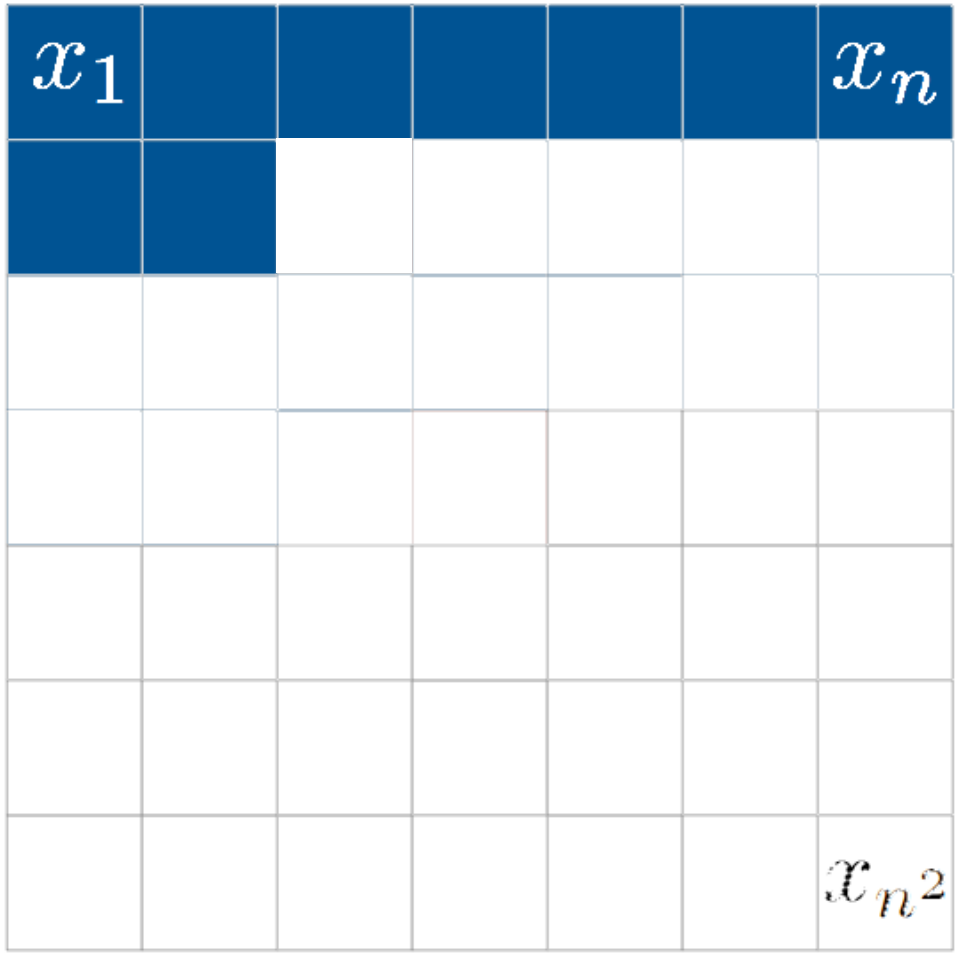

x_1
x_i
x_{n^2}

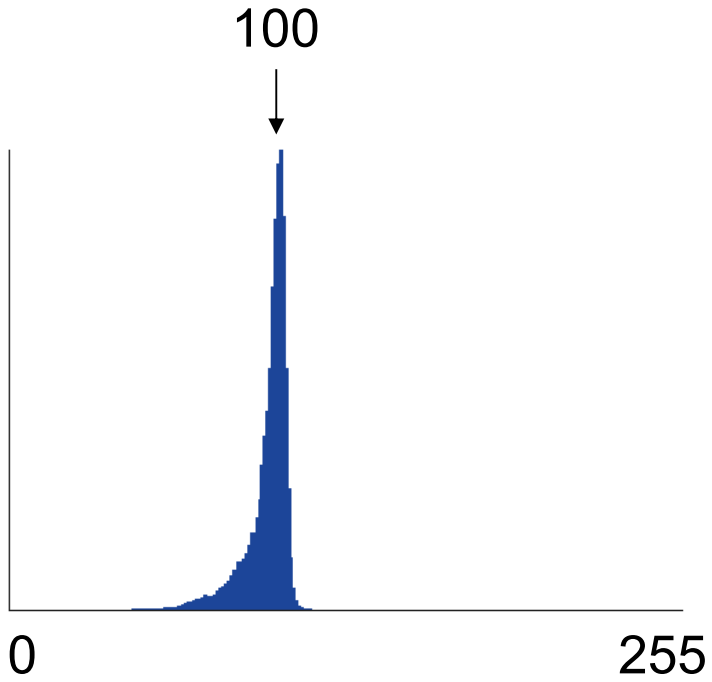

x_1
x_i
x_{n^2}


x_1
x_i
x_{n^2}

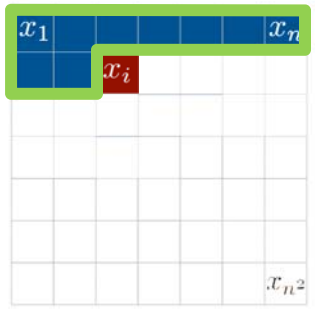

x_1
x_n
x_i
x_{n^2}

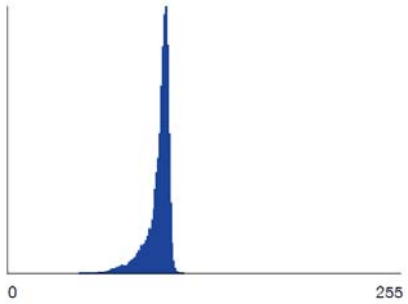

x_1
x_n
x_i
x_{n^2}


x_1
x_n
x_{n^2}

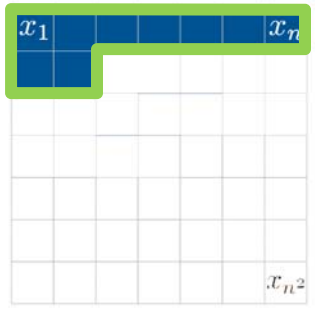

100
0
255

R

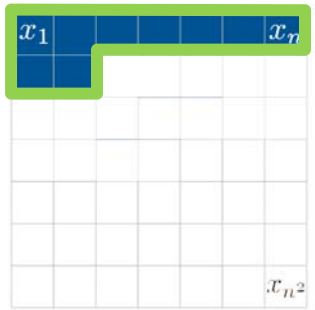




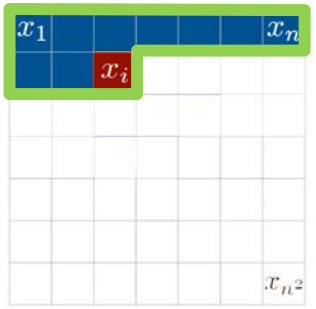
G

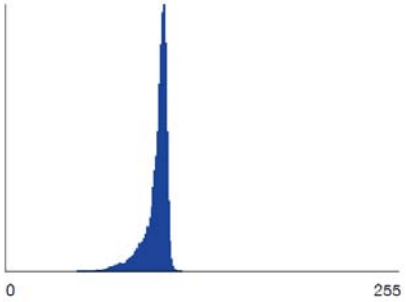


B

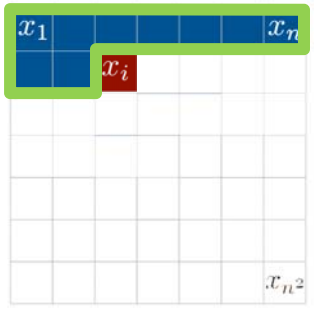


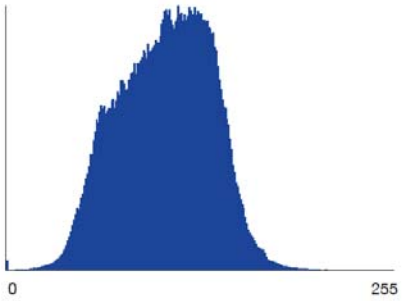
R



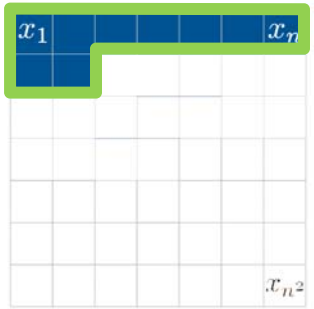


G

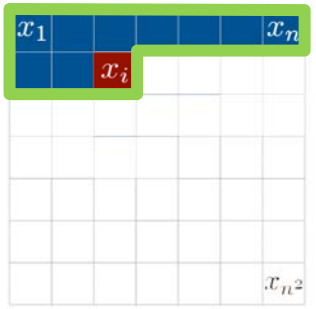


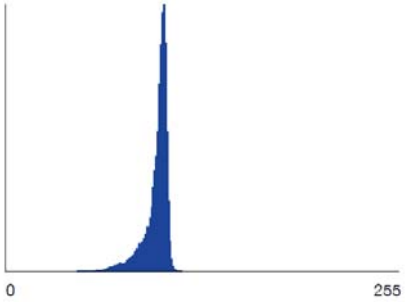


B

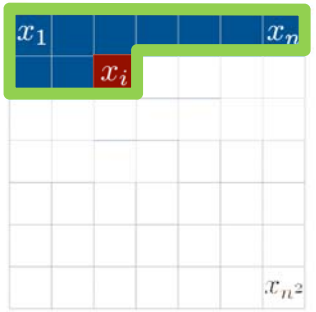


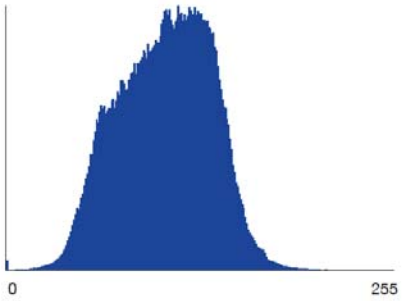
R



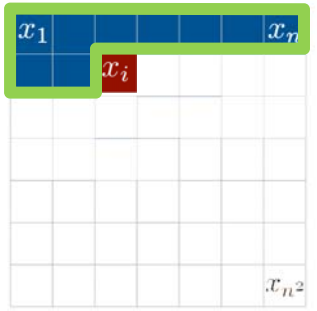


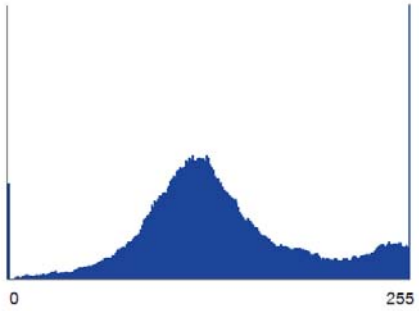
G



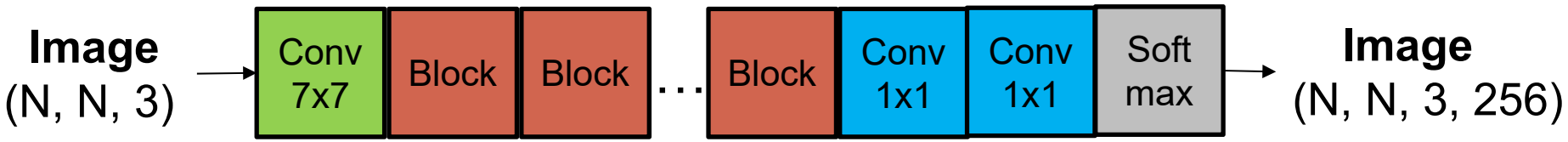


B

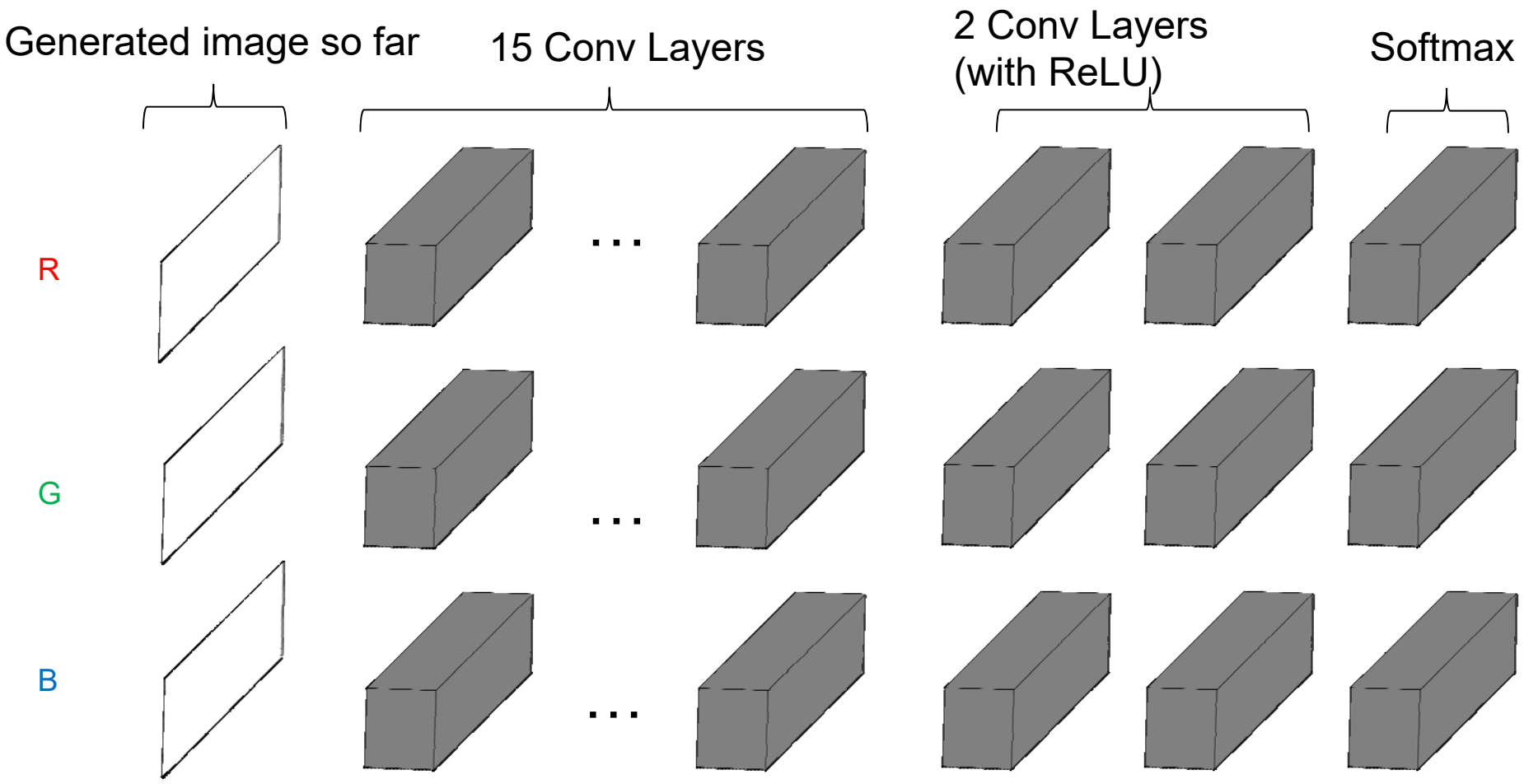


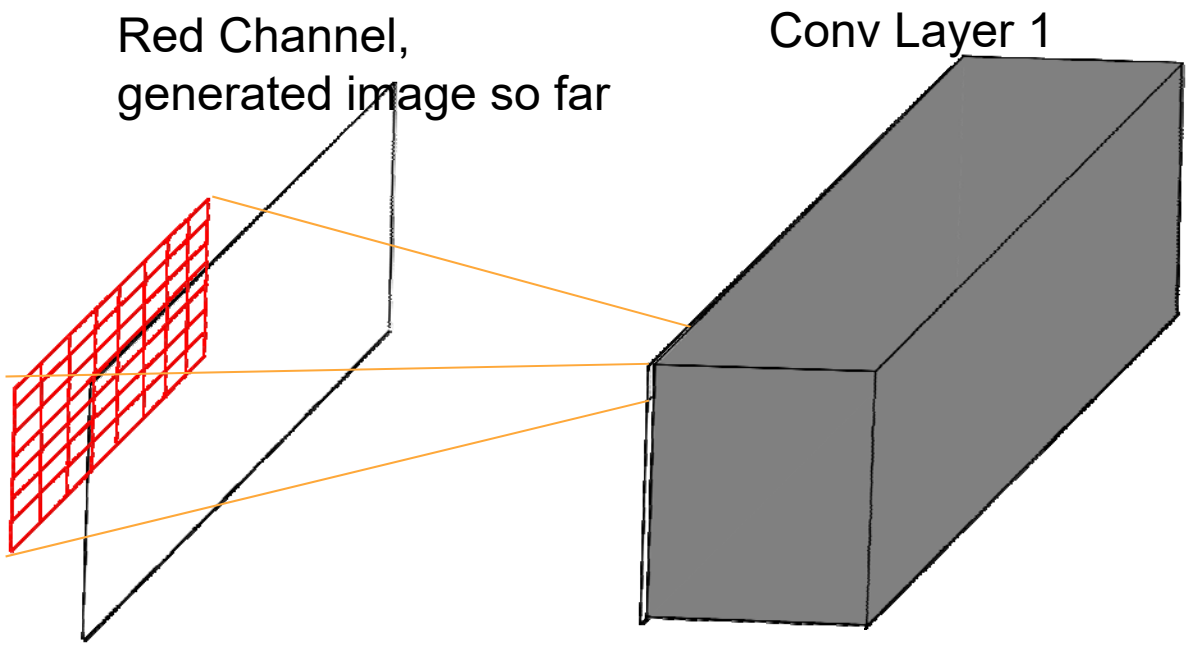


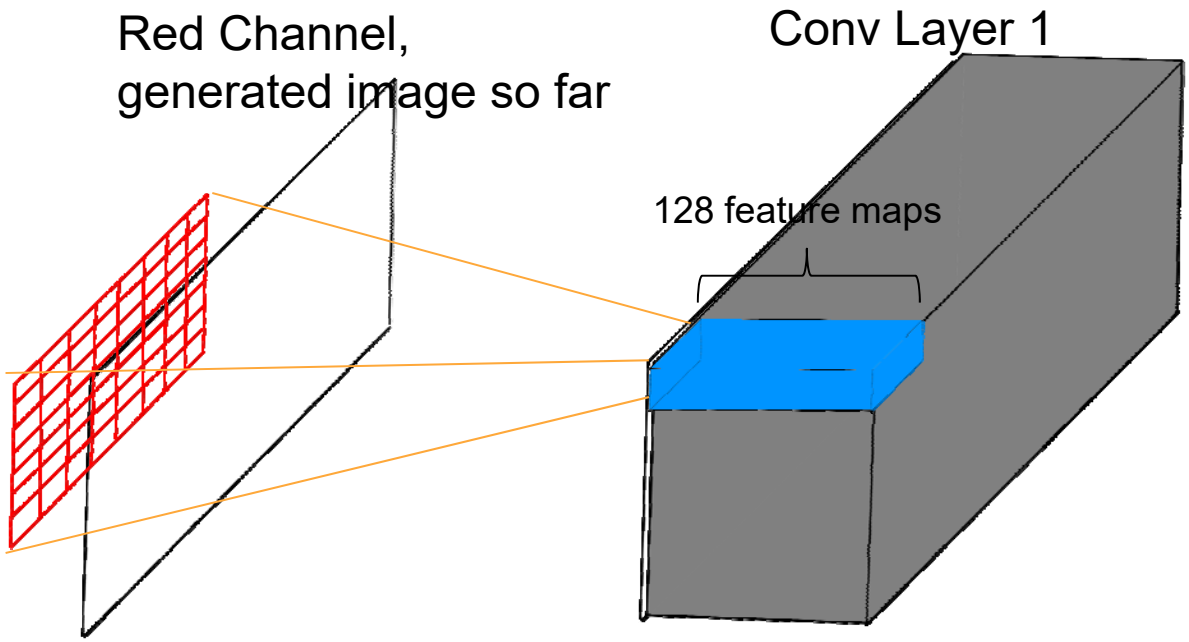
Typical Architecture

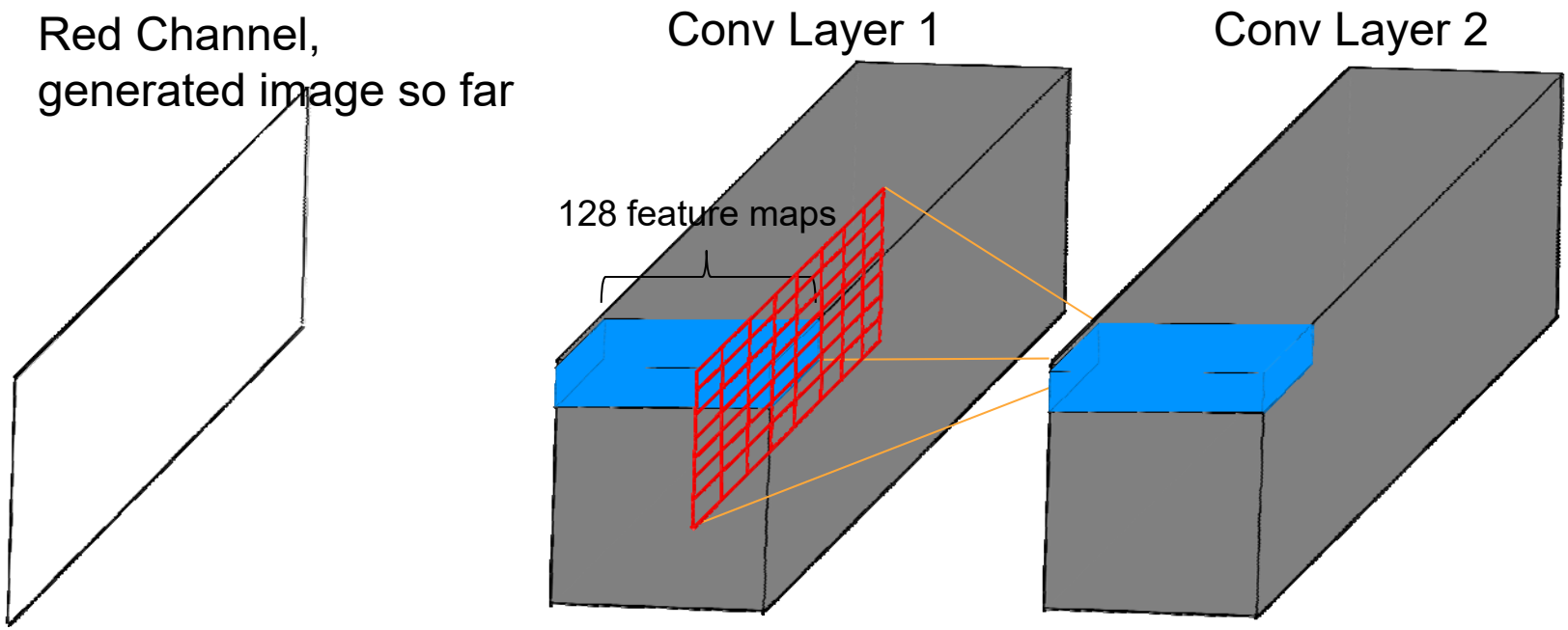


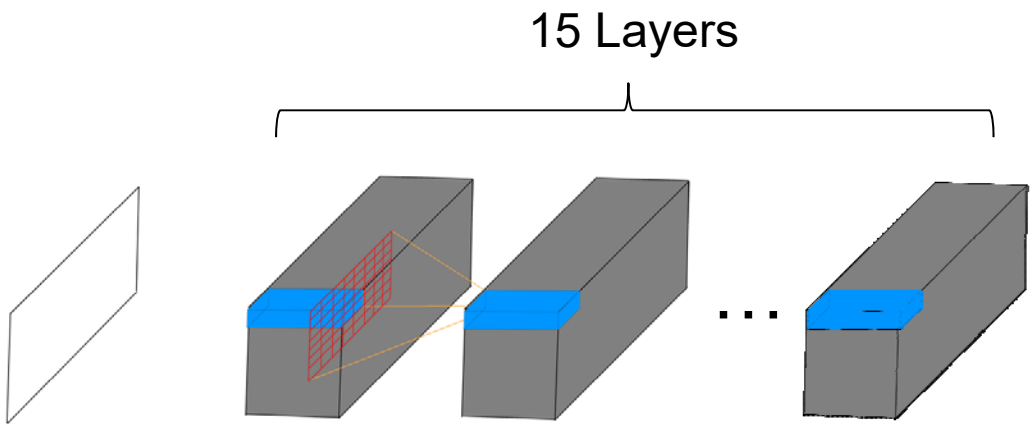
PixelCNN
(Generation Process)

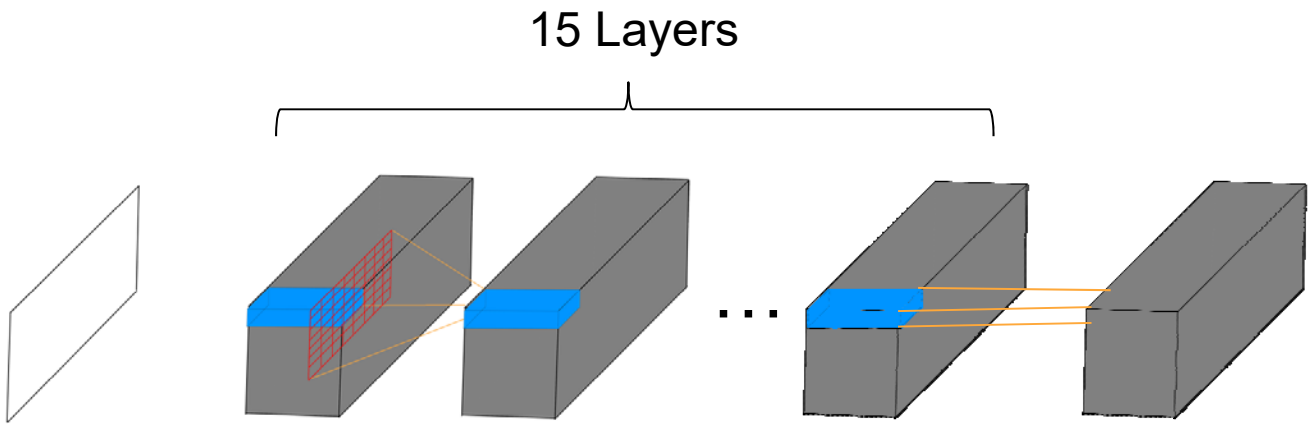

Generated image so far
15 Conv Layers
2 Conv Layers
(with ReLU)
Softmax
R
G
B

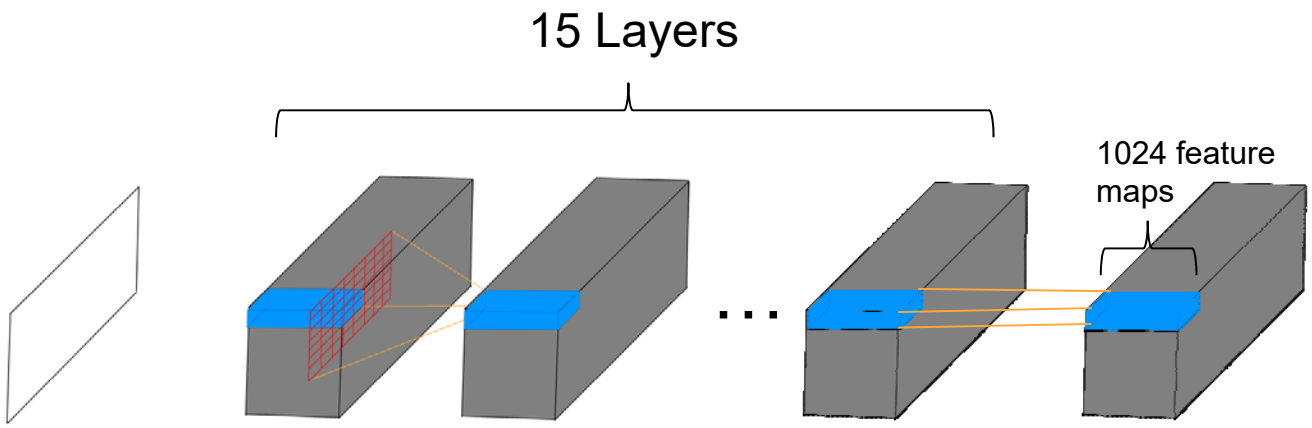

Red Channel,
generated image so far
Conv Layer 1

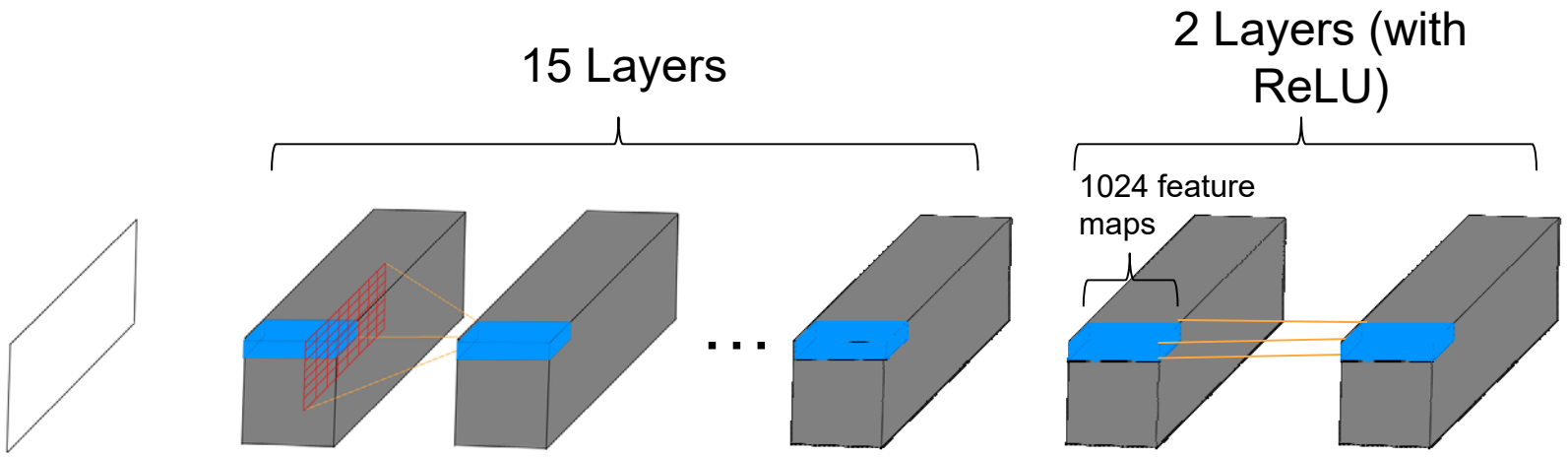

Red Channel,
generated image so far
Conv Layer 1
128 feature maps

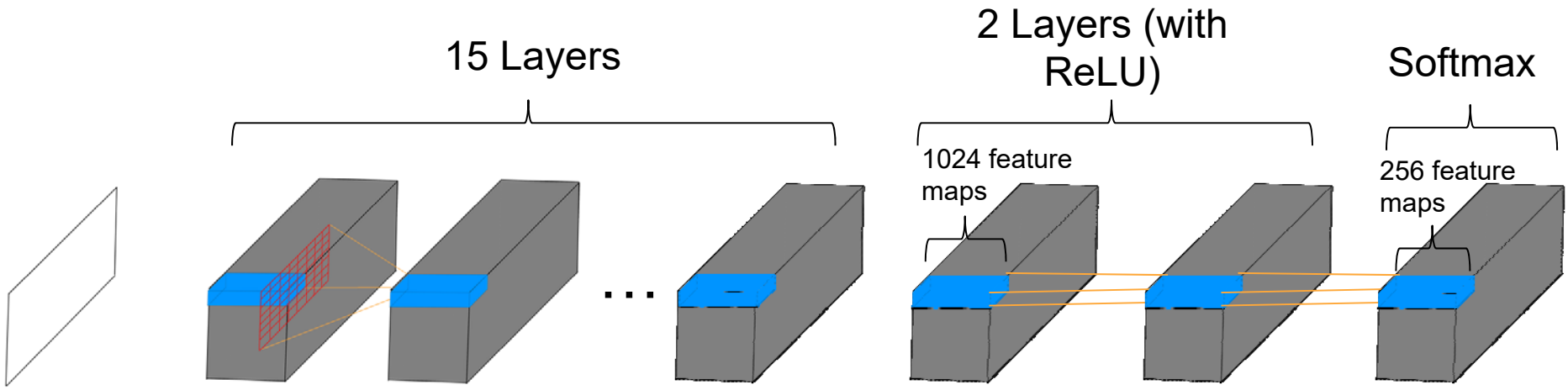

Red Channel,
generated image so far
Conv Layer 1
Conv Layer 2
128 feature maps

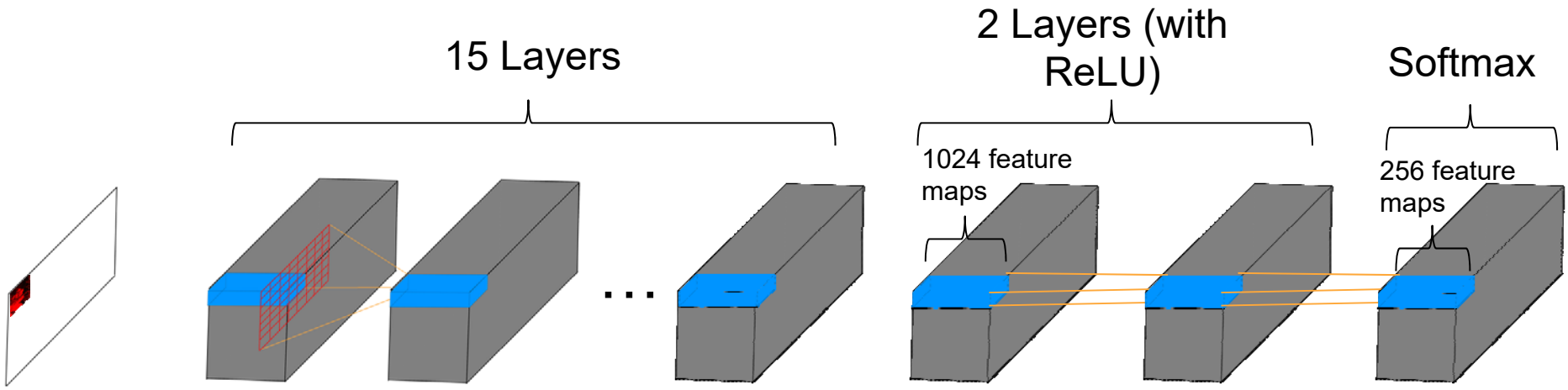

15 Layers
...

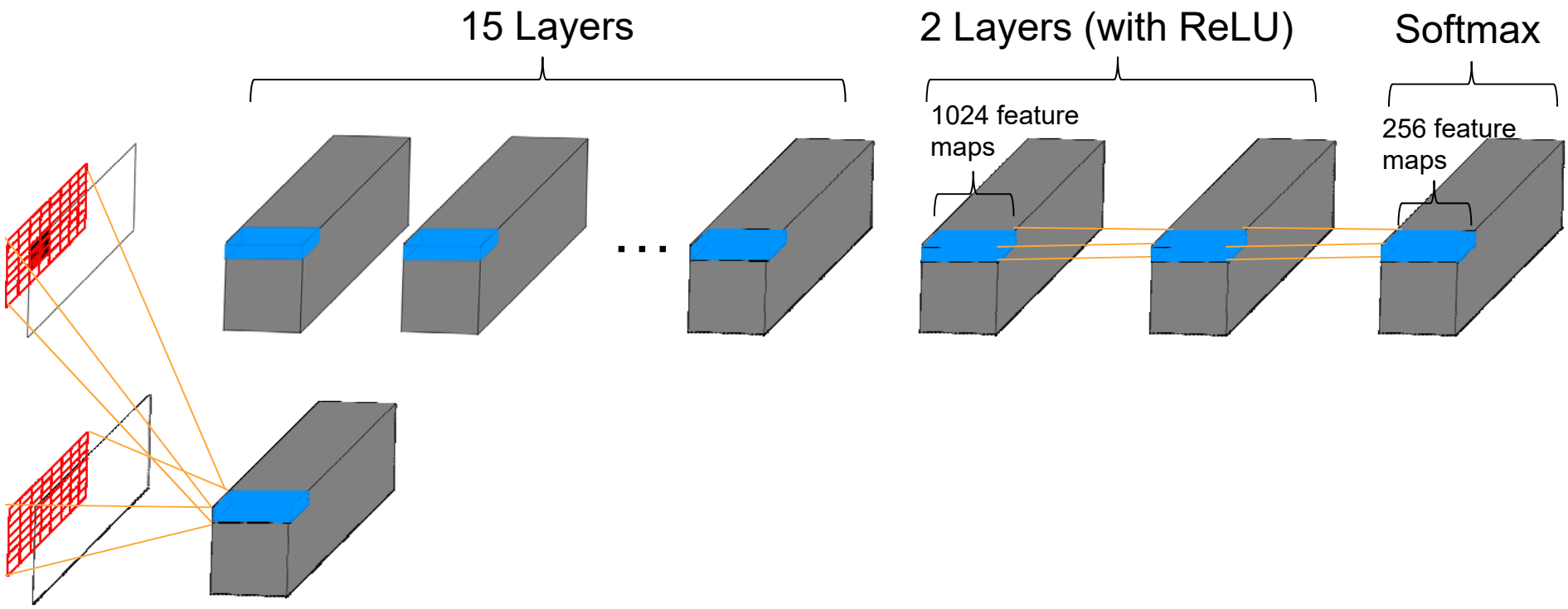

15 Layers

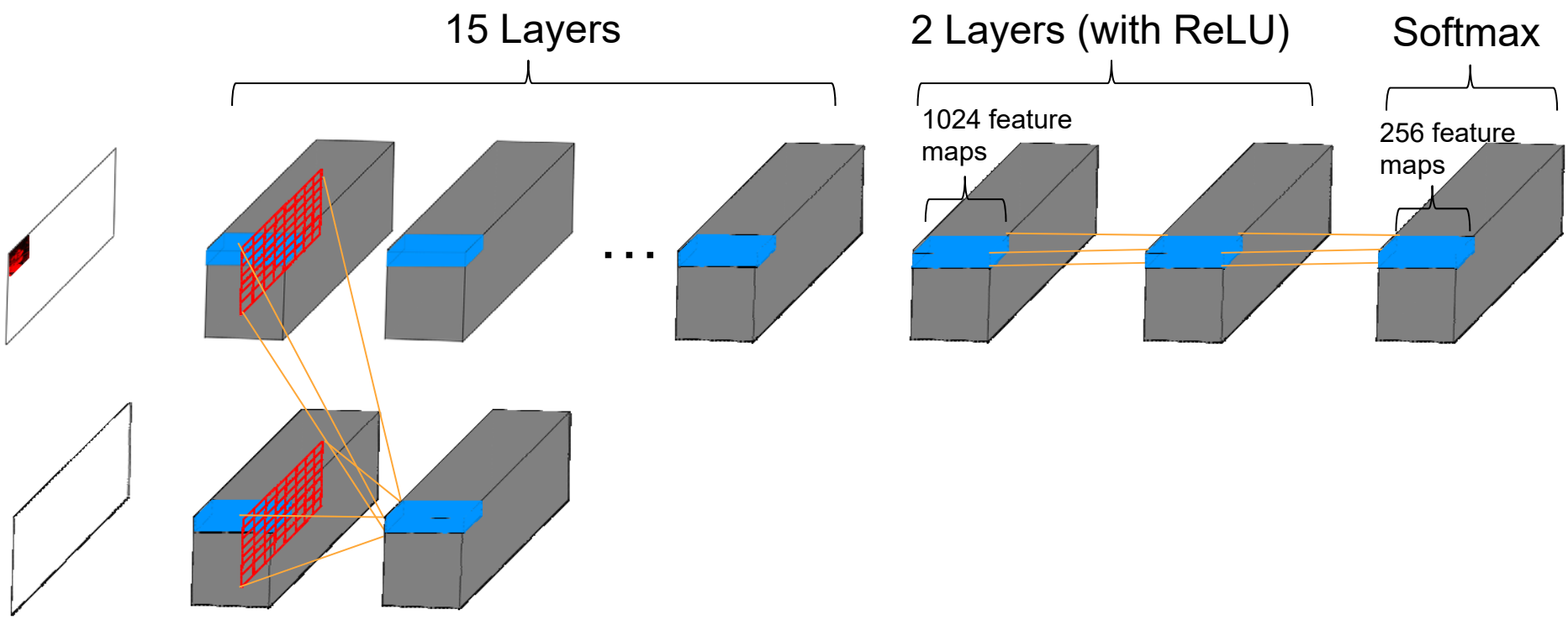

15 Layers
1024 feature maps

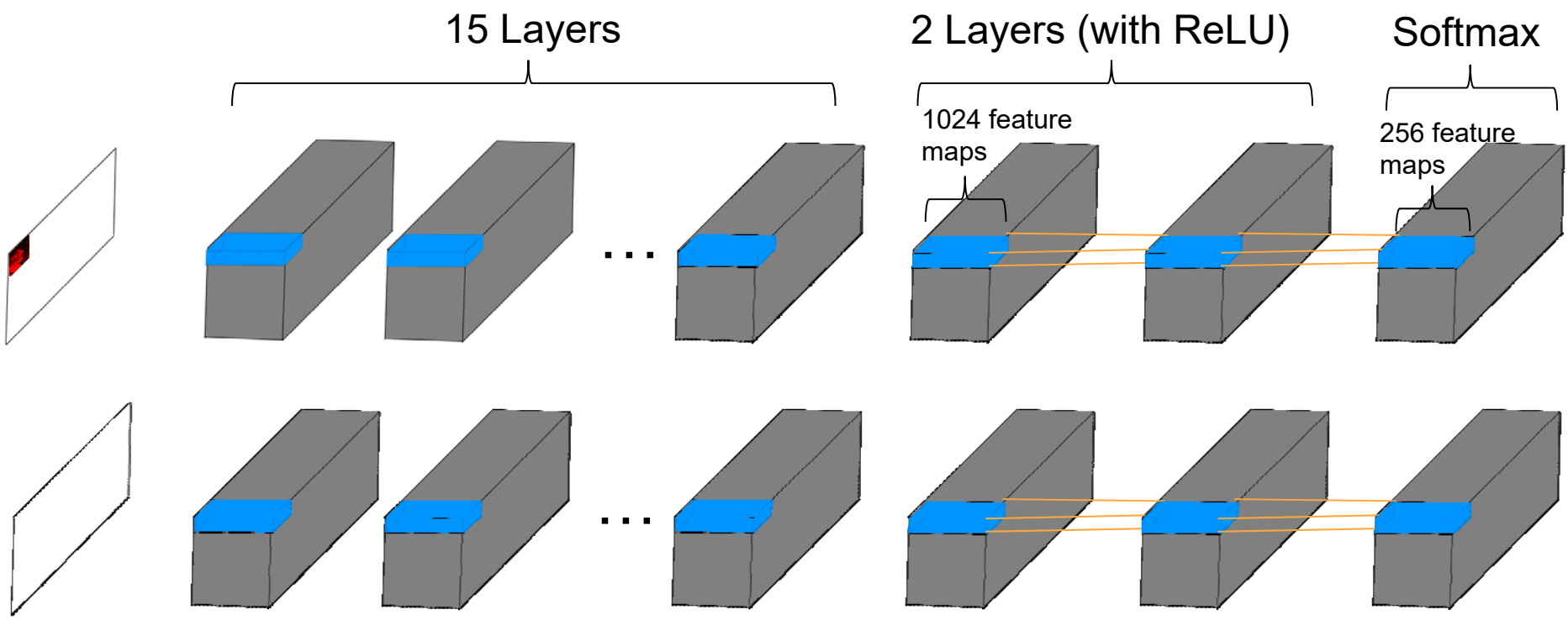

15 Layers
2 Layers (with ReLU)
1024 feature maps

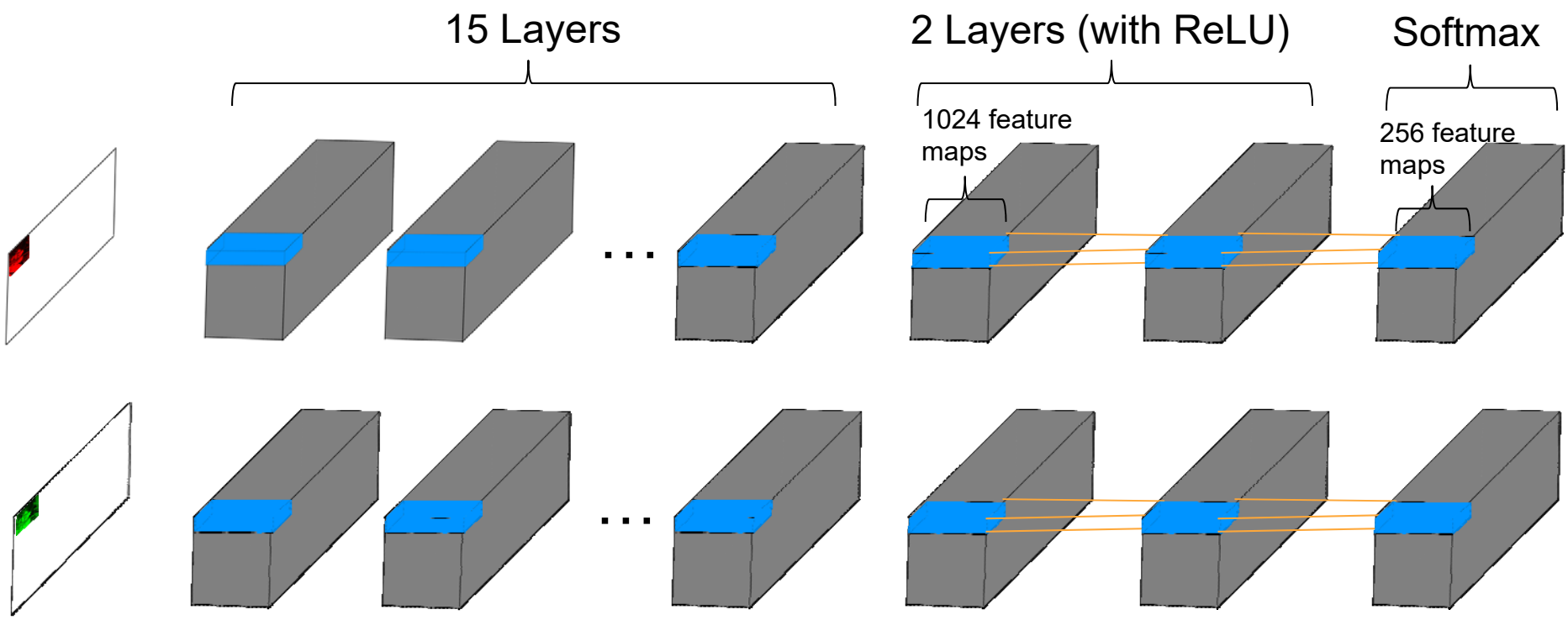

15 Layers
2 Layers (with ReLU)
Softmax
1024 feature maps
256 feature maps

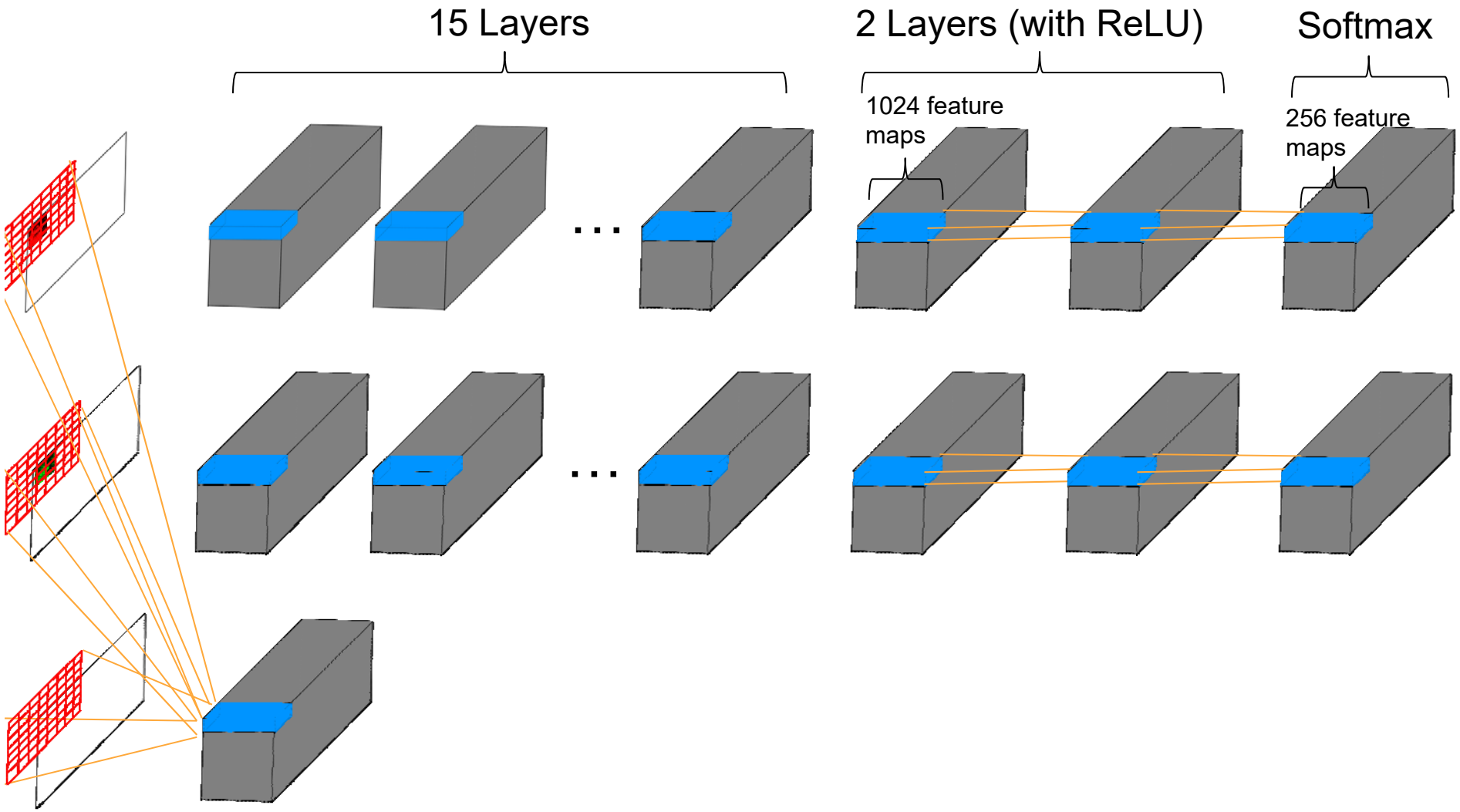

15 Layers
2 Layers (with ReLU)
Softmax
1024 feature maps
256 feature maps

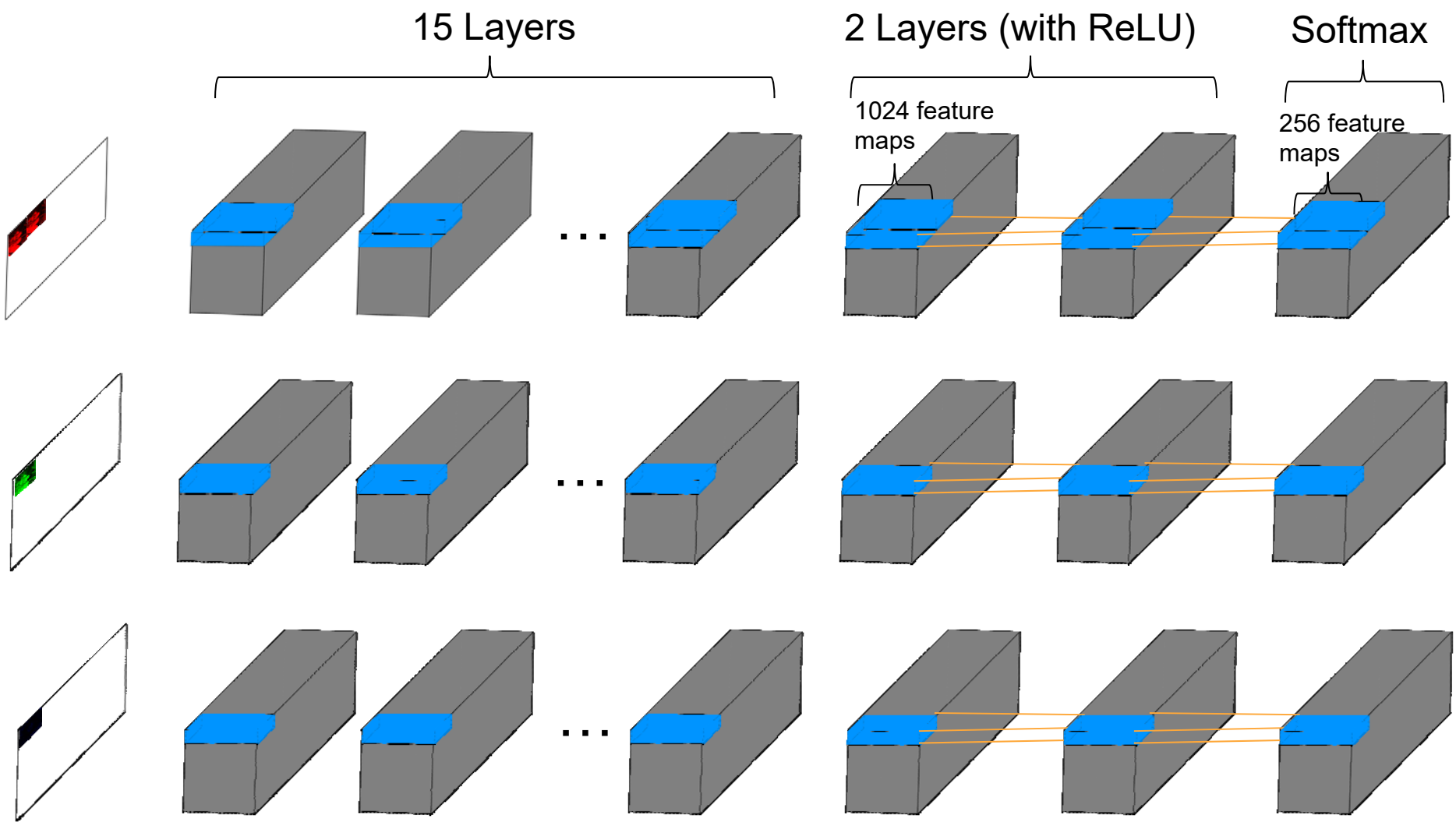

15 Layers
2 Layers (with ReLU)
Softmax
1024 feature maps
256 feature maps

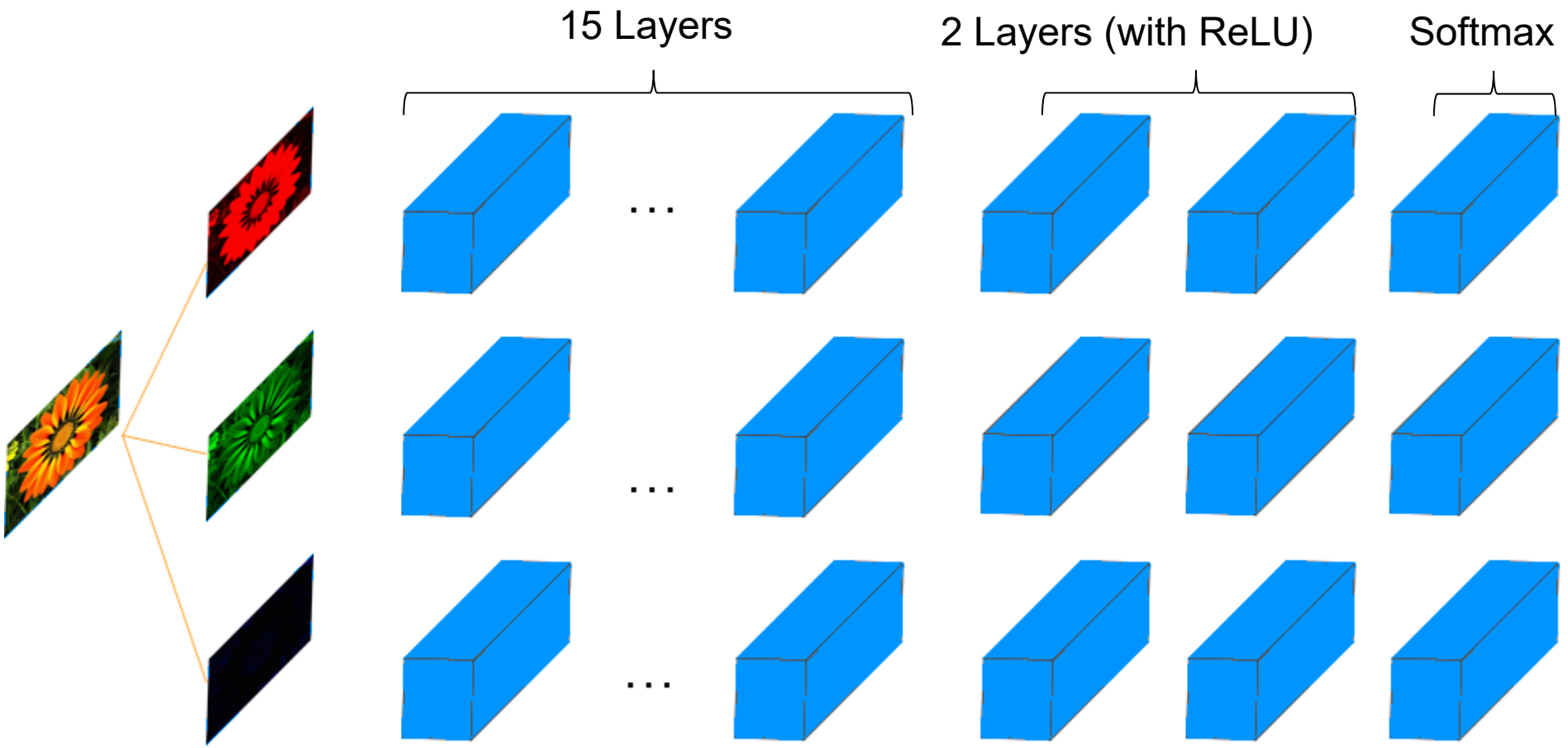

15 Layers
2 Layers (with ReLU)
Softmax
1024 feature maps
256 feature maps


15 Layers
2 Layers (with ReLU)
Softmax
1024 feature maps
256 feature maps

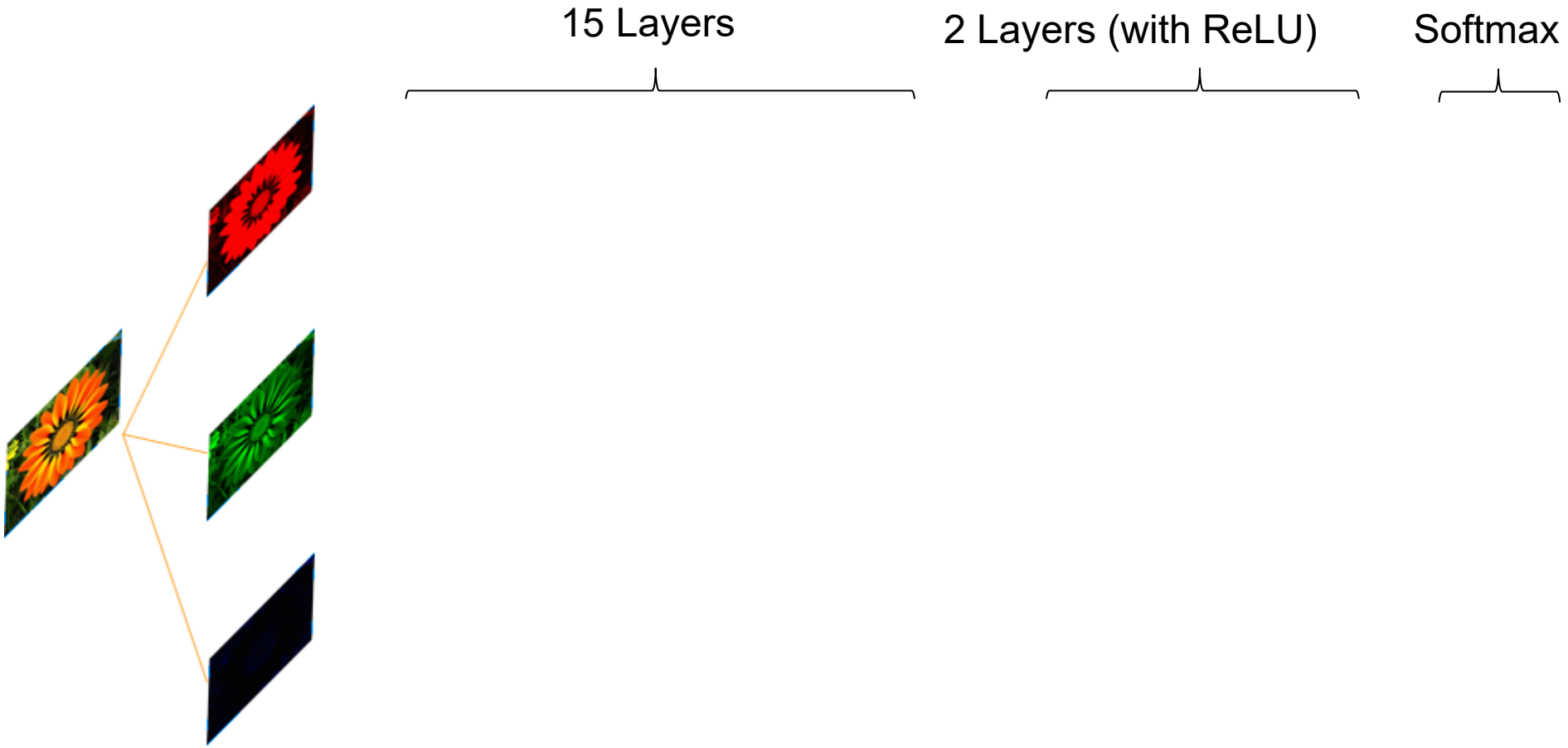

15 Layers
2 Layers (with ReLU)
Softmax
1024 feature maps
256 feature maps


15 Layers
2 Layers (with ReLU)
Softmax
1024 feature maps
256 feature maps


15 Layers
2 Layers (with ReLU)
Softmax
1024 feature maps
256 feature maps


15 Layers
2 Layers (with ReLU)
Softmax

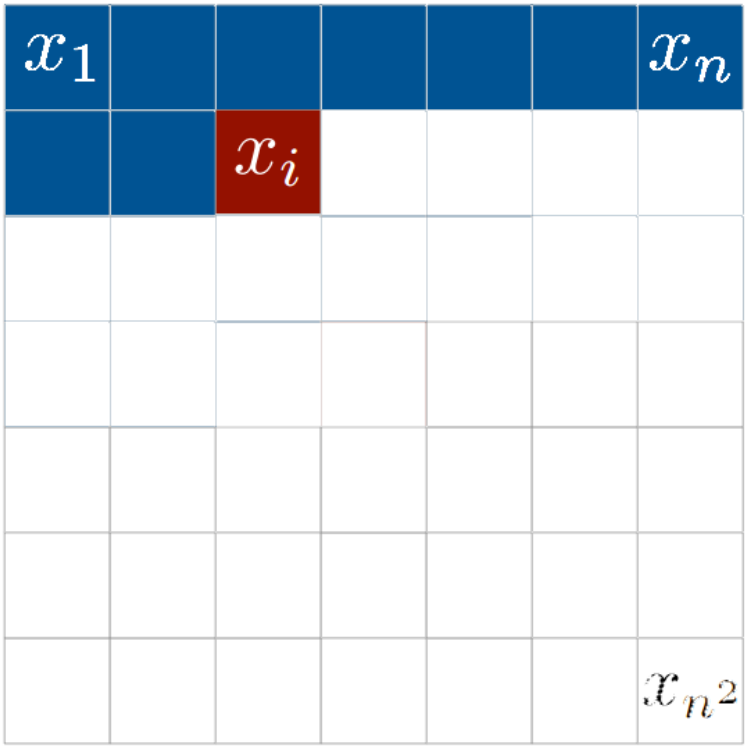
PixelCNN
(Training)


15 Layers
2 Layers (with ReLU)
Softmax

Kernel mask

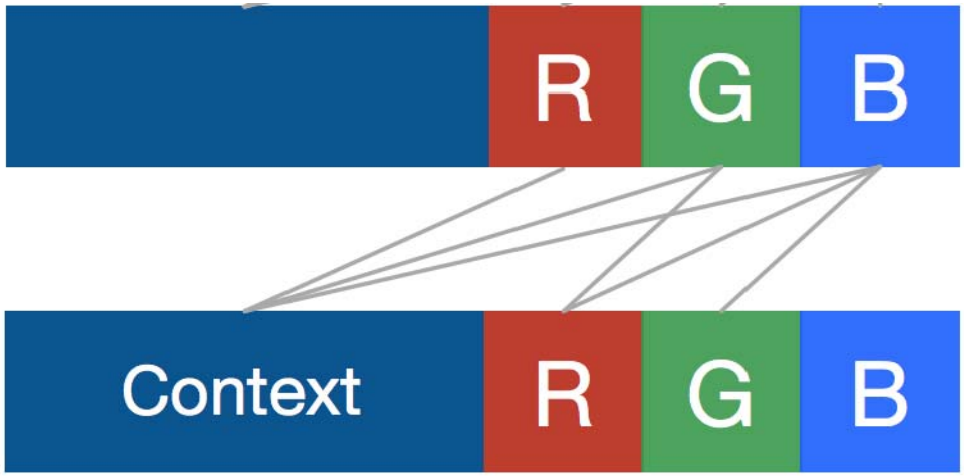
1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

Kernel mask



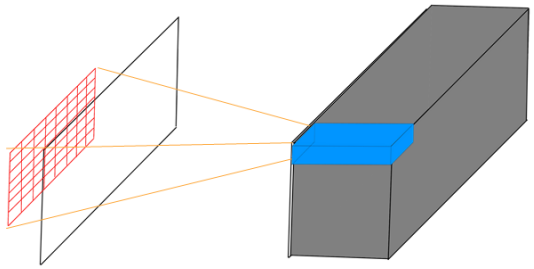
1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

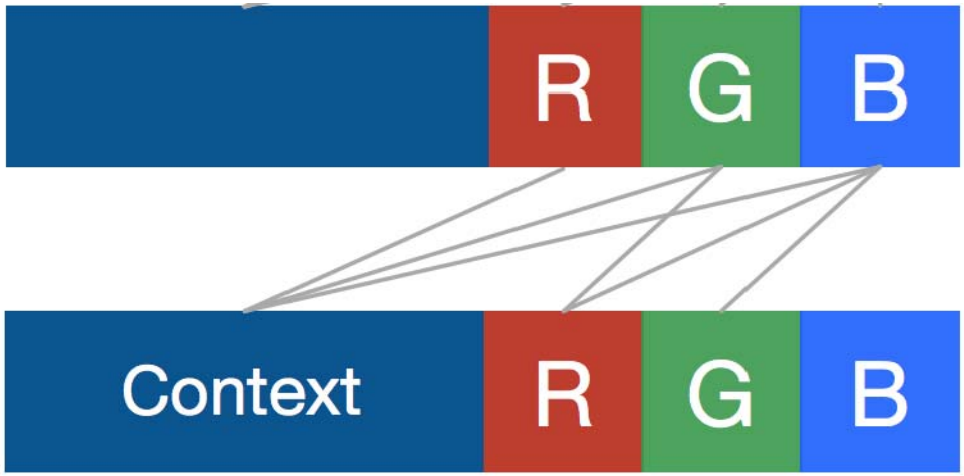
Masks



Mask A

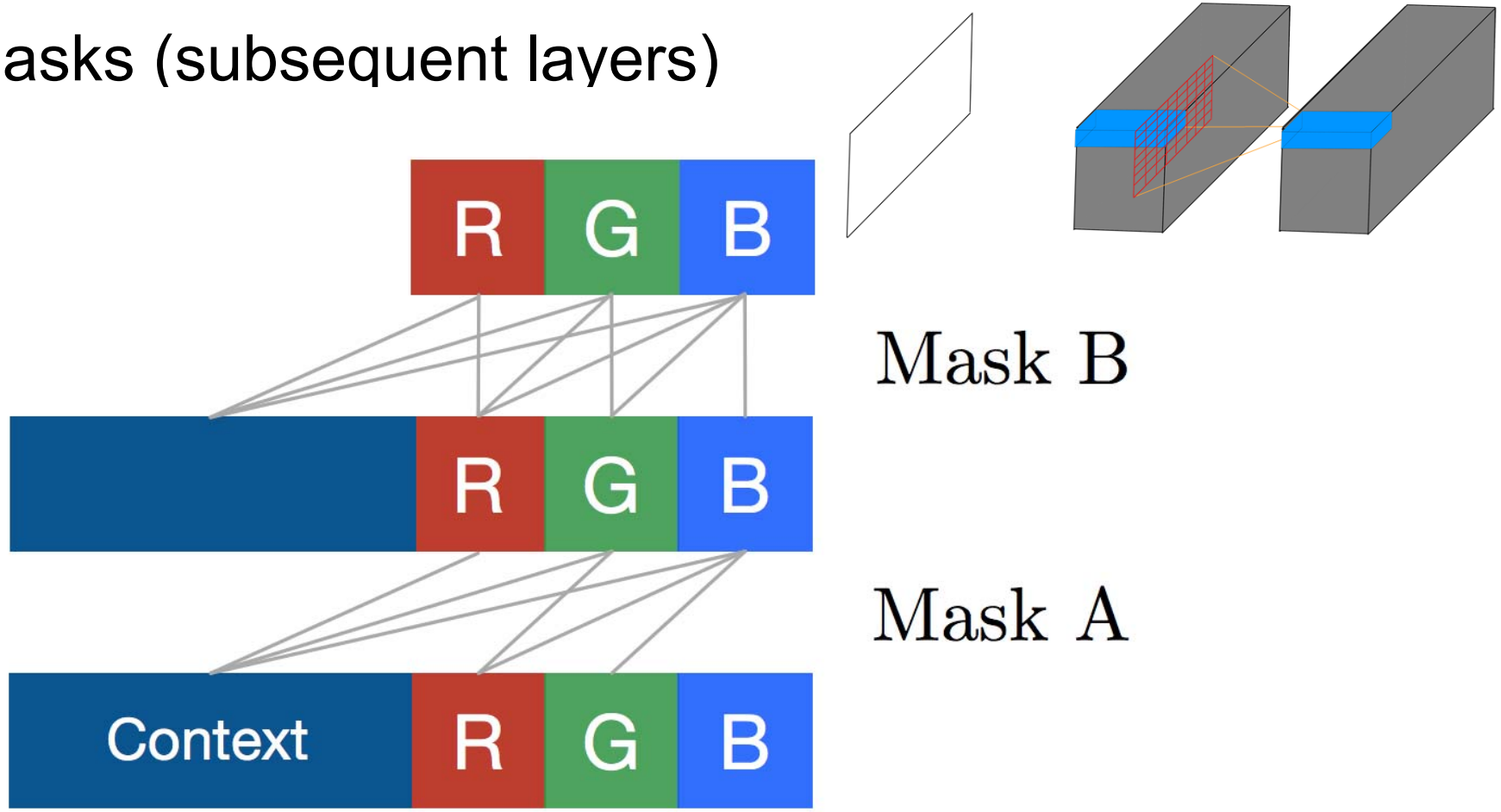
Masks





Mask A

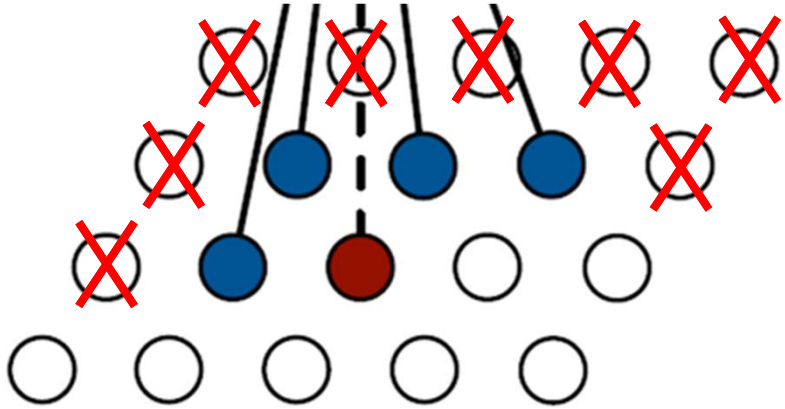
Masks (subsequent layers)



PixelCNN Advantages/Disadvantages

- Fastest to train

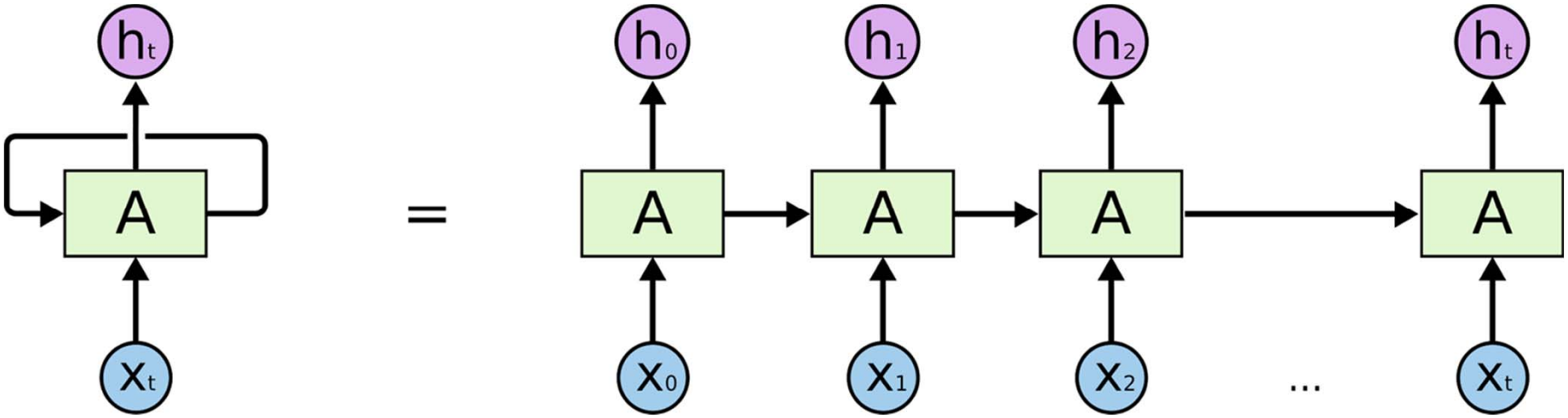
- Smallest receptive field
- Does not use all available context



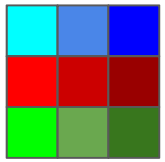
RNN Models

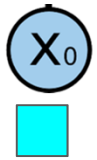
RNN Review

Sequence of data

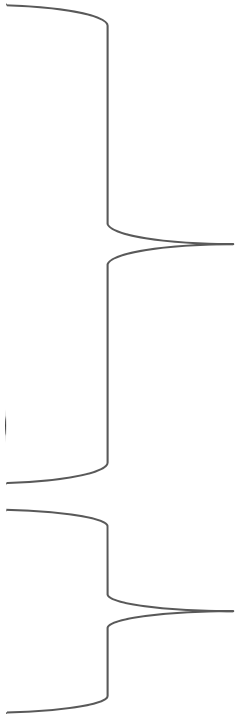


RNN for Image Generation





LSTM Equations



Gates - Control how much information is allowed through

States - Hold information about all time steps up till now {0, i, …, i-1, i}

LSTM Equations

$$i = \sigma(x_i U^i + \mathbf{h}_{i-1} W^i)$$
$$f = \sigma(x_i U^f + \mathbf{h}_{i-1} W^f)$$
$$o = \sigma(x_i U^o + \mathbf{h}_{i-1} W^o)$$
$$g = tanh(x_i U^g + \mathbf{h}_{i-1} W^g)$$

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

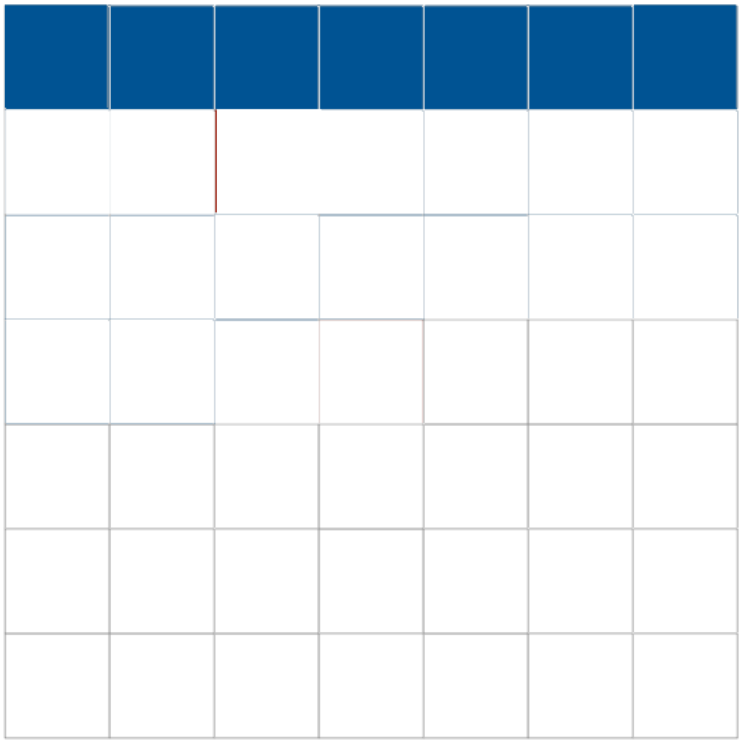
$$c_i = c_{i-1} \circ f + g \circ i$$
$$\mathbf{h}_i = \tanh(c_i) \circ o$$

$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i$$
$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$$

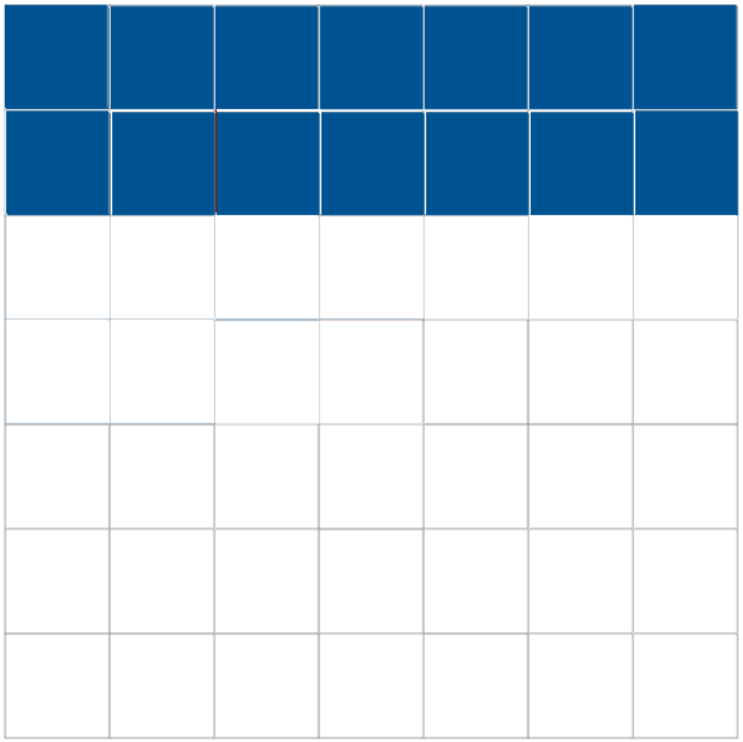
Like Convolutional LSTM - replaced fully-connected layer with convolutional layer

Row LSTM

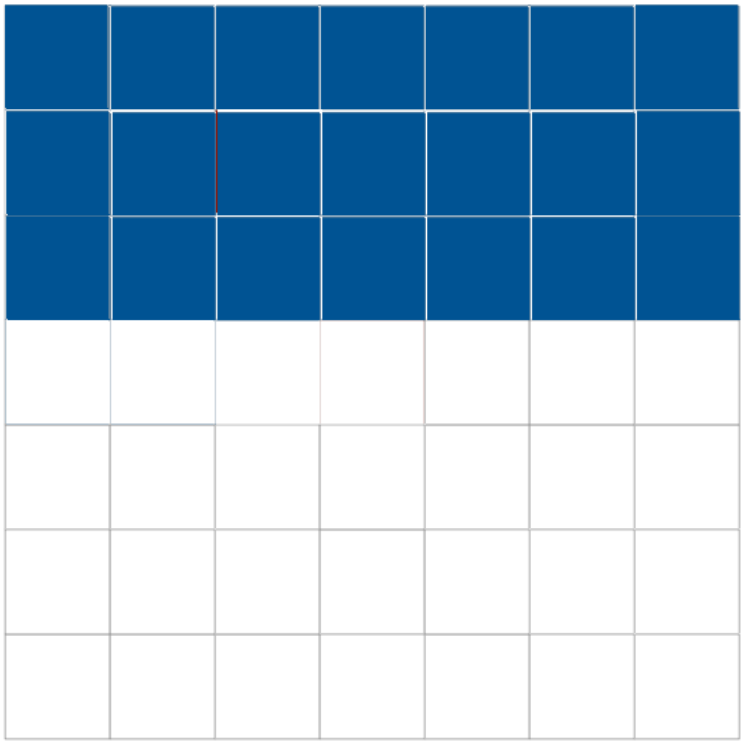
Row LSTM



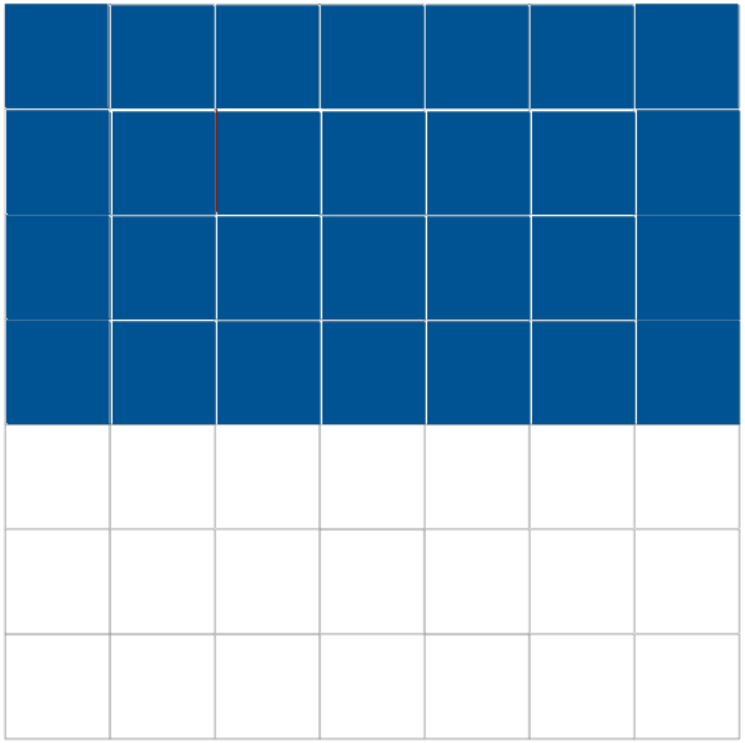
Row LSTM



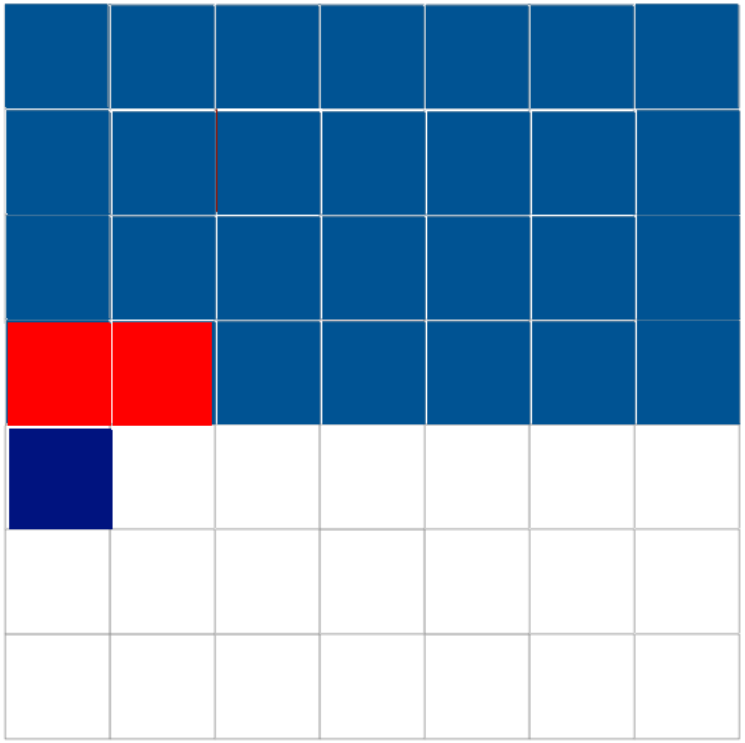
Row LSTM



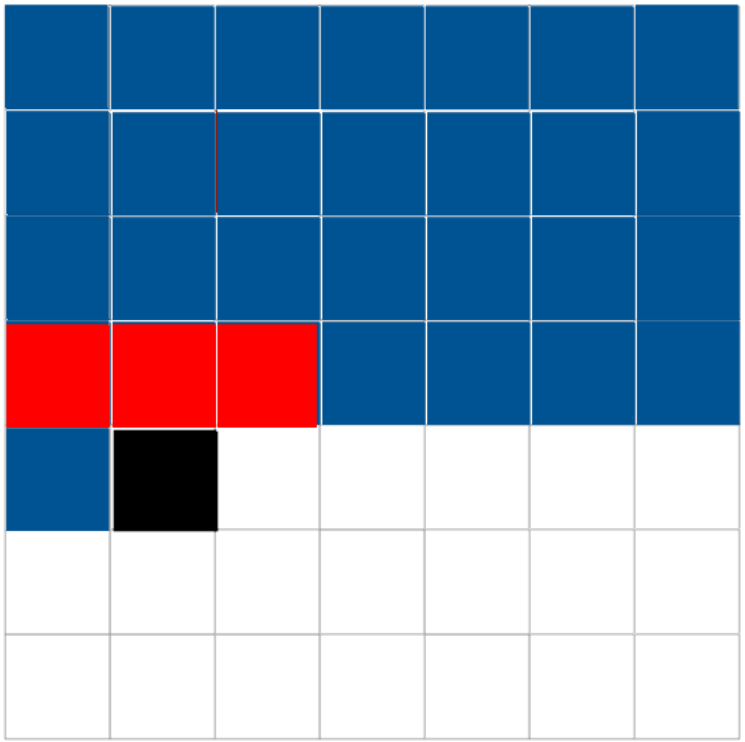
Row LSTM



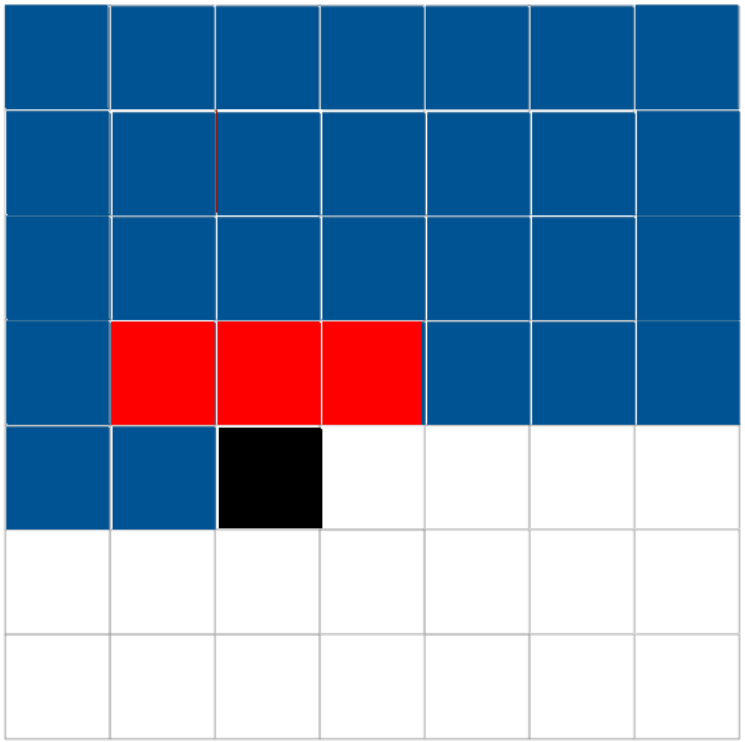
Row LSTM



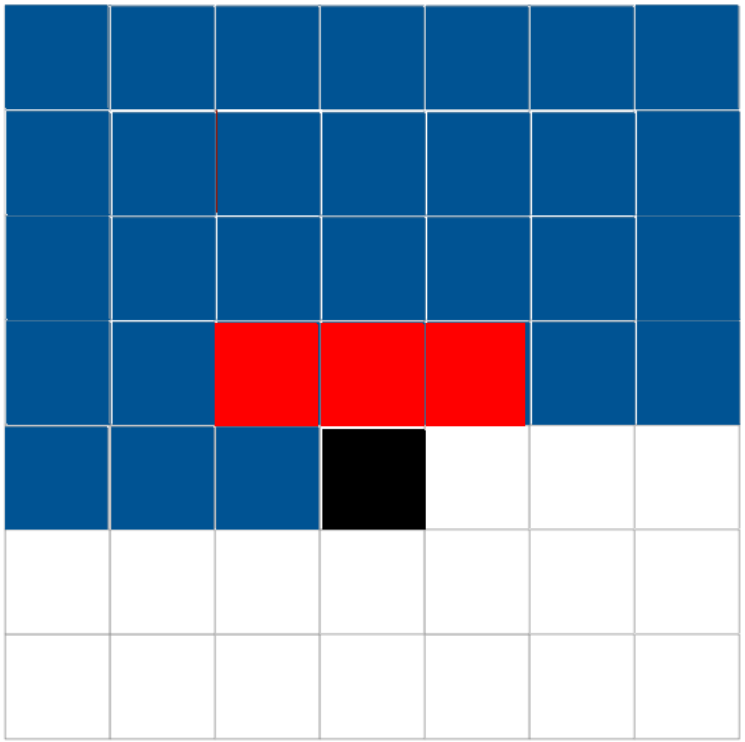
Row LSTM



Row LSTM

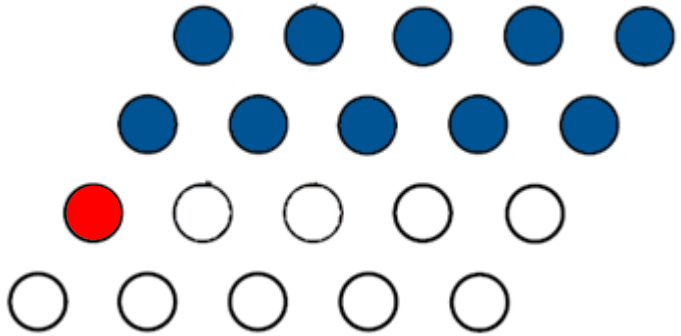


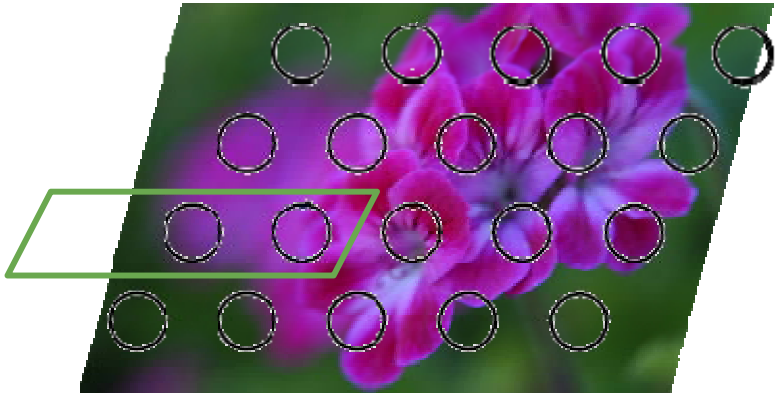
Row LSTM



Input-to-State Component

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

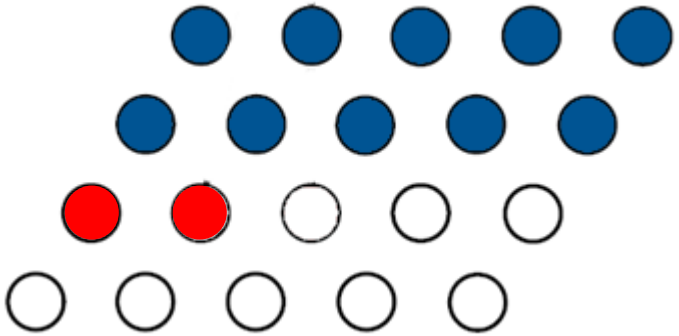


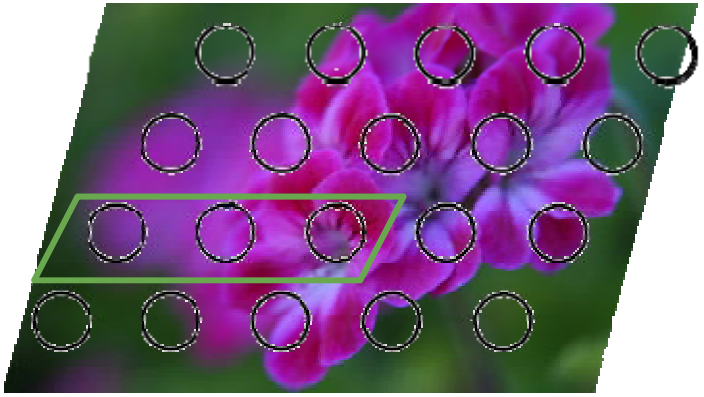


Image

Input-to-State Component

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

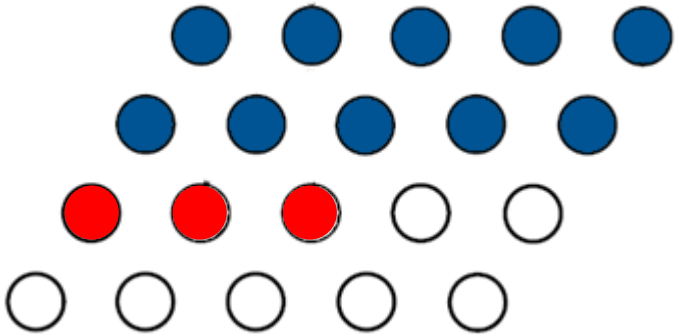


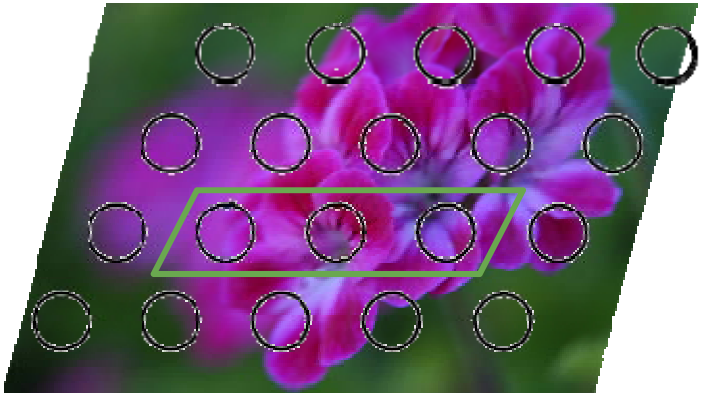


Image

Input-to-State Component

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

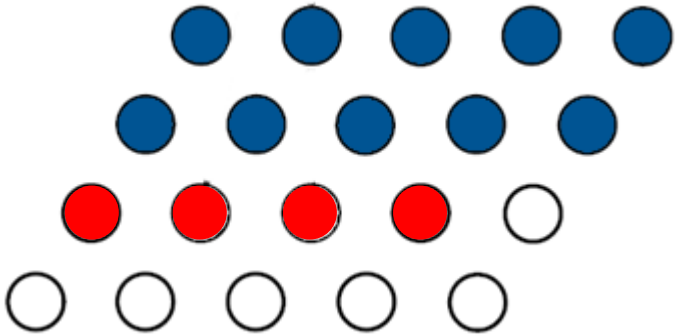




Image

Input-to-State Component

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

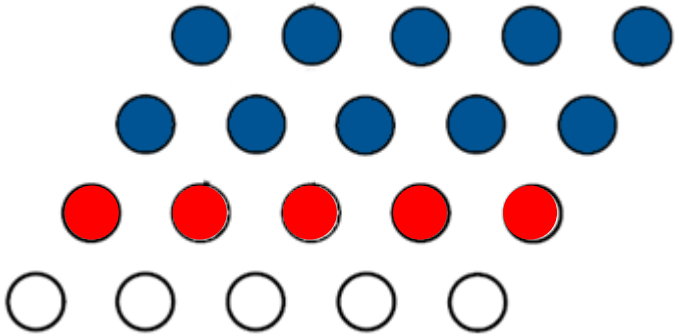




Image

Input-to-State Component

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

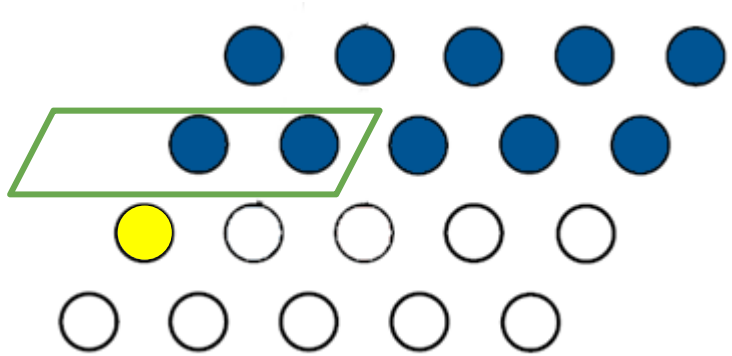




Image

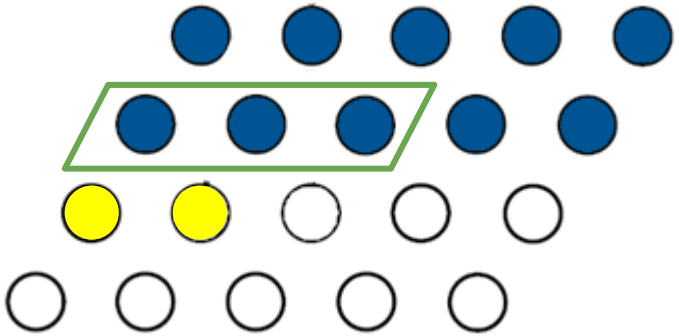
State-to-State Component

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



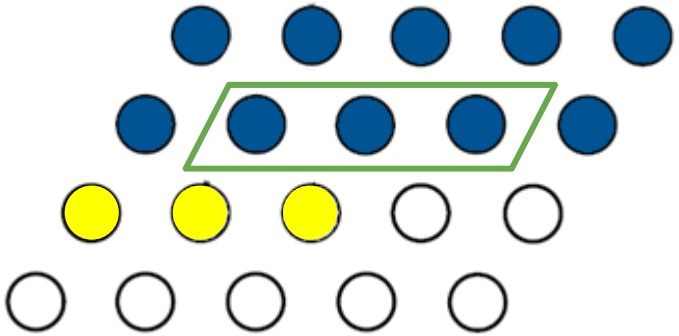
State-to-State Component

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



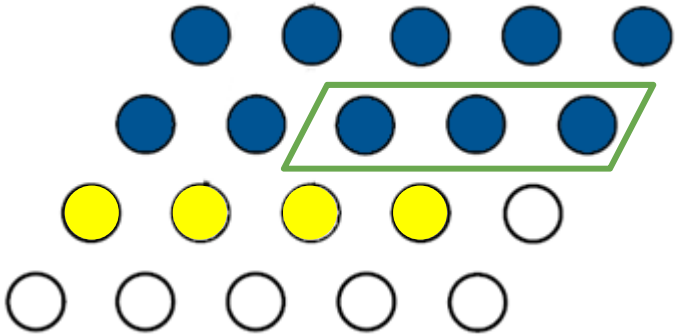
State-to-State Component

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



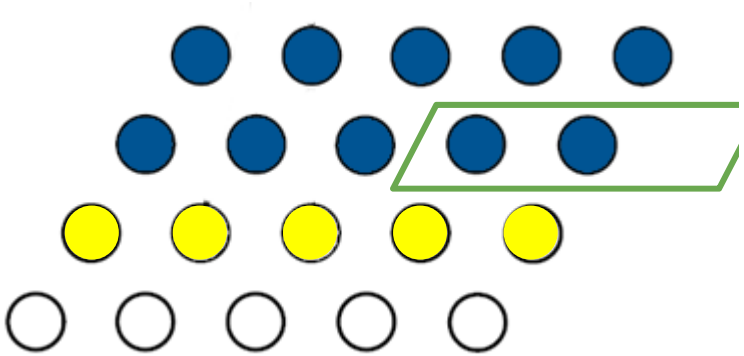
State-to-State Component

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



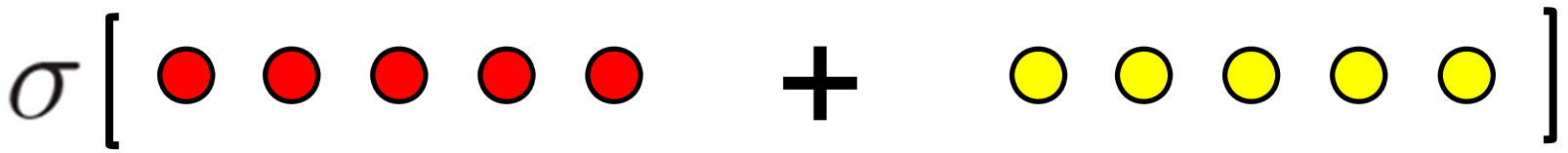
State-to-State Component

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

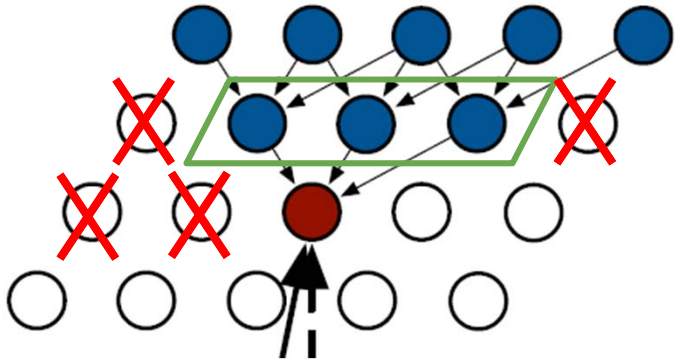


Combine State Components

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



Advantages and Disadvantages

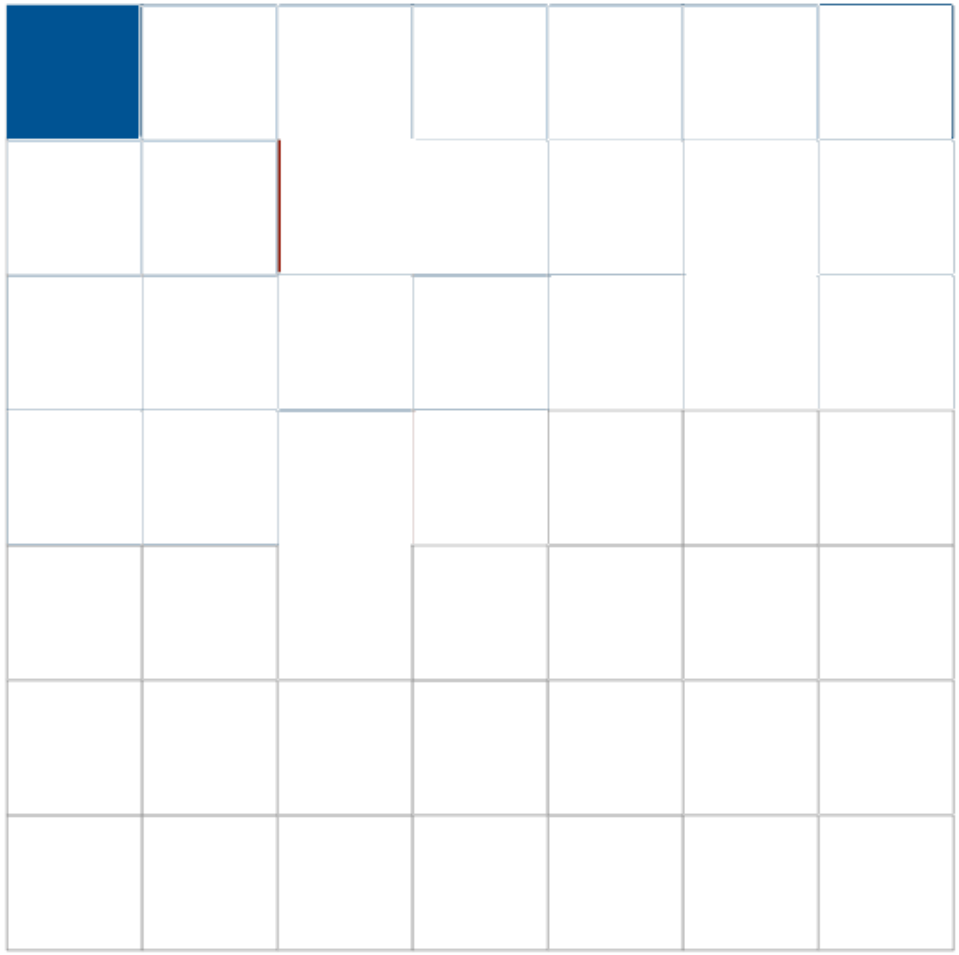


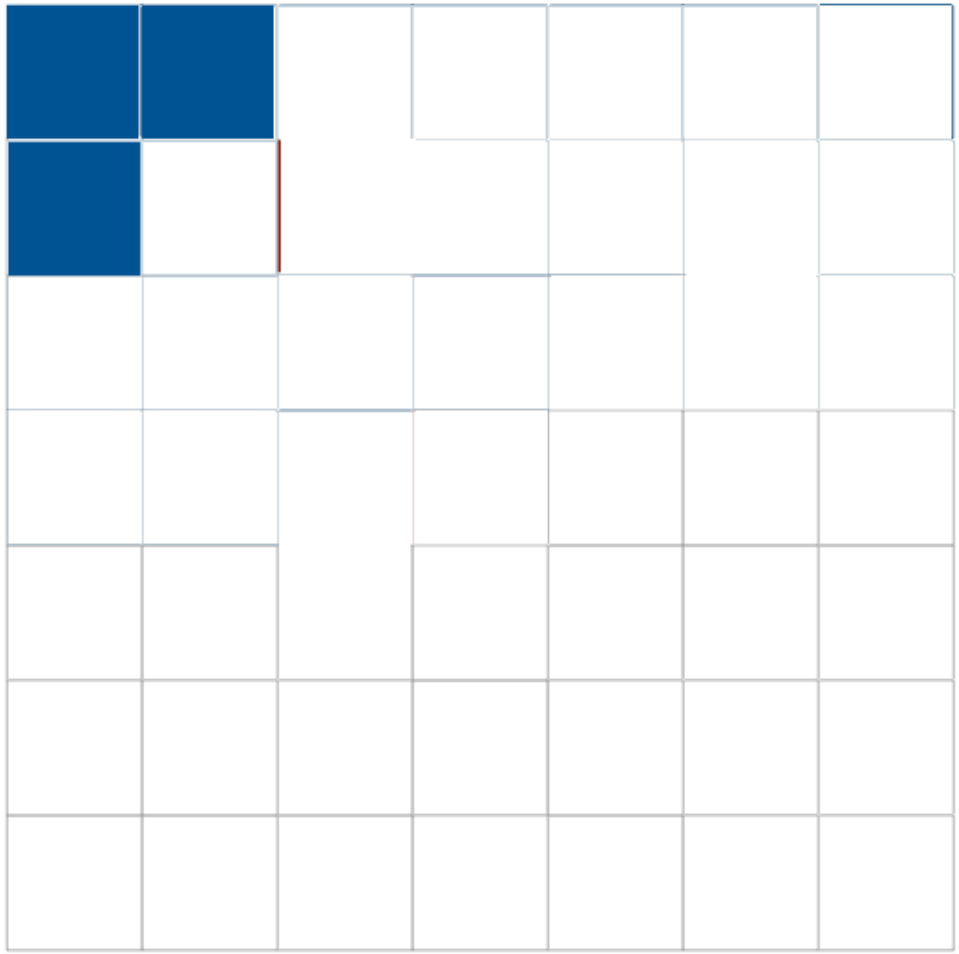
Compute the state for an entire row at once

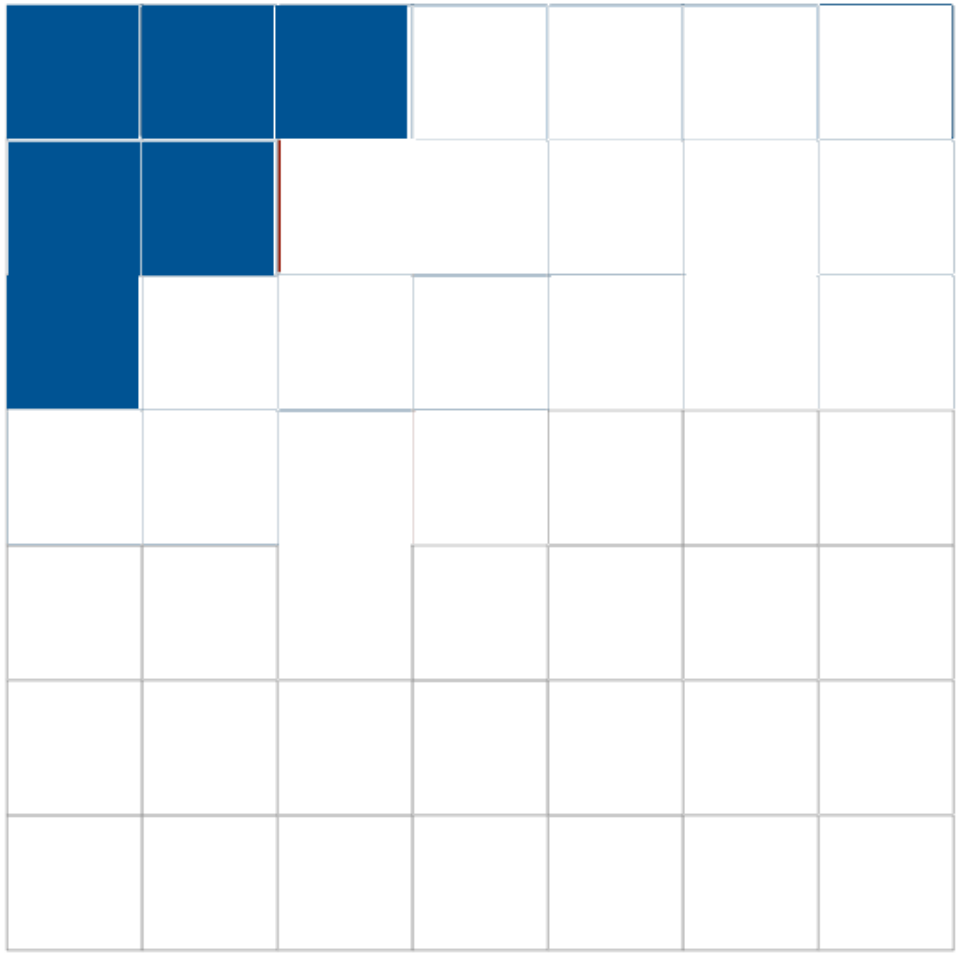
Triangular receptive field

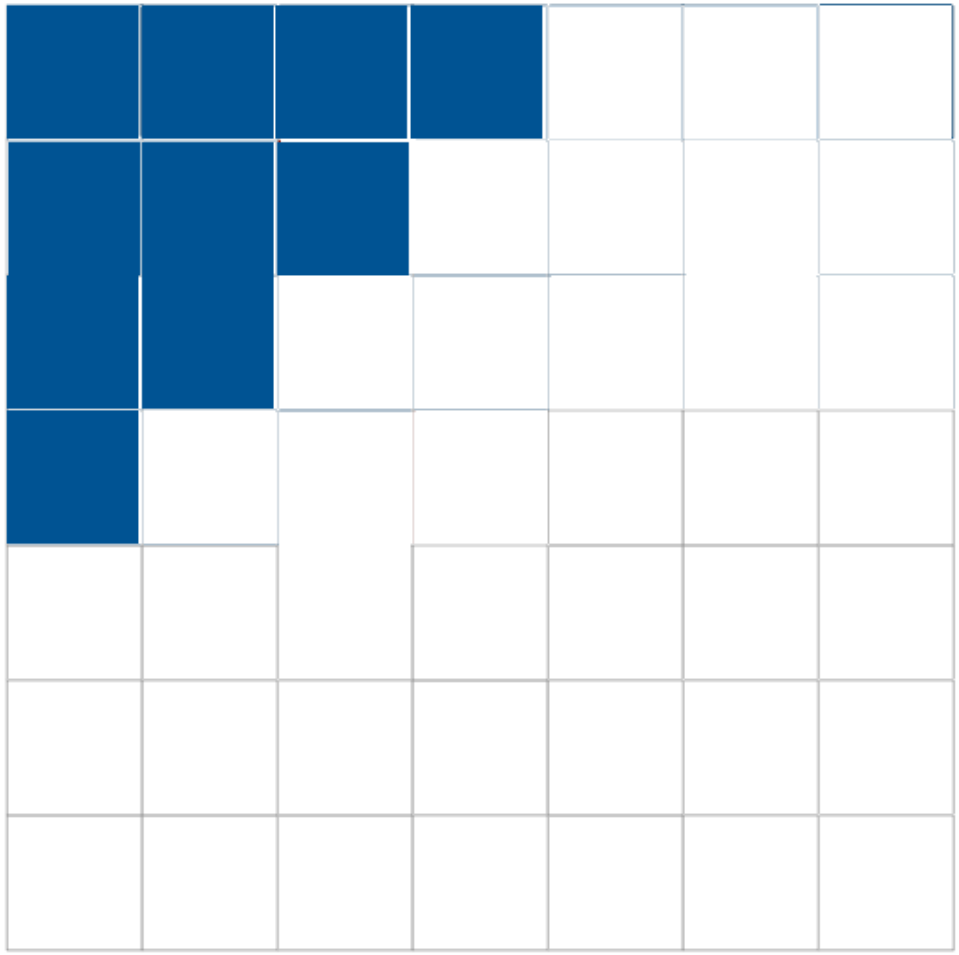
Does not use all available context

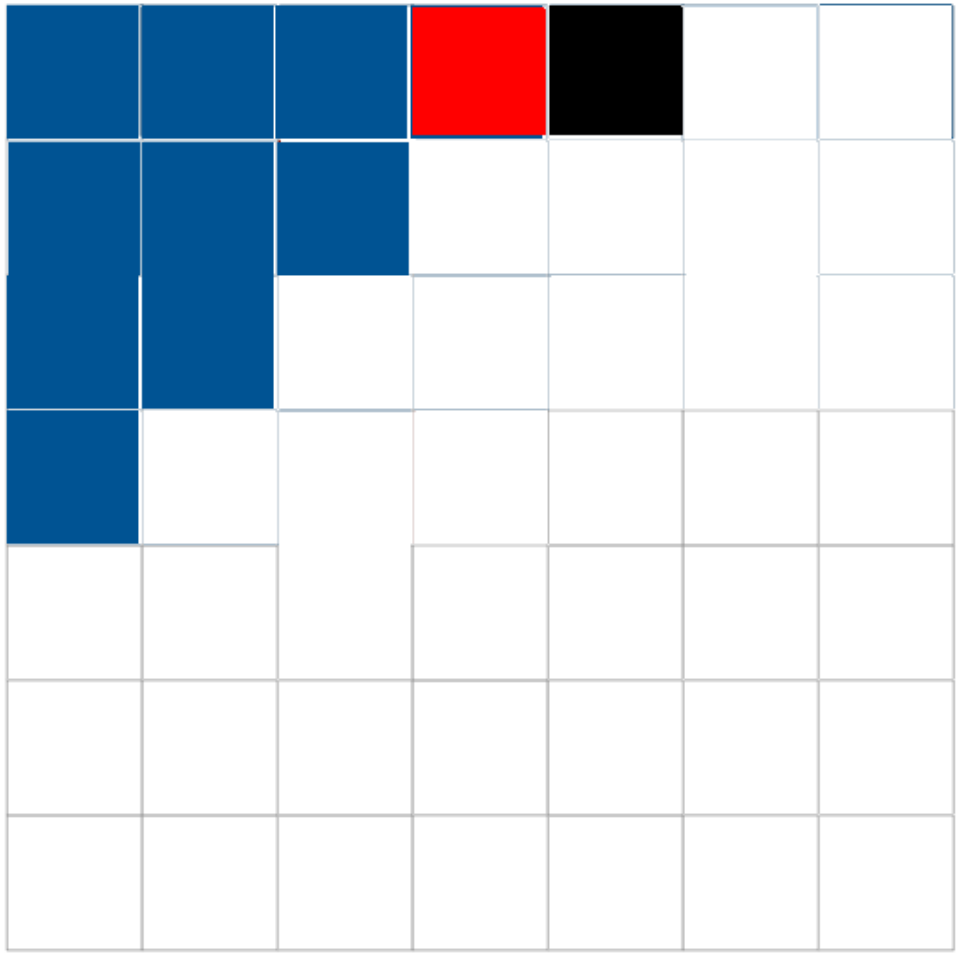
Diagonal BiLSTM

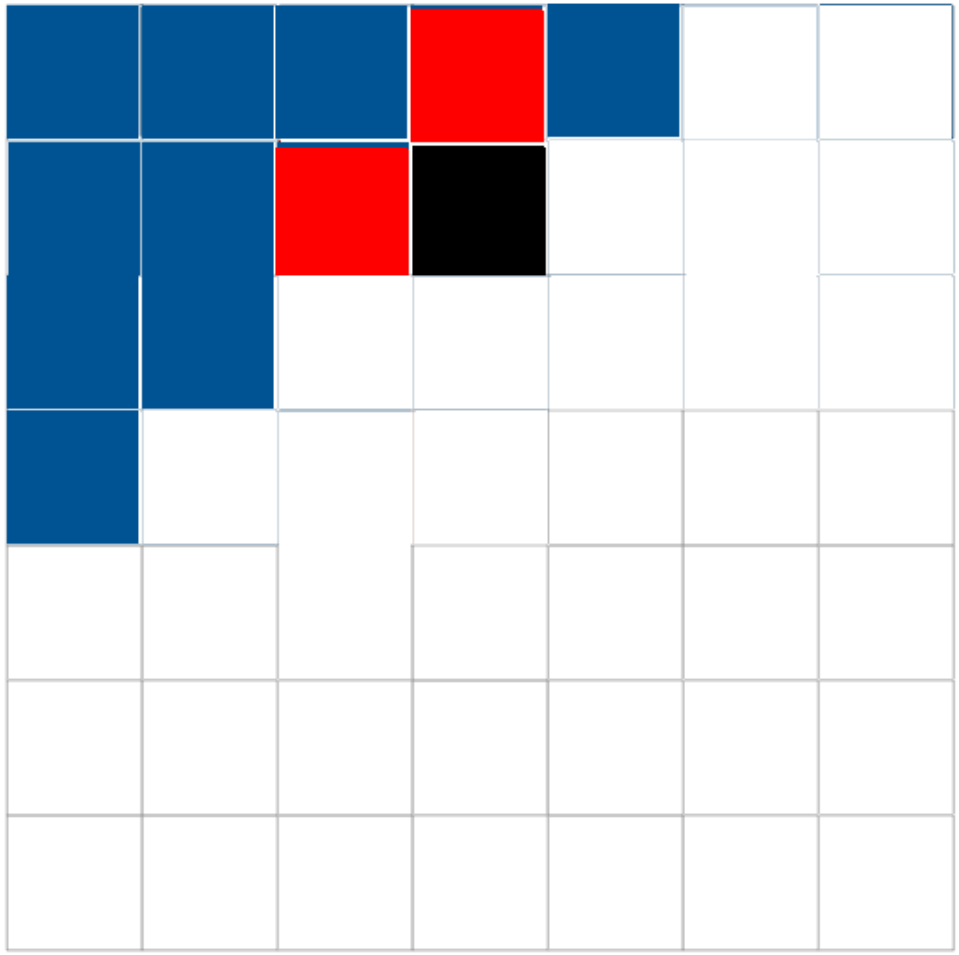


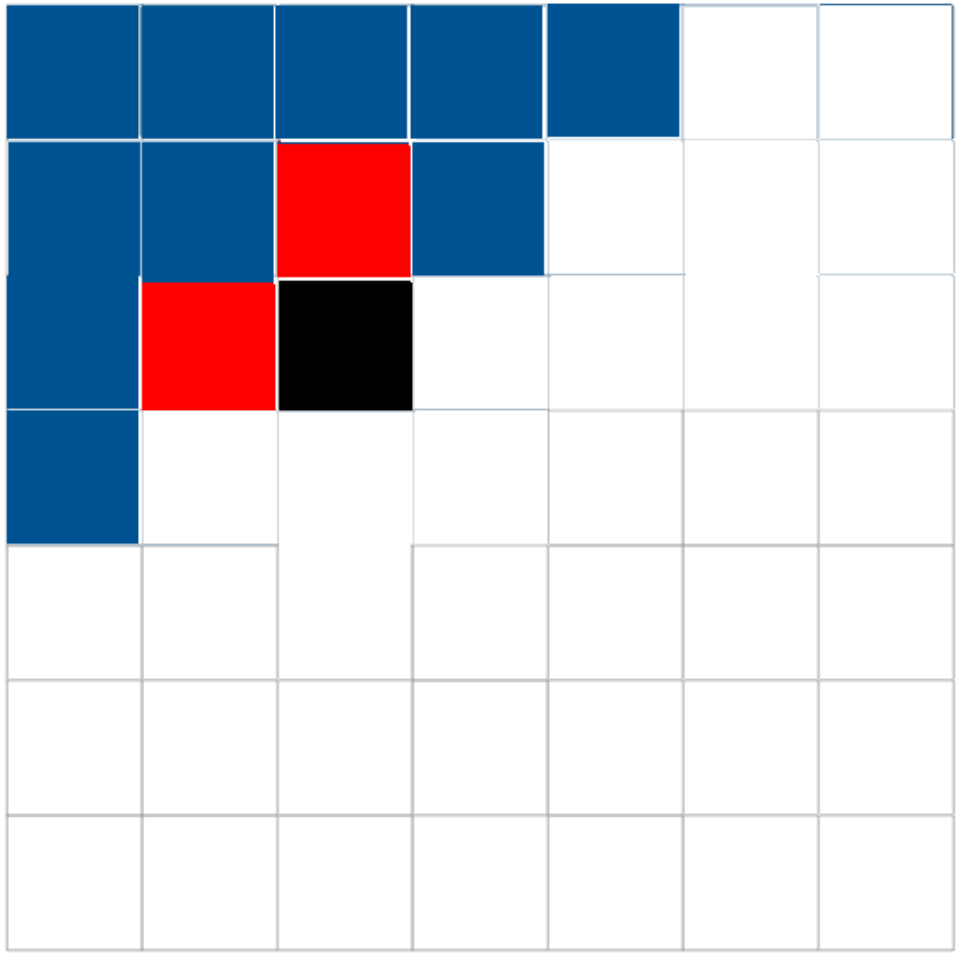


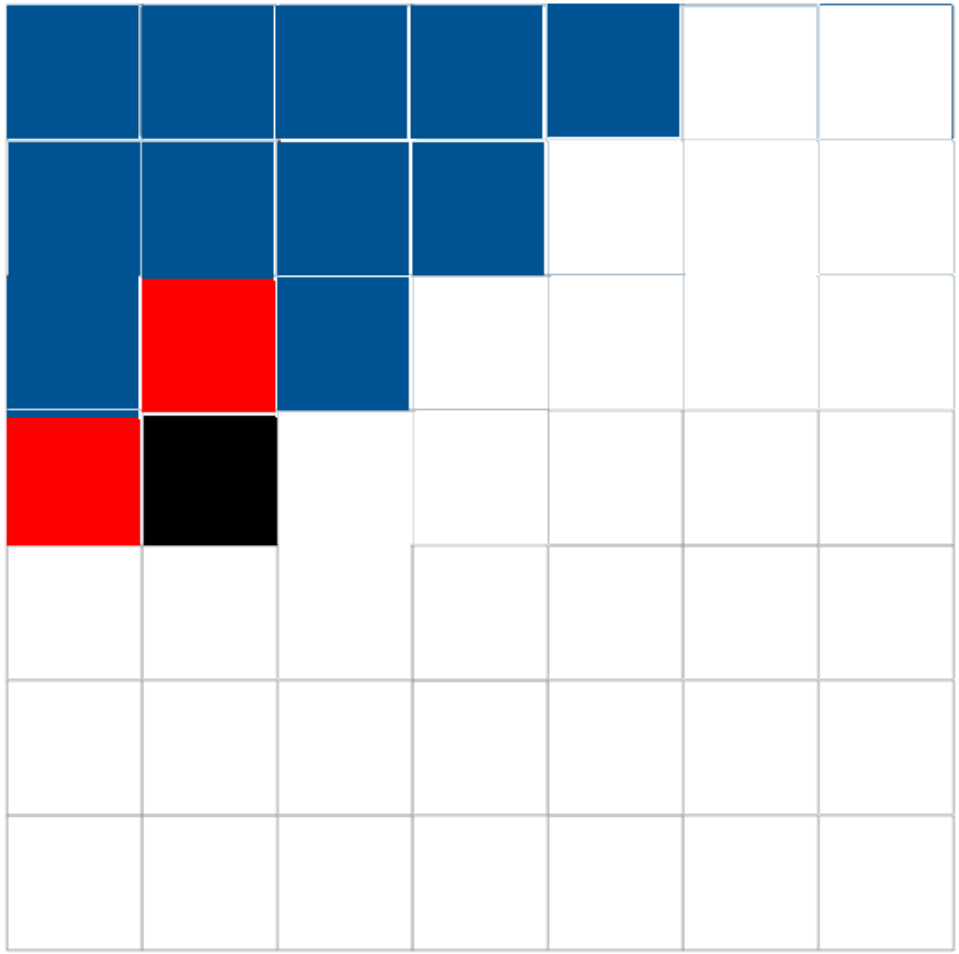


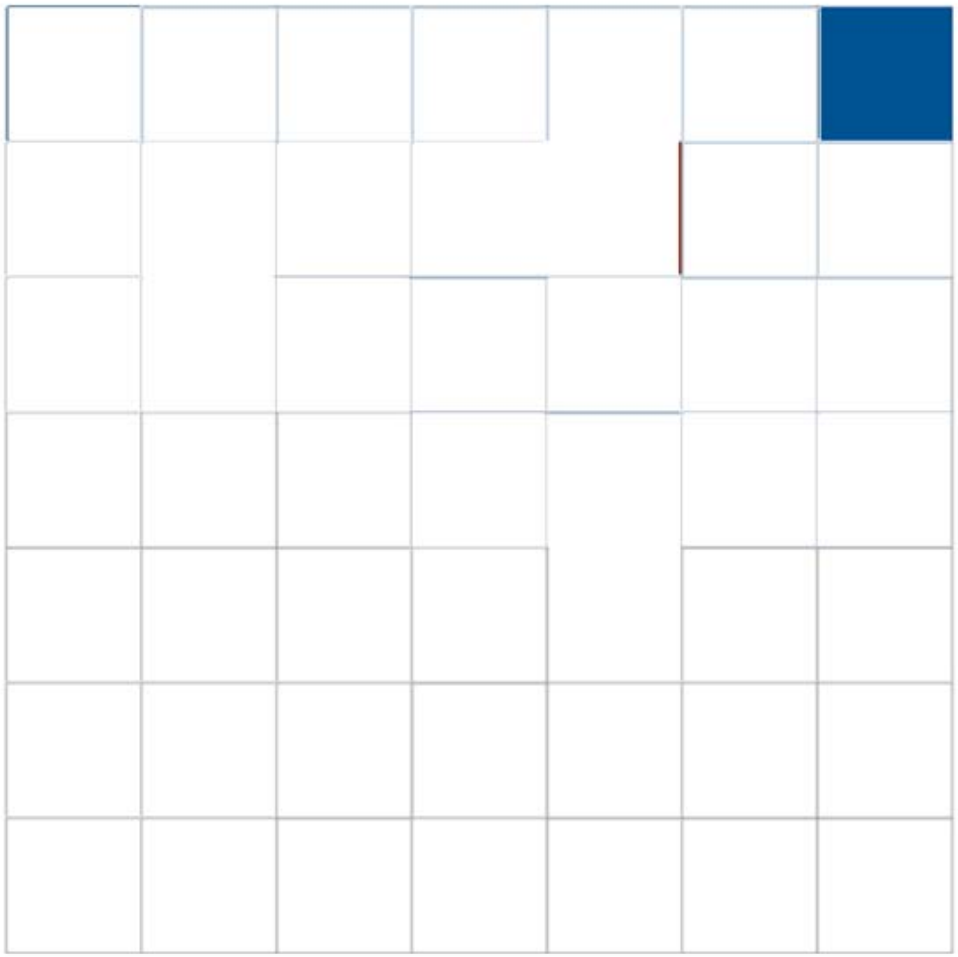


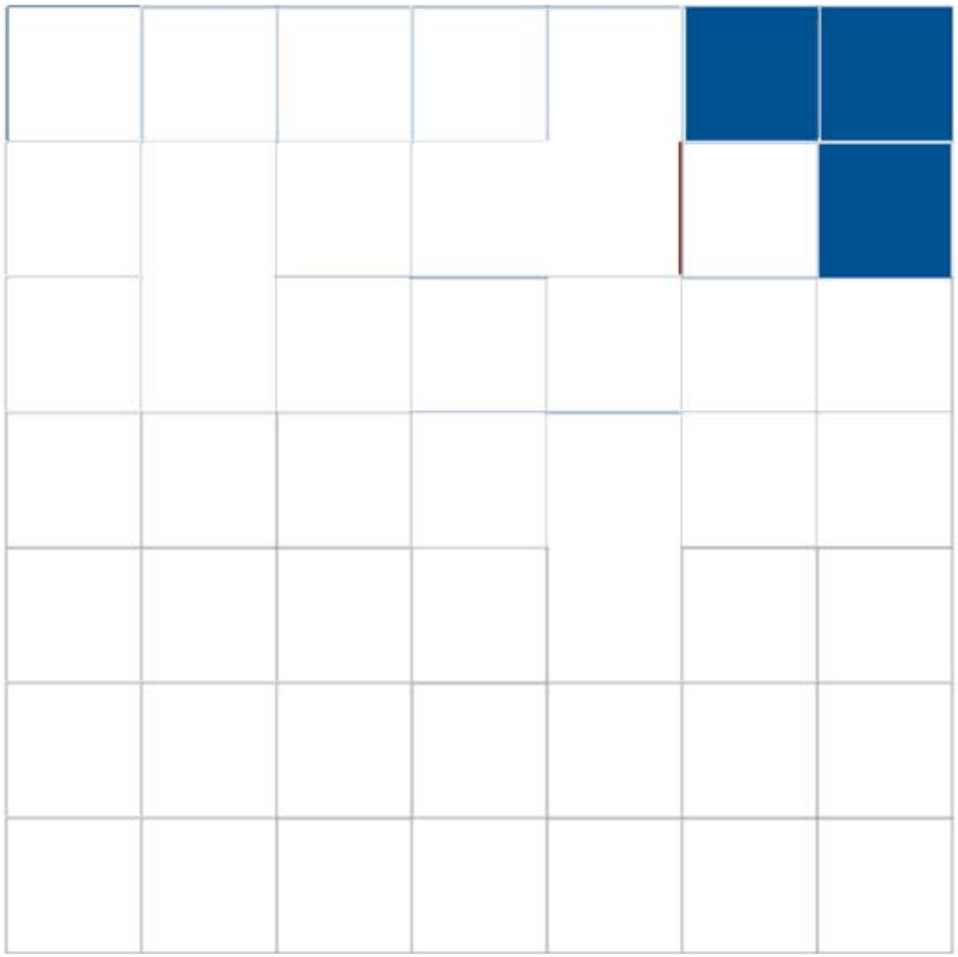


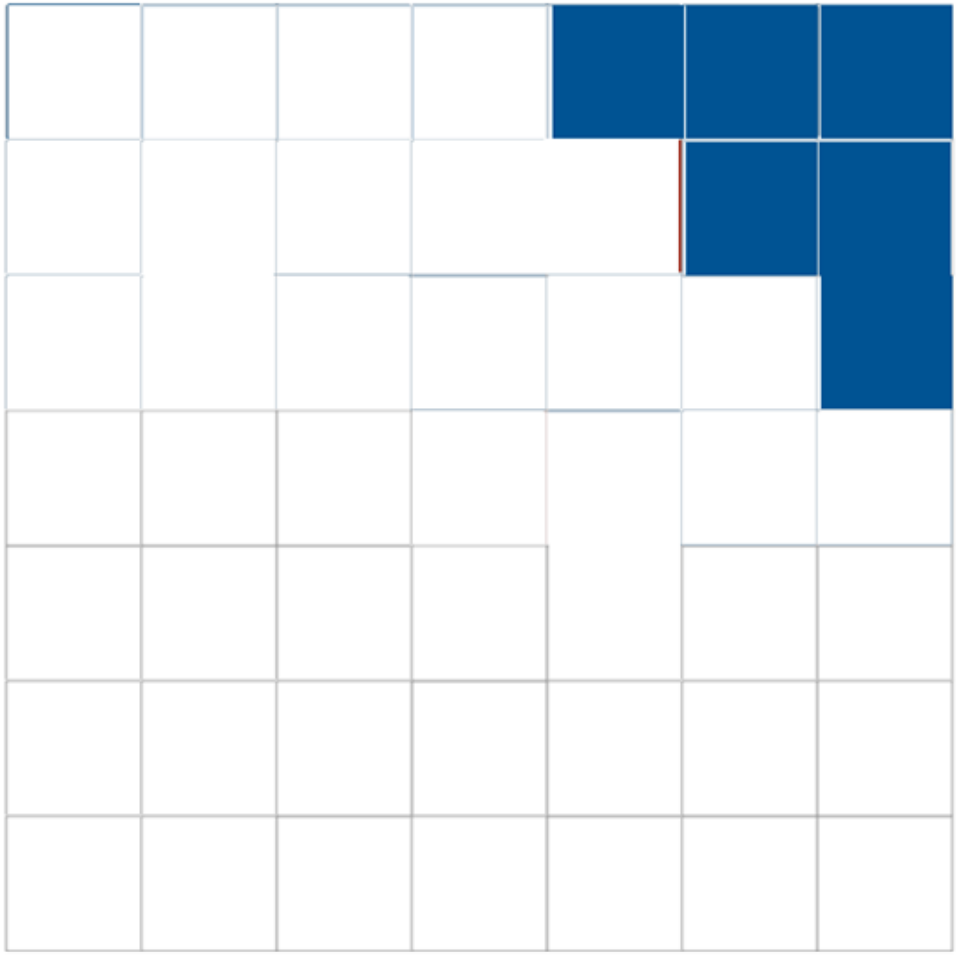


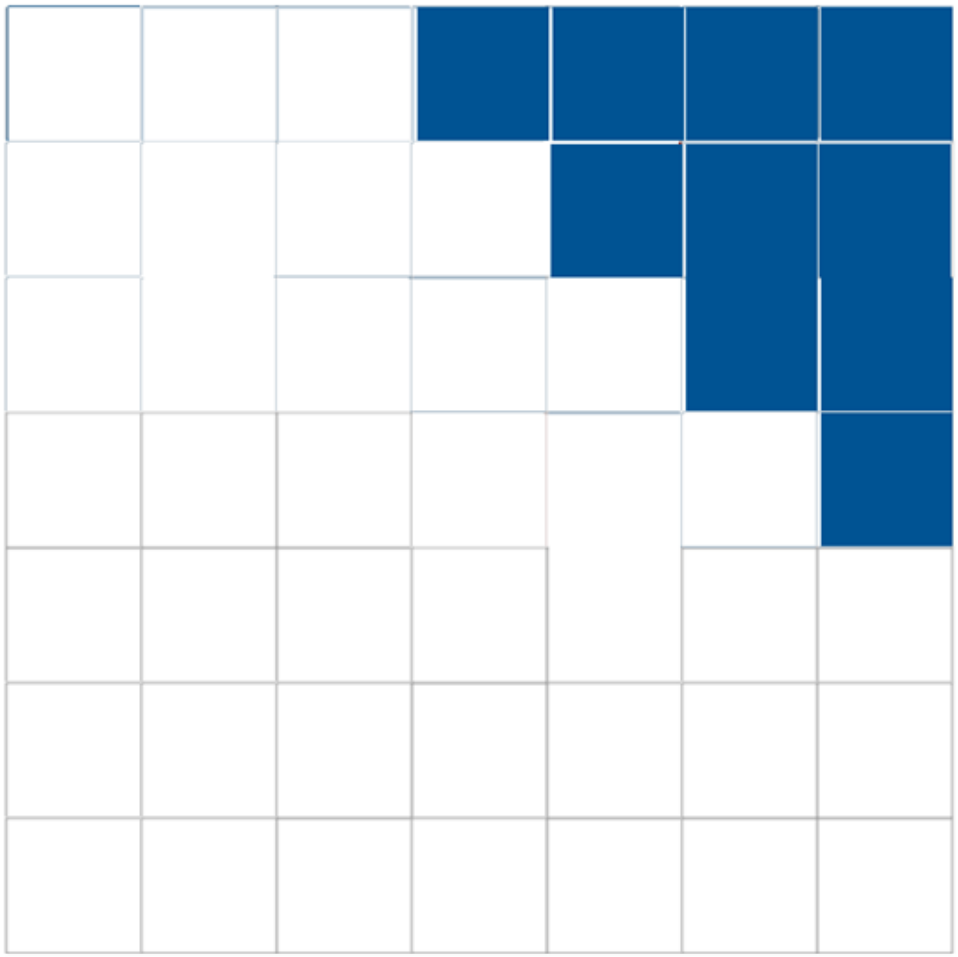




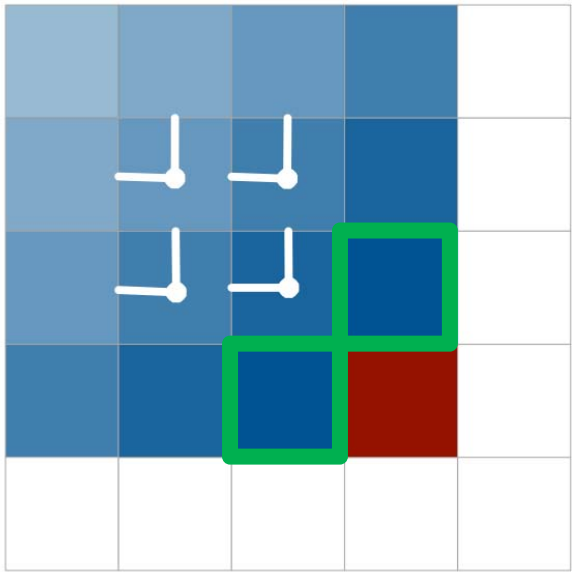




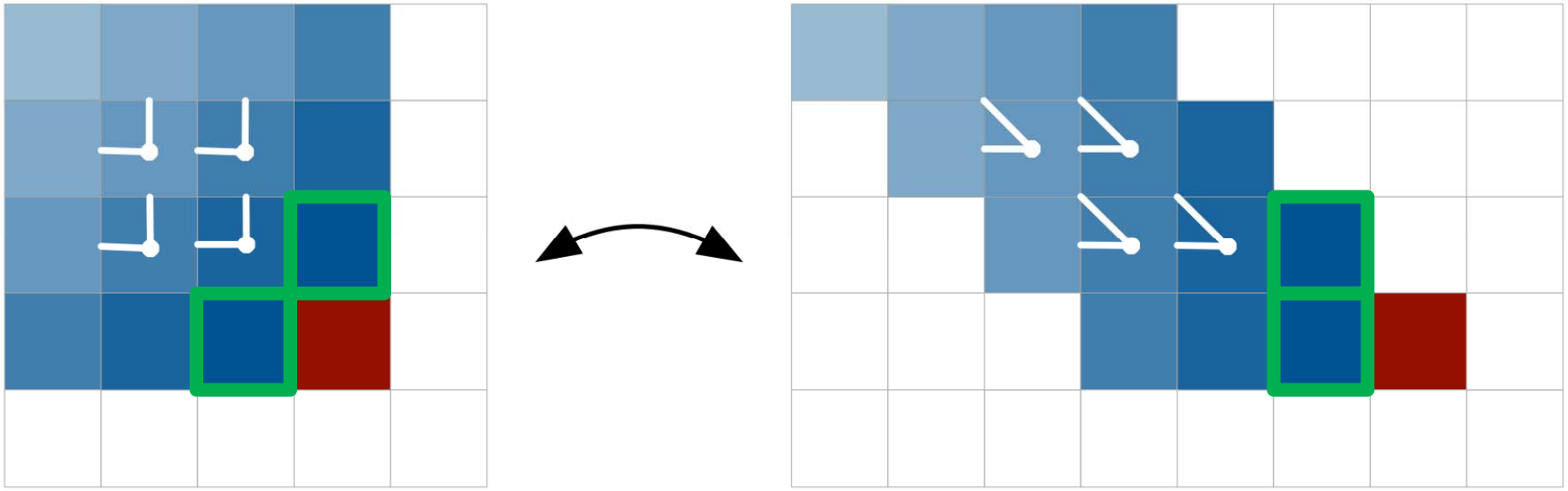




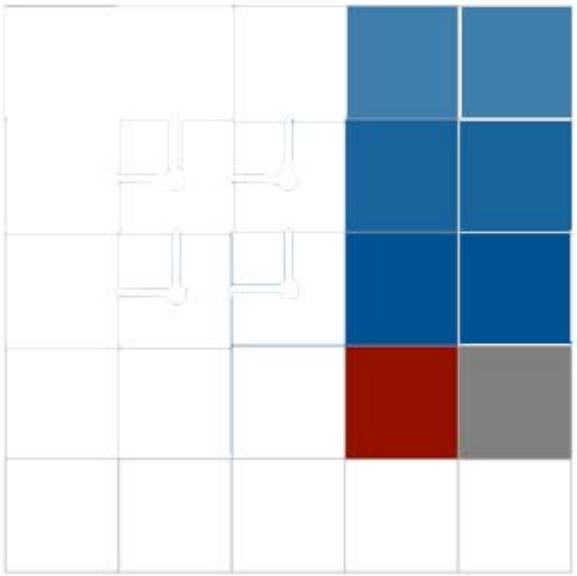
Skew



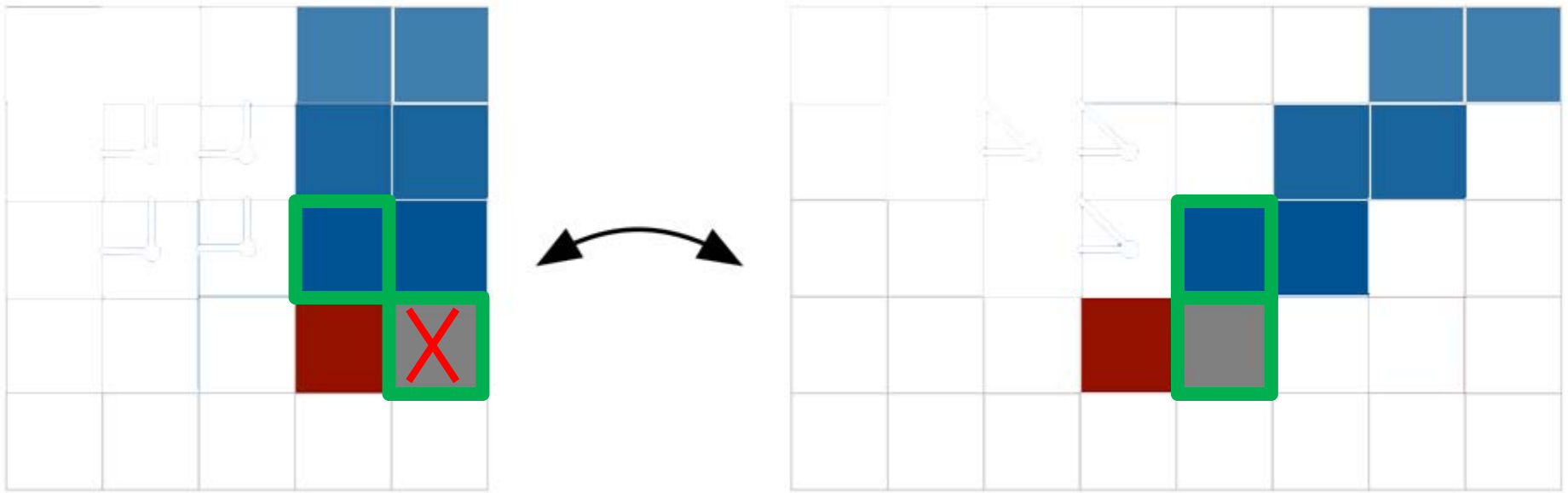
Skew



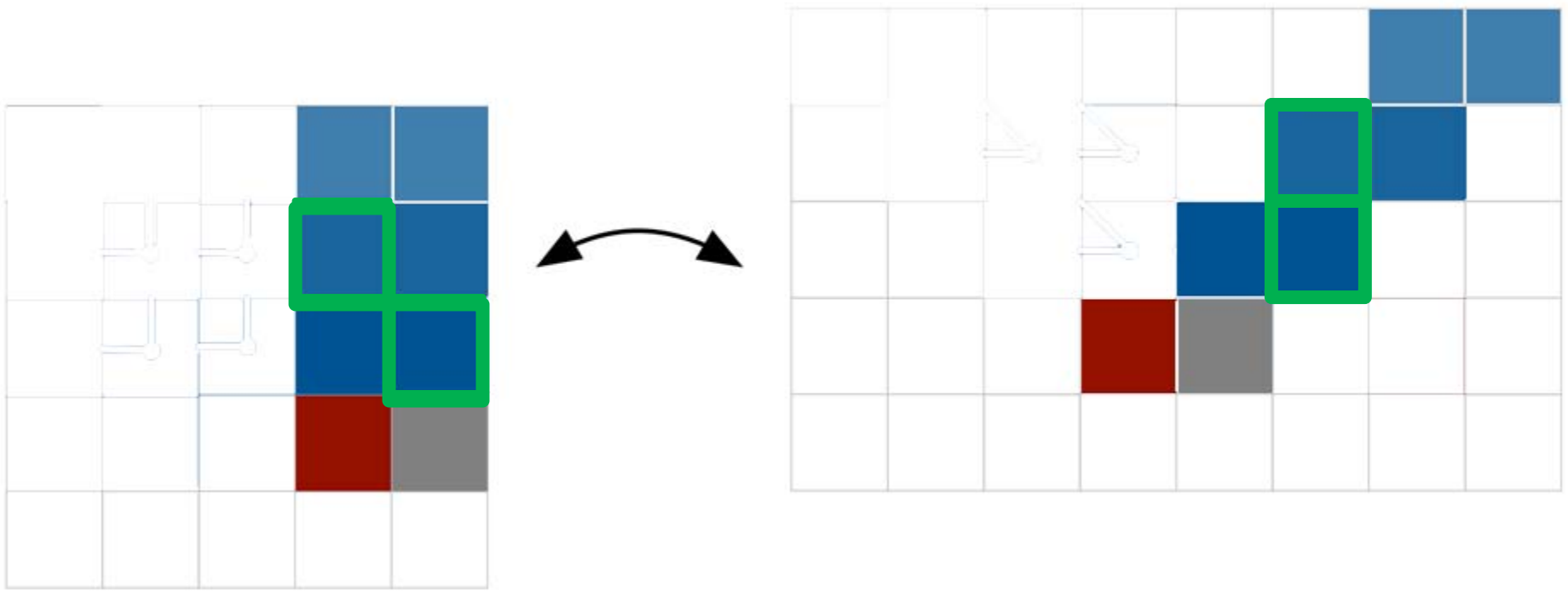
Skew



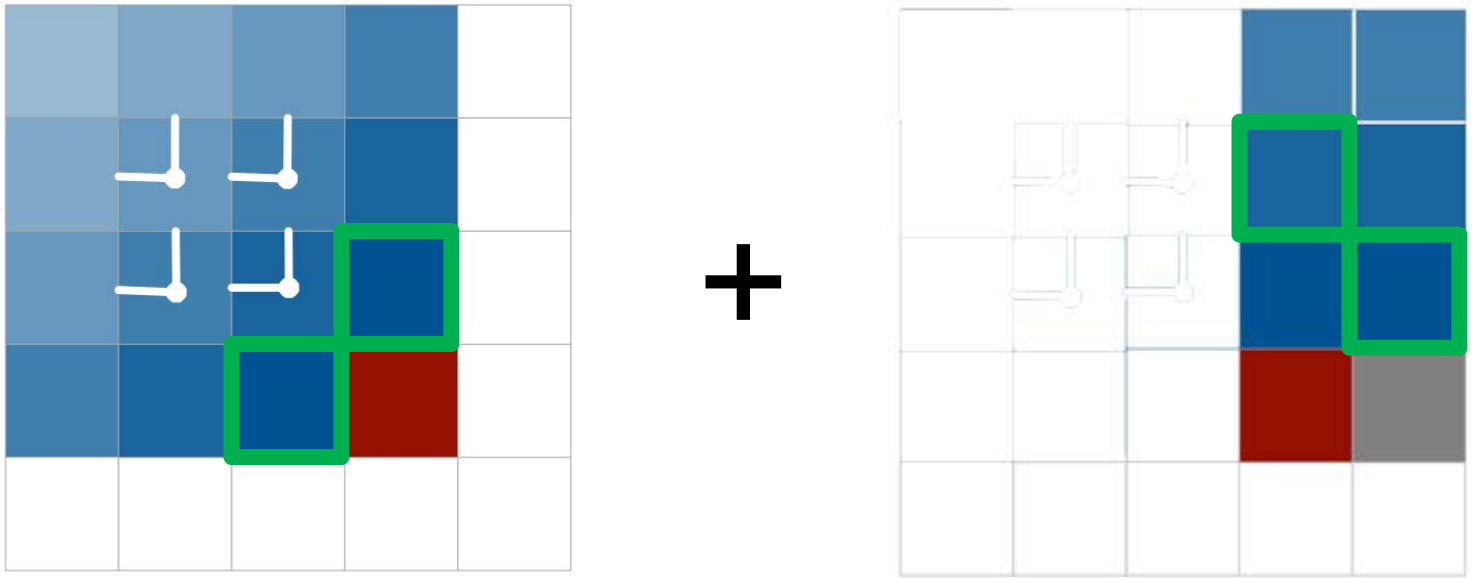
Skew



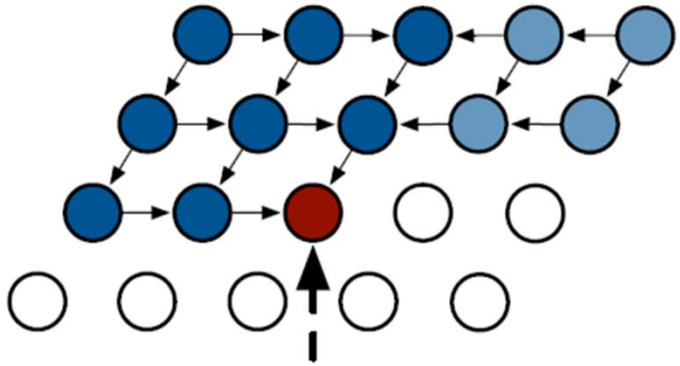
Skew



Finished State-to-State Component



Advantages



Compute the state for an entire diagonal at once

Global receptive field

Uses all available context

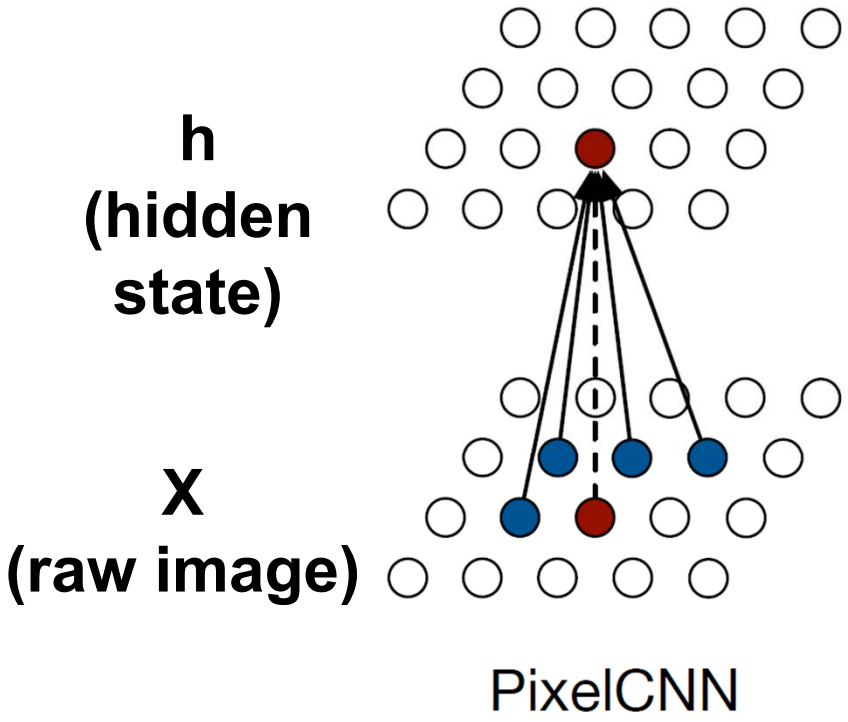
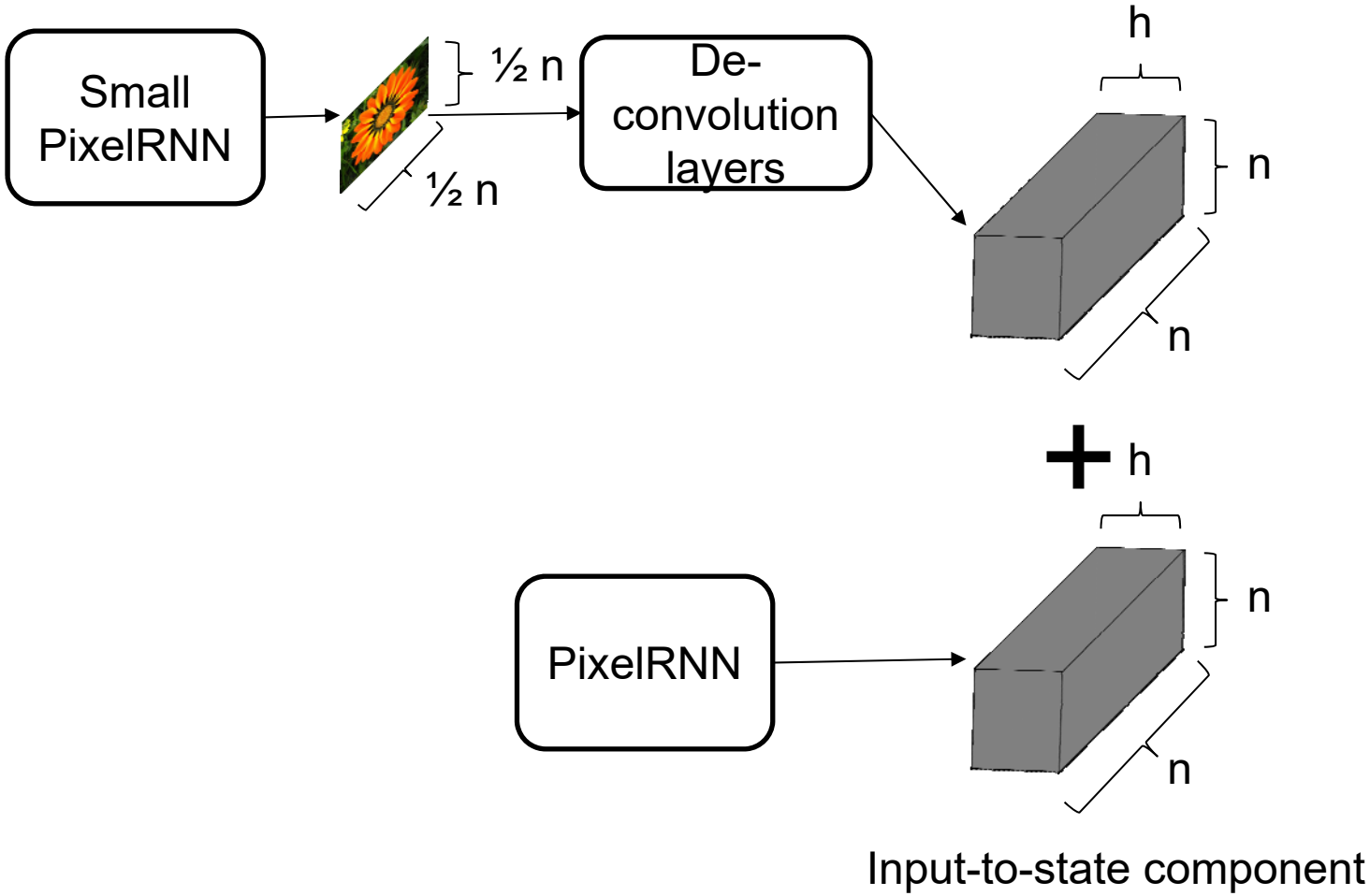


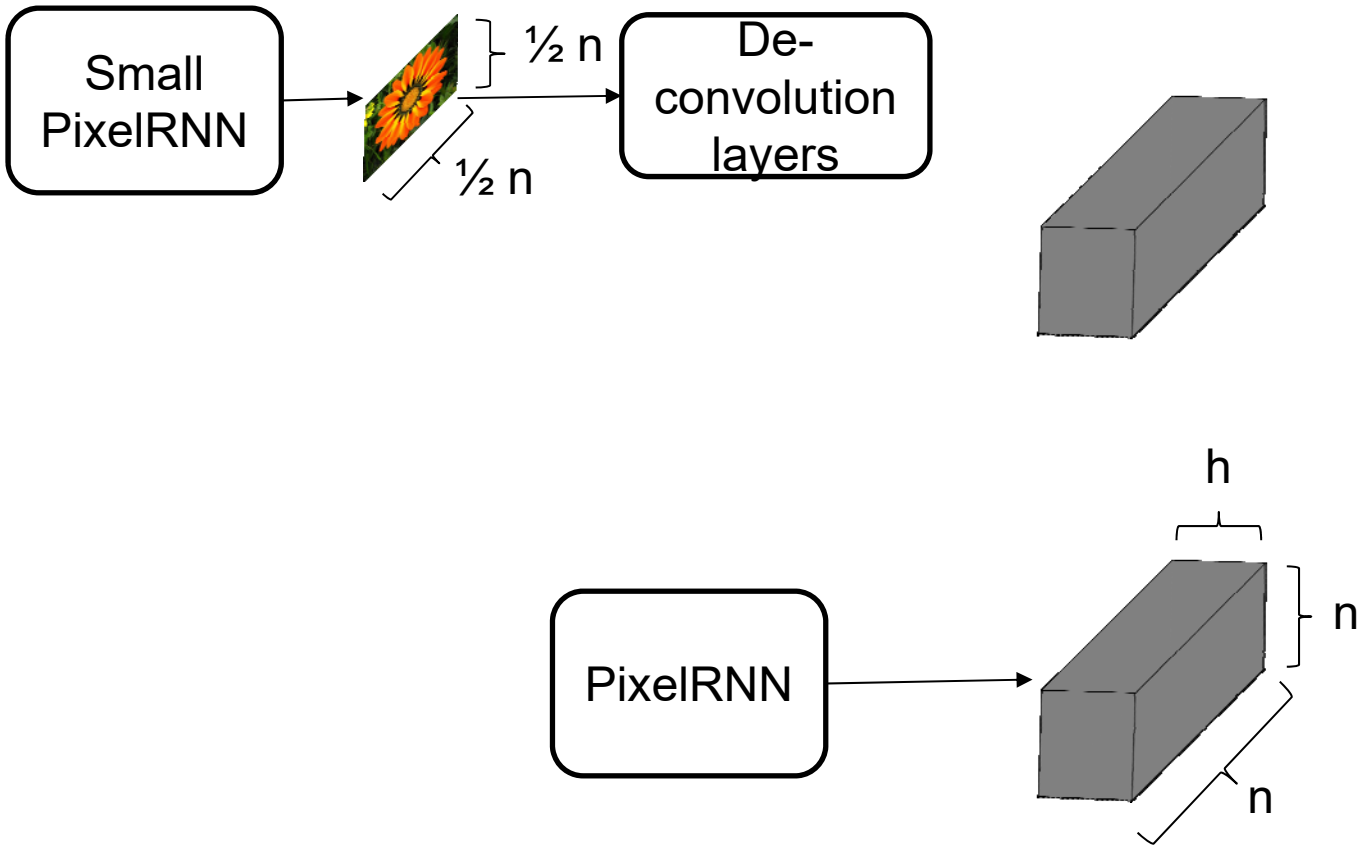
Figure 4. Visualization of the input-to-state and state-to-state mappings for the three proposed architectures.

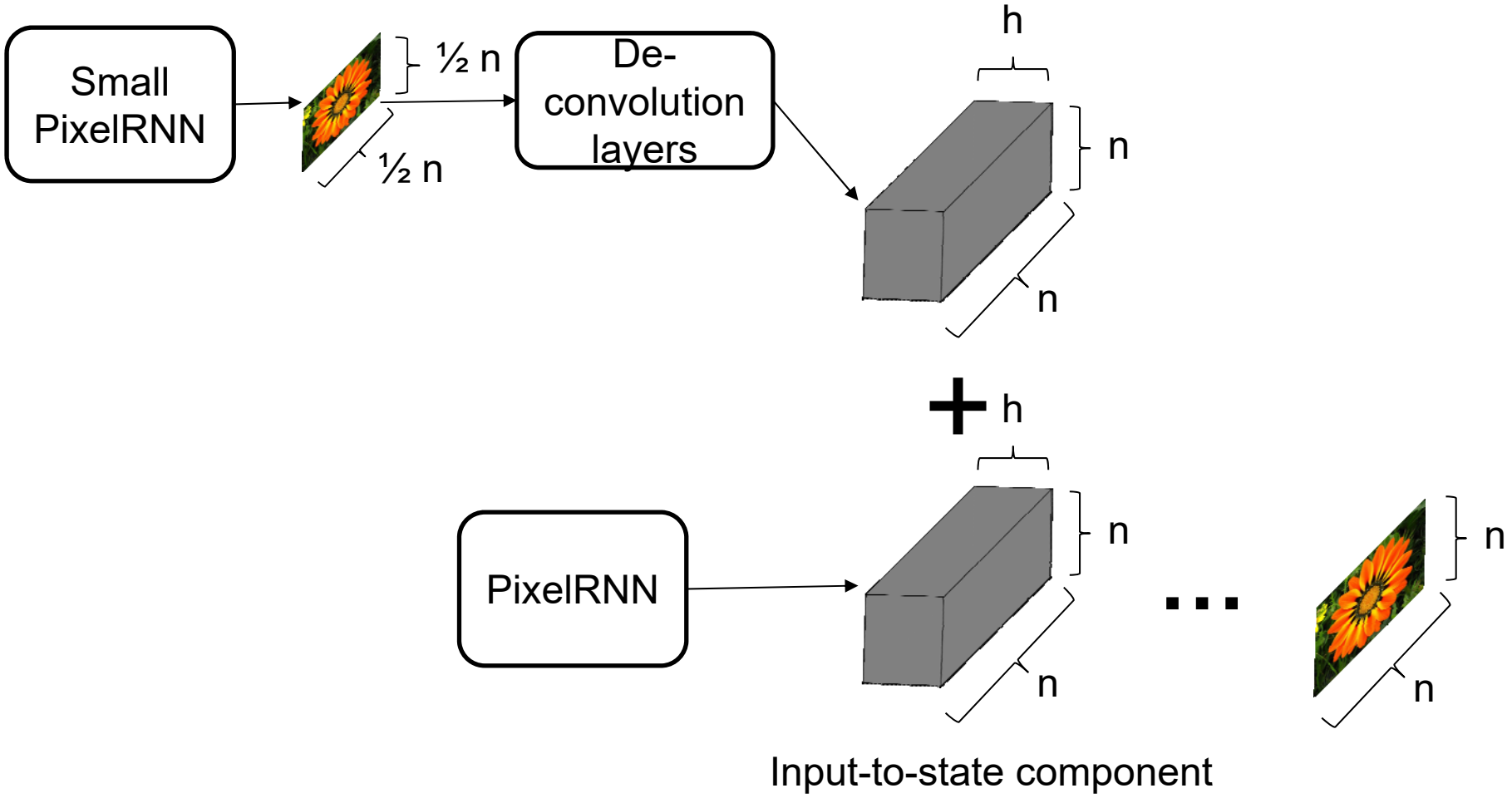
Model Architectures

PixelCNN		
Multiple residual blocks: (see fig 5)		

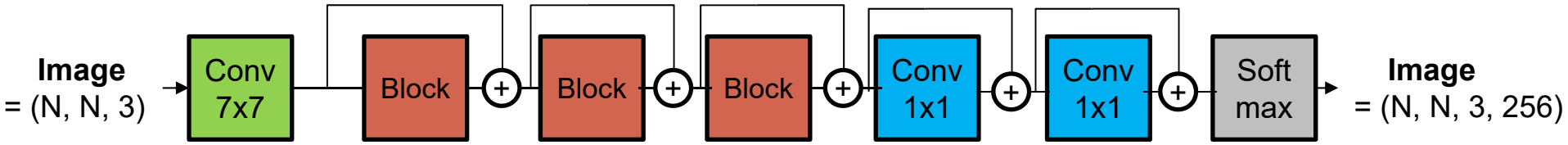
Multi-Scale PixelRNN


Small PixelRNN
½ n
½ n
De-convolution layers
h
n
n
+
h
PixelRNN
n
n
Input-to-state component

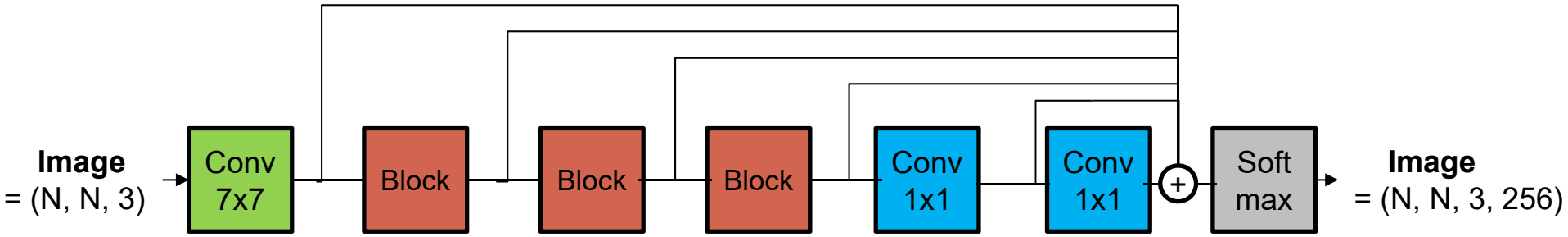

Small PixelRNN
½ n
½ n
De-convolution layers
h
n
n
PixelRNN
Input-to-state component


Small PixelRNN
½ n
½ n
De-convolution layers
h
n
n
+
h
PixelRNN
n
n
n
n
Input-to-state component

Residual Connections



Skip Connections



Results

Ablation Results

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

In bits/dim

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

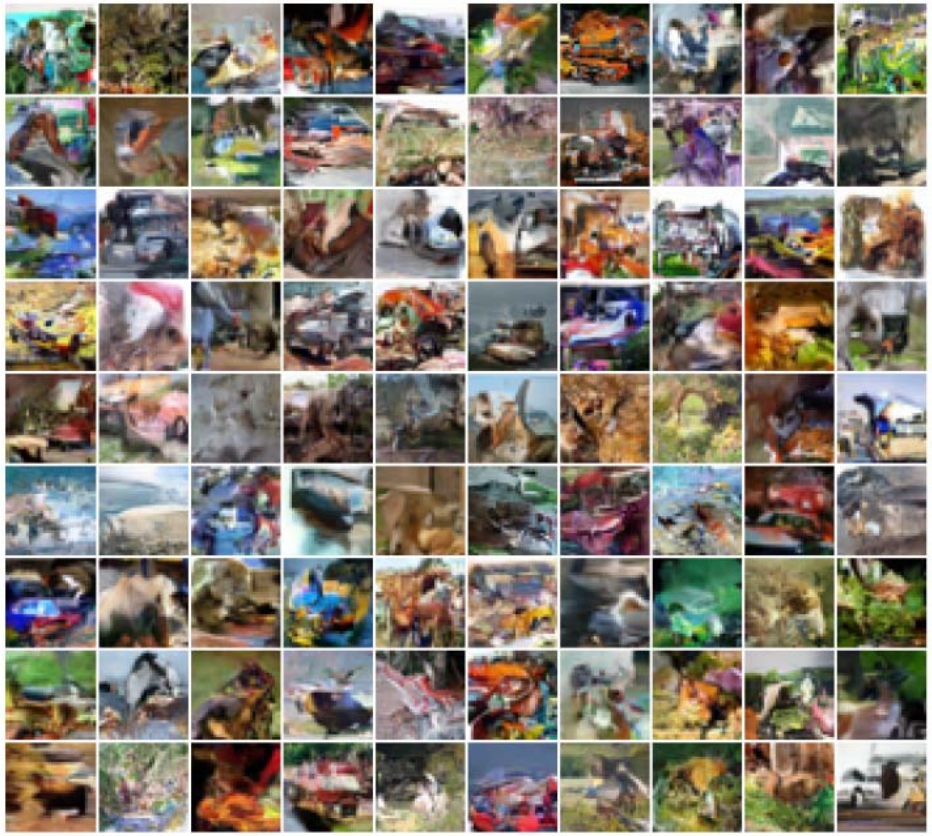
In bits/dim

CIFAR-10 Results

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)

In bits/dim

CIFAR-10

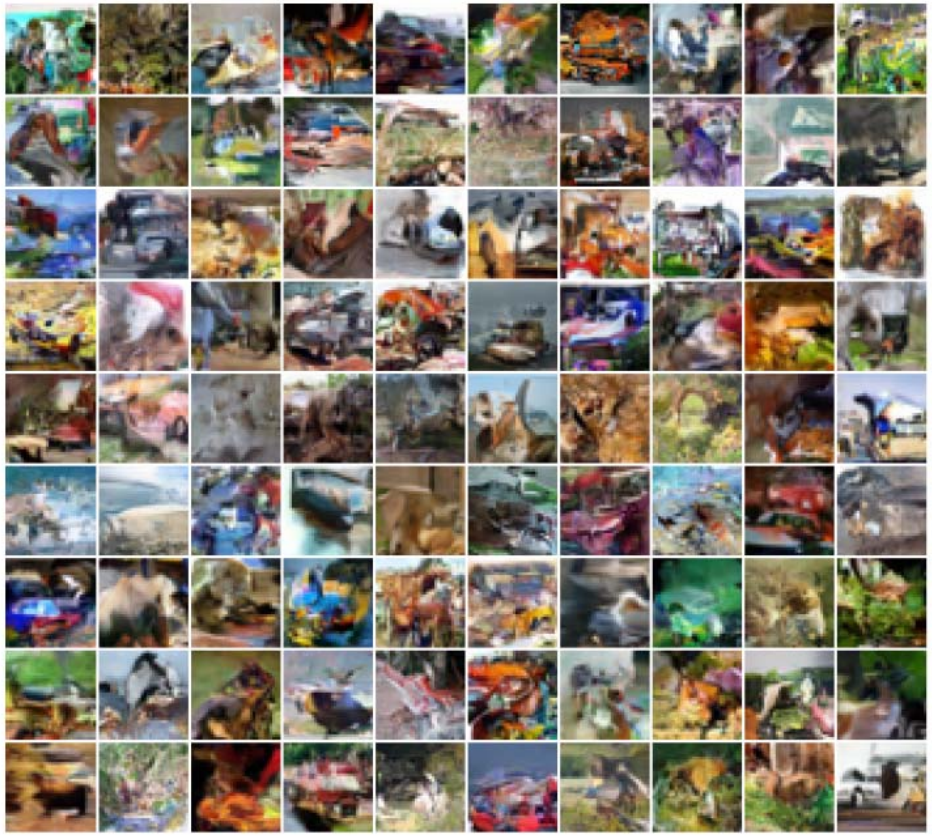


ImageNet Results

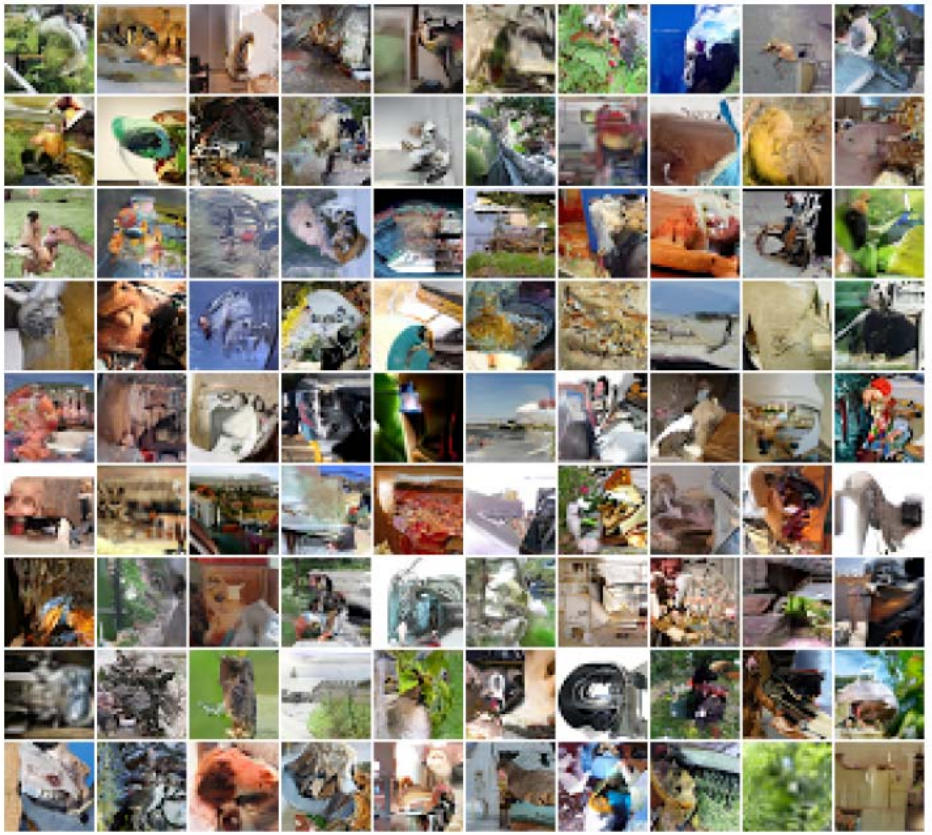
Image size	NLL Validation (Train)
32x32:	3.86 (3.83)
64x64:	3.63 (3.57)

In bits/dim

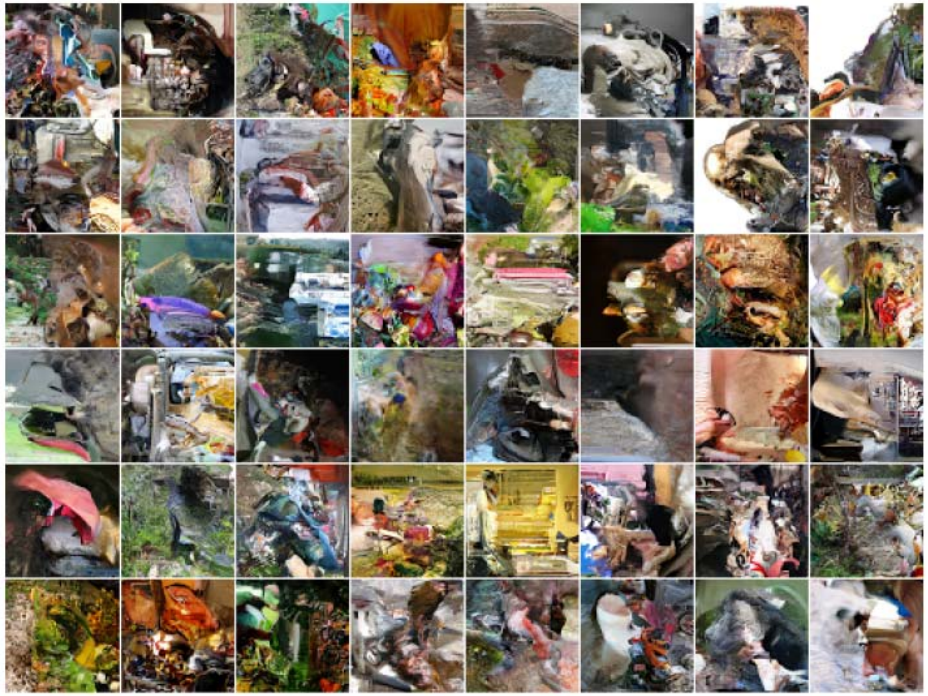
CIFAR-10



ImageNet



PixelRNN



Multi-Scale Pixel RNN

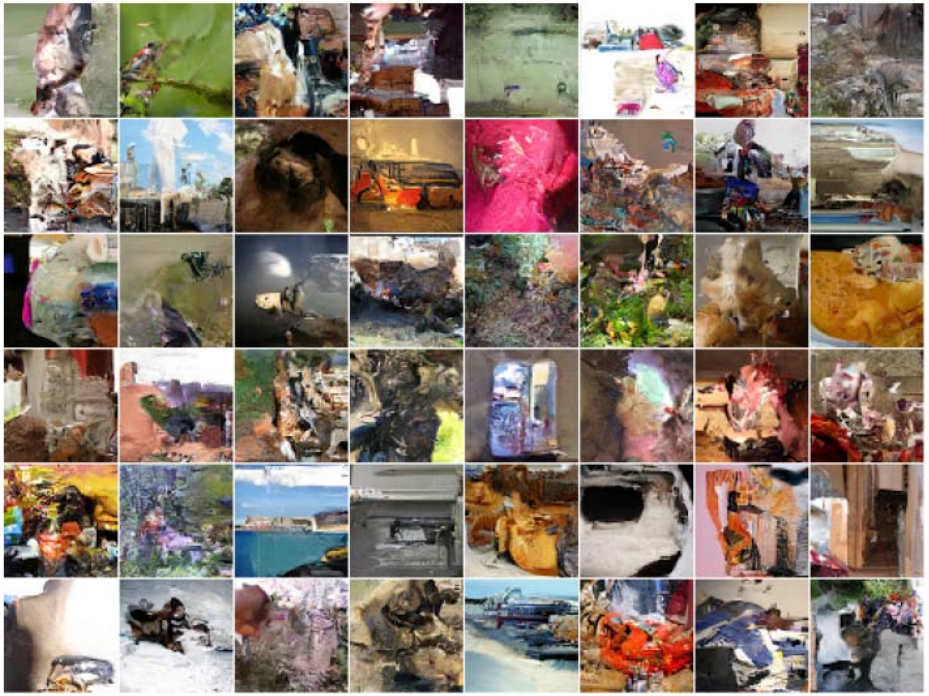
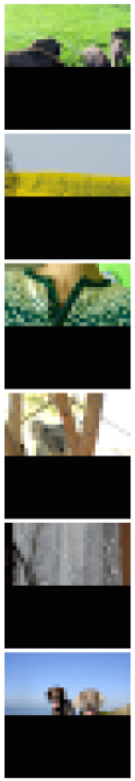
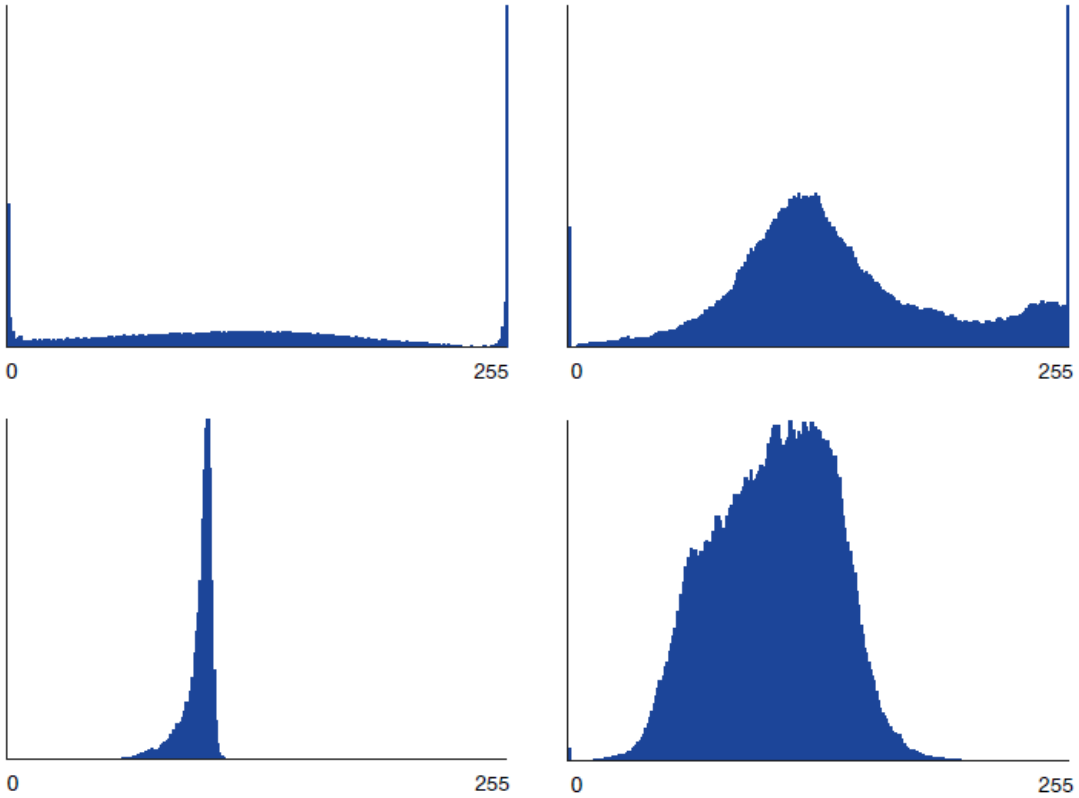


Image Completions

occluded completions original



Discrete Softmax Distribution



MNIST 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**

In nats

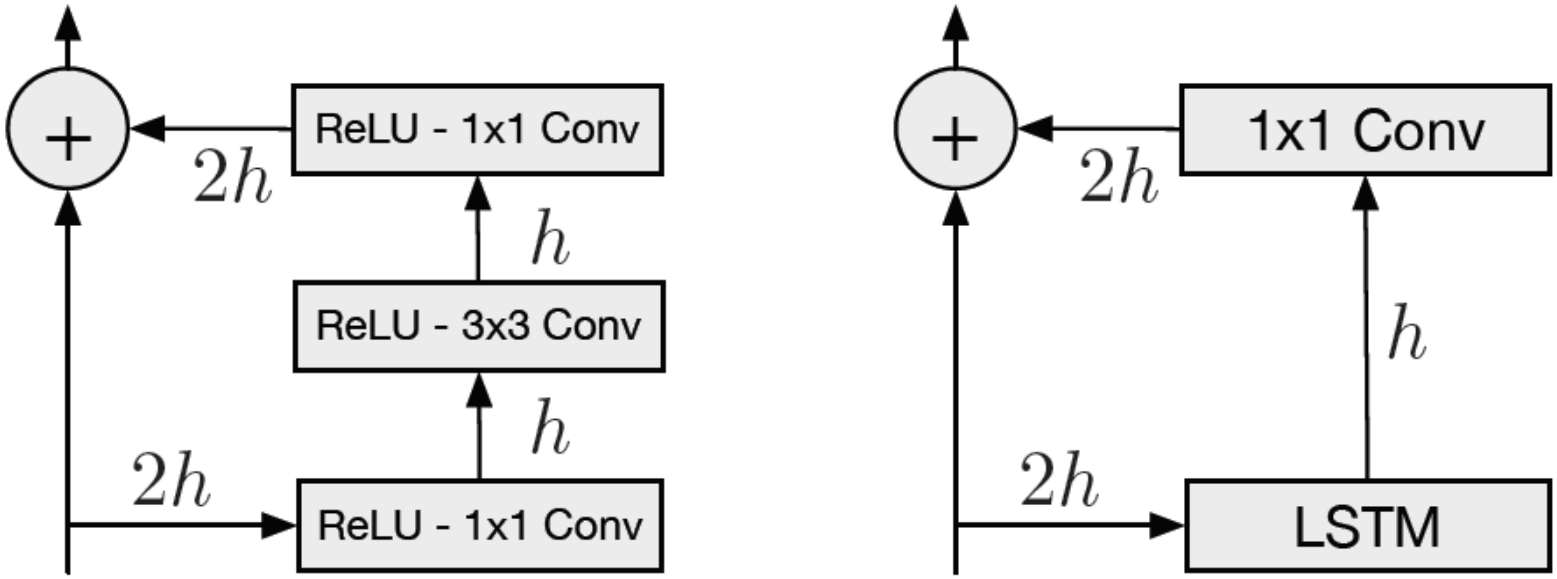
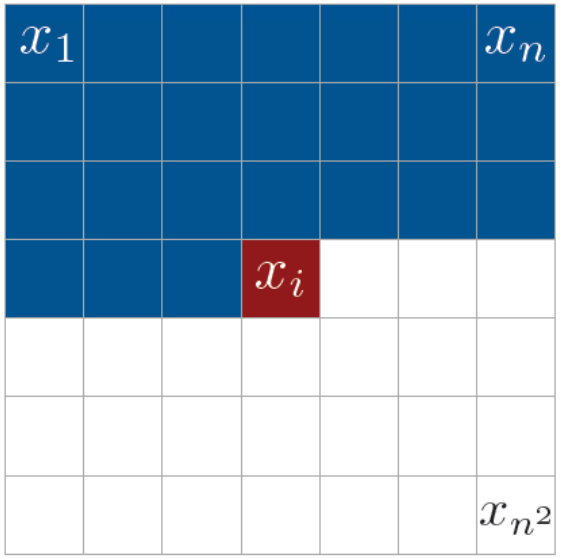
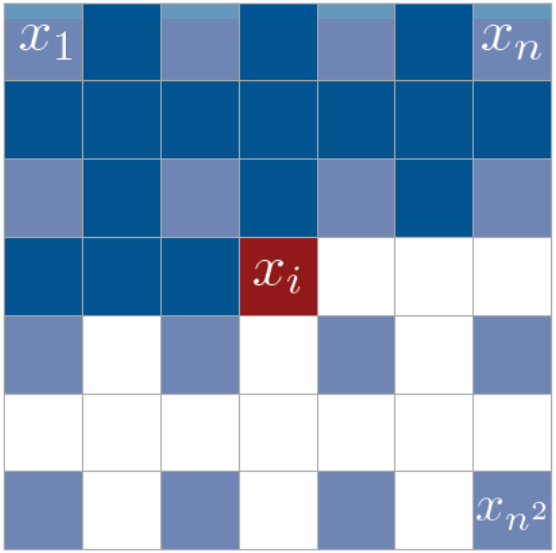


Figure 5. Residual blocks for a PixelCNN (left) and PixelRNNs.



Context



Multi-scale context

