Understand And Design Convolutional Neural Networks

Neural Networks Design And Application

LeNet-5 in 1999

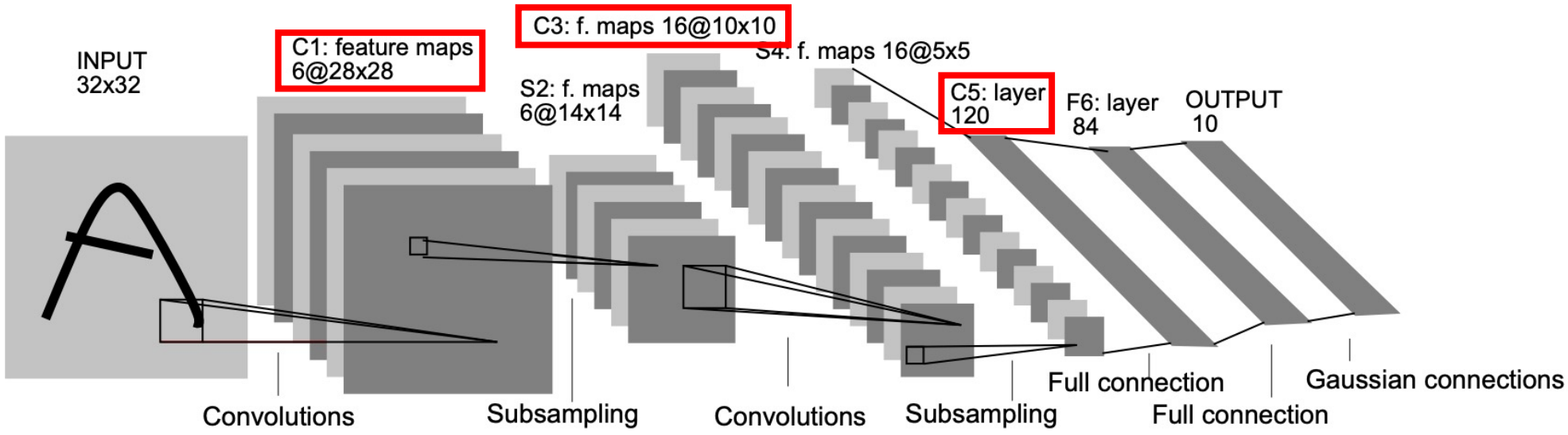


Fig. 1. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

LeCun, Yann, Patrick Haffner, Léon Bottou, and Yoshua Bengio. "Object recognition with gradient-based learning." In *Shape, contour and grouping in computer vision*, pp. 319-345. Springer, Berlin, Heidelberg, 1999.

LeNet-5 in 1999

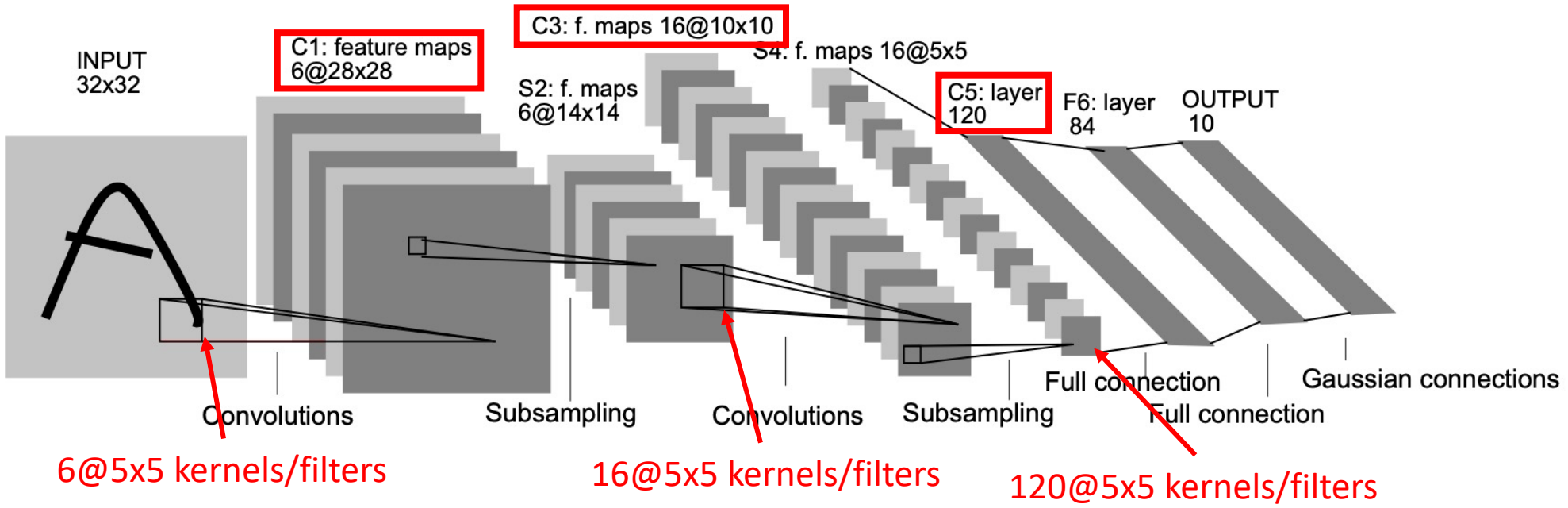


Fig. 1. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

LeCun, Yann, Patrick Haffner, Léon Bottou, and Yoshua Bengio. "Object recognition with gradient-based learning." In *Shape, contour and grouping in computer vision*, pp. 319-345. Springer, Berlin, Heidelberg, 1999.

LeNet-5 in 1999

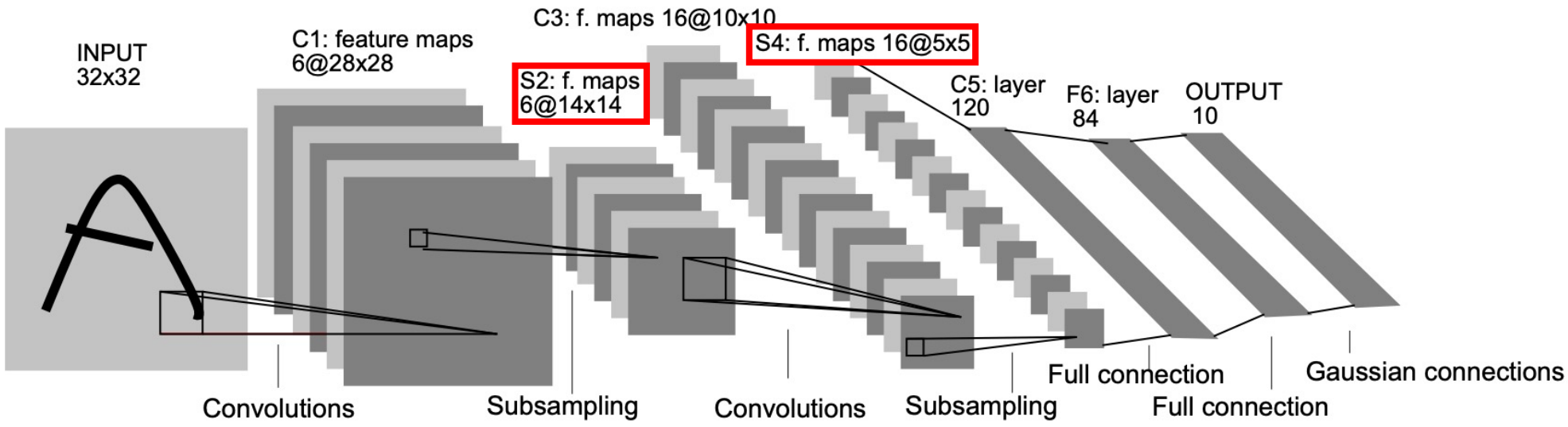


Fig. 1. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

LeCun, Yann, Patrick Haffner, Léon Bottou, and Yoshua Bengio. "Object recognition with gradient-based learning." In *Shape, contour and grouping in computer vision*, pp. 319-345. Springer, Berlin, Heidelberg, 1999.

LeNet-5 in 1999

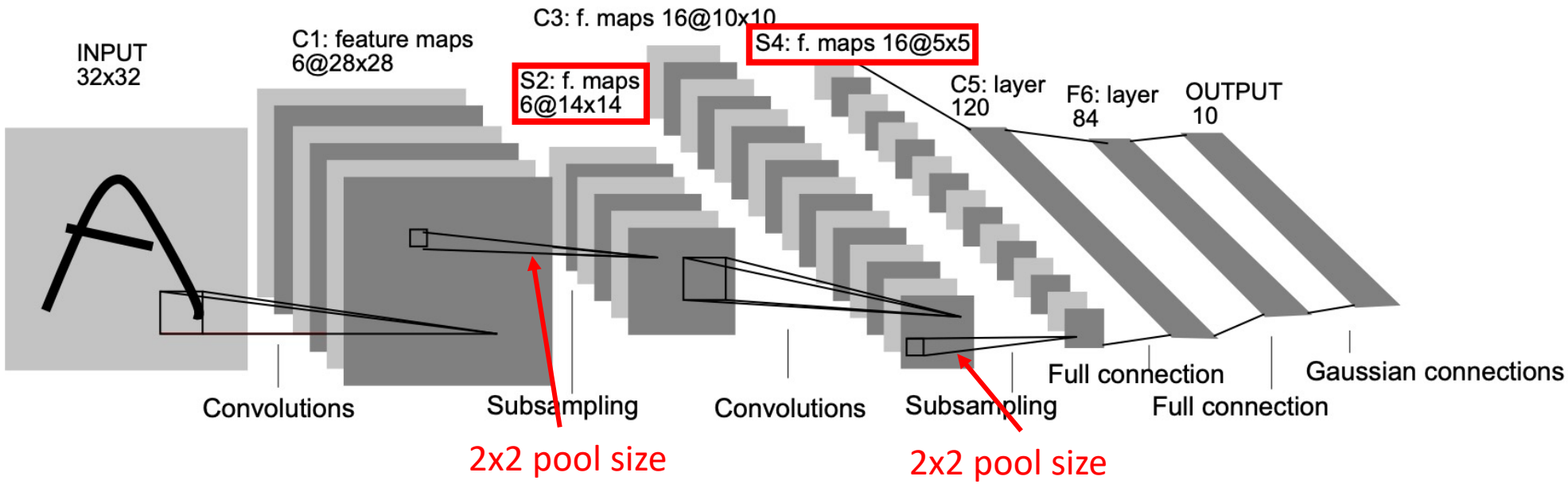
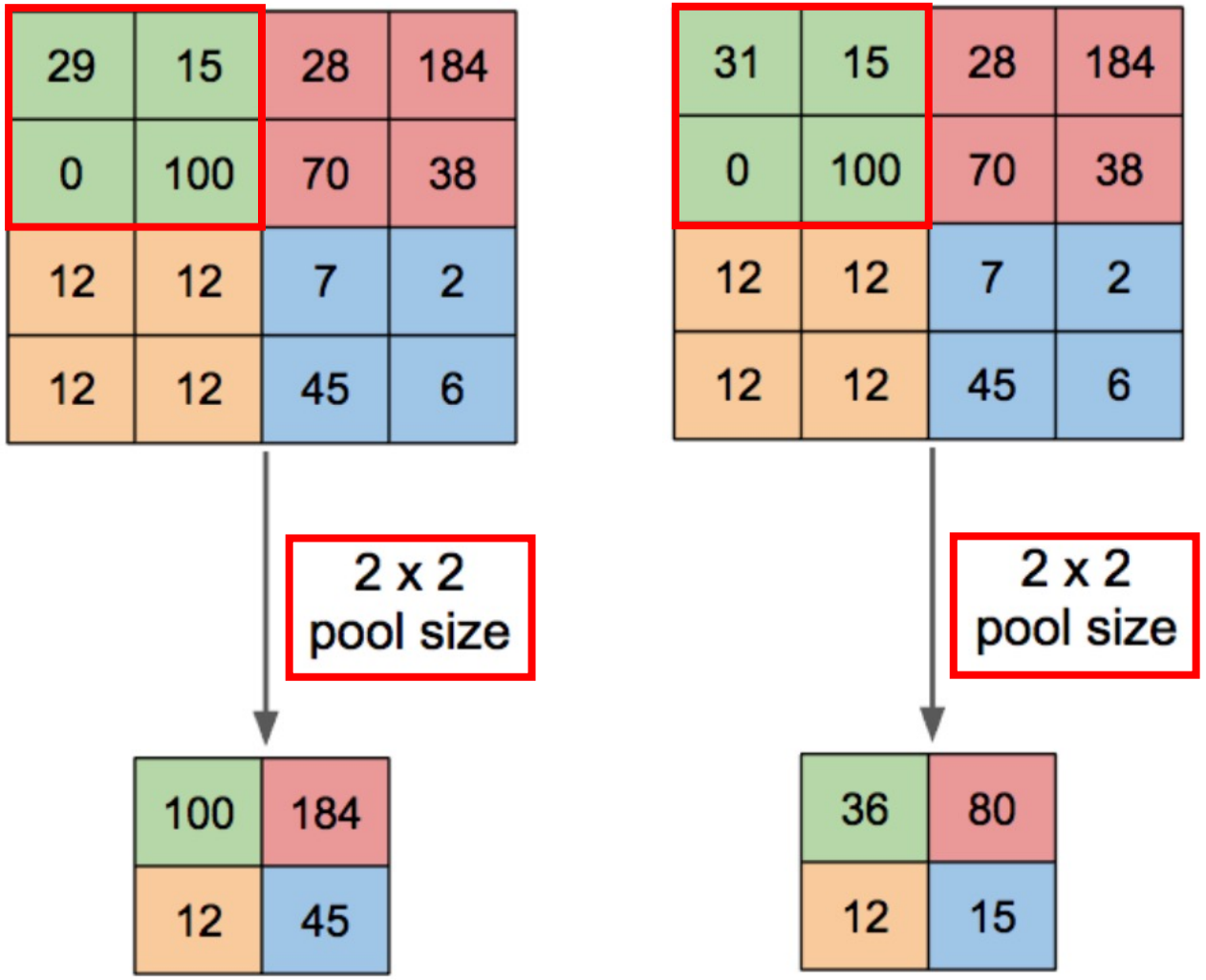


Fig. 1. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

LeCun, Yann, Patrick Haffner, Léon Bottou, and Yoshua Bengio. "Object recognition with gradient-based learning." In *Shape, contour and grouping in computer vision*, pp. 319-345. Springer, Berlin, Heidelberg, 1999.

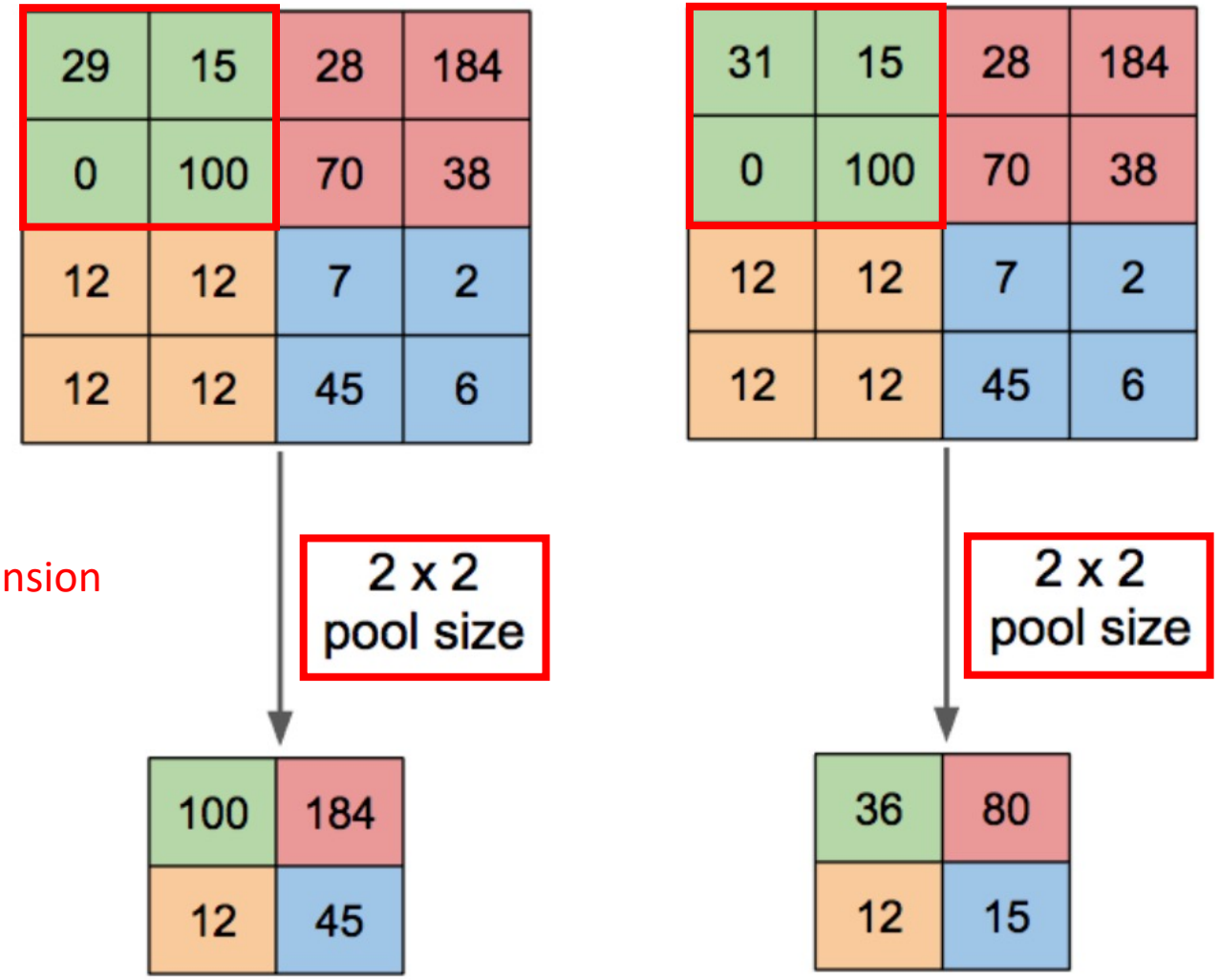
Subsampling operations

- Max pooling
- Average pooling



Subsampling operations

- Max pooling
- Average pooling

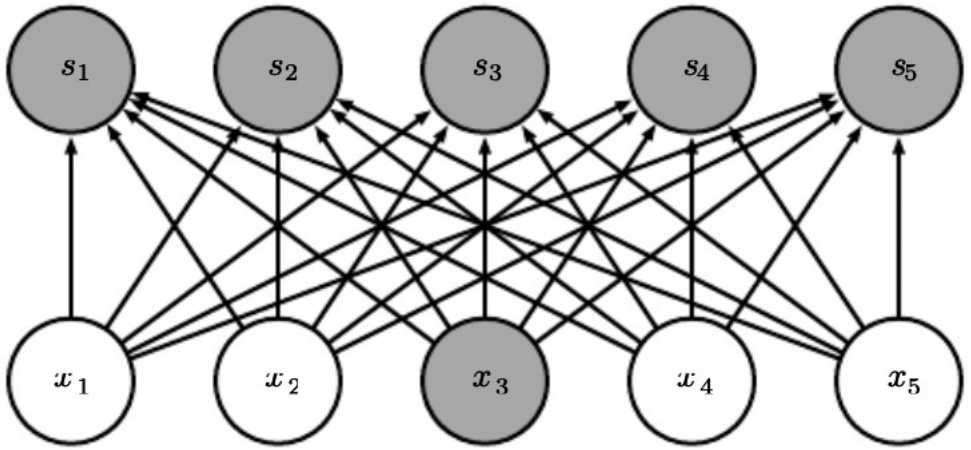


Halve the dimension

Difference between ConvNet and MLP

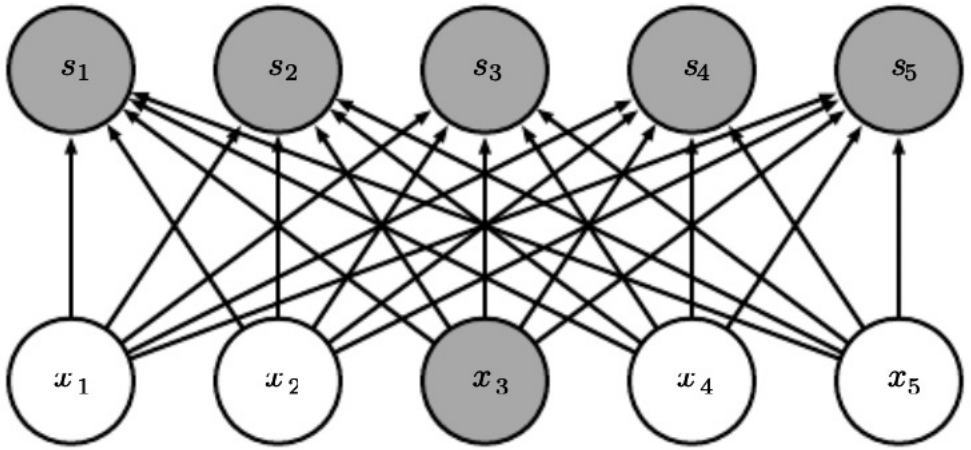
- Sparse connectivity
- Parameter sharing
- Equivariant representations

Sparse connectivity of convolution



Feedforward network (fully connected layer)

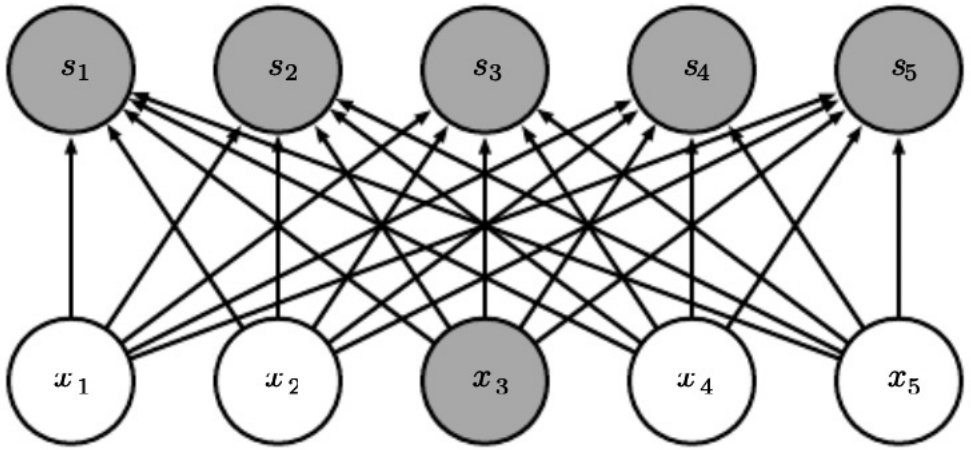
Sparse connectivity of convolution



Feedforward network (fully connected layer)

Q: how many arrows we have?

Sparse connectivity of convolution



Feedforward network (fully connected layer)

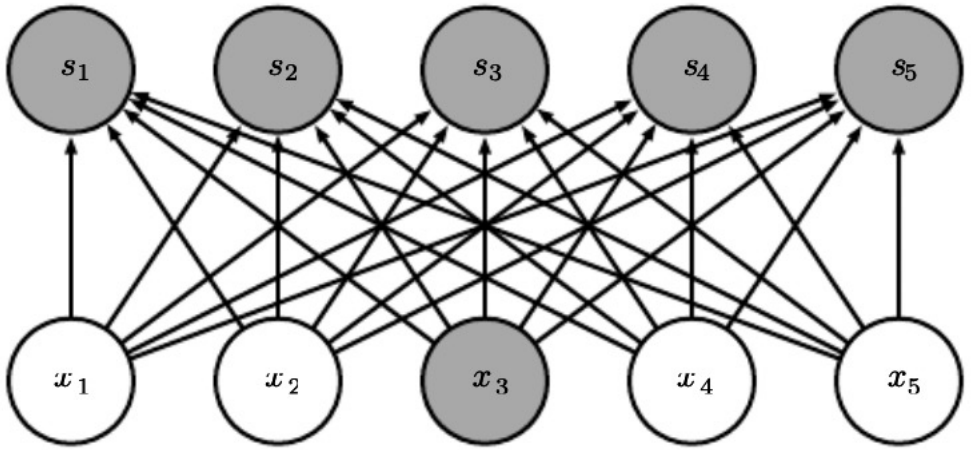
Q: how many arrows we have?

For each x_i: from s_1 to s_5→5 arrows

$x_1, \ldots, x_5$→25 arrows

Sparse connectivity of convolution

$x'w \rightarrow s$



Feedforward network (fully connected layer)

Q: how many arrows we have?

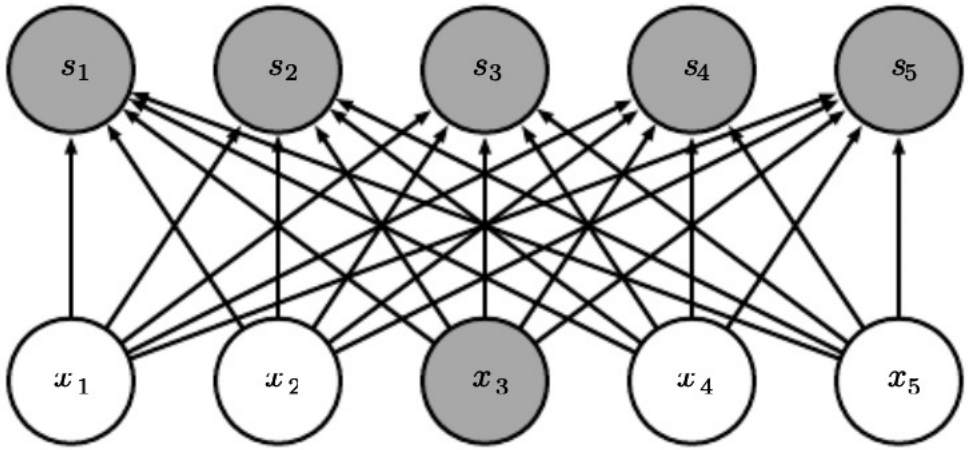
For each x_i: from s_1 to s_5→5 arrows

$x_1, \ldots, x_5$→25 arrows

Sparse connectivity of convolution

x and w are vectors; s is a scalar number

$x'w \rightarrow s$



Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5→5 arrows

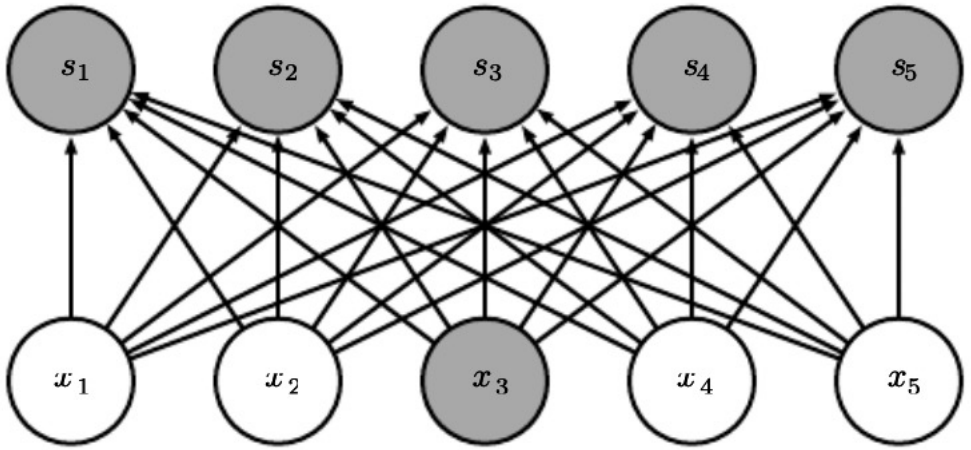
$x_1, \ldots, x_5$→25 arrows

Sparse connectivity of convolution

x and w are vectors; s is a scalar number

$$x'w \to s$$

$$x'w_1 = \sum_{i=1}^{5} x_i w_{1,i}$$



Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5→5 arrows

$x_1, \ldots, x_5$→25 arrows

Sparse connectivity of convolution

$$x'w \rightarrow s$$

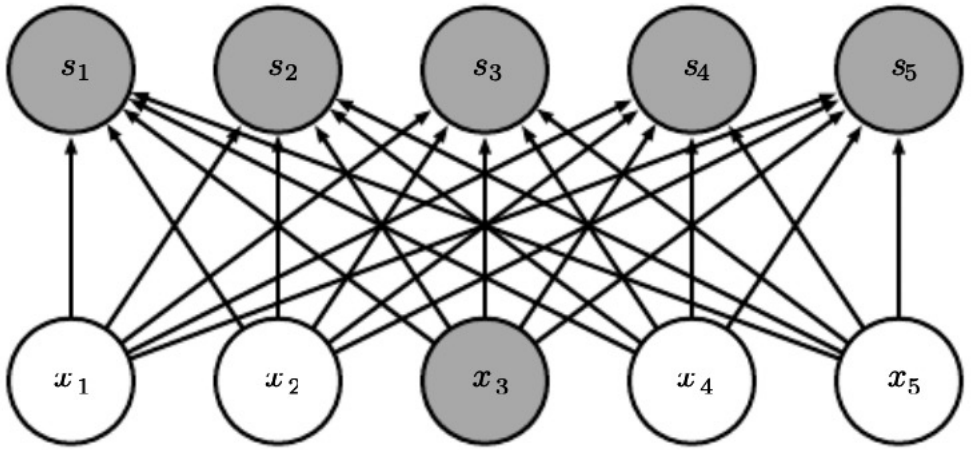
$$x'w_1 = \sum_{i=1}^{5} x_i w_{1,i}$$

$$x'w_2 = \sum_{i=1}^{5} x_i w_{2,i}$$

$$x'w_3 = \sum_{i=1}^{5} x_i w_{3,i}$$

$$x'w_4 = \sum_{i=1}^{5} x_i w_{4,i}$$

$$x'w_5 = \sum_{i=1}^{5} x_i w_{5,i}$$



Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5→5 arrows

$x_1, \ldots, x_5$→25 arrows

Sparse connectivity of convolution

$$x'w \rightarrow s$$

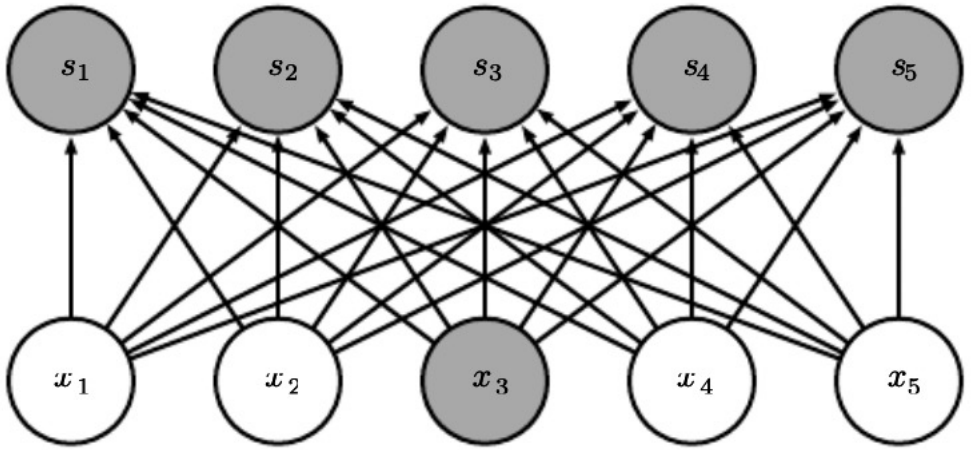
$$x'w_1 = \sum_{i=1}^{5} x_i w_{1,i}$$

$$x'w_2 = \sum_{i=1}^{5} x_i w_{2,i}$$

$$x'w_3 = \sum_{i=1}^{5} x_i w_{3,i}$$

$$x'w_4 = \sum_{i=1}^{5} x_i w_{4,i}$$

$$x'w_5 = \sum_{i=1}^{5} x_i w_{5,i}$$



Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5→5 arrows

$x_1, \ldots, x_5$→25 arrows

Sparse connectivity of convolution

$$x'w \rightarrow s$$

$$x'w_1 = \sum_{i=1}^{5} x_i w_{1,i}$$

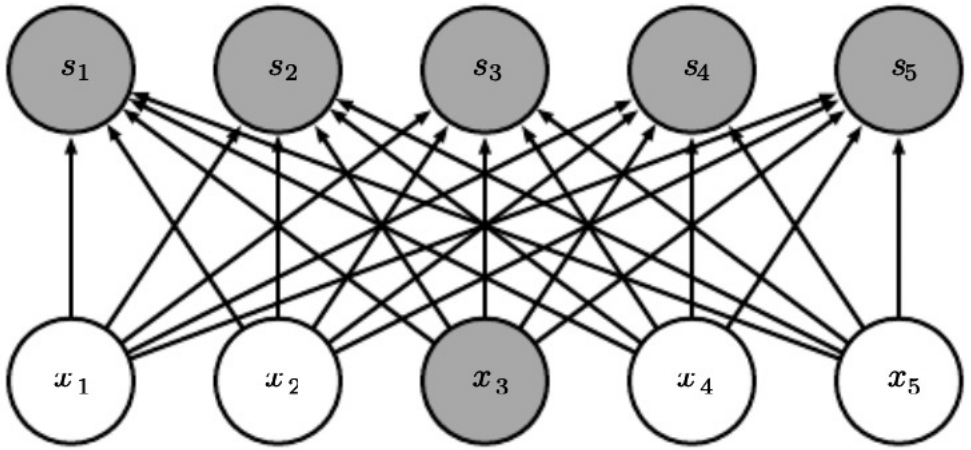
$$x'w_2 = \sum_{i=1}^{5} x_i w_{2,i}$$

$$x'w_3 = \sum_{i=1}^{5} x_i w_{3,i}$$

$$x'w_4 = \sum_{i=1}^{5} x_i w_{4,i}$$

$$x'w_5 = \sum_{i=1}^{5} x_i w_{5,i}$$

$$W = [w_1, w_2, w_3, w_4, w_5]$$



Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5→5 arrows

$x_1, \ldots, x_5$→25 arrows

Sparse connectivity of convolution

$x'w \rightarrow s$

$$x'w_1 = \sum_{i=1}^{5} x_i w_{1,i}$$

$$x'w_2 = \sum_{i=1}^{5} x_i w_{2,i}$$

$$x'w_3 = \sum_{i=1}^{5} x_i w_{3,i}$$

$$x'w_4 = \sum_{i=1}^{5} x_i w_{4,i}$$

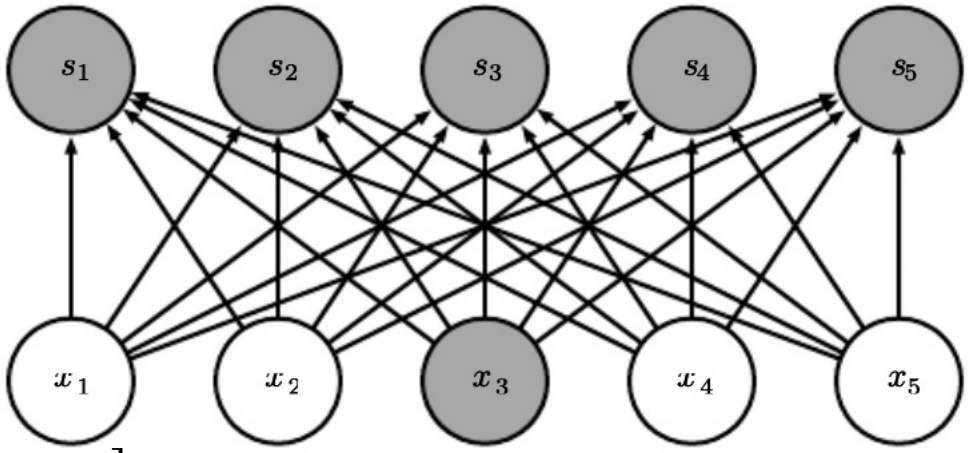
$$x'w_5 = \sum_{i=1}^{5} x_i w_{5,i}$$

W

=

$[w_1, w_2, w_3, w_4, w_5]$

(5x5 weight matrix)



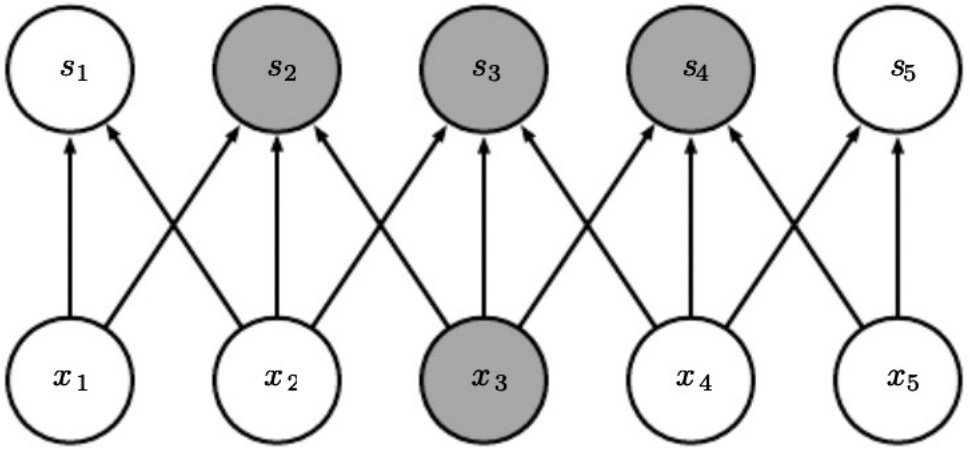
Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5 → 5 arrows

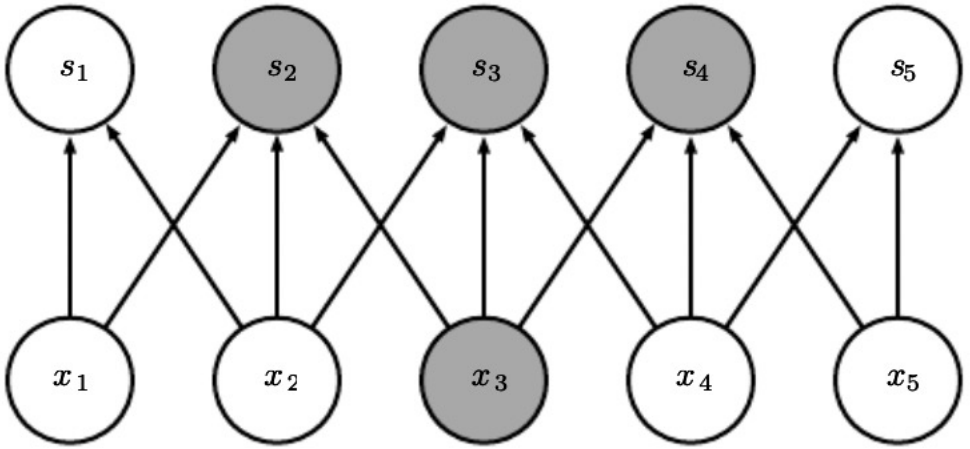
$x_1, \ldots, x_5$ → 25 arrows

Sparse connectivity of convolution



Convolutional layers

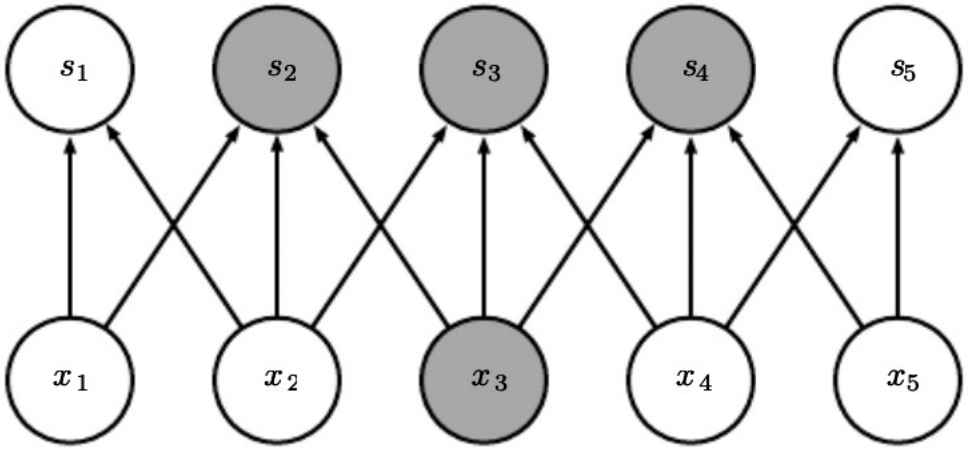
Sparse connectivity of convolution



Convolutional layers

Q: how many arrows we have?

Sparse connectivity of convolution

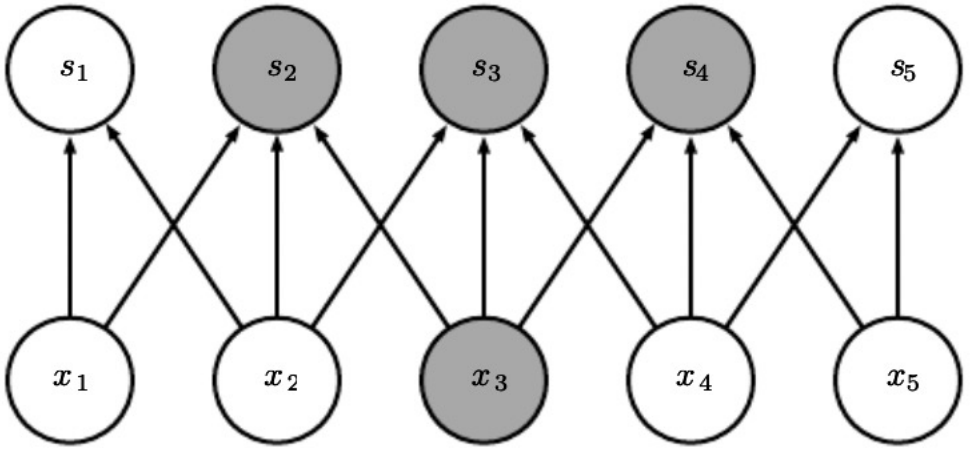


Convolutional layers

Q: how many arrows we have?

For each x_i: connect to 3 s outputs

Sparse connectivity of convolution



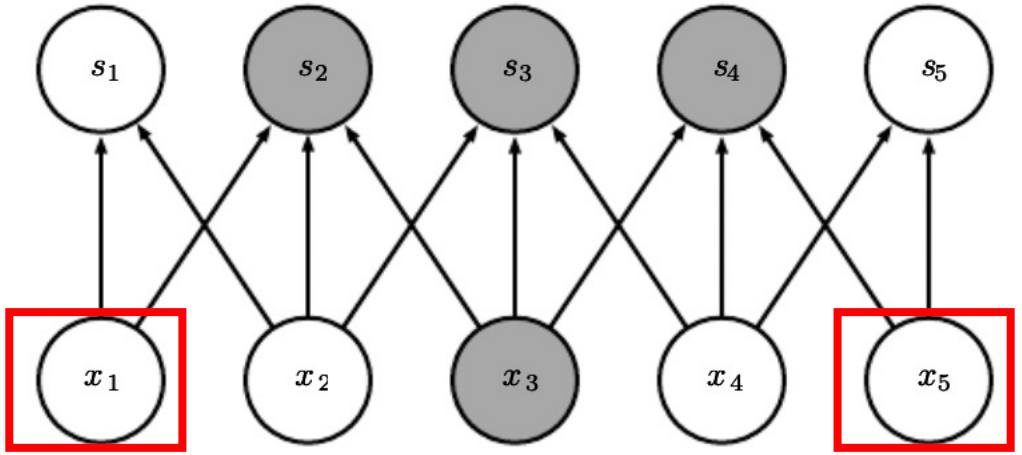
Convolutional layers

Q: how many arrows we have?

For each x_i: connect to 3 s outputs

$x_1, \ldots, x_5$ → 3x5-2=13 arrows

Sparse connectivity of convolution



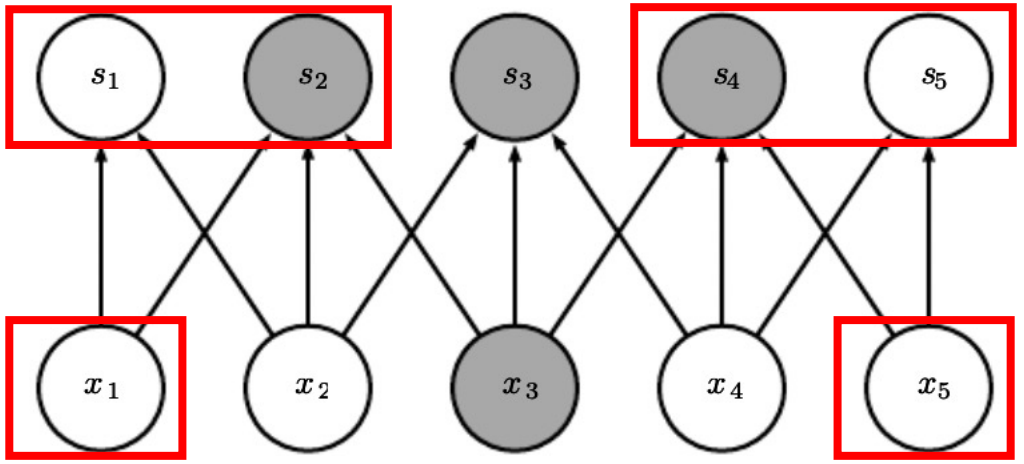
Convolutional layers

Q: how many arrows we have?

For each x_i: connect to 3 s outputs

$x_1, \ldots, x_5$ → 3x5-2=13 arrows

Sparse connectivity of convolution



Convolutional layers

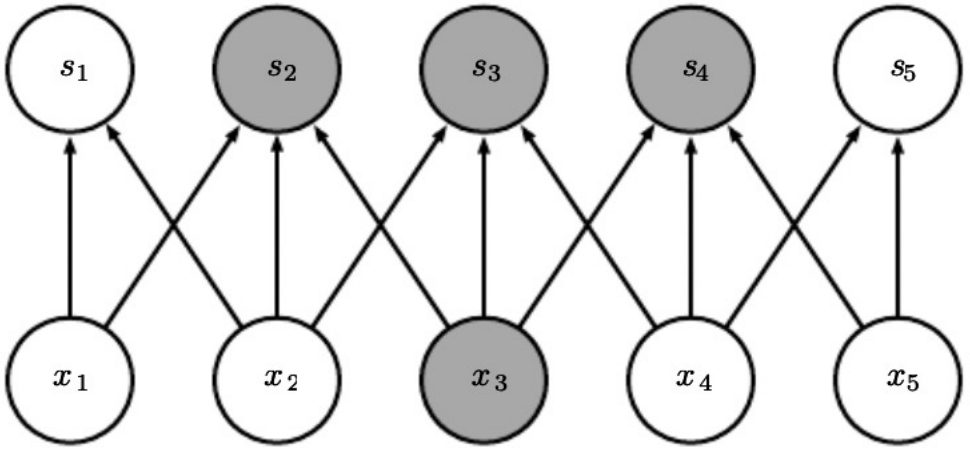
Q: how many arrows we have?

For each x_i: connect to 3 s outputs

$x_1, \ldots, x_5$ → 3x5-2=13 arrows

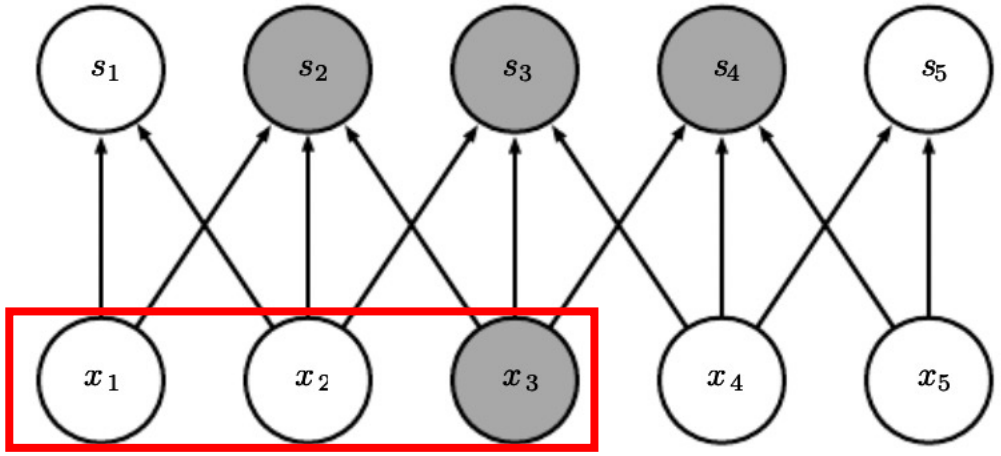
Sparse connectivity of convolution

The view of convolutional kernel/filter



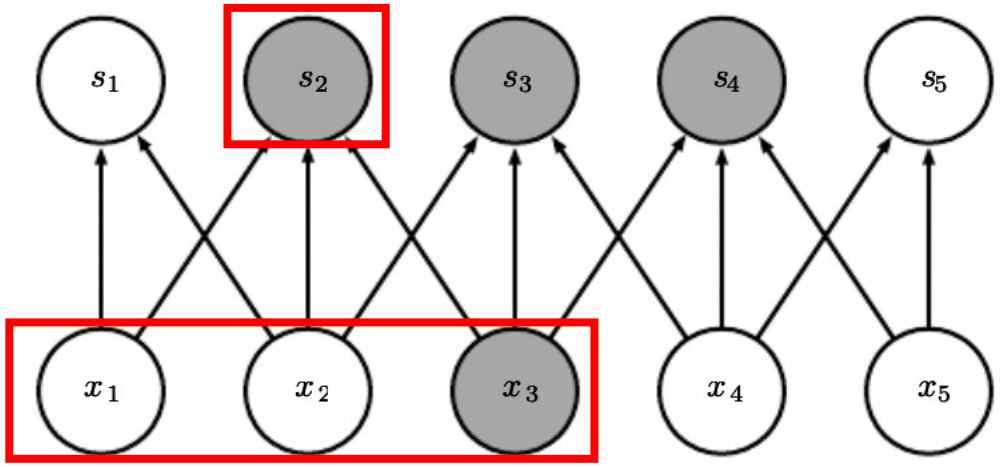
Sparse connectivity of convolution

The view of convolutional kernel/filter



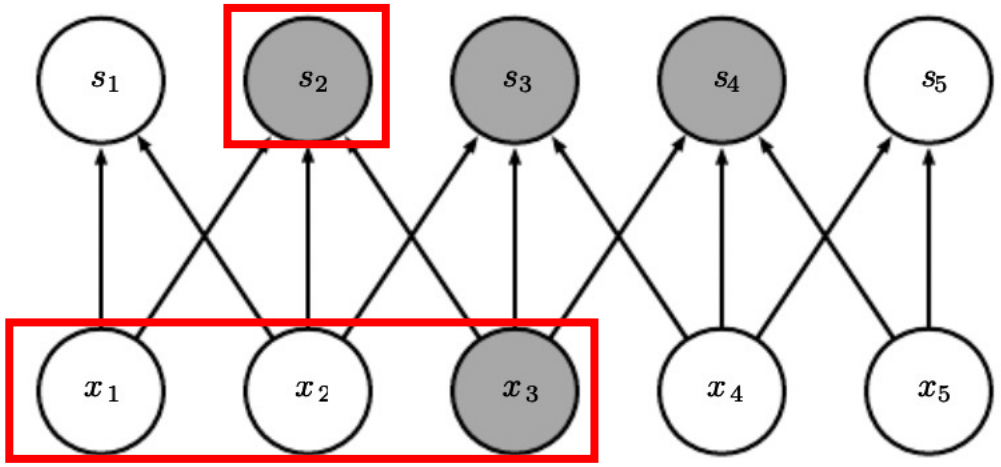
Sparse connectivity of convolution

The view of convolutional kernel/filter



Sparse connectivity of convolution

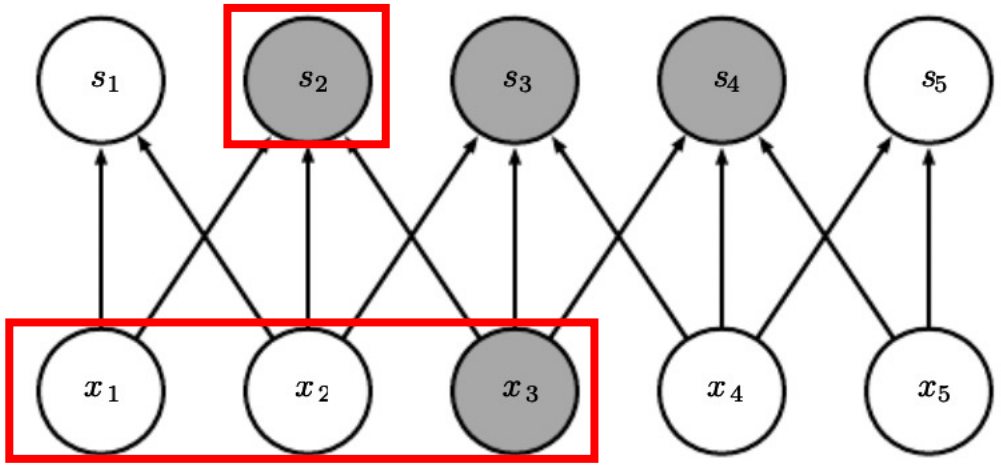
The view of convolutional kernel/filter



Q: filter size and stride?

Sparse connectivity of convolution

The view of convolutional kernel/filter

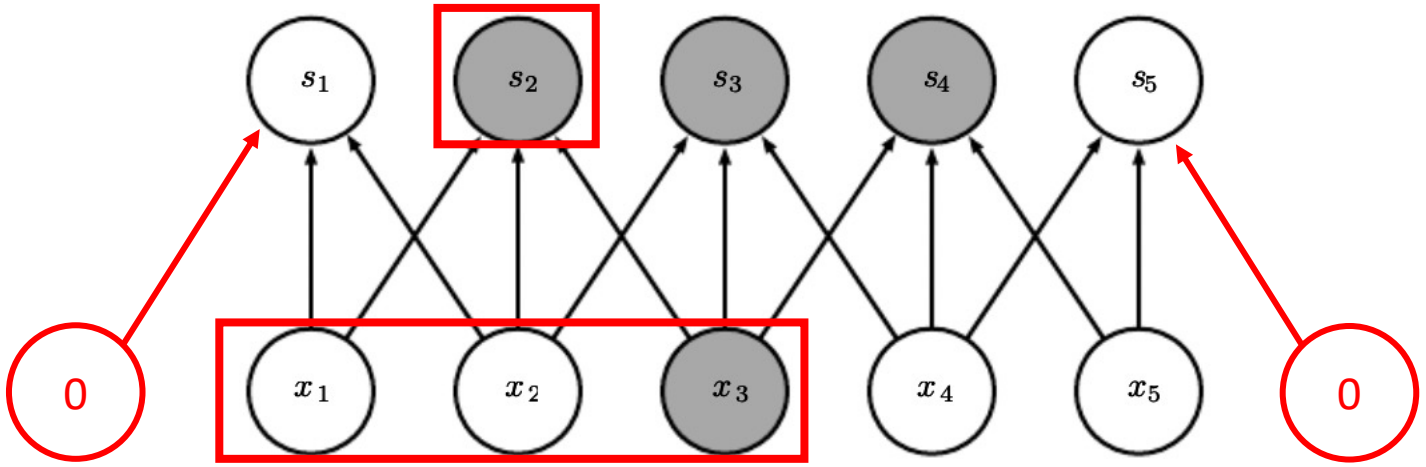


Q: filter size and stride?

Filter size = 3 + stride = 1 with 0-pading

Sparse connectivity of convolution

The view of convolutional kernel/filter

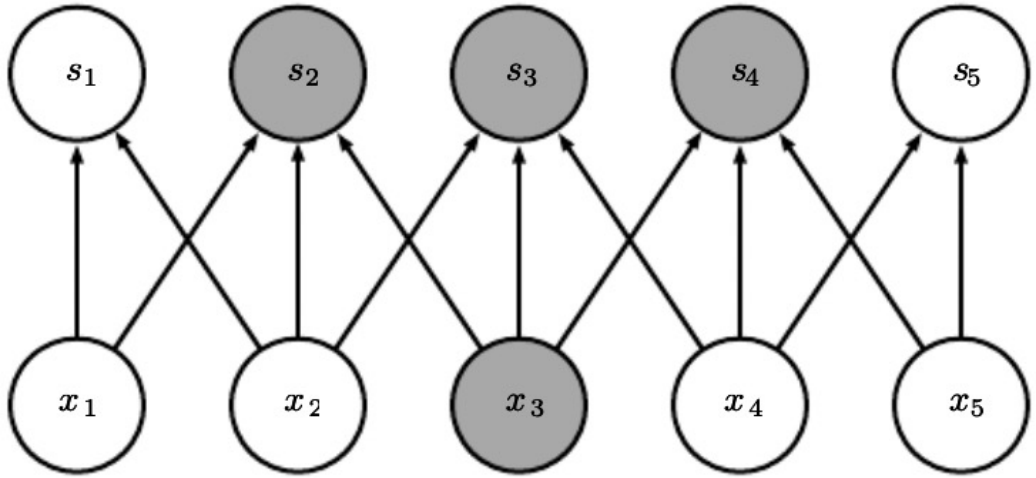


Q: filter size and stride?

Filter size = 3 + stride = 1 with 0-pading

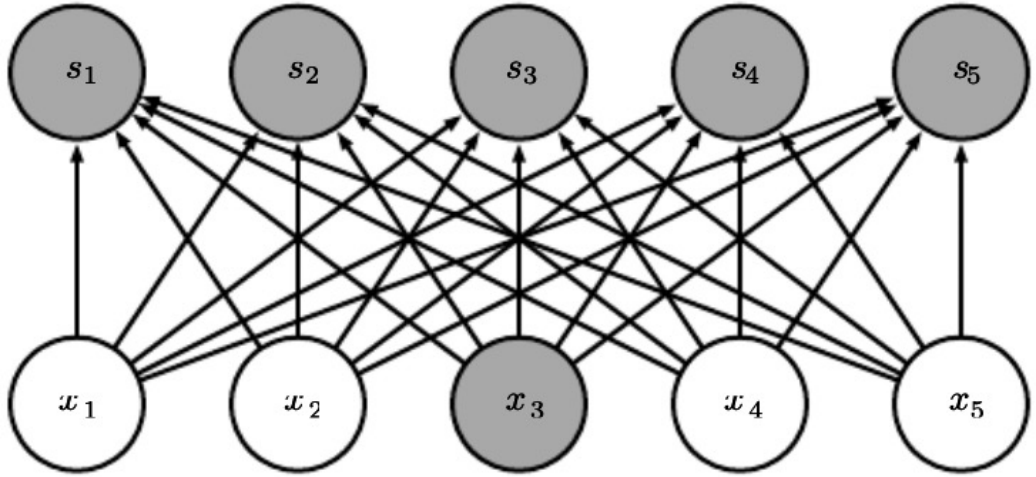
Sparse connectivity of convolution

Sparse connection



Weights→13 scalar numbers

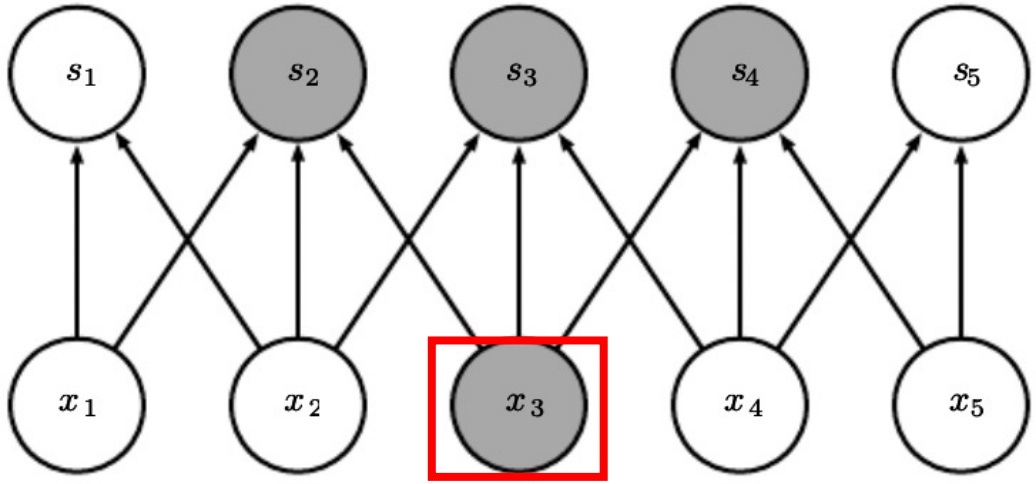
Dense connection



Weights→25 scalar numbers

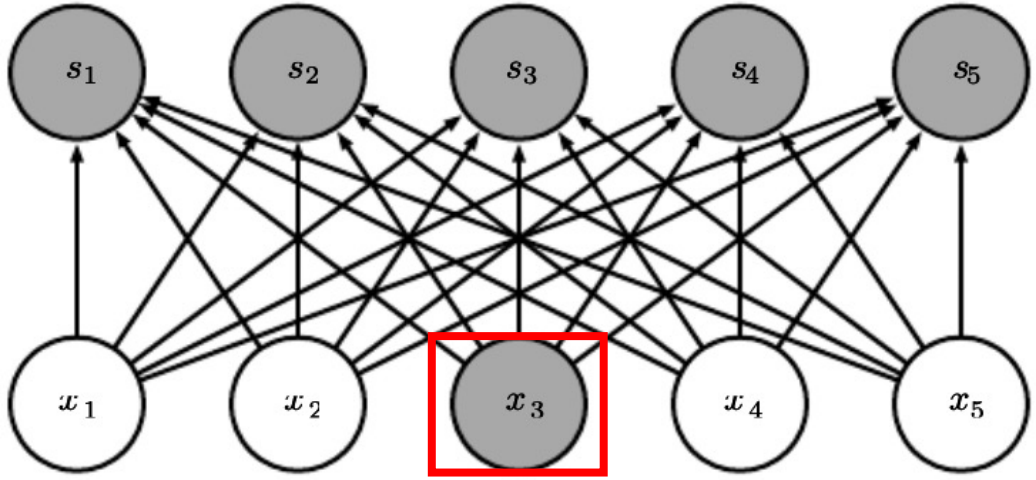
Sparse connectivity of convolution

Sparse connection



Weights→13 scalar numbers

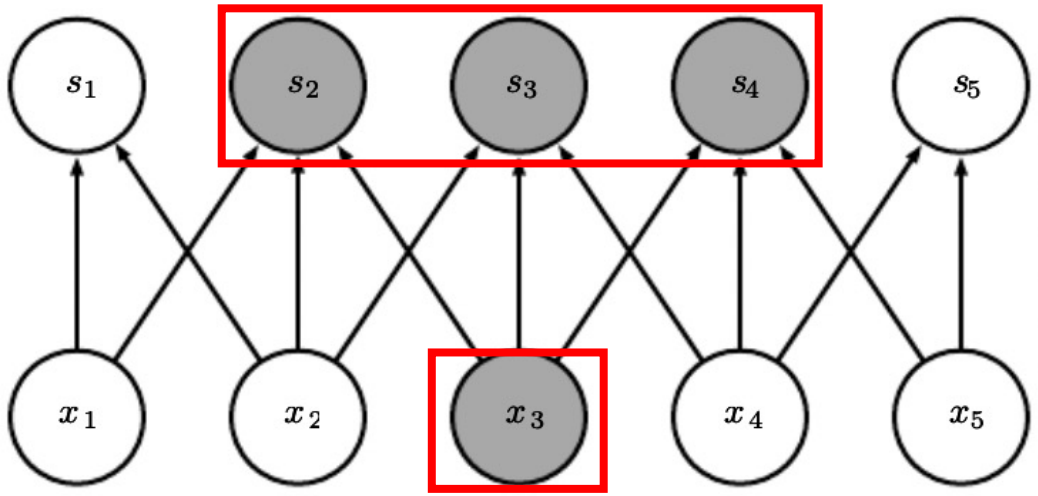
Dense connection



Weights→25 scalar numbers

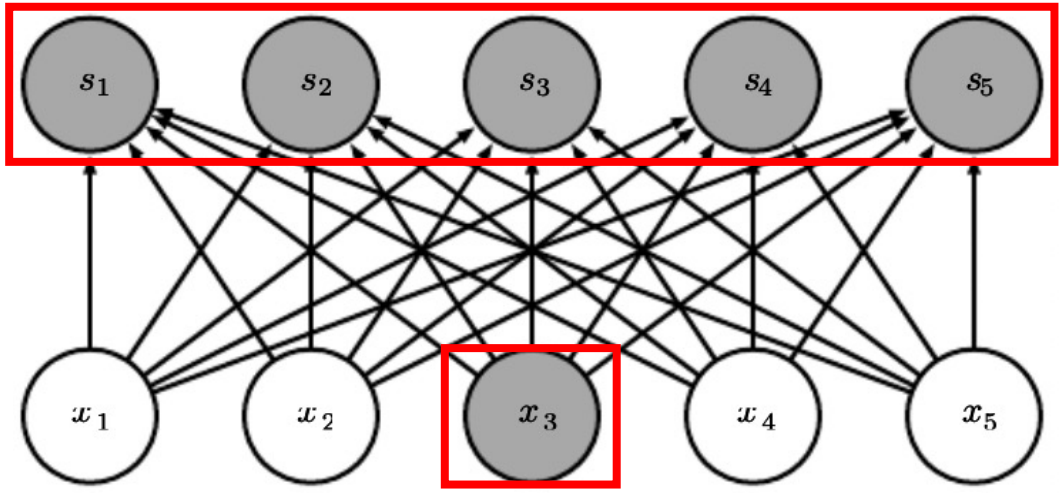
Sparse connectivity of convolution

Sparse connection



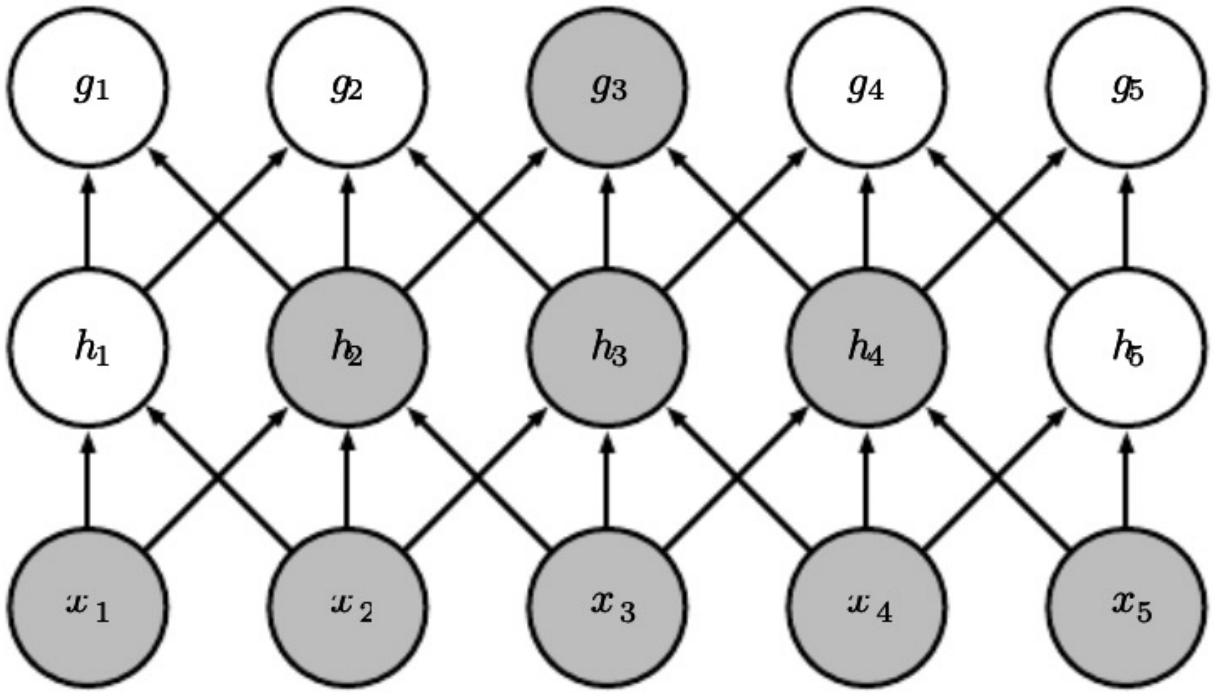
Weights→13 scalar numbers

Dense connection



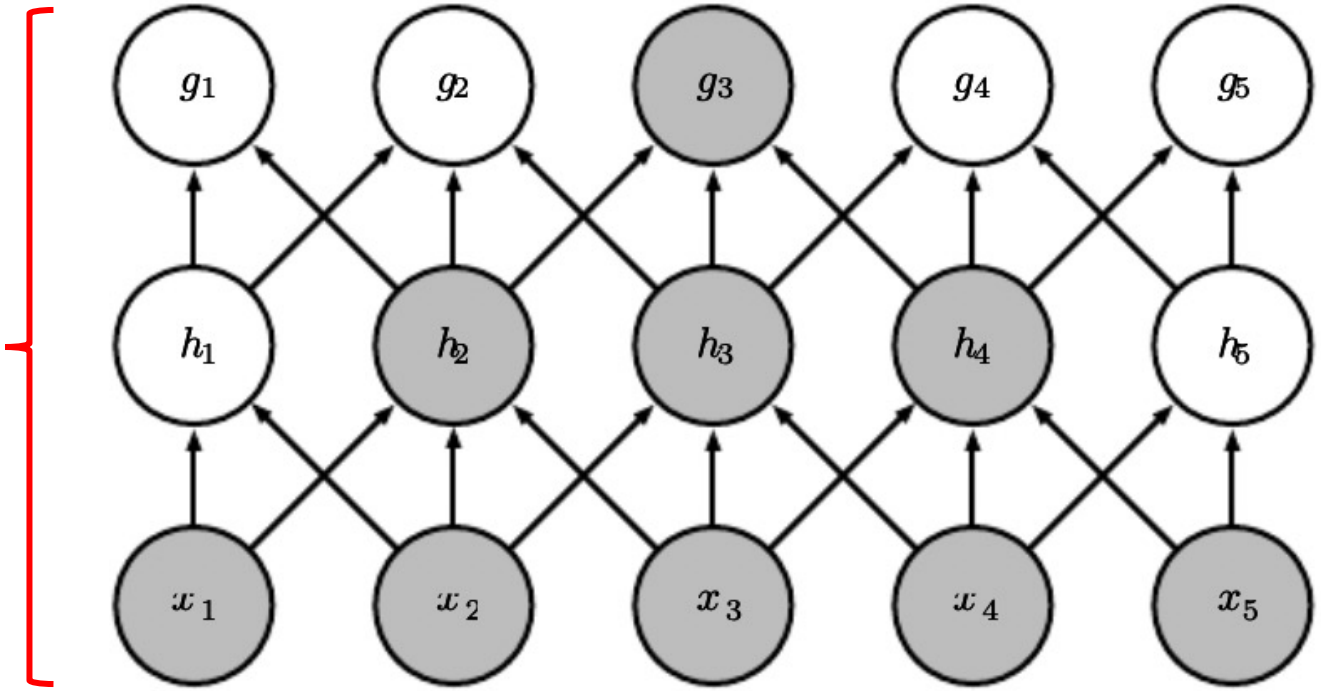
Weights→25 scalar numbers

Sparse connectivity of convolution

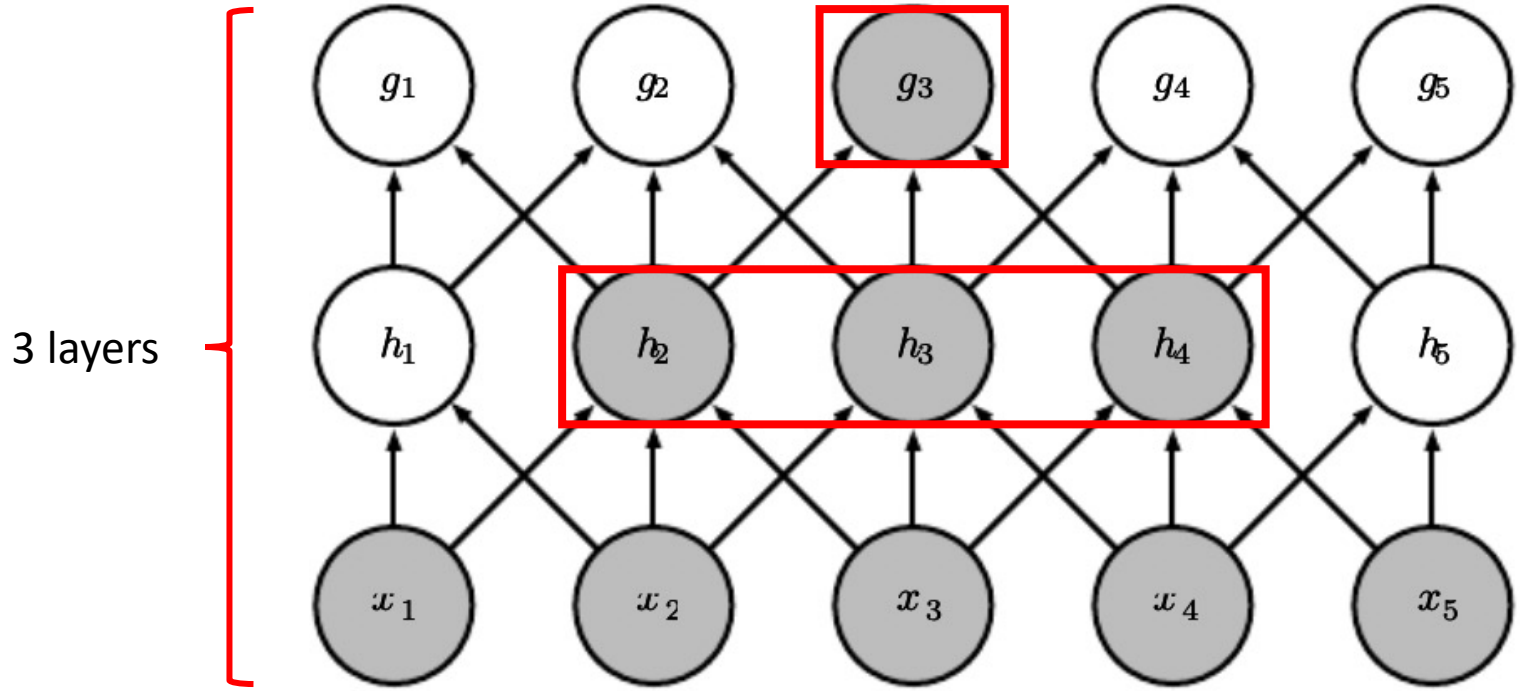


Sparse connectivity of convolution

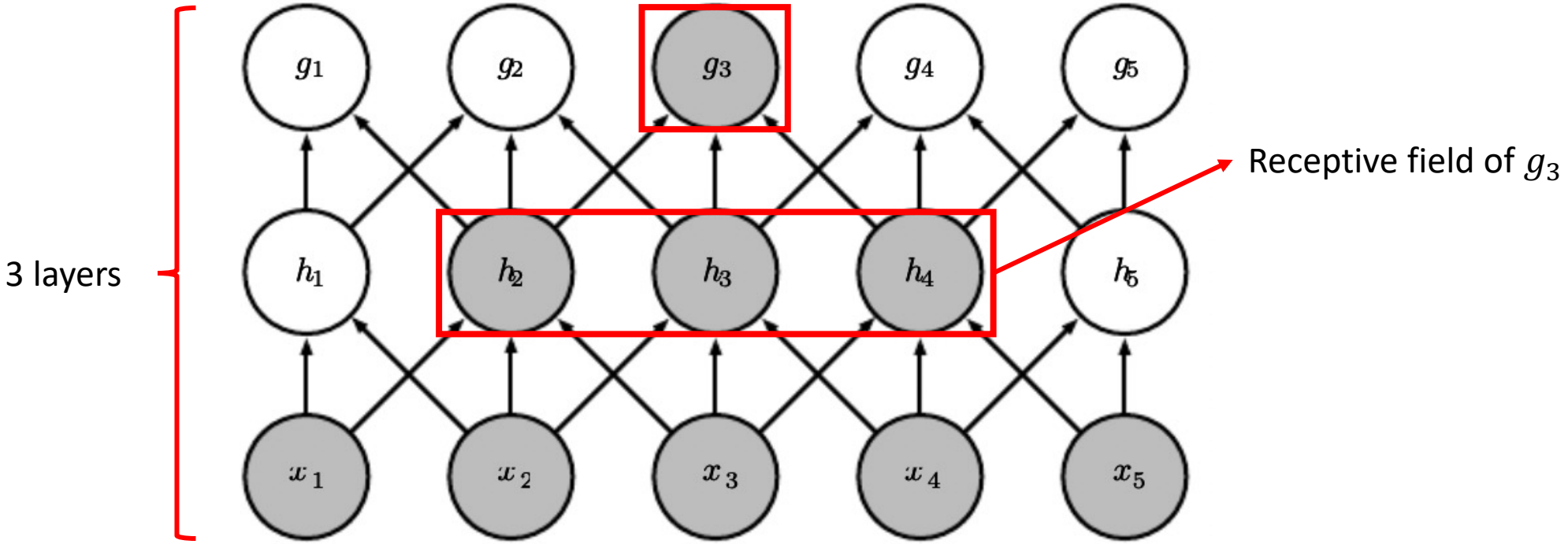
3 layers



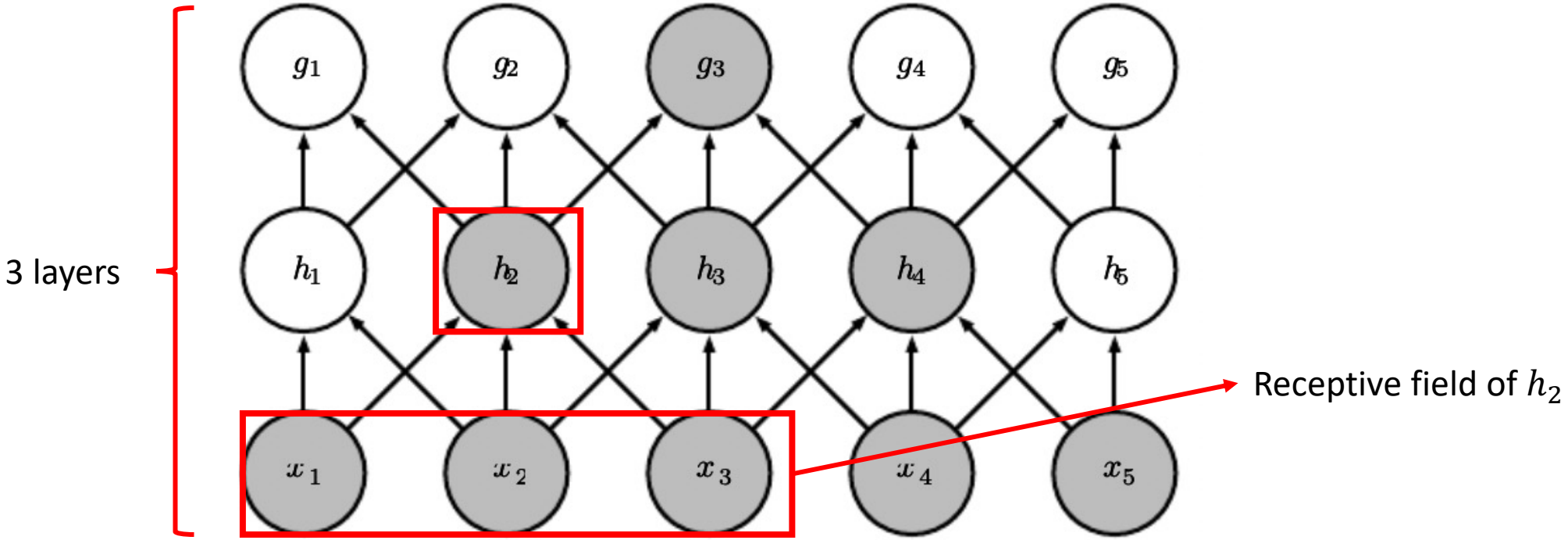
Sparse connectivity of convolution



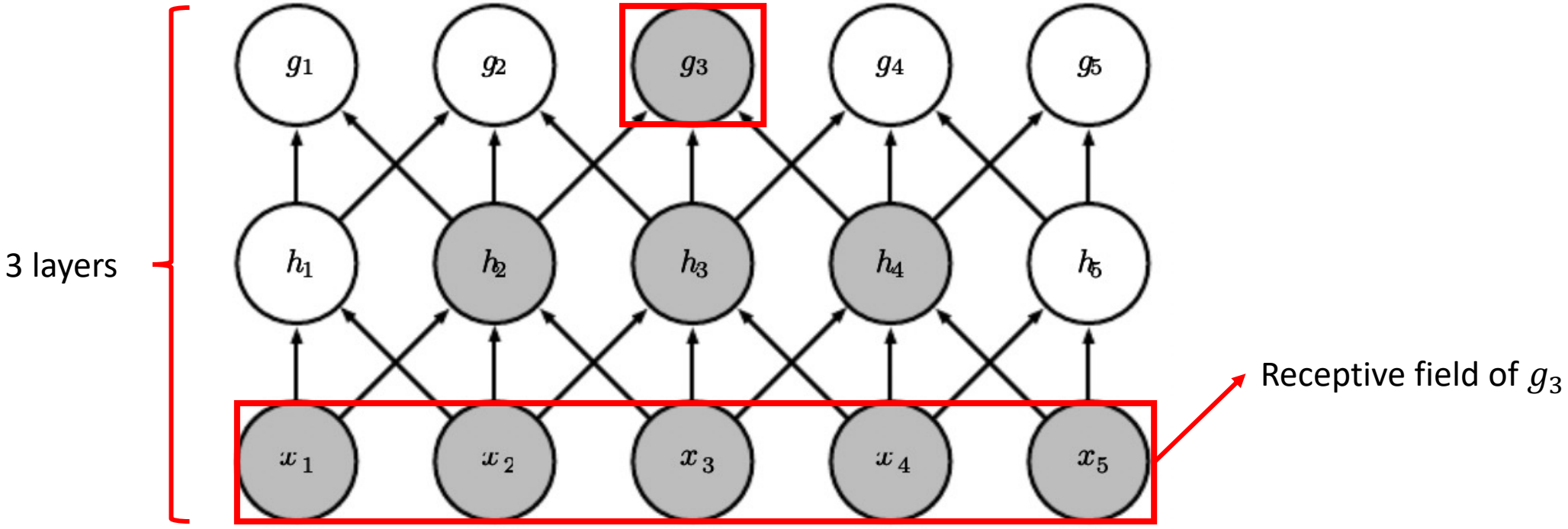
Sparse connectivity of convolution



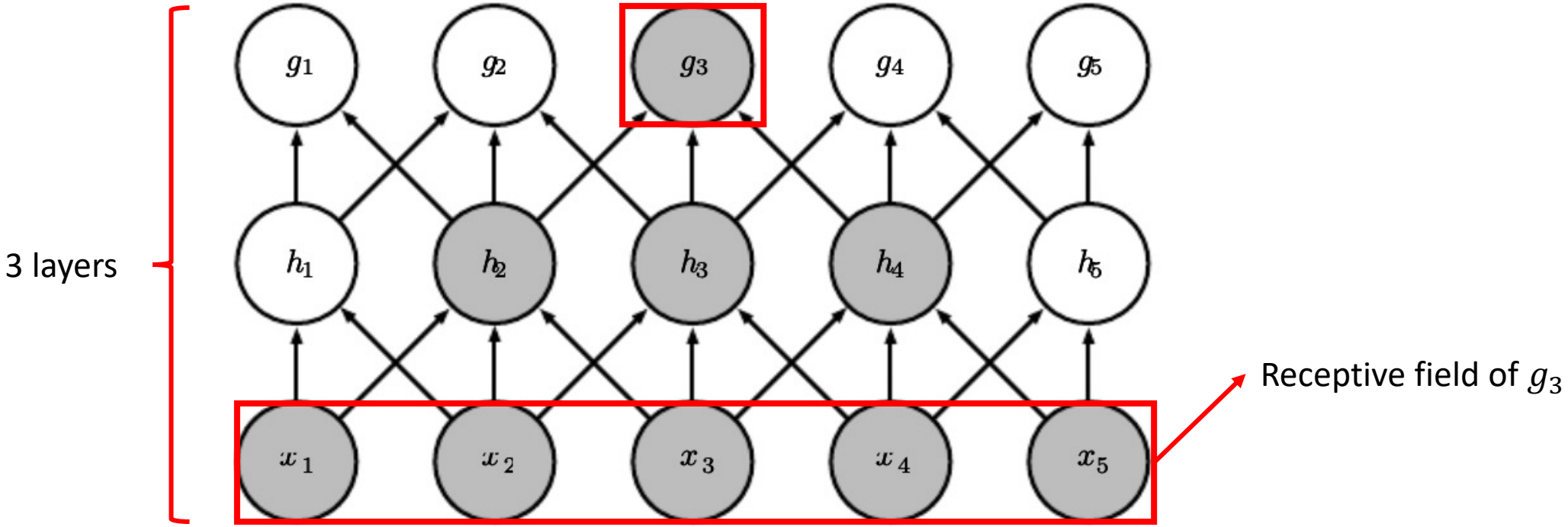
Sparse connectivity of convolution



Sparse connectivity of convolution



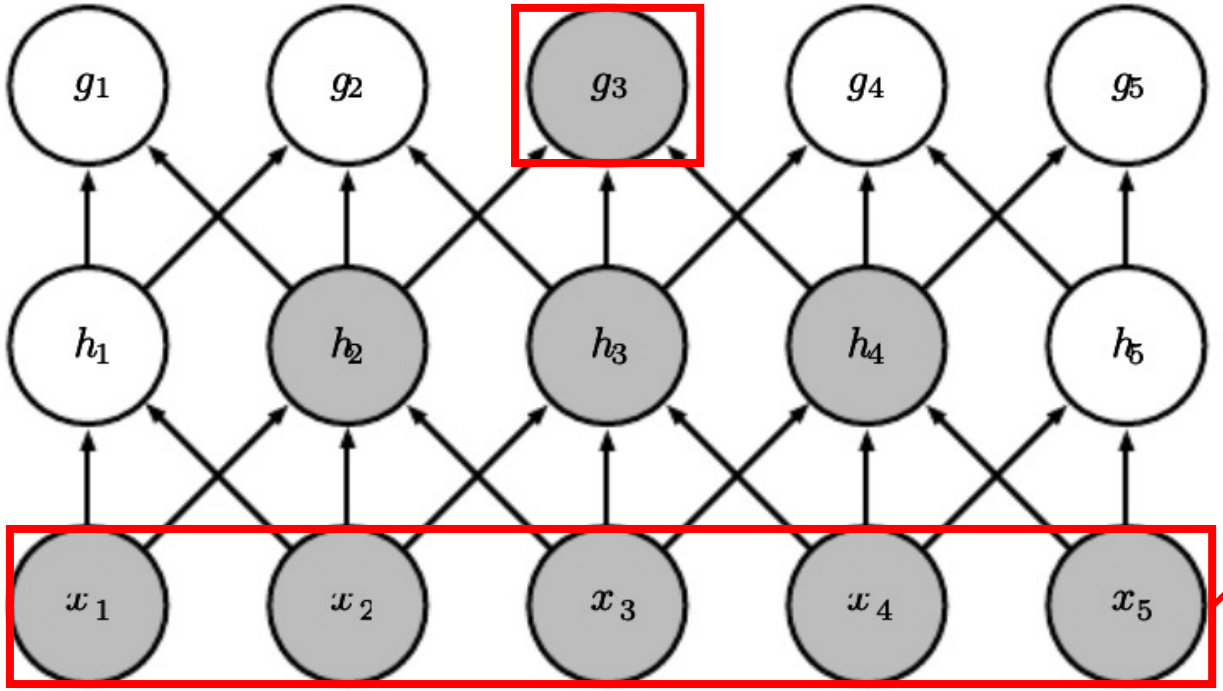
Sparse connectivity of convolution



Deep layers has larger receptive field than shallow layers

Sparse connectivity of convolution

3 layers



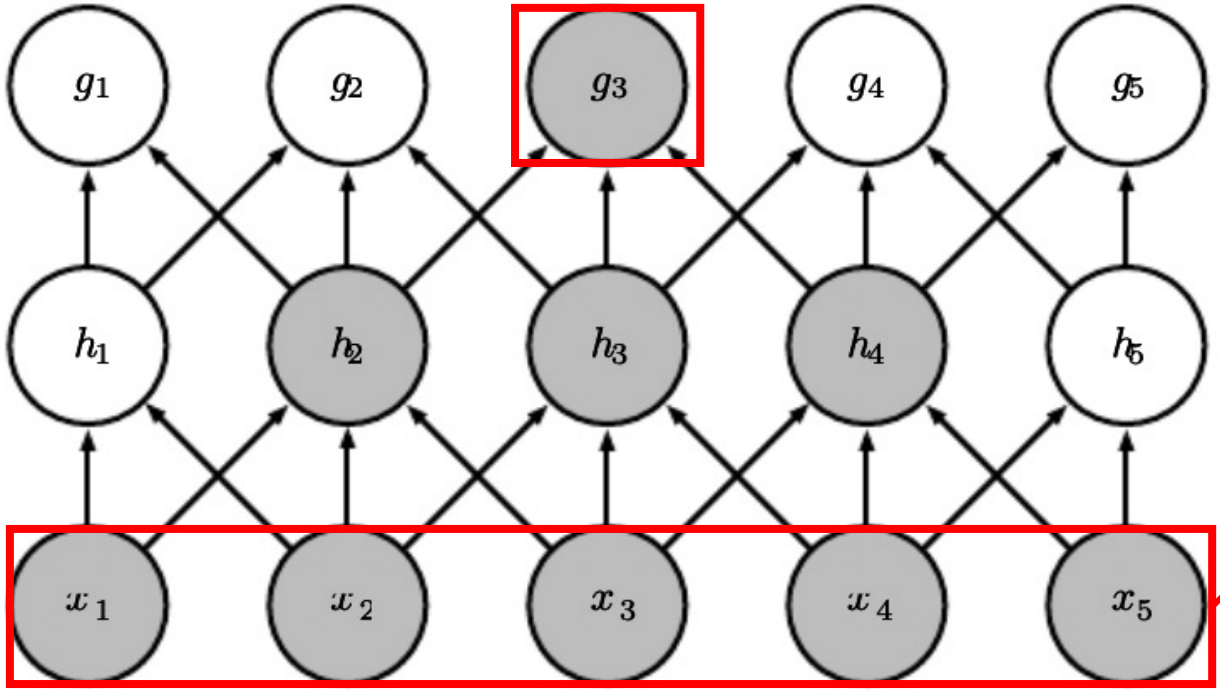
Receptive field of g_3

Units in deep layer indirectly connect to all/most input image

Deep layers has larger receptive field than shallow layers

Sparse connectivity of convolution

3 layers



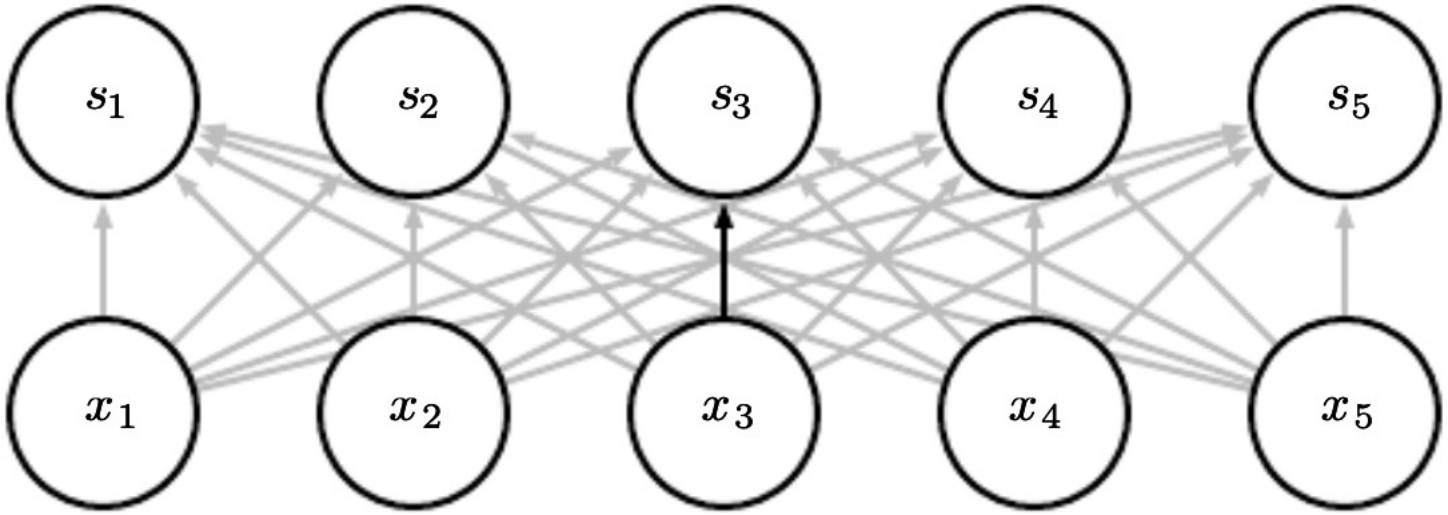
Receptive field of g_3

Units in deep layer indirectly connect to all/most input image

Deep layers has larger receptive field than shallow layers

Q: larger stride of convolution filter → increase receptive field?

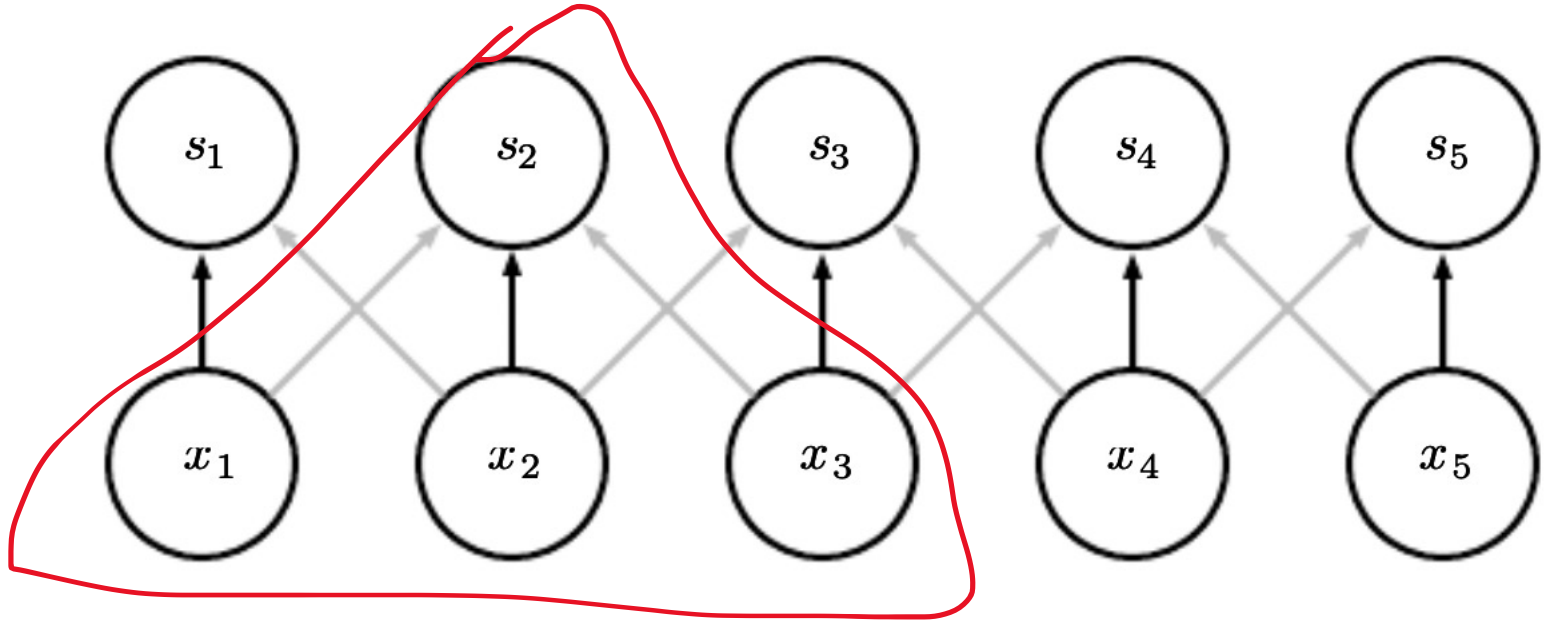
Parameter sharing



In MLP (FC layer): $w^T x$

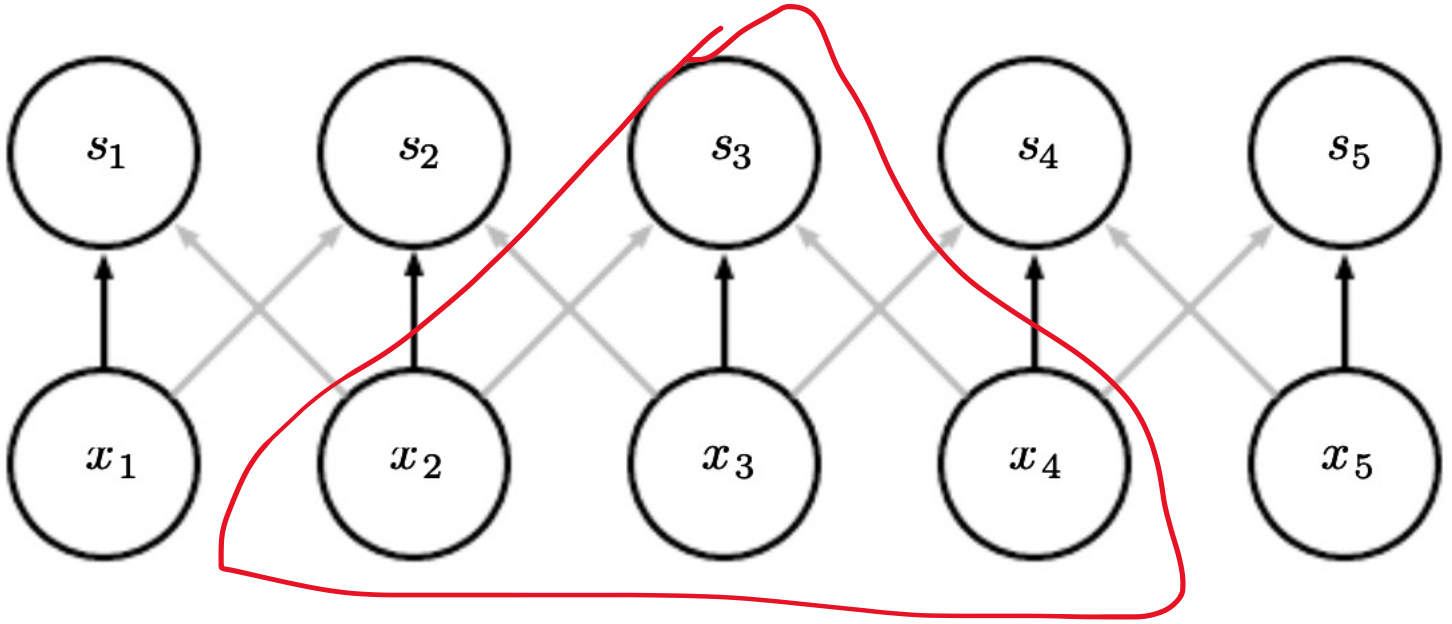
Parameter sharing

Consider the same filter



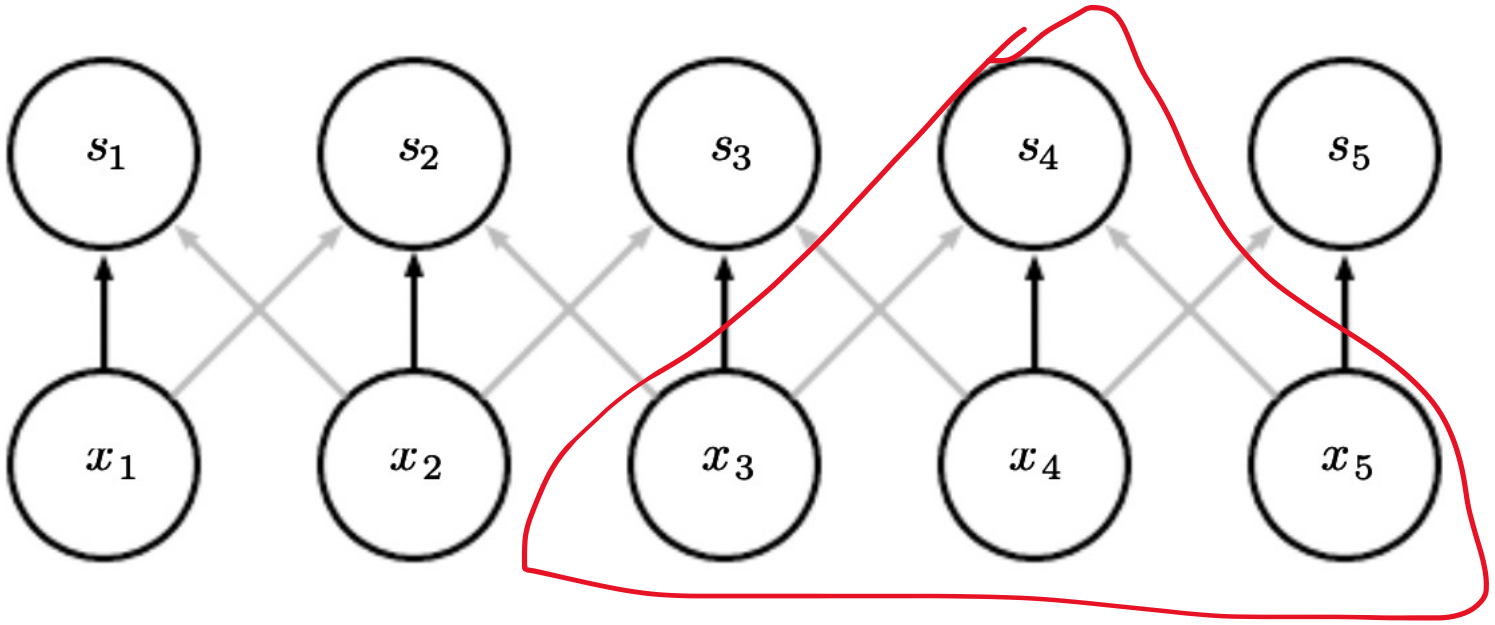
Parameter sharing

Consider the same filter



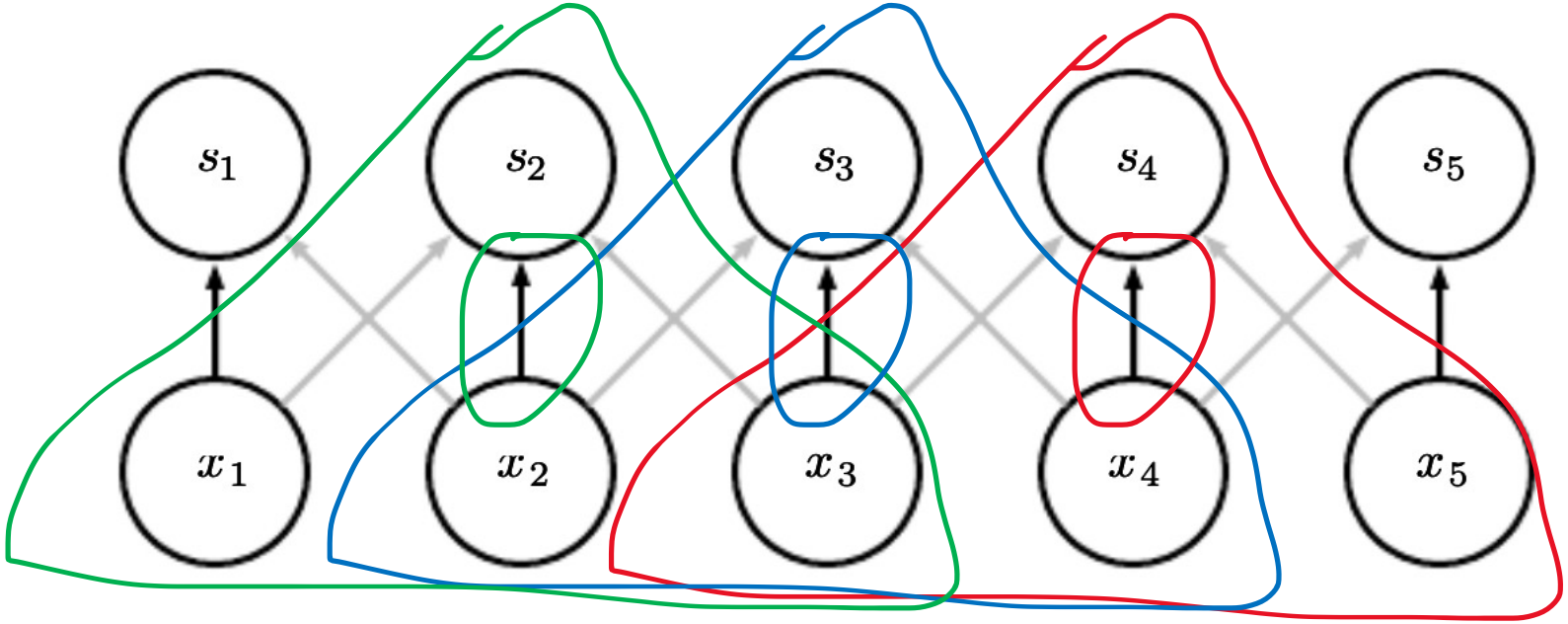
Parameter sharing

Consider the same filter



Parameter sharing

Consider the same filter (but different part of input feat. map)



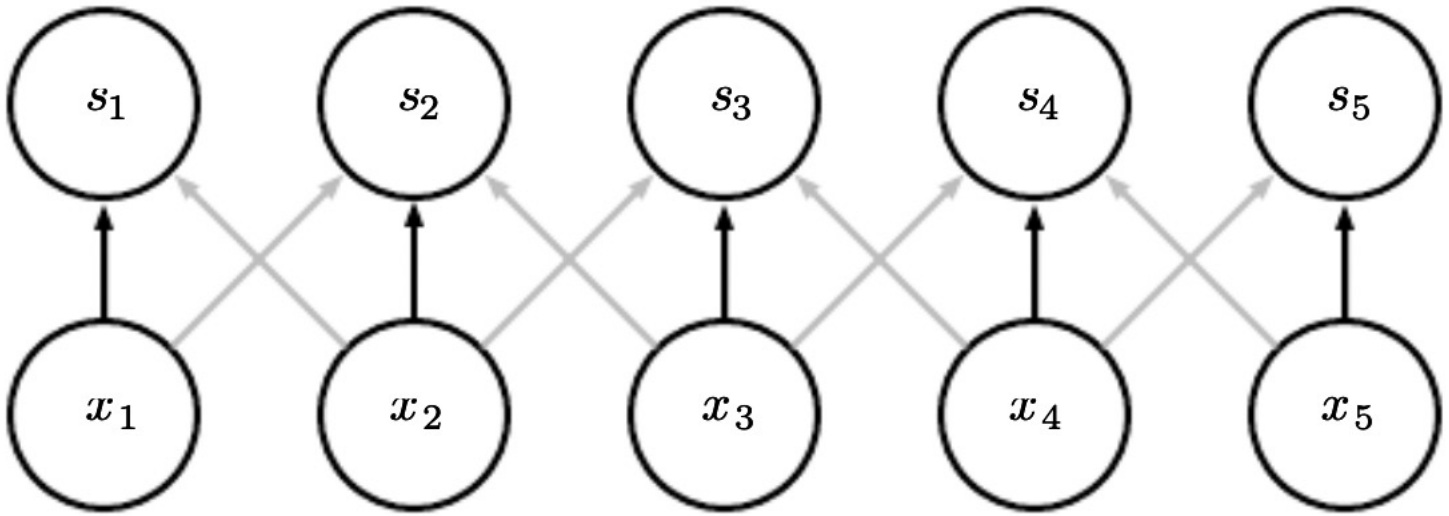
In convlayer:

$w^T(x_1; x_2; x_3)$

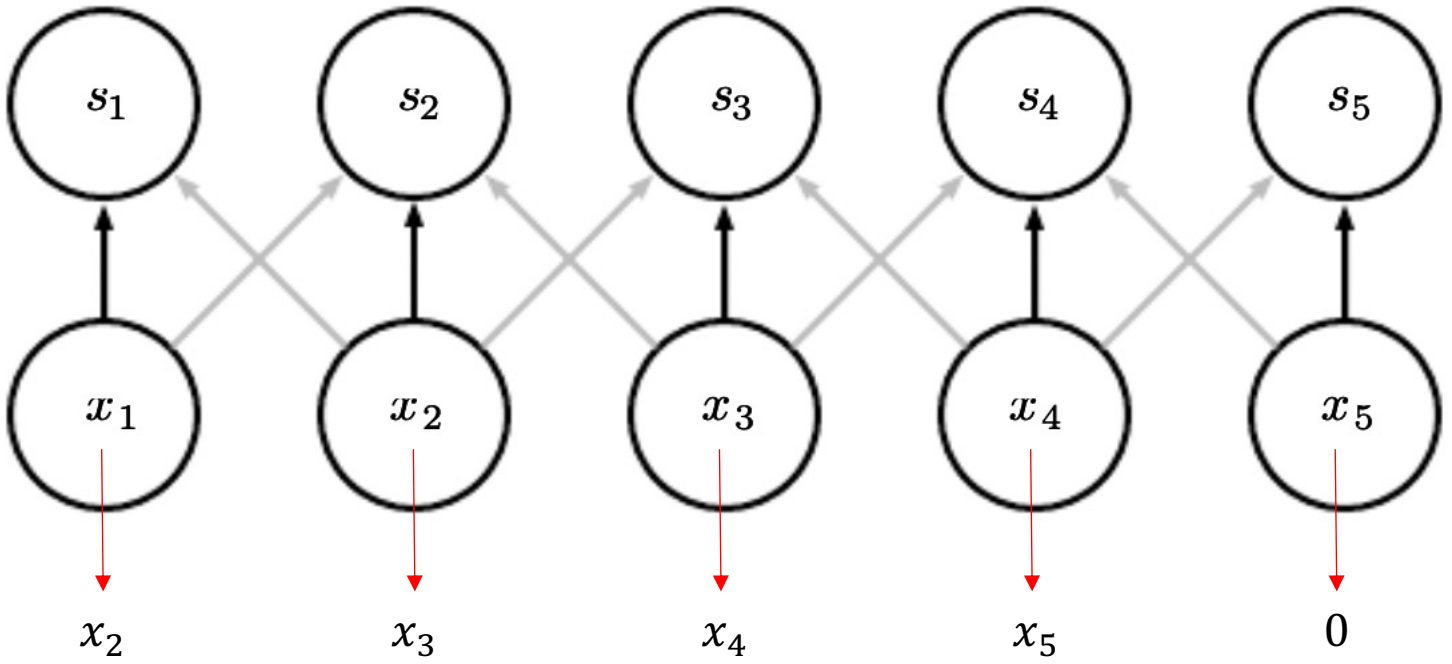
$w^T(x_2; x_3; x_4)$

$w^T(x_3; x_4; x_5)$

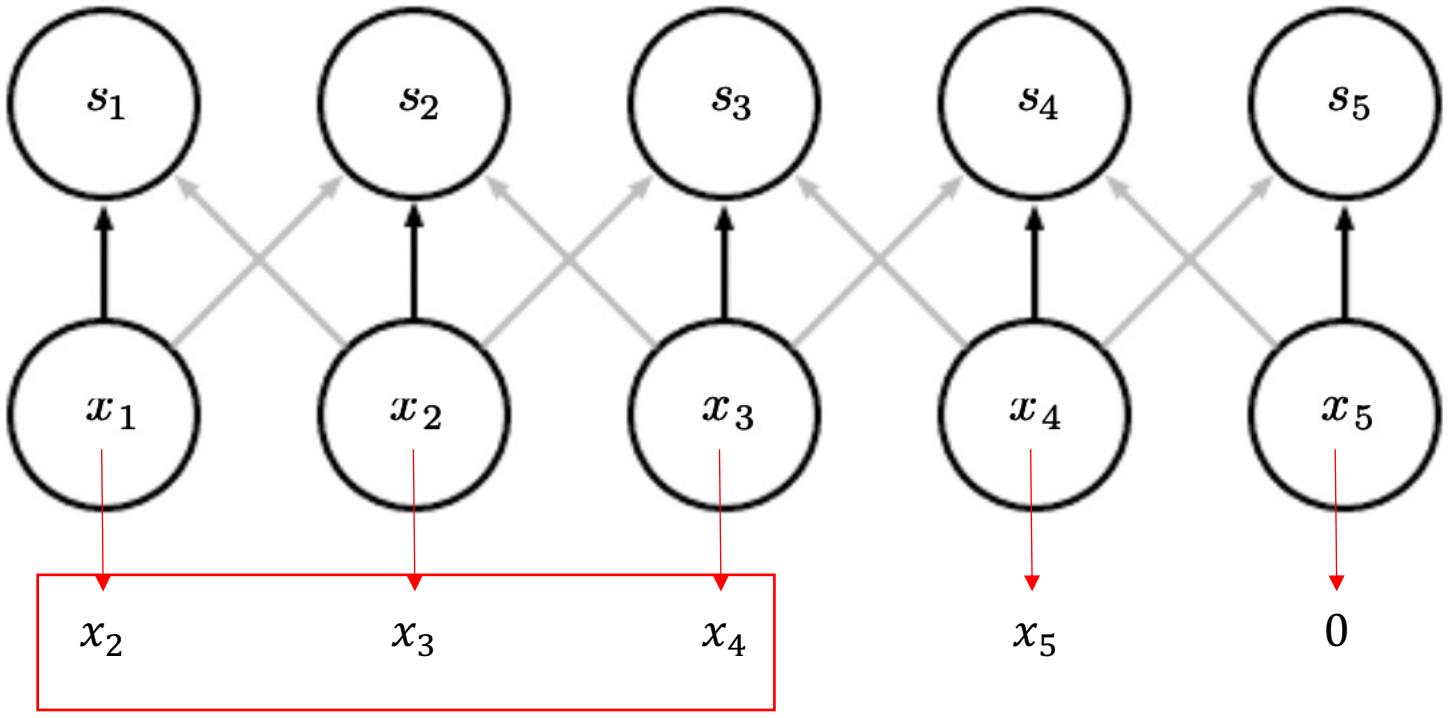
Equivariance



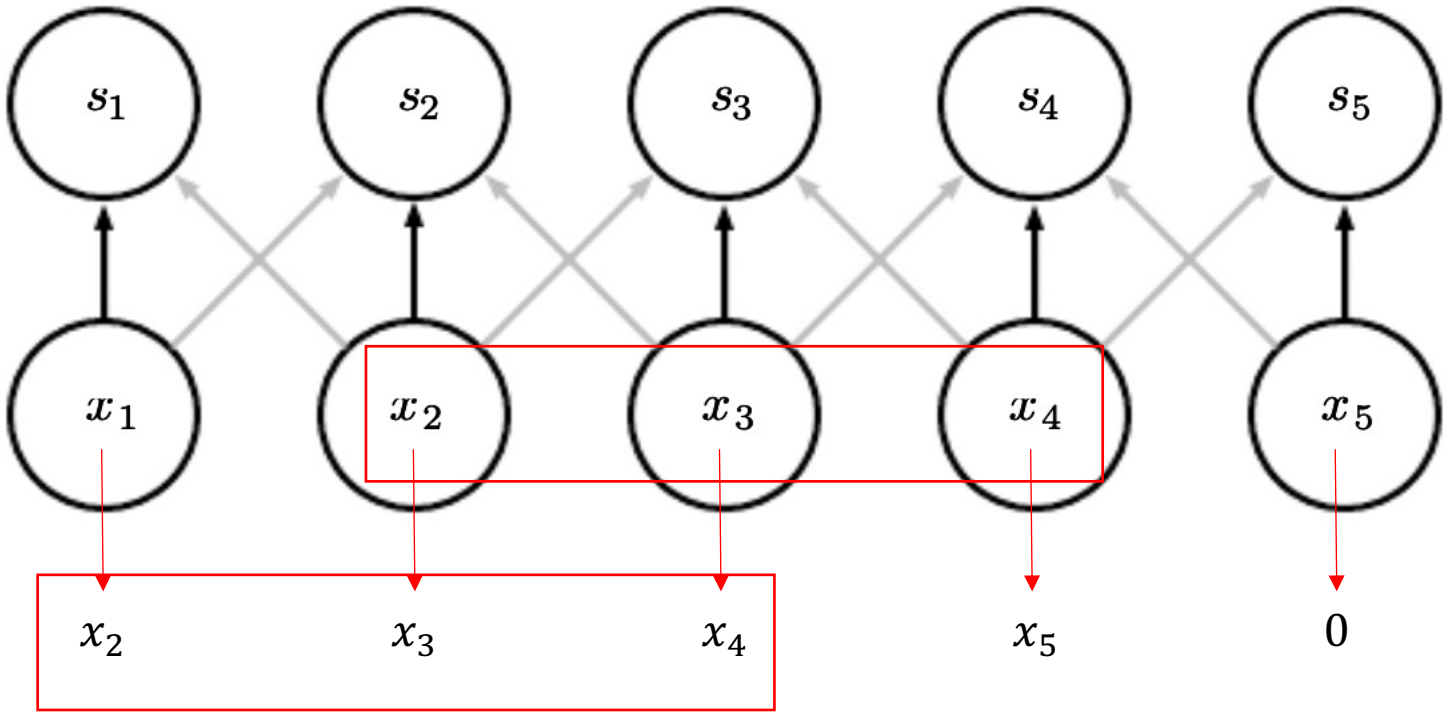
Equivariance



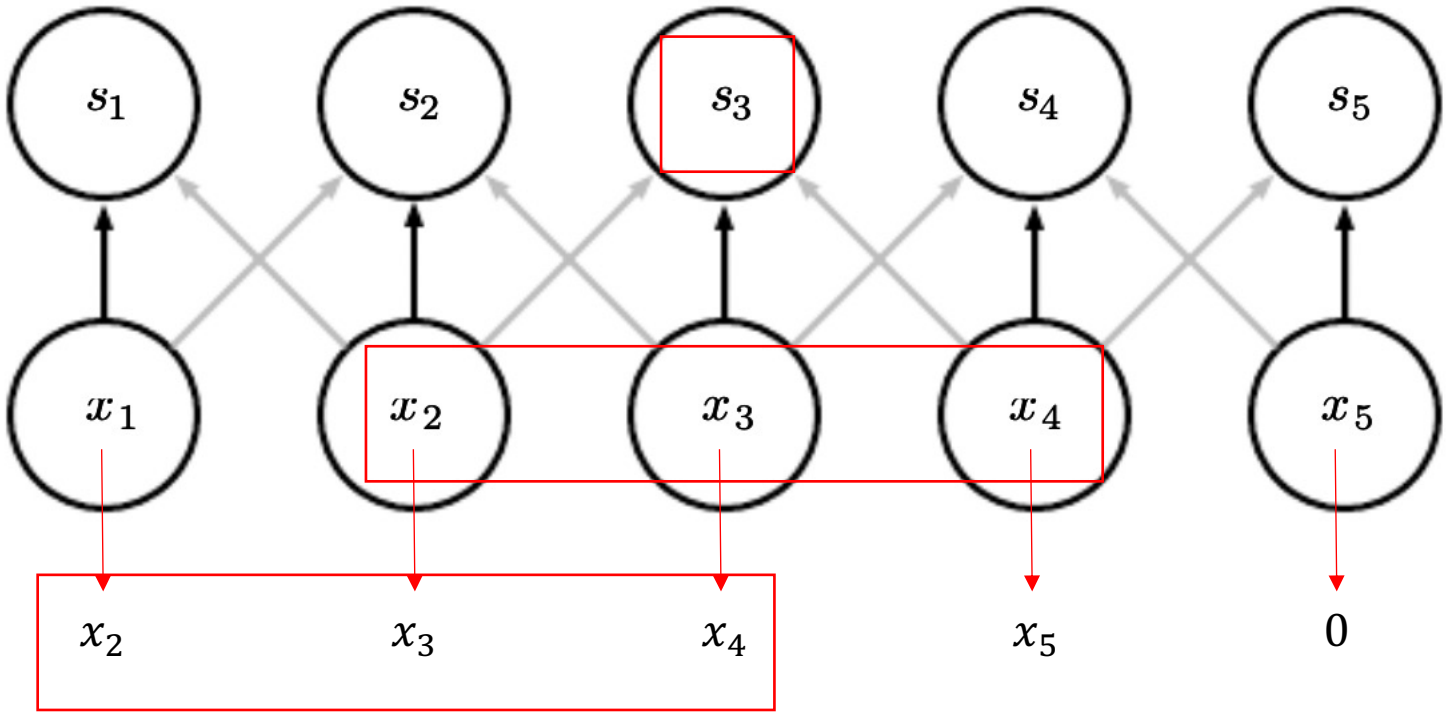
Equivariance



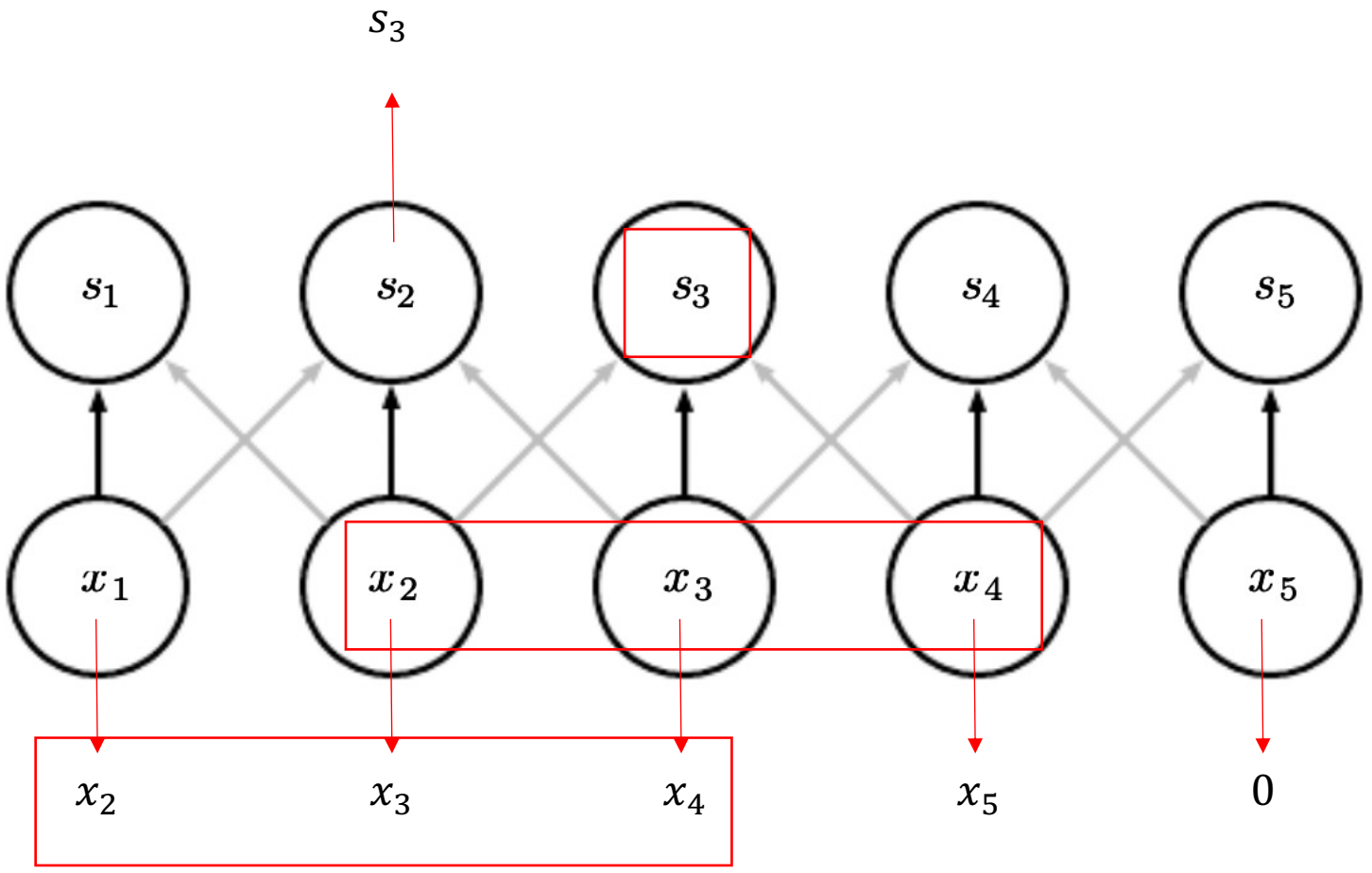
Equivariance



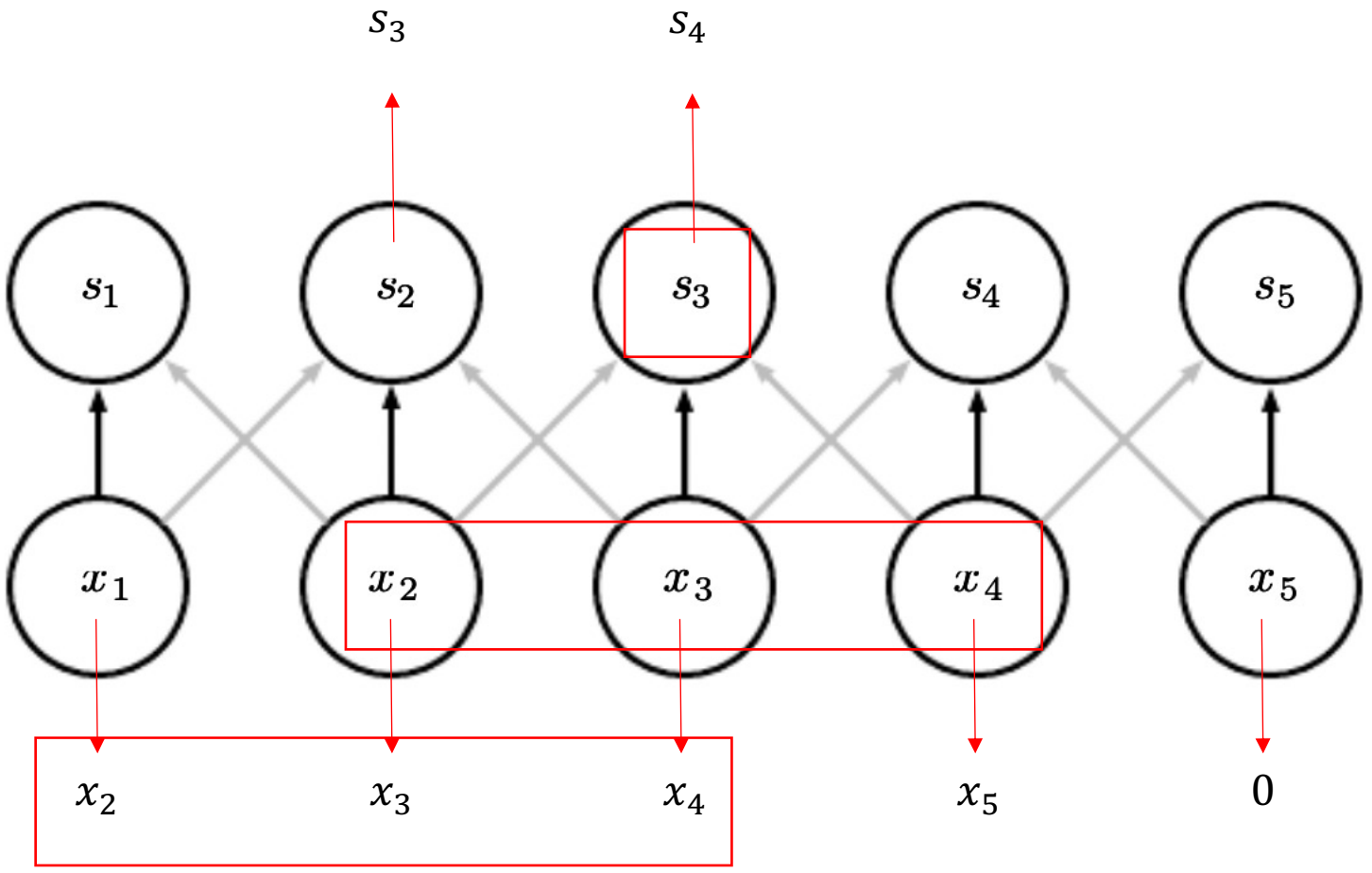
Equivariance



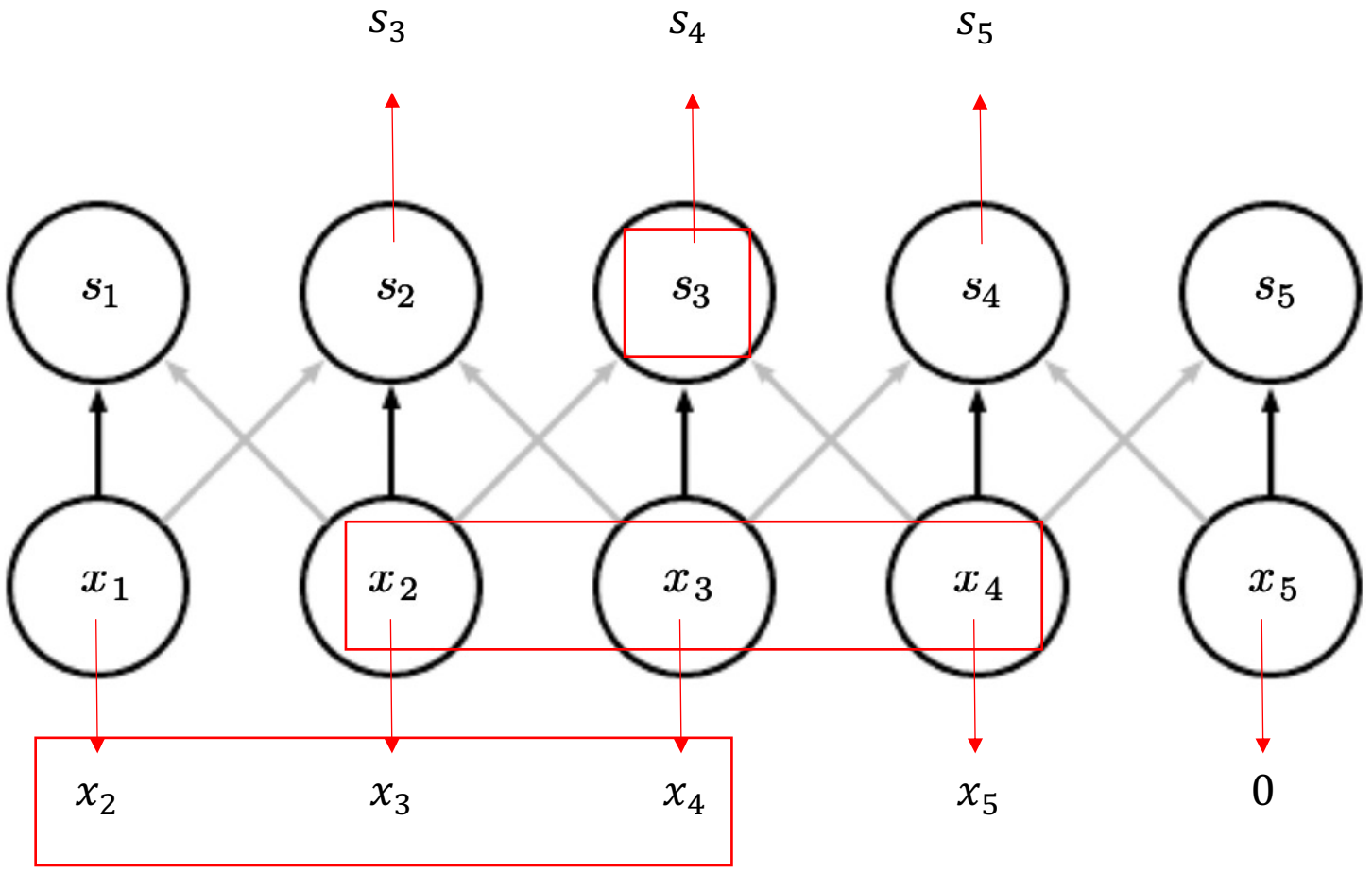
Equivariance



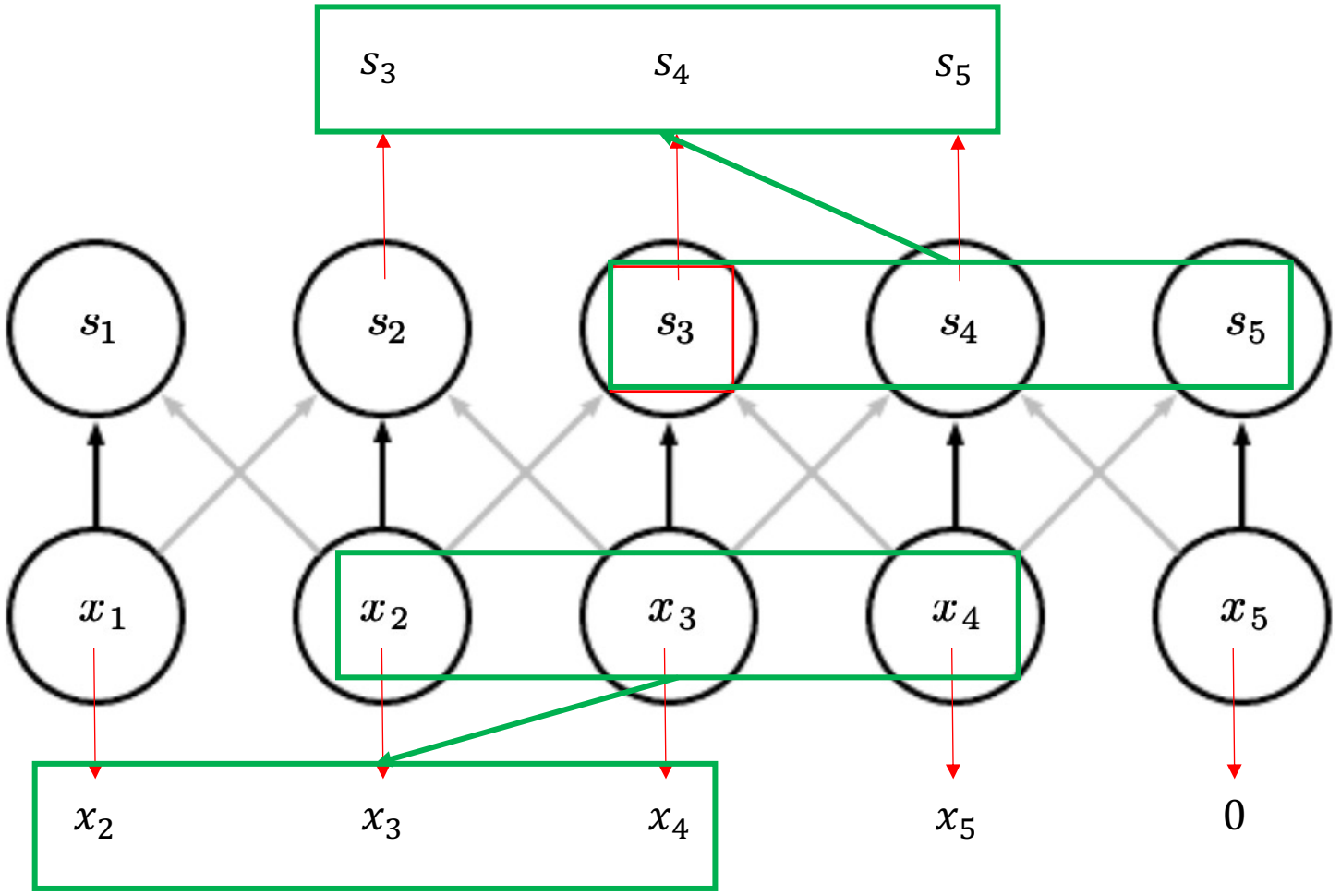
Equivariance



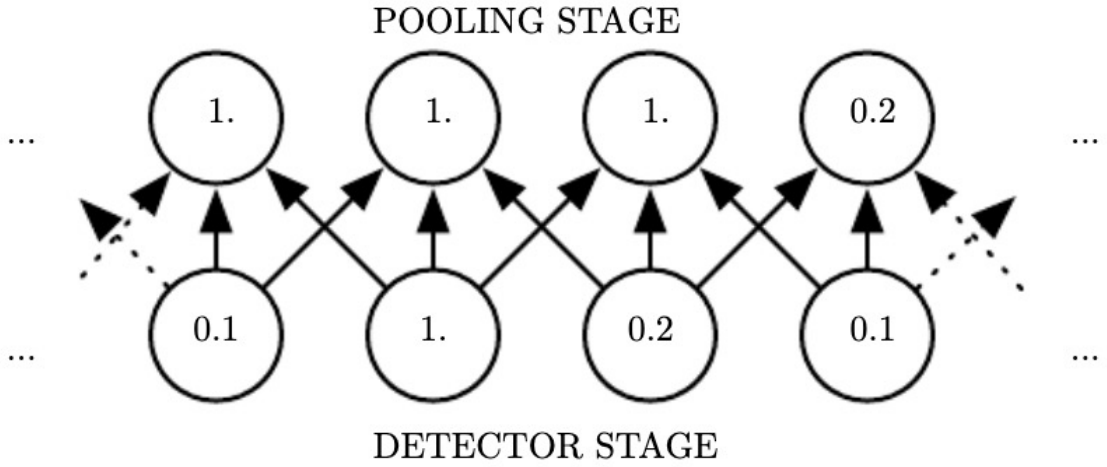
Equivariance

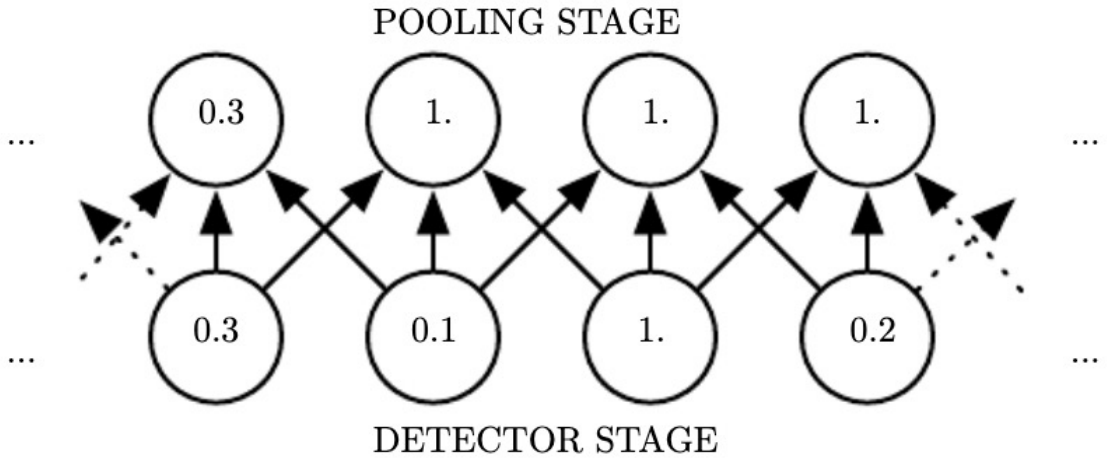


Equivariance

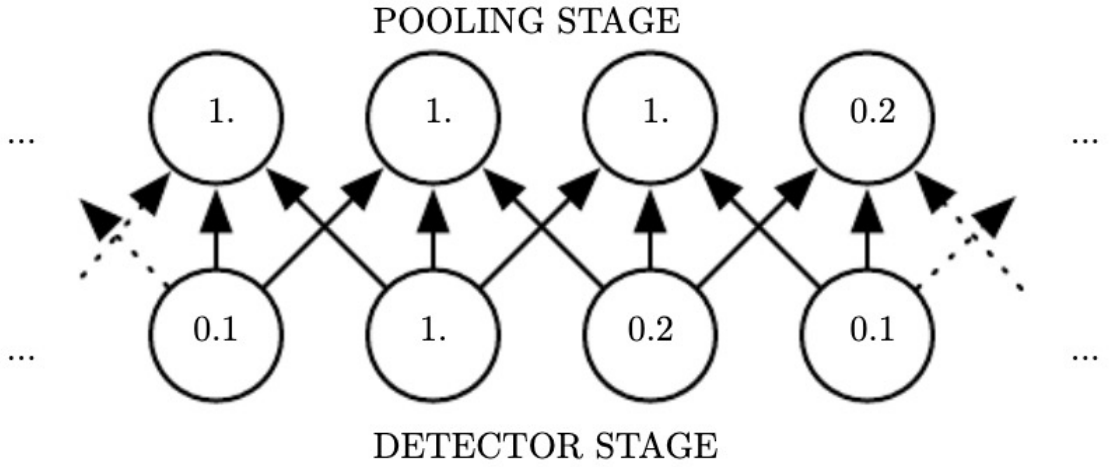


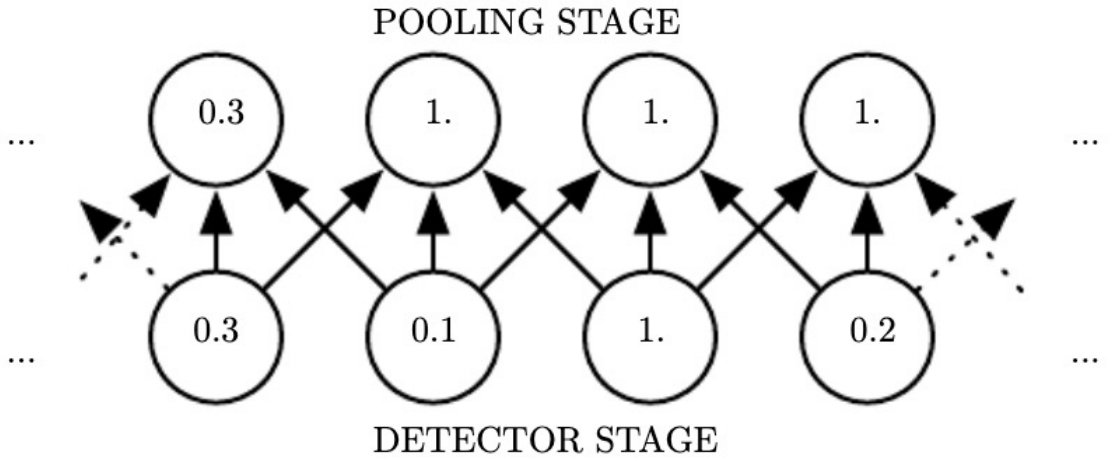
Pooling: invariance to small translation





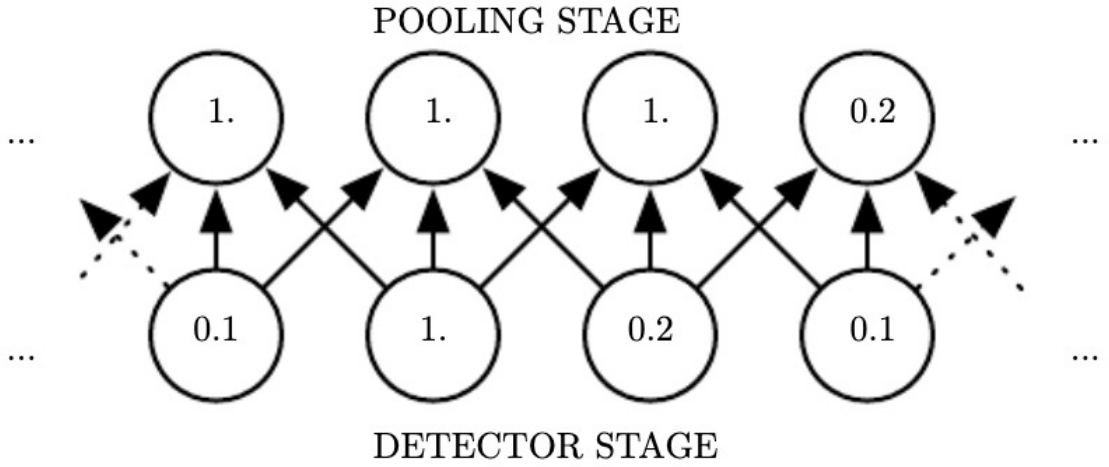
Pooling: invariance to small translation

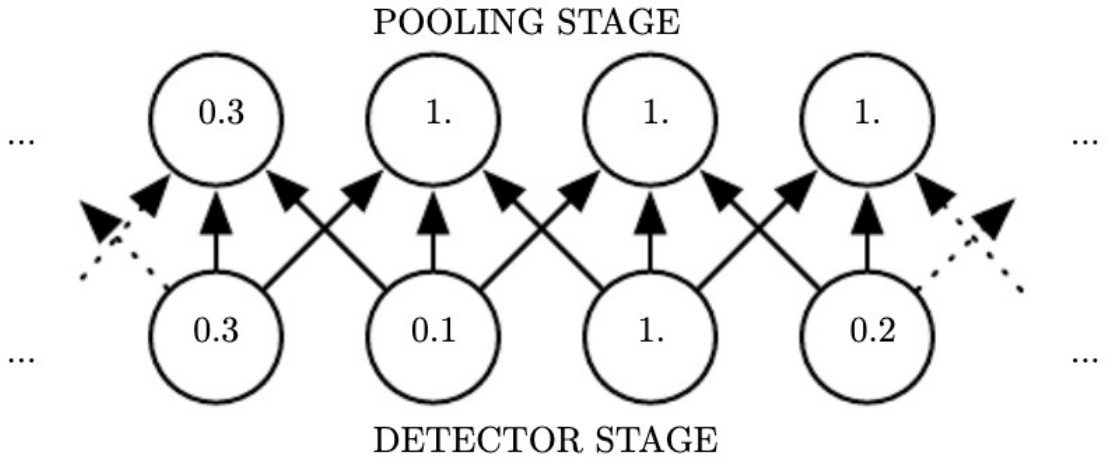




Q: what is type of pooling?
Max or average pooling?

Pooling: invariance to small translation

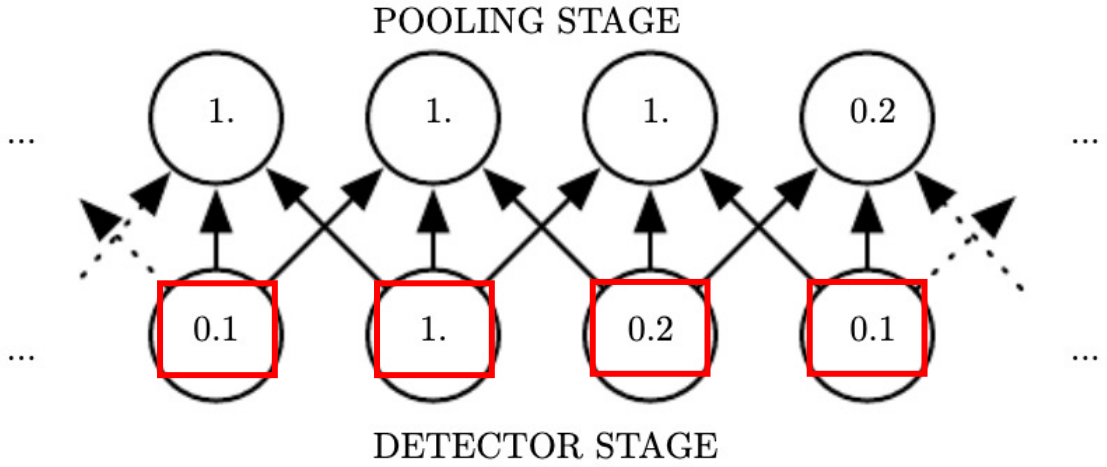


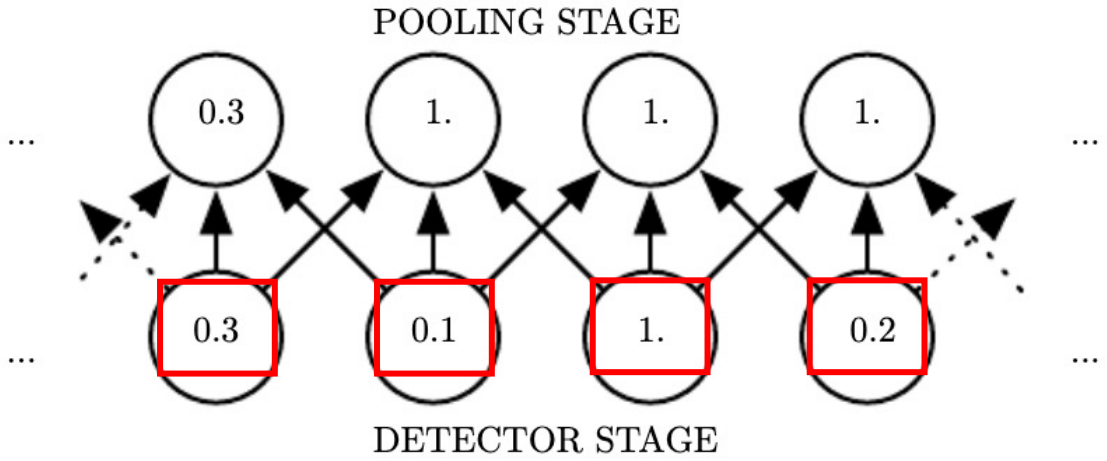


Q: what is type of pooling?
Max or average pooling?

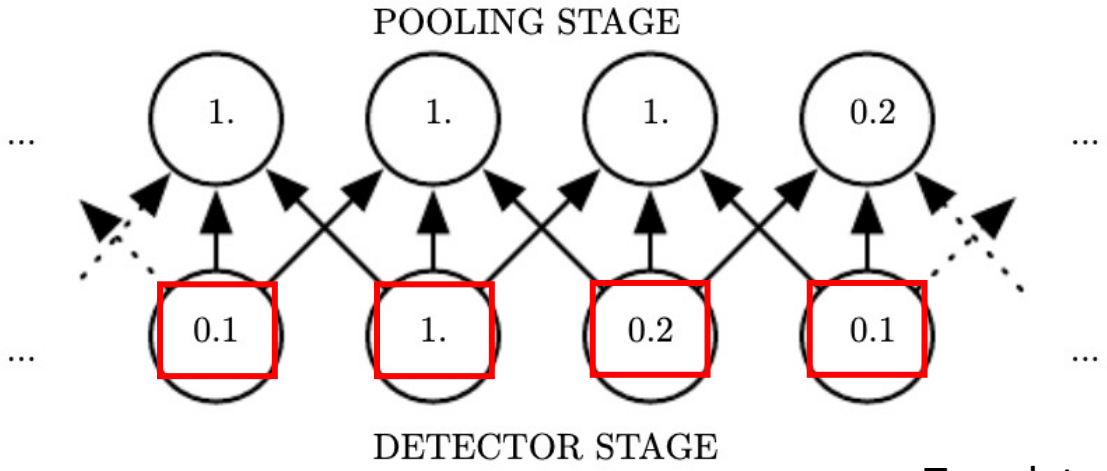
Max pooling

Pooling: invariance to small translation

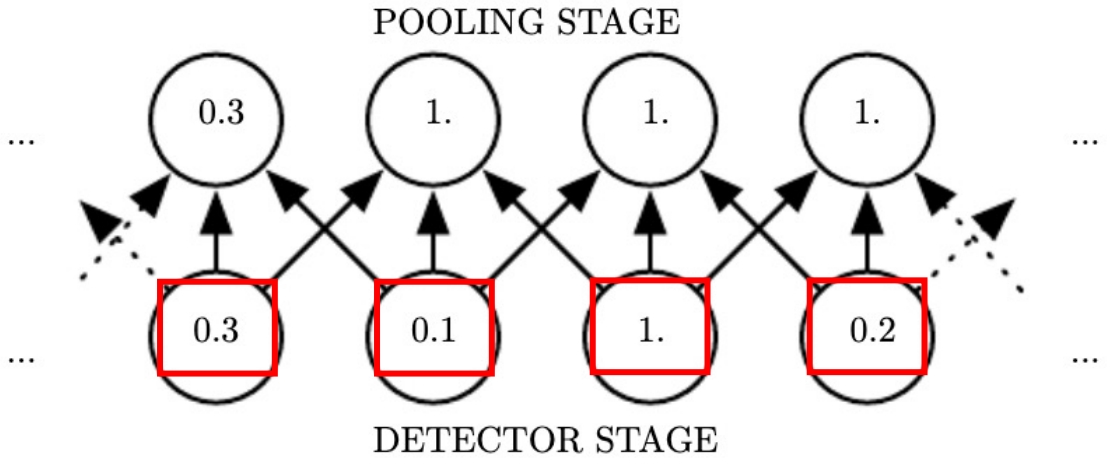




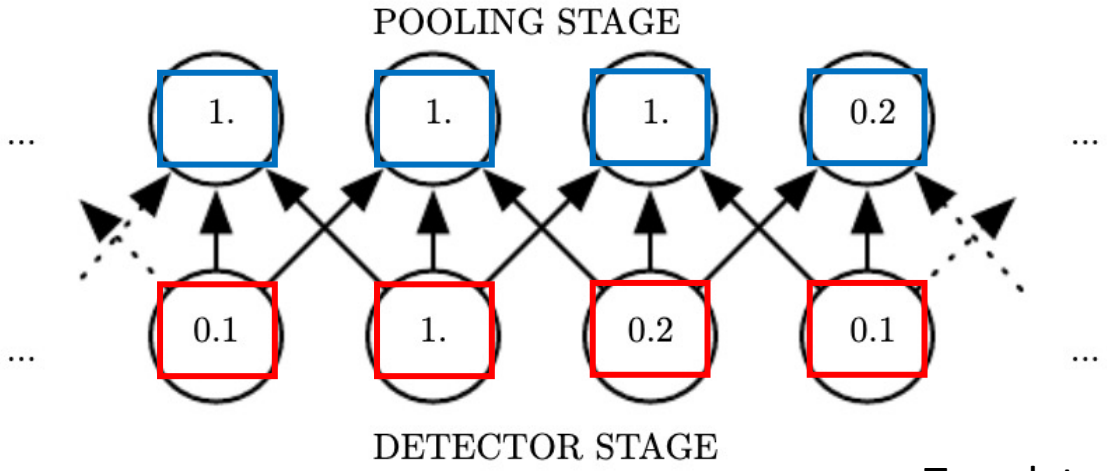
Pooling: invariance to small translation



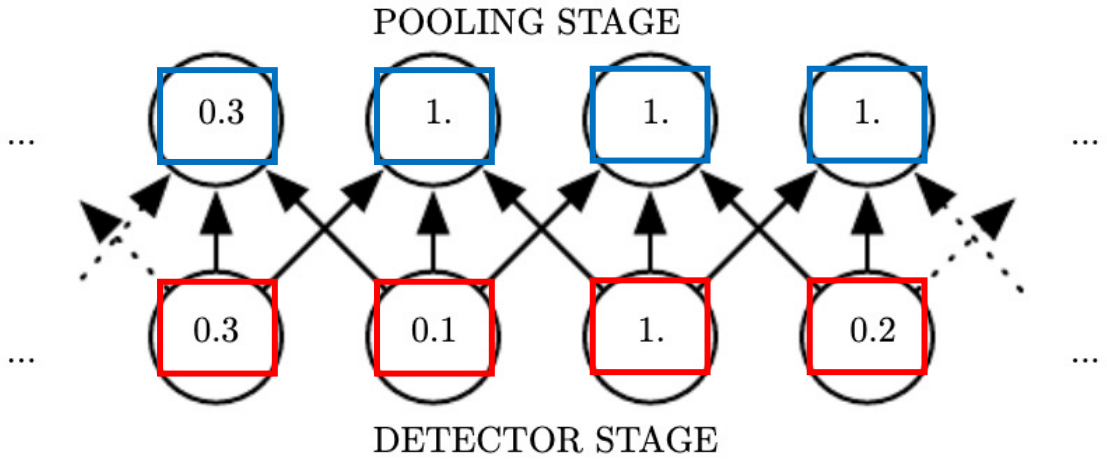
Translate: change the input value a little bit
+ change their positions



Pooling: invariance to small translation



Translate: change the input value a little bit
+ change their positions



Seagull = 1?

Seagull = 1?

Seagull is in the center

Seagull = 1?

Seagull = 1?

Seagulls are present, but not in the center

Seagull = 1?

Seagulls are present, but not in the center

Seagull = 1?

Seagulls are present, but not in the center

CNNs can tell:
whether seagulls are present
Not tell:
their positions in the image

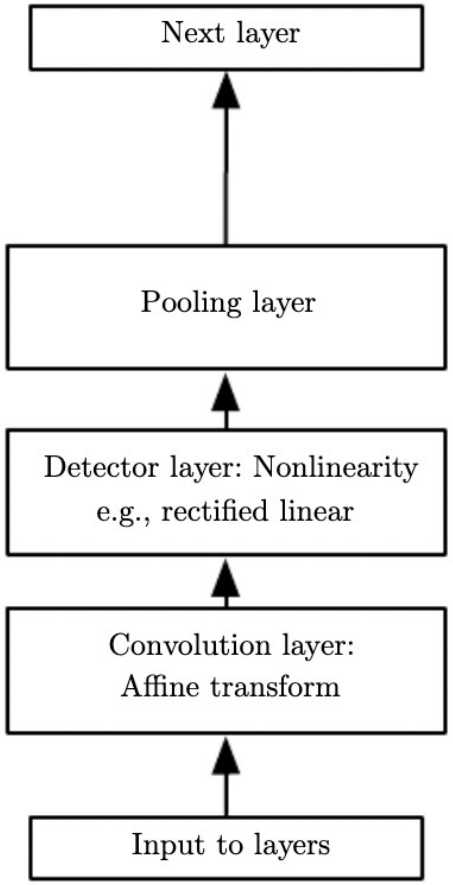
Seagull = 1?

Seagulls are present, but not in the center

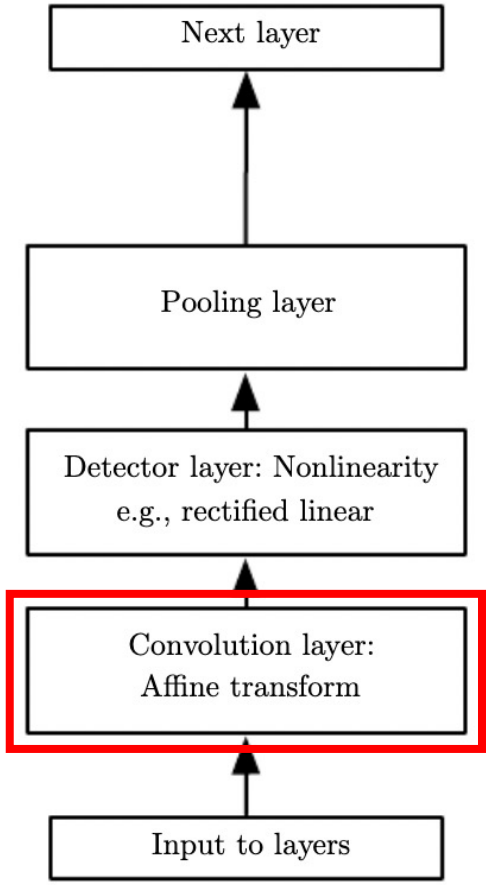
CNNs can tell:
whether seagulls are present
Not tell:
their positions in the image

Invariant to small translations

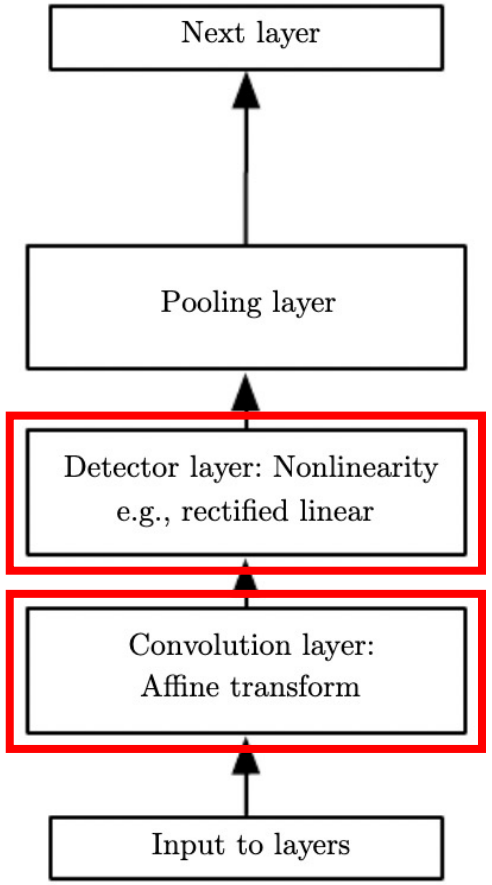
A typical convolutional layer



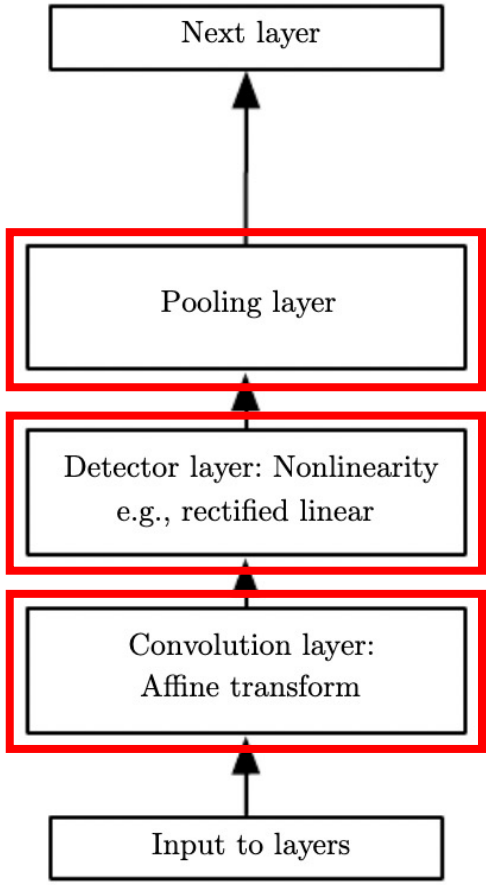
A typical convolutional layer



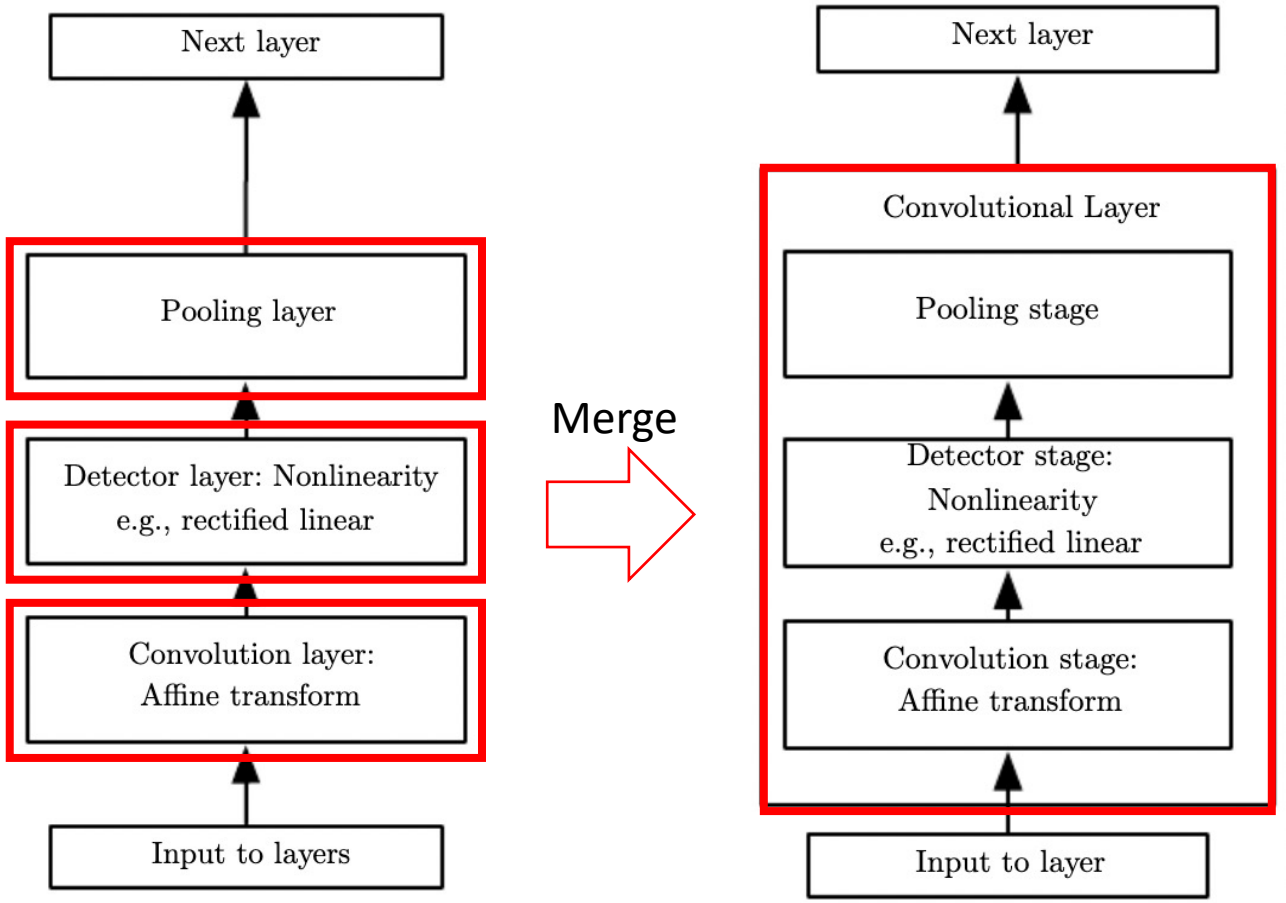
A typical convolutional layer

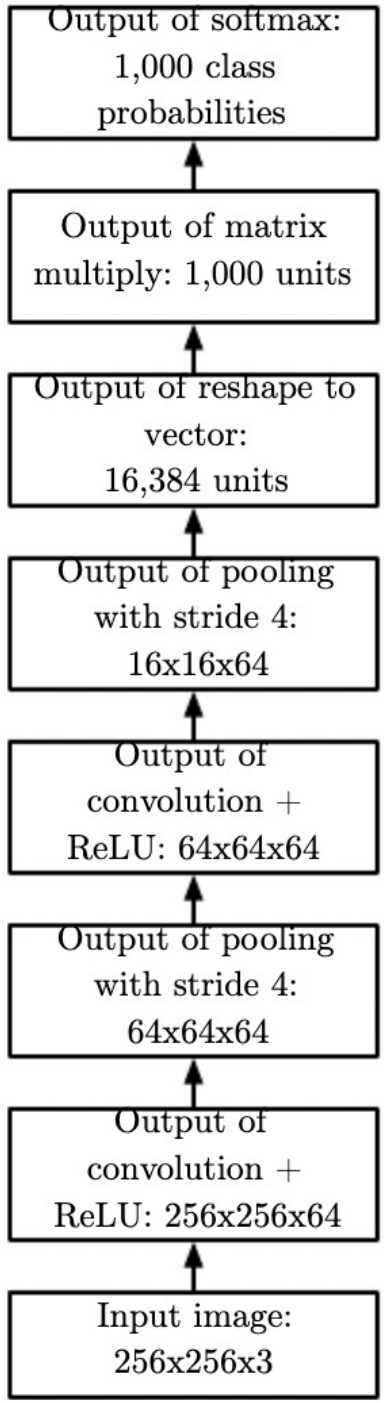


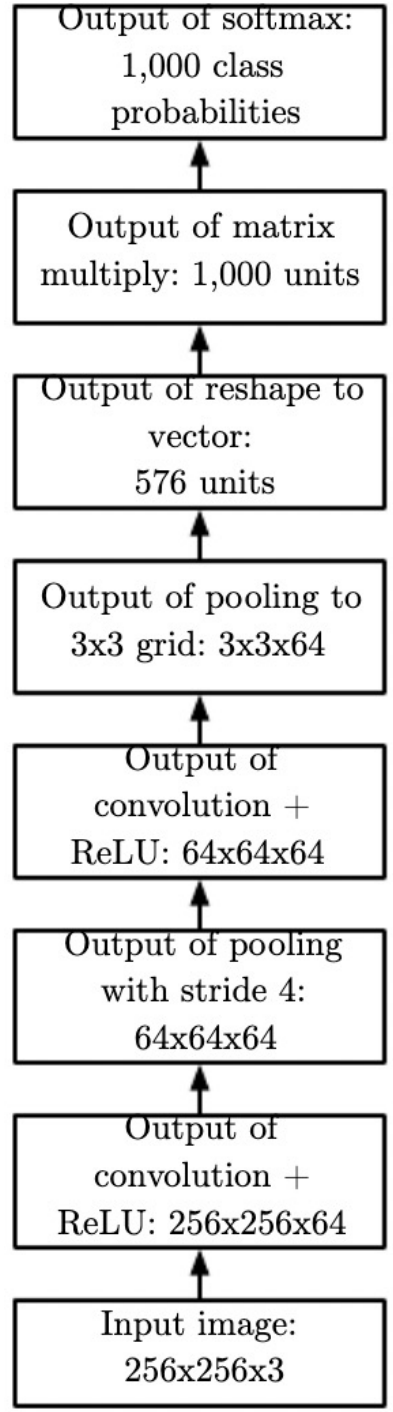
A typical convolutional layer

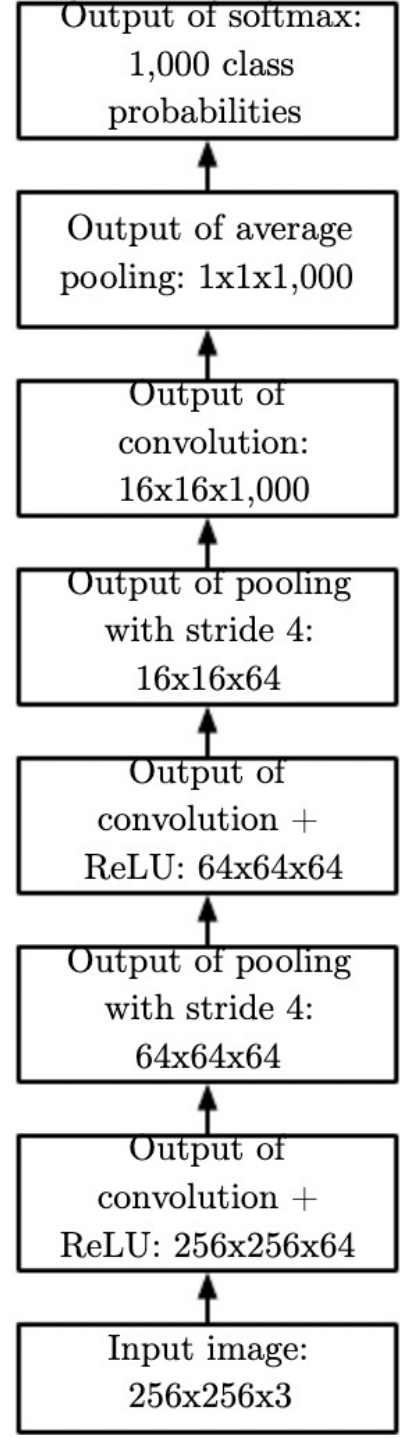


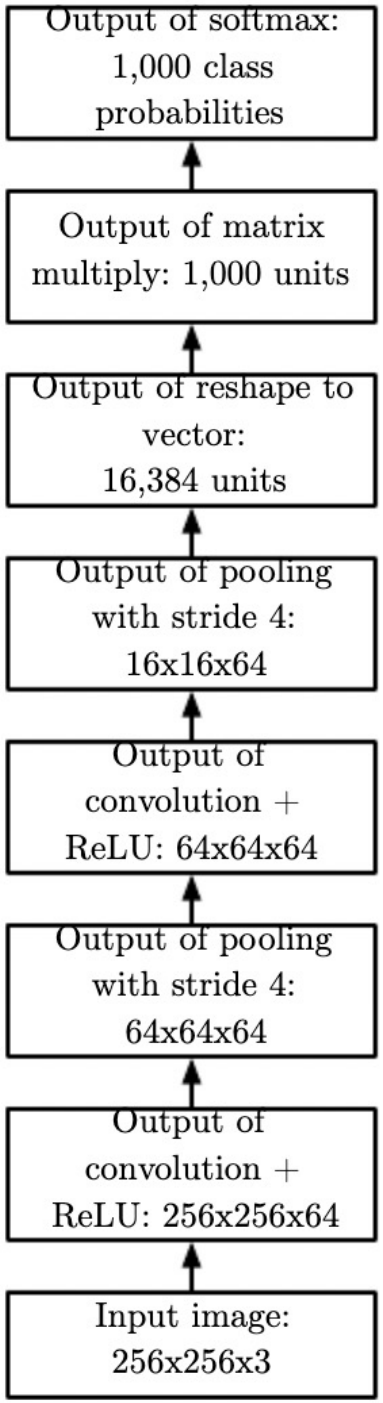
A typical convolutional layer

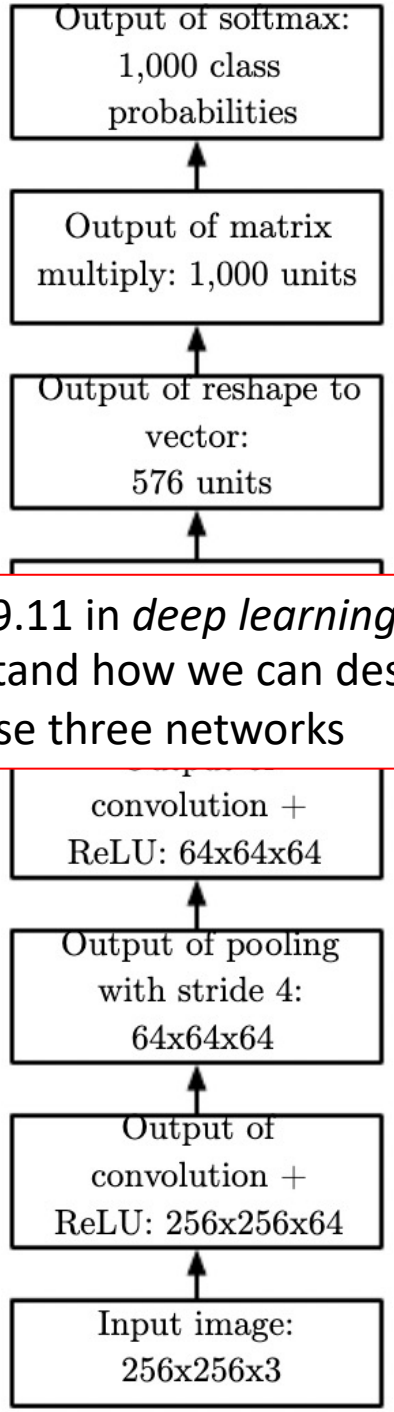



Output of softmax: 1,000 class probabilities
Output of matrix multiply: 1,000 units
Output of reshape to vector: 16,384 units
Output of pooling with stride 4: 16x16x64
Output of convolution + ReLU: 64x64x64
Output of pooling with stride 4: 64x64x64
Output of convolution + ReLU: 256x256x64
Input image: 256x256x3


Output of softmax: 1,000 class probabilities
Output of matrix multiply: 1,000 units
Output of reshape to vector: 576 units
Output of pooling to 3x3 grid: 3x3x64
Output of convolution + ReLU: 64x64x64
Output of pooling with stride 4: 64x64x64
Output of convolution + ReLU: 256x256x64
Input image: 256x256x3


Output of softmax: 1,000 class probabilities
Output of average pooling: 1x1x1,000
Output of convolution: 16x16x1,000
Output of pooling with stride 4: 16x16x64
Output of convolution + ReLU: 64x64x64
Output of pooling with stride 4: 64x64x64
Output of convolution + ReLU: 256x256x64
Input image: 256x256x3





Read: Figure 9.11 in *deep learning* book
Try to understand how we can design architecture and build these three networks