

# Convolutional Networks

Lecture slides for Chapter 9 of *Deep Learning*  
Ian Goodfellow  
2016-09-12

# Convolutional Networks

- Scale up neural networks to process very large images / video sequences
  - Sparse connections
  - Parameter sharing
- Automatically generalize across spatial translations of inputs
- Applicable to any input that is laid out on a grid (1-D, 2-D, 3-D, ...)

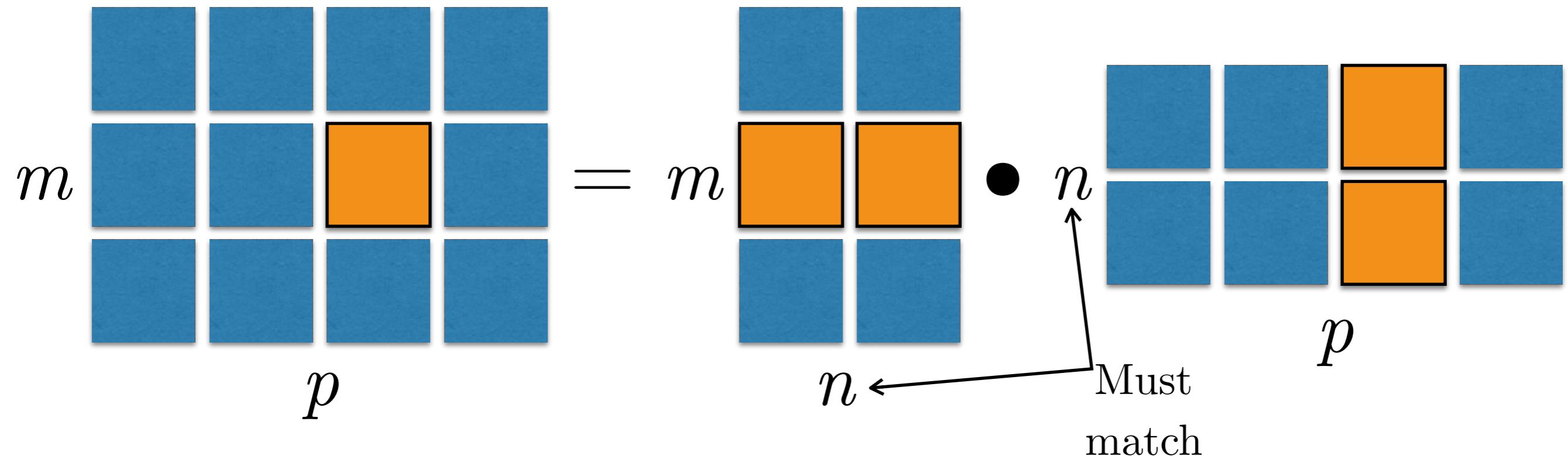
# Key Idea

- Replace matrix multiplication in neural nets with convolution
- Everything else stays the same
  - Maximum likelihood
  - Back-propagation
  - etc.

# Matrix (Dot) Product

$$C = AB. \quad (2.4)$$

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}. \quad (2.5)$$



# Matrix Transpose

$$(\mathbf{A}^\top)_{i,j} = A_{j,i}. \quad (2.3)$$

The diagram shows a 3x2 matrix  $A$  with elements  $A_{1,1}, A_{1,2}, A_{2,1}, A_{2,2}, A_{3,1}, A_{3,2}$ . A curved arrow points from the element  $A_{1,2}$  to its transpose position  $A_{2,1}$ , indicating the swap of indices. To the right, the transpose matrix  $\mathbf{A}^\top$  is shown as a 2x3 matrix with elements  $A_{1,1}, A_{2,1}, A_{3,1}, A_{1,2}, A_{2,2}, A_{3,2}$ .

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow \mathbf{A}^\top = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

Figure 2.1: The transpose of the matrix can be thought of as a mirror image across the main diagonal.

$$(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top. \quad (2.9)$$

# 2D Convolution

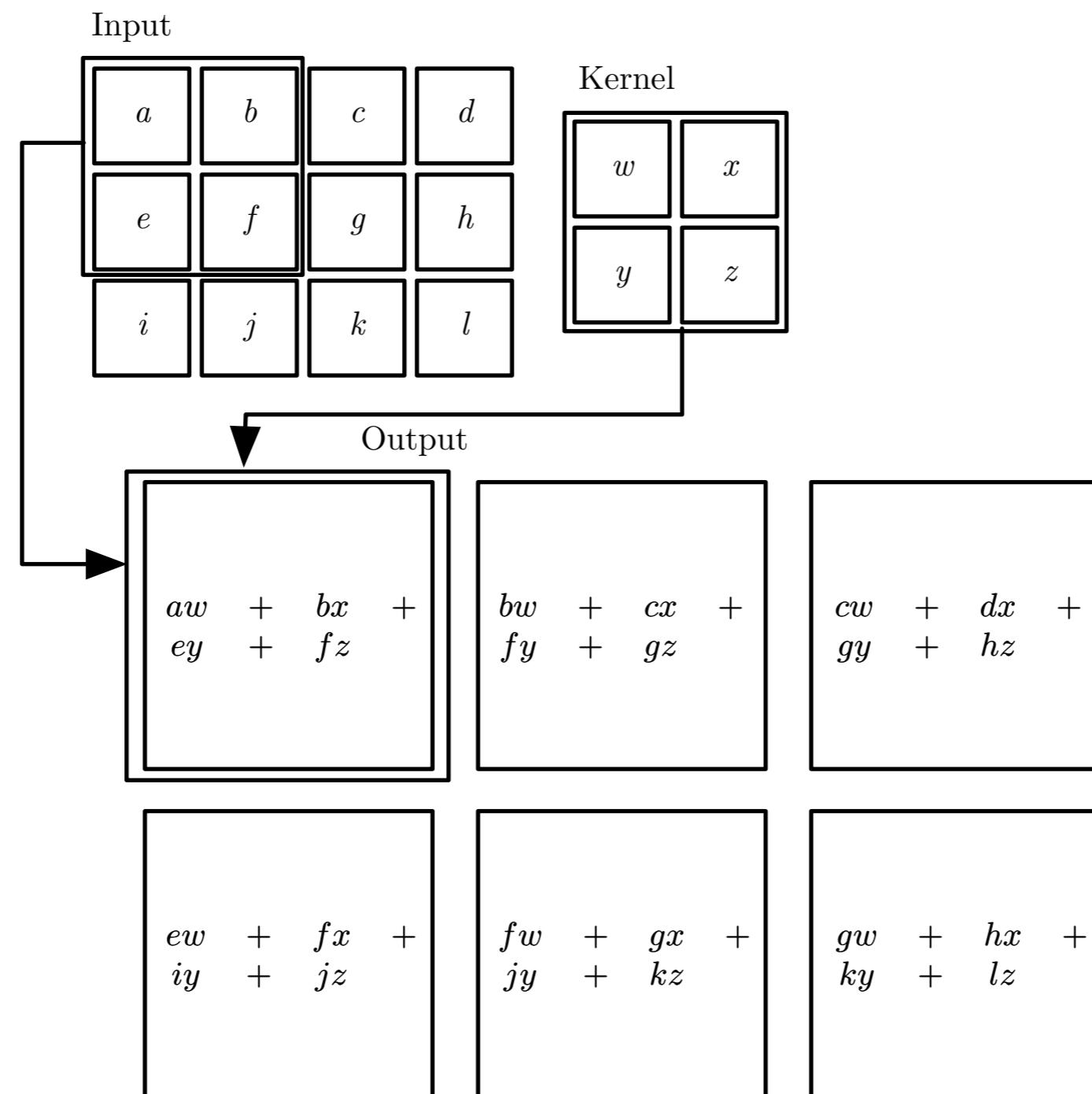


Figure 9.1

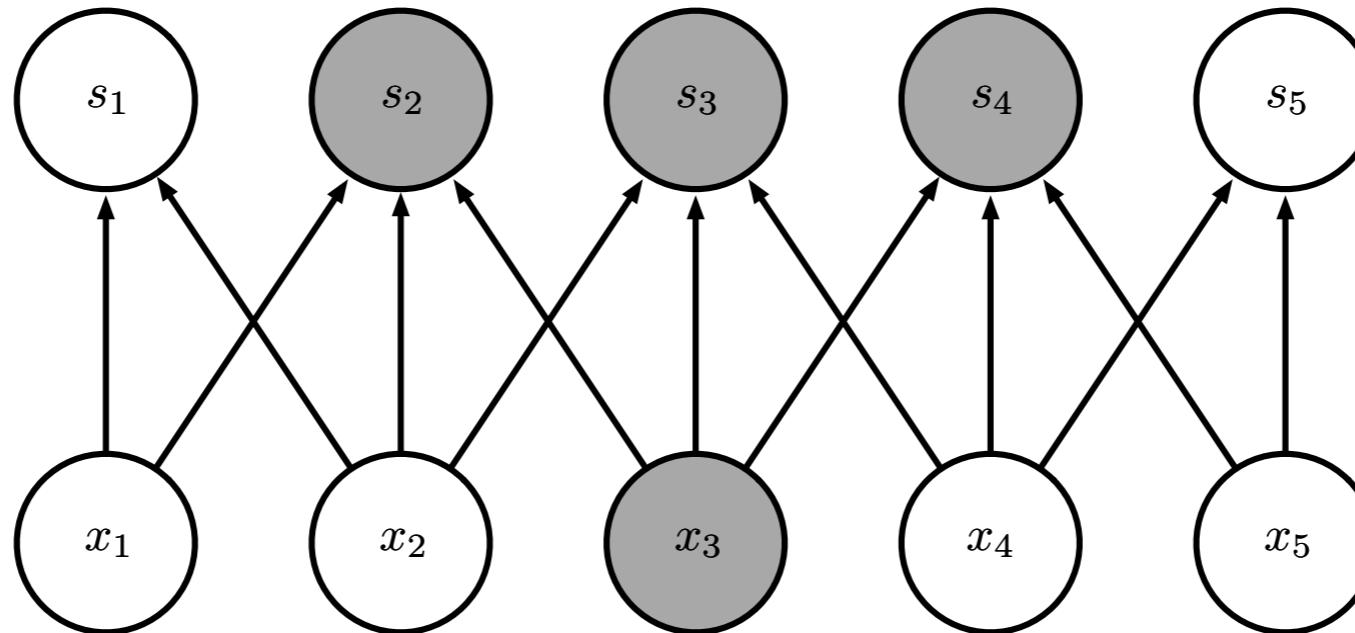
(Goodfellow 2016)

# Three Operations

- Convolution: like matrix multiplication
  - Take an input, produce an output (hidden layer)
- “Deconvolution”: like multiplication by transpose of a matrix
  - Used to back-propagate error from output to input
  - Reconstruction in autoencoder / RBM
- Weight gradient computation
  - Used to backpropagate error from output to weights
  - Accounts for the parameter sharing

# Sparse Connectivity

Sparse  
connections  
due to small  
convolution  
kernel



Dense  
connections

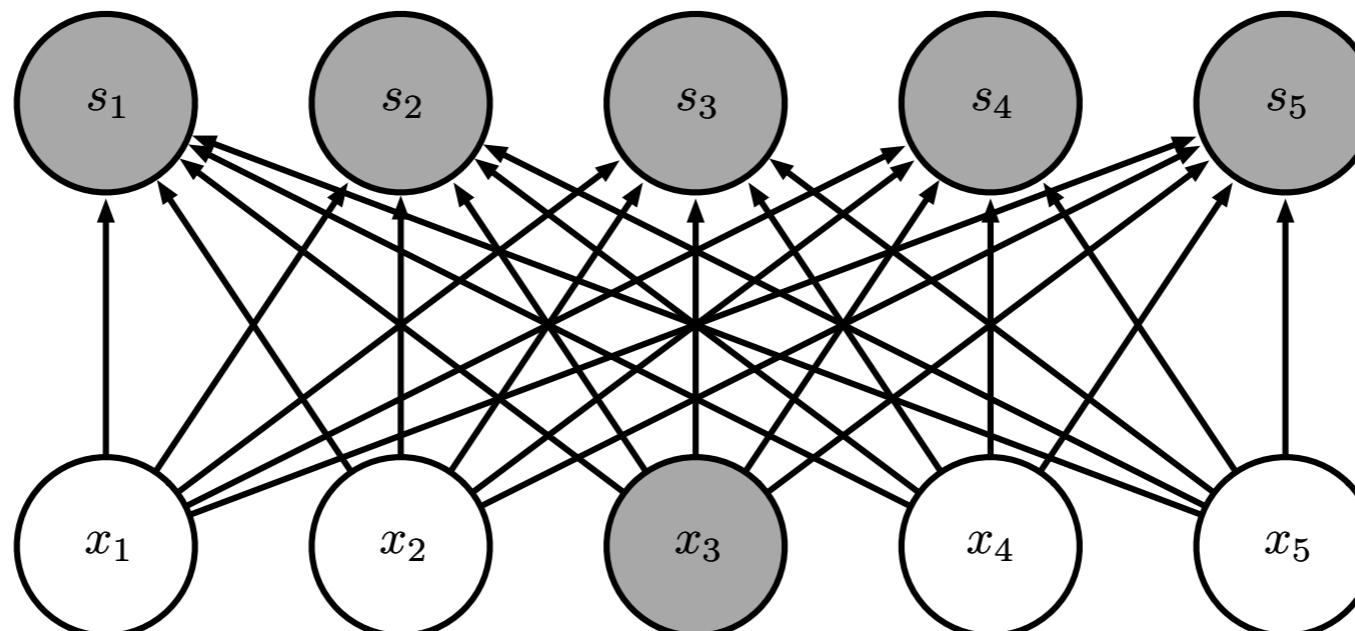
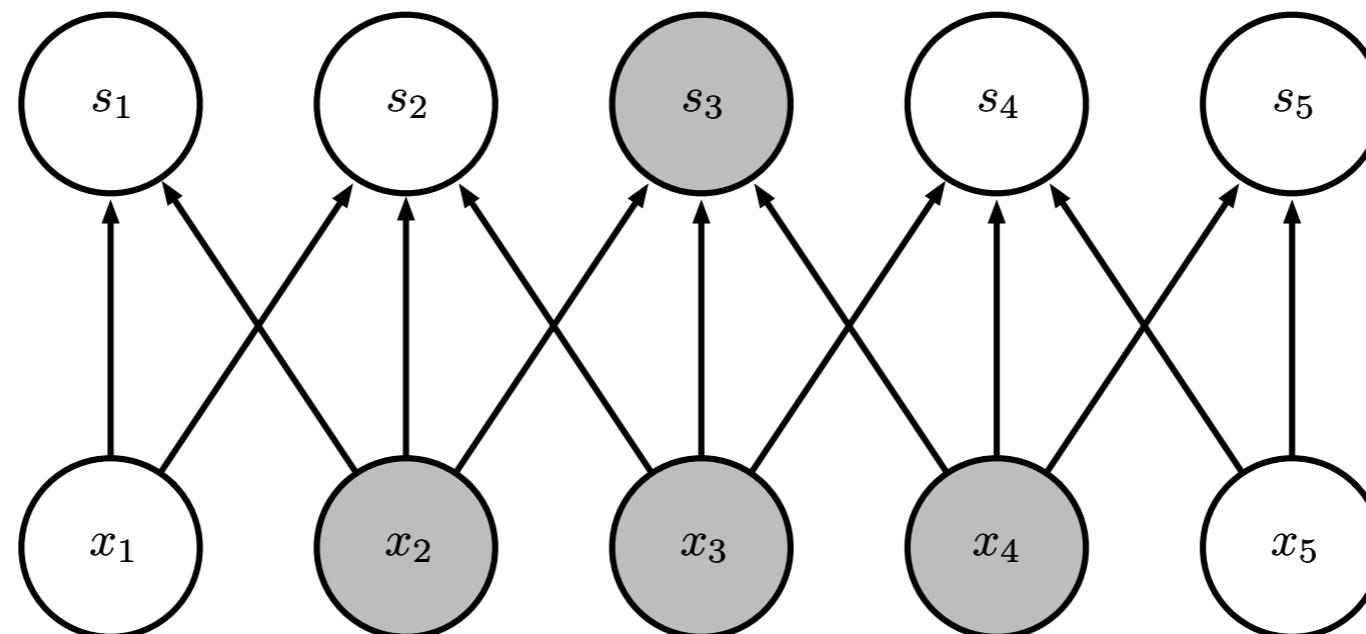


Figure 9.2

# Sparse Connectivity

Sparse  
connections  
due to small  
convolution  
kernel



Dense  
connections

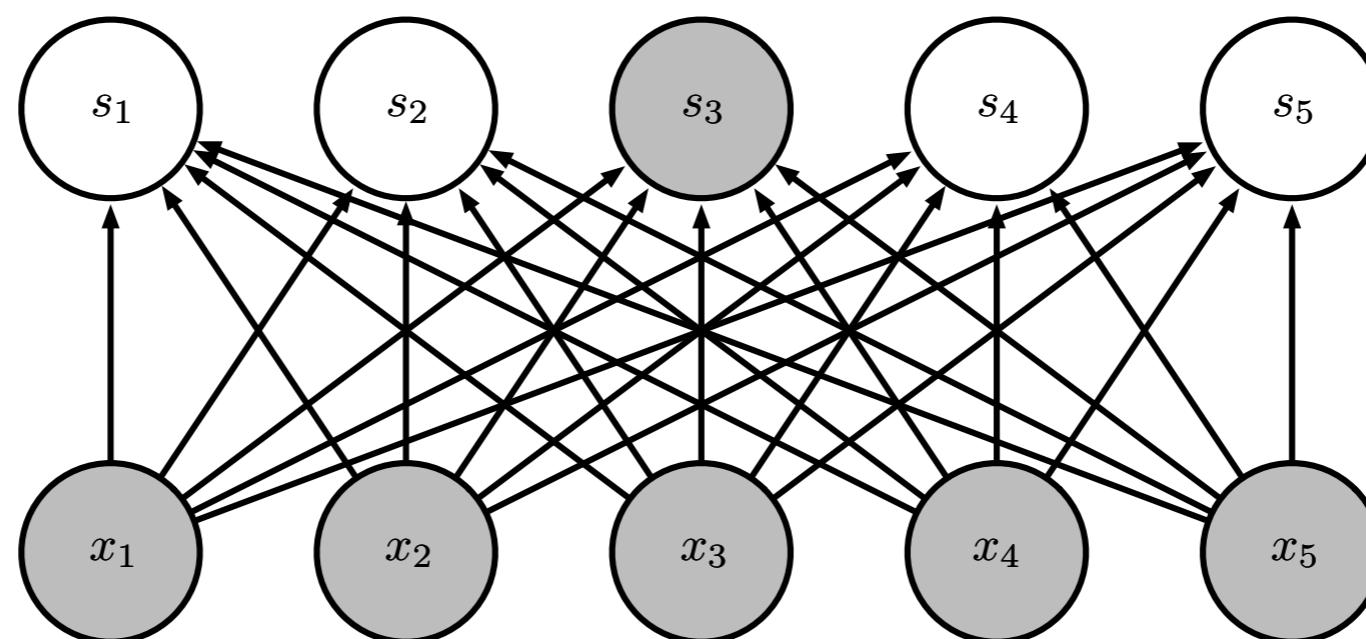


Figure 9.3

# Growing Receptive Fields

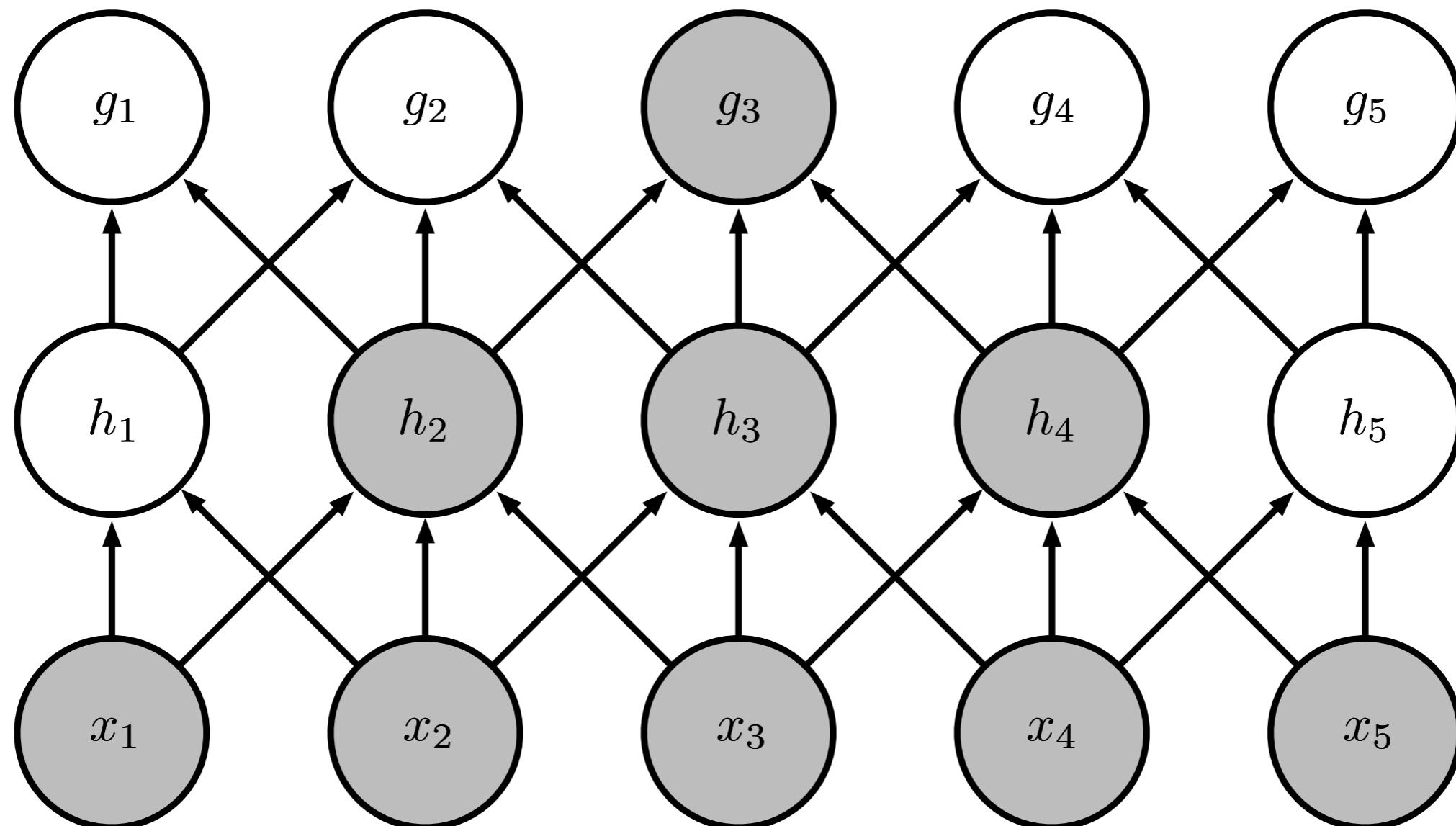


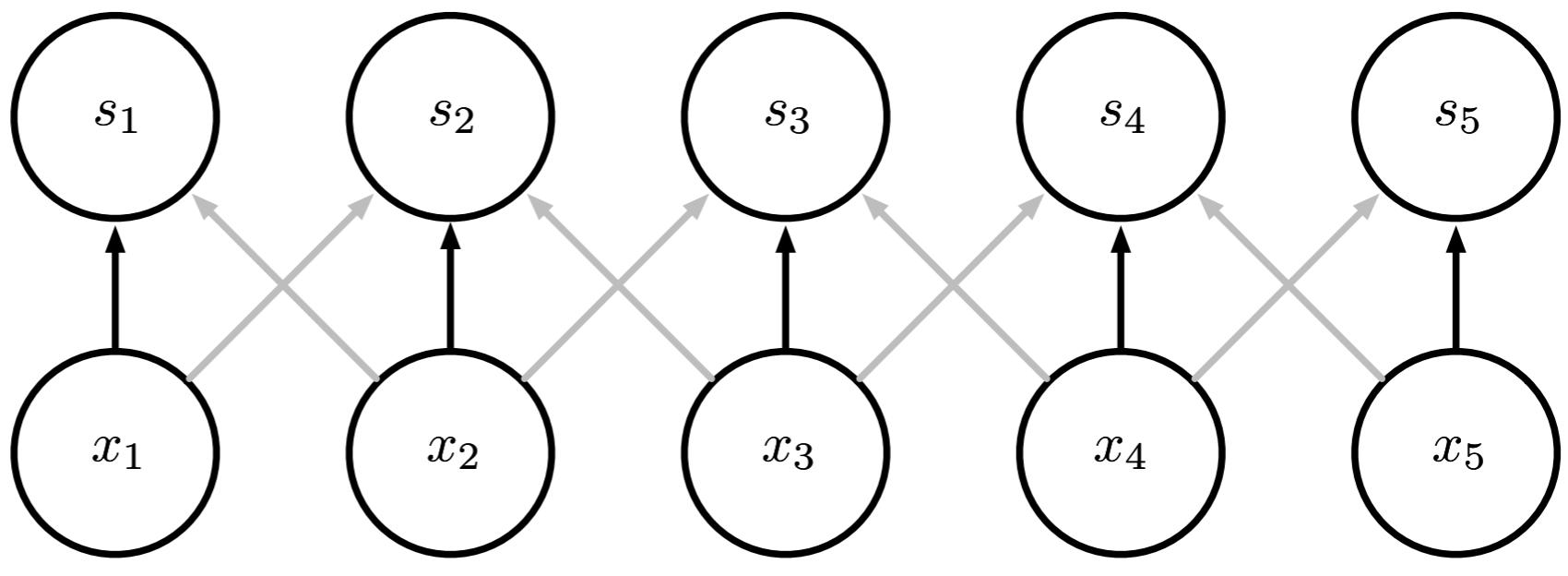
Figure 9.4

(Goodfellow 2016)

# Parameter Sharing

Convolution  
shares the same  
parameters  
across all spatial  
locations

---



Traditional  
matrix  
multiplication  
does not share  
any parameters

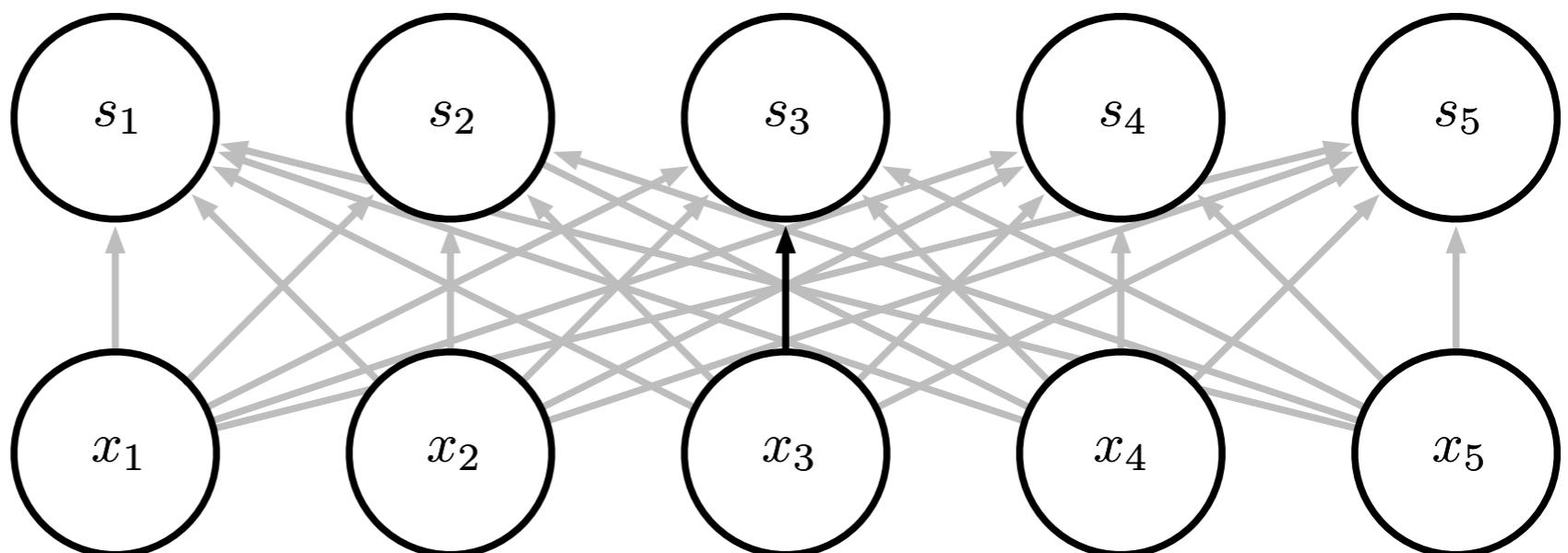


Figure 9.5

# Edge Detection by Convolution

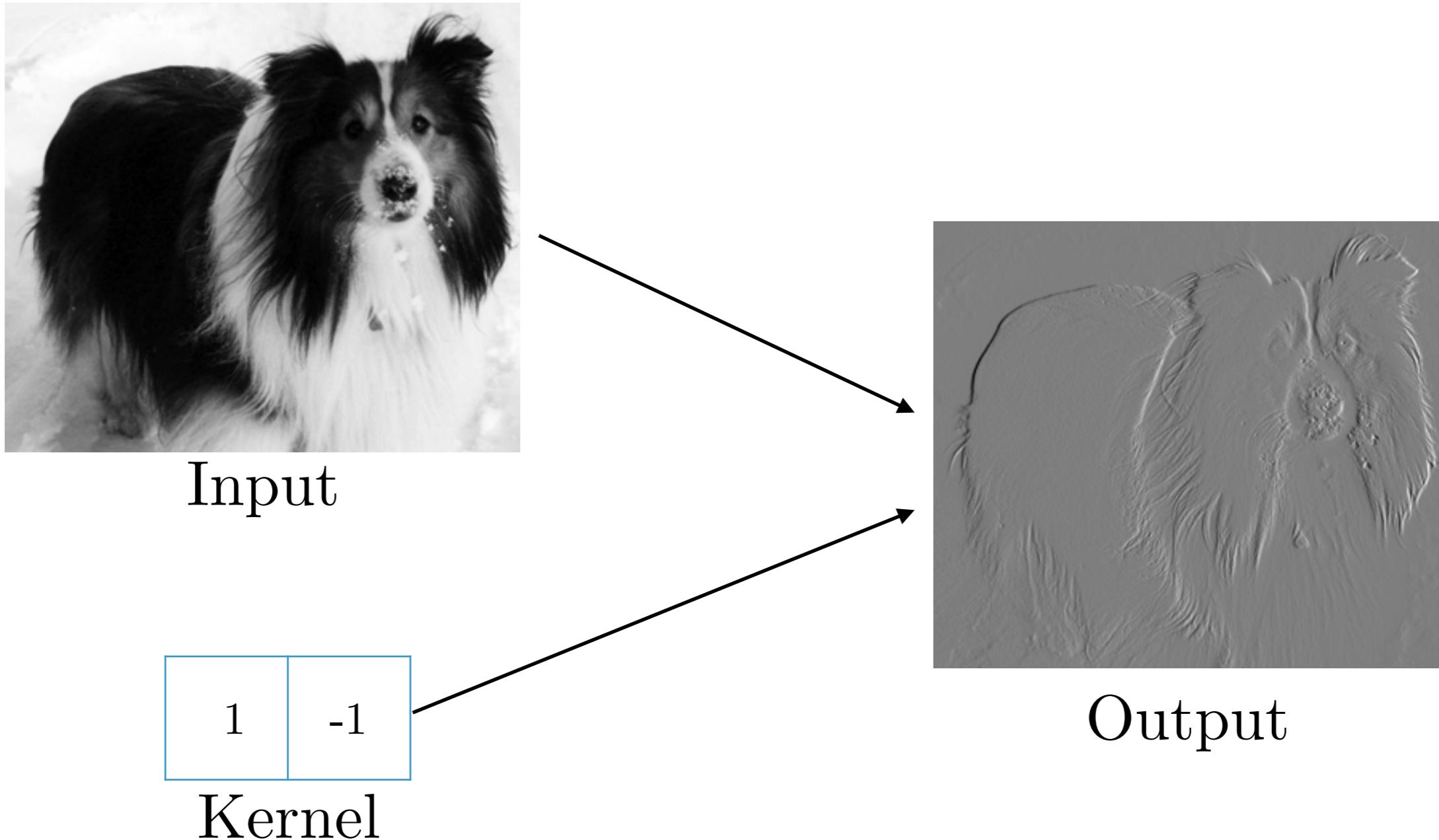


Figure 9.6

(Goodfellow 2016)

# Efficiency of Convolution

Input size: 320 by 280

Kernel size: 2 by 1

Output size: 319 by 280

	Convolution	Dense matrix	Sparse matrix
Stored floats	2 <sup>319*280*320*280</sup> $> 8e9$	$319*280*320*280$ $> 8e9$	$2*319*280 =$ 178,640
Float muls or adds	$319*280*3 =$ 267,960	$> 16e9$	Same as convolution (267,960)

# Convolutional Network Components

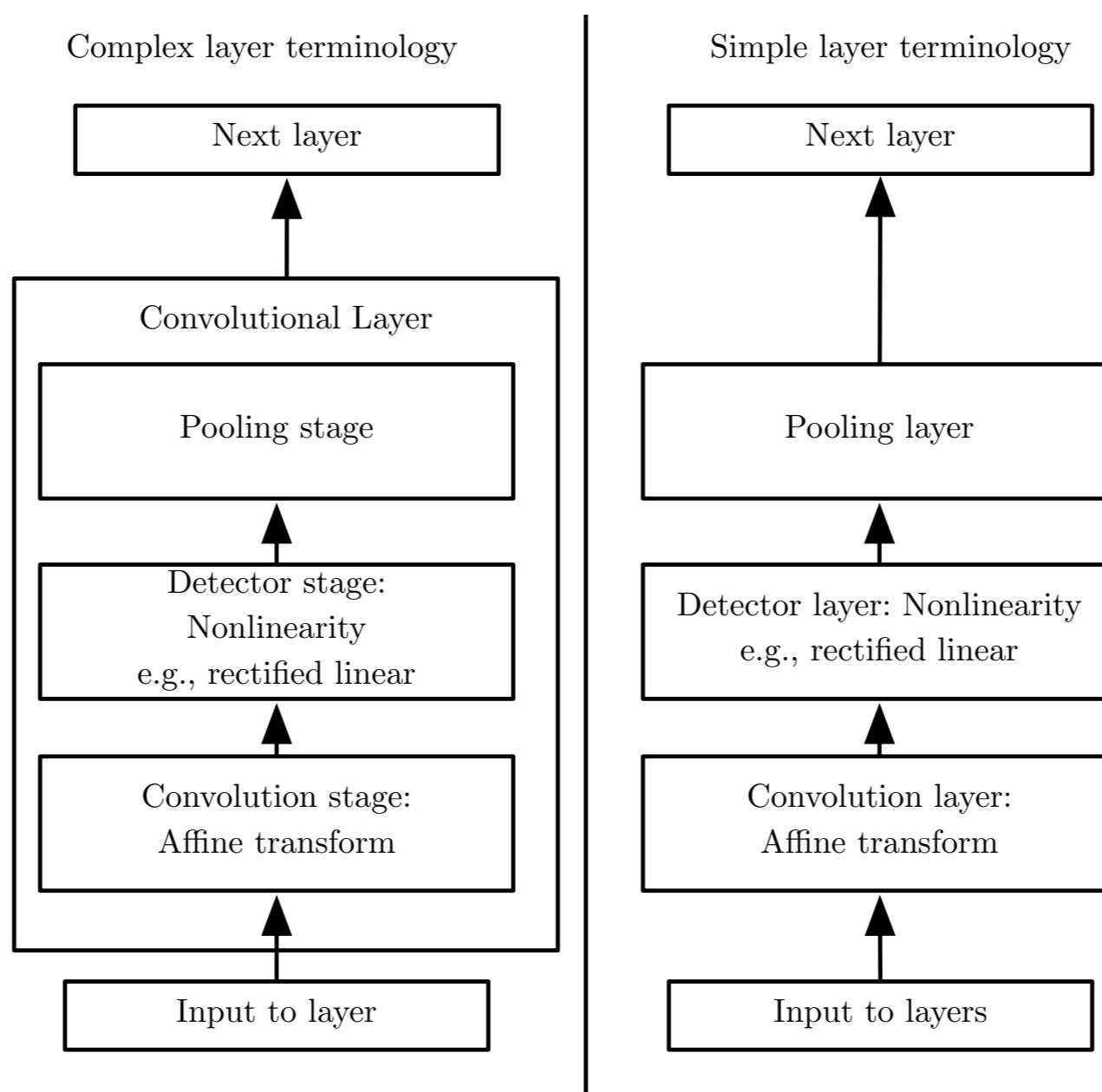


Figure 9.7

(Goodfellow 2016)

# Max Pooling and Invariance to Translation

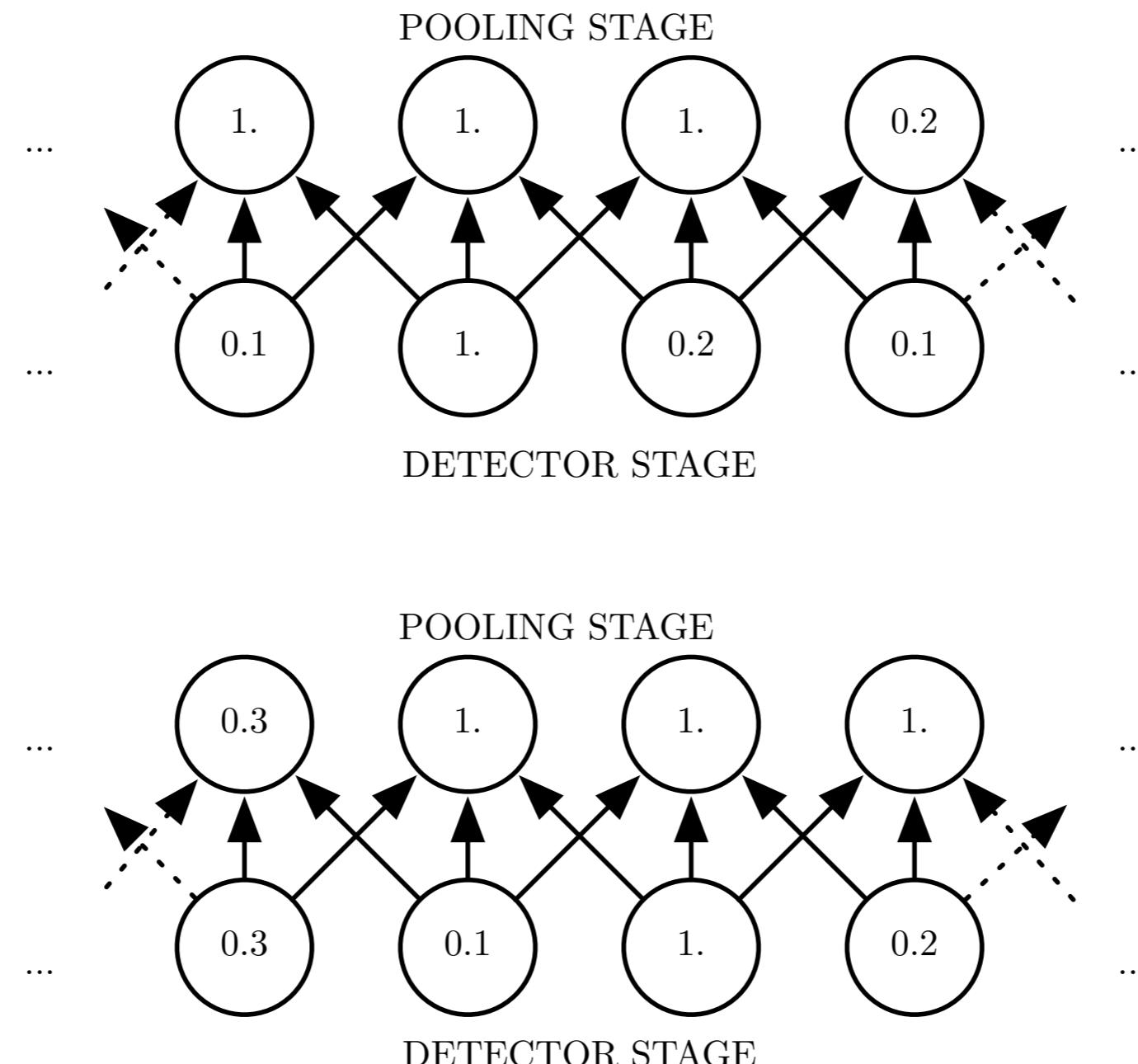


Figure 9.8

(Goodfellow 2016)

# Cross-Channel Pooling and Invariance to Learned Transformations

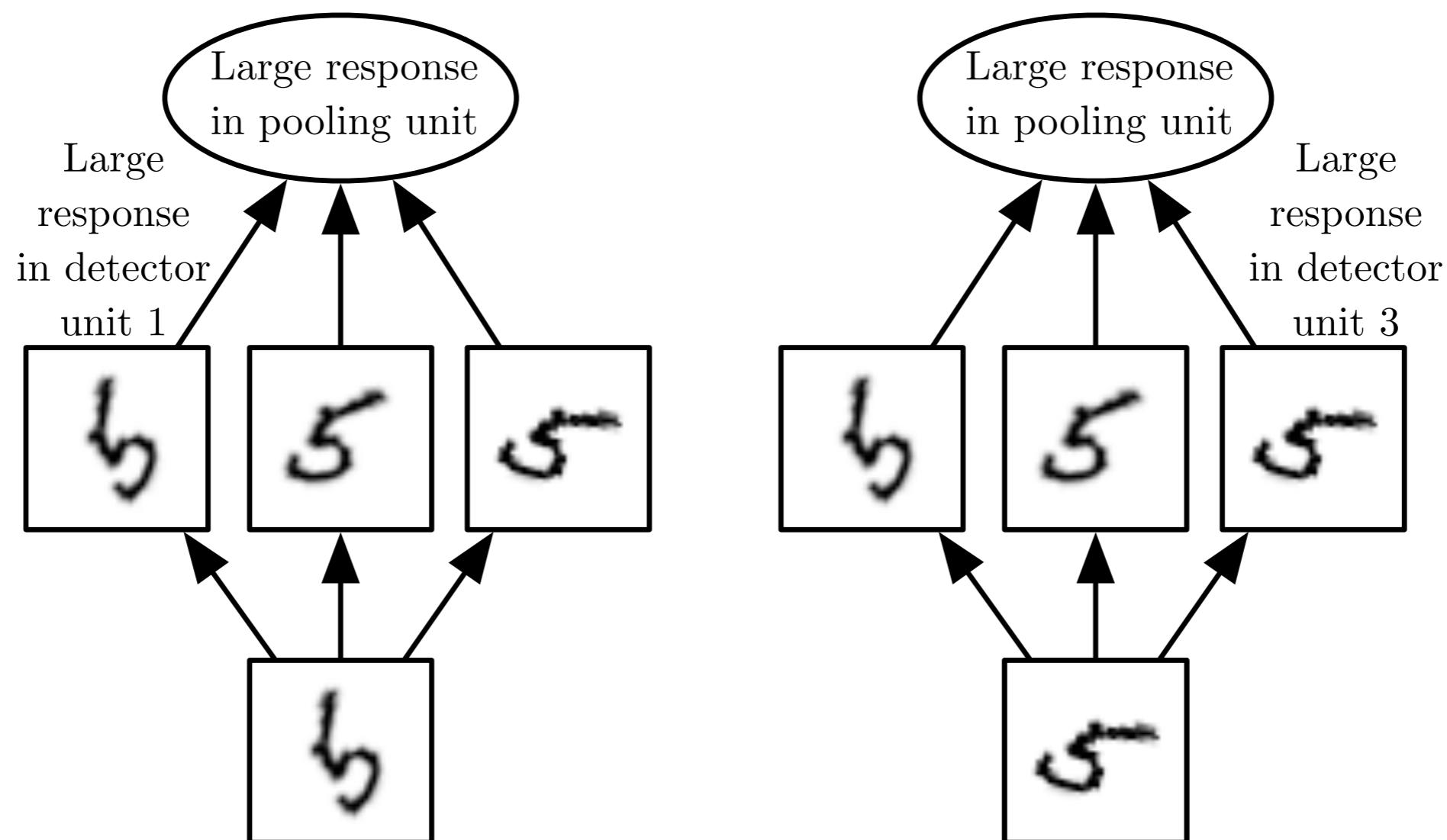


Figure 9.9

(Goodfellow 2016)

# Pooling with Downsampling

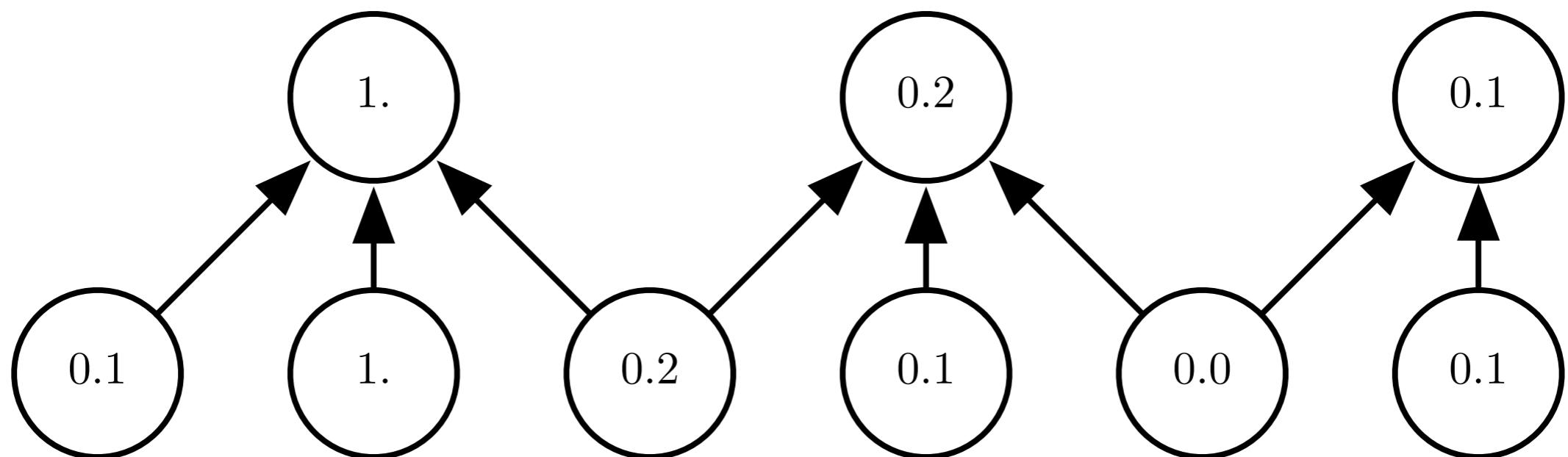


Figure 9.10

(Goodfellow 2016)

# Example Classification Architectures

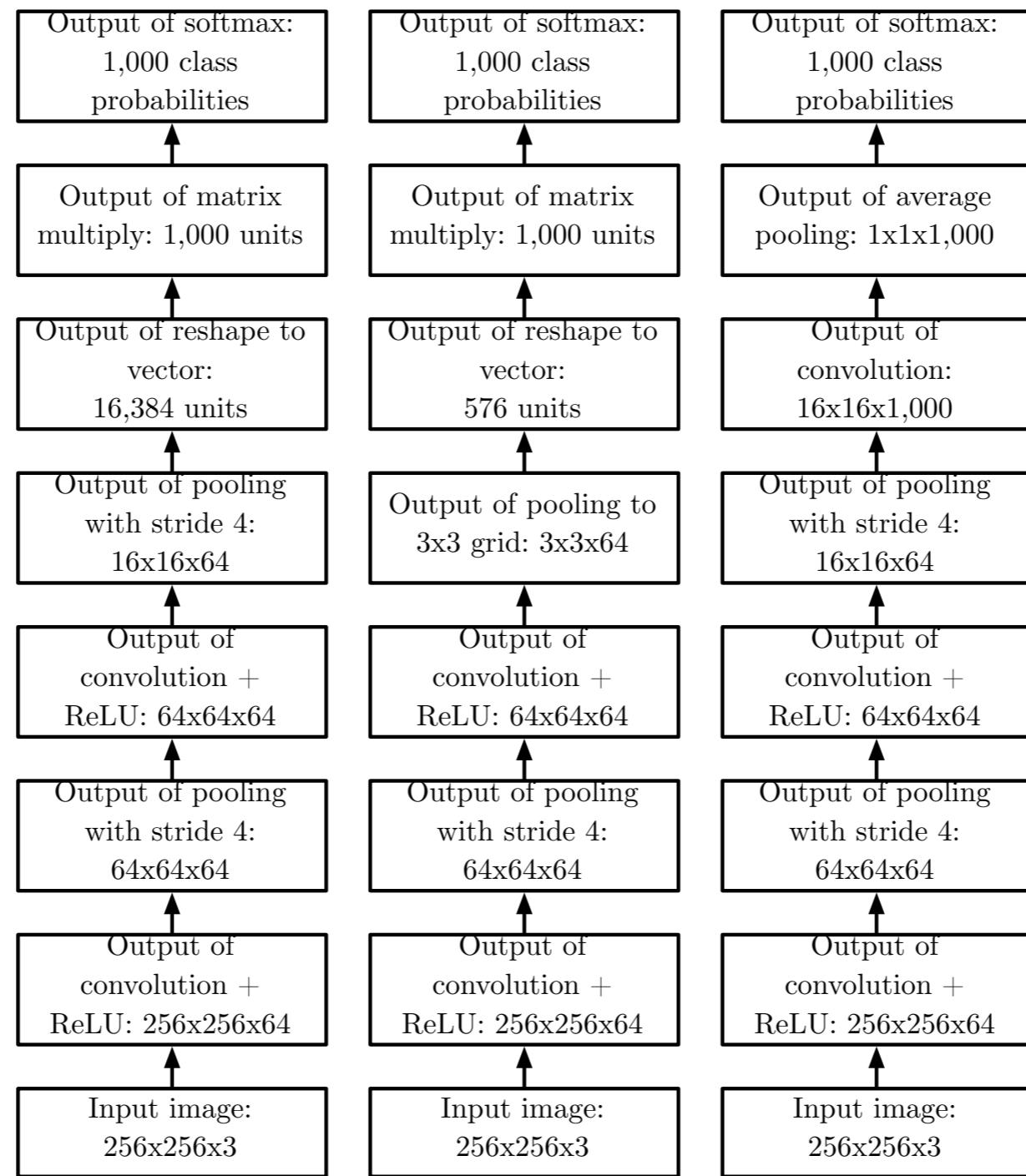


Figure 9.11

# Convolution with Stride

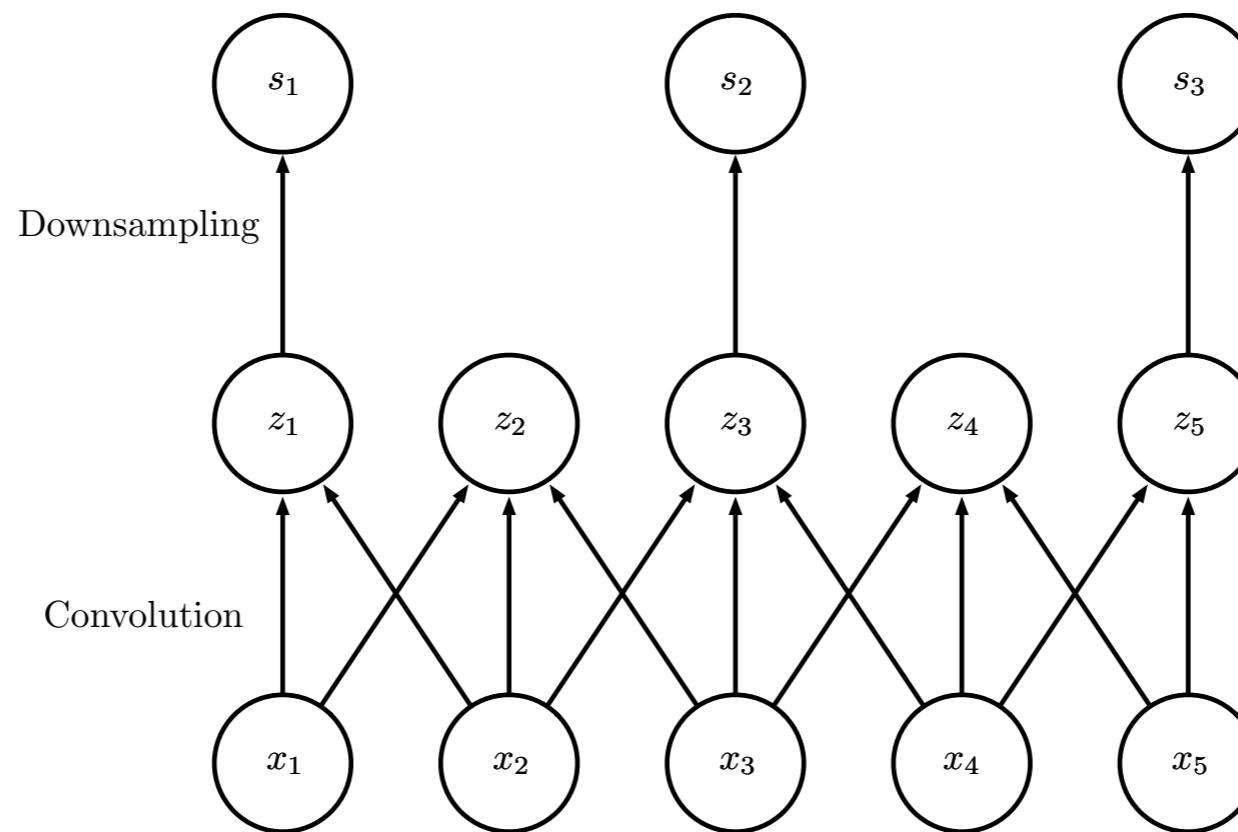
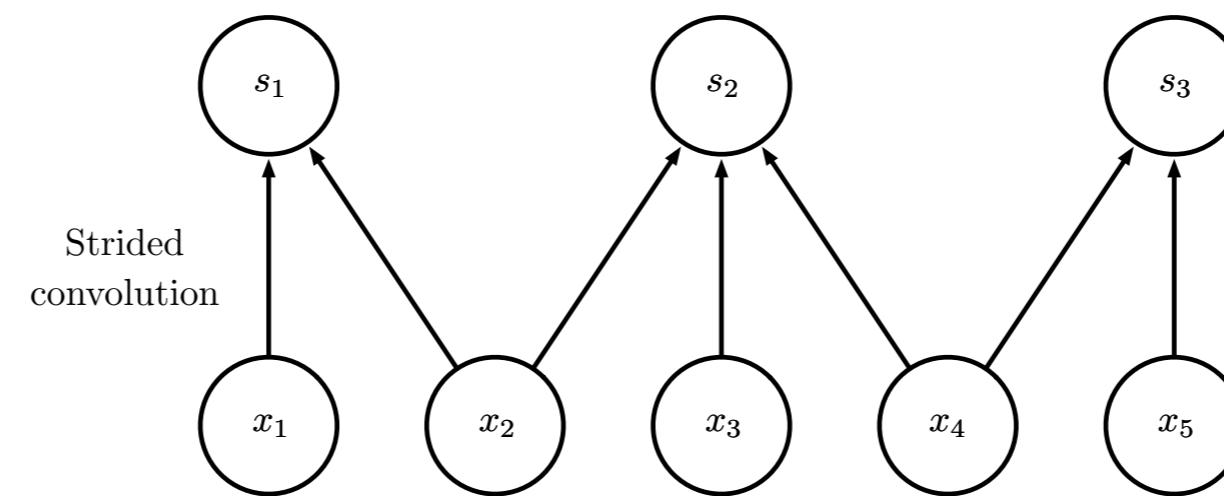


Figure 9.12

# Zero Padding Controls Size

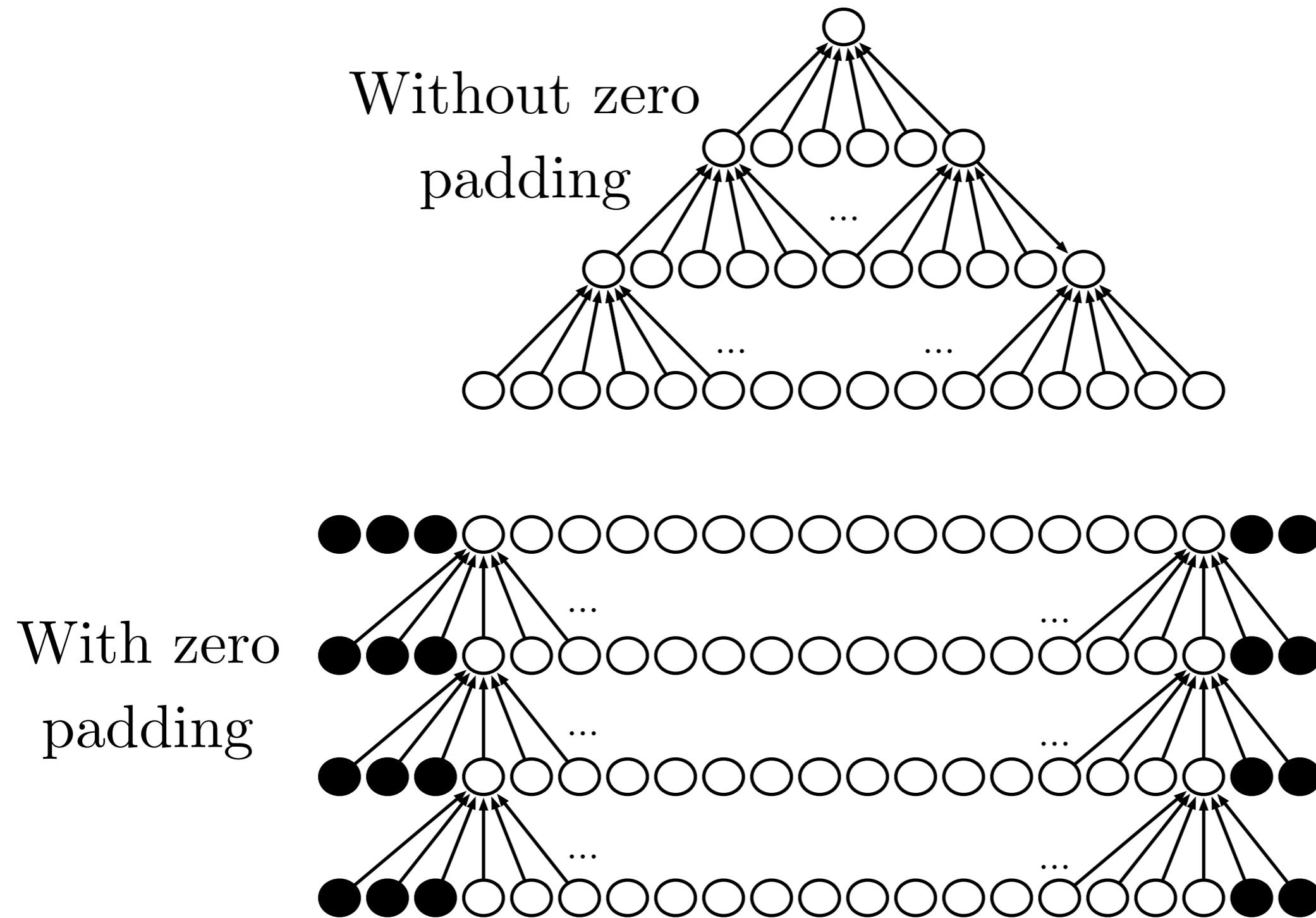
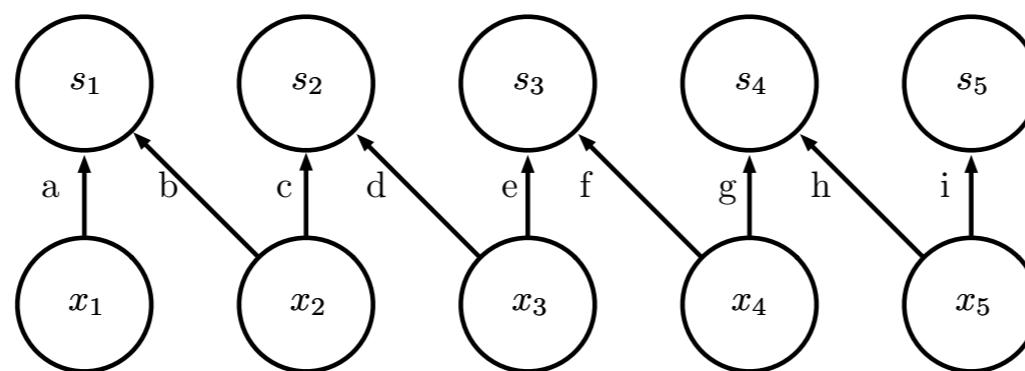
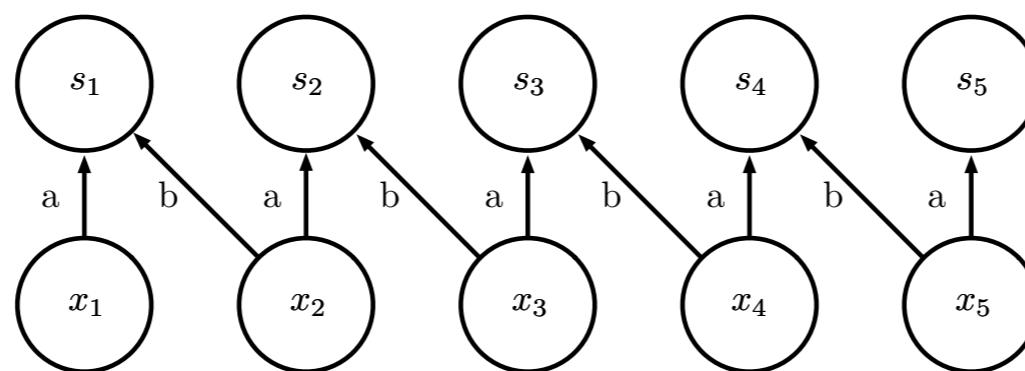


Figure 9.13

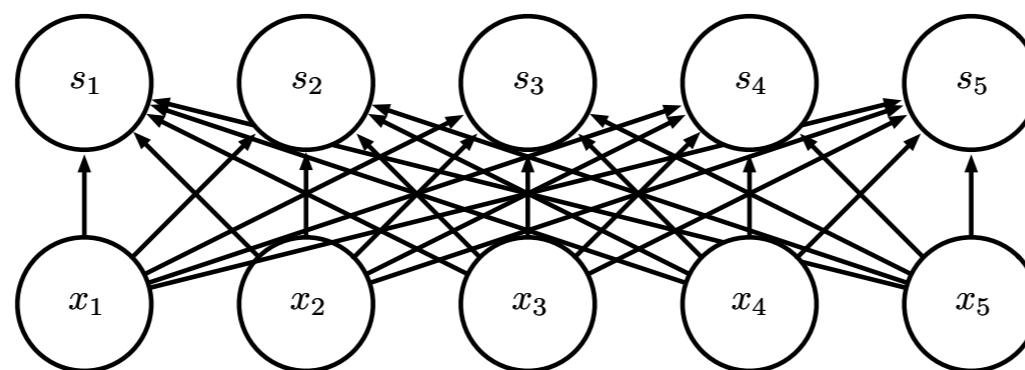
# Kinds of Connectivity



Local connection:  
like convolution,  
but no sharing



Convolution



Fully connected

Figure 9.14

# Partial Connectivity Between Channels

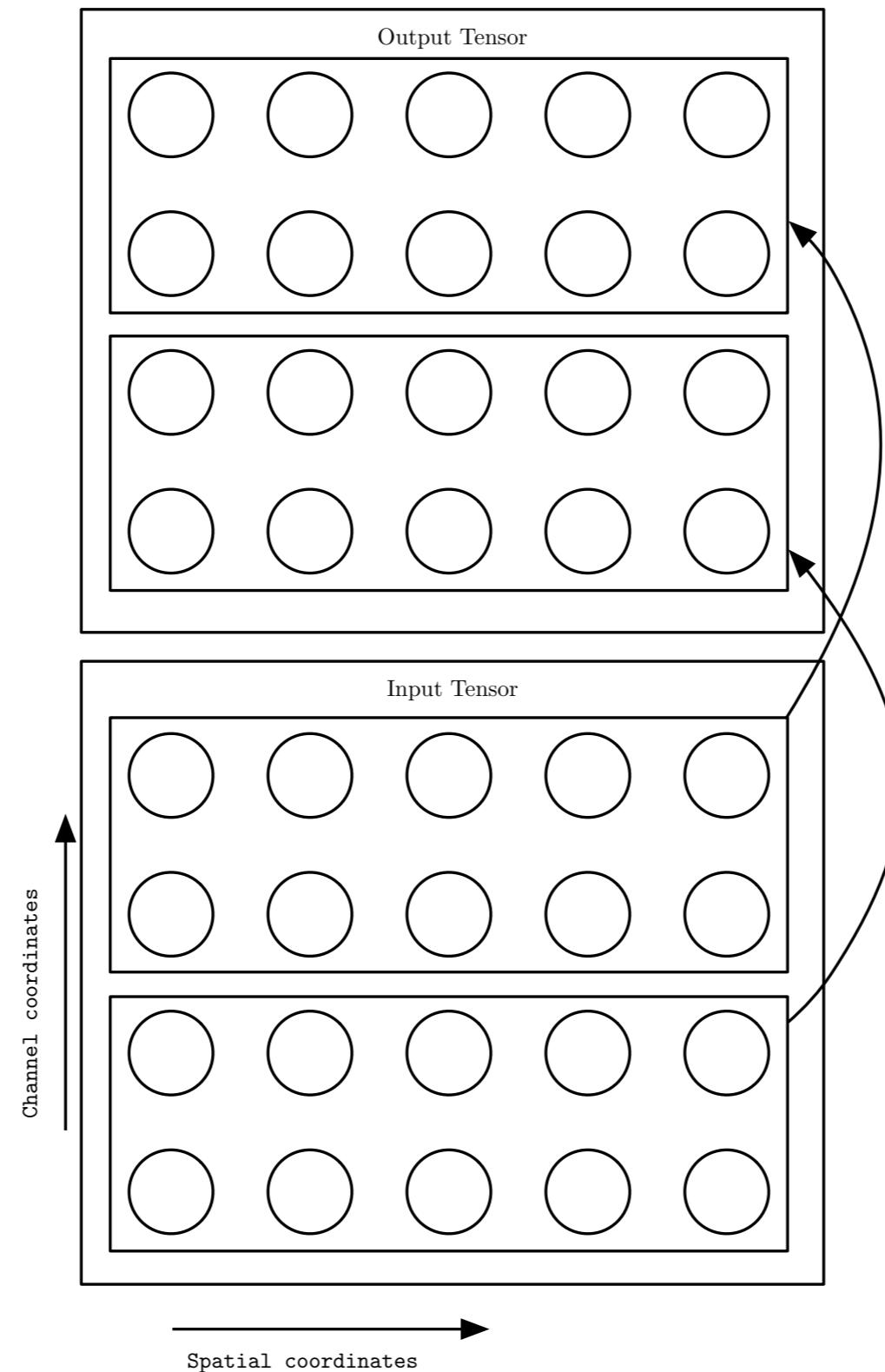
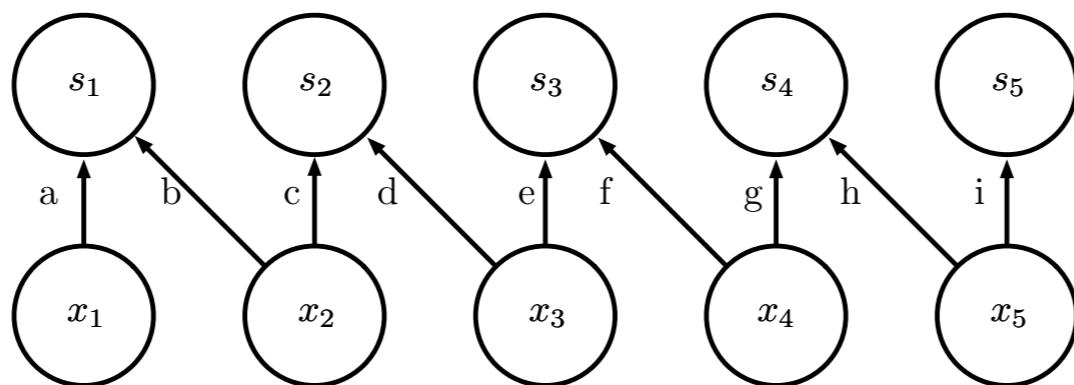
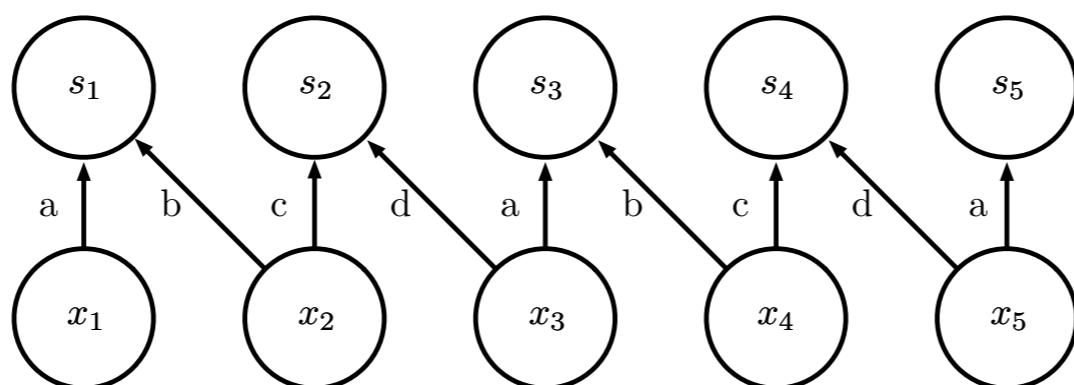


Figure 9.15

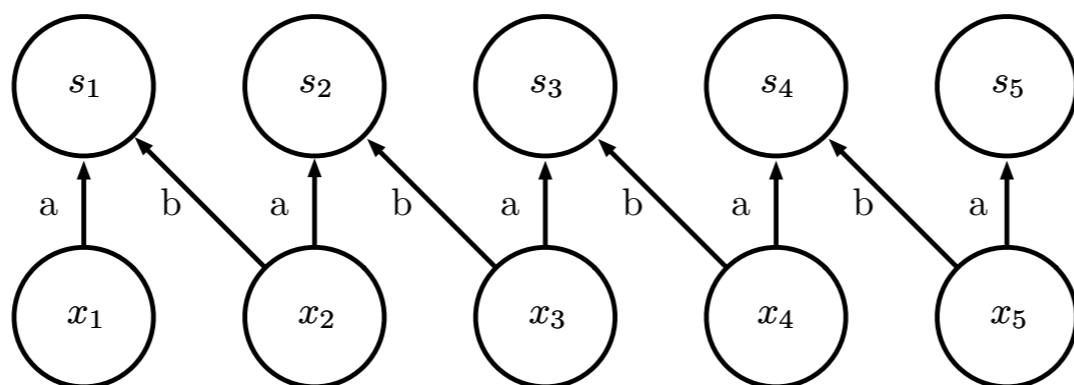
# Tiled convolution



Local connection  
(no sharing)



Tiled convolution  
(cycle between  
groups of shared  
parameters)



Convolution  
(one group shared  
everywhere)

Figure 9.16

# Recurrent Pixel Labeling

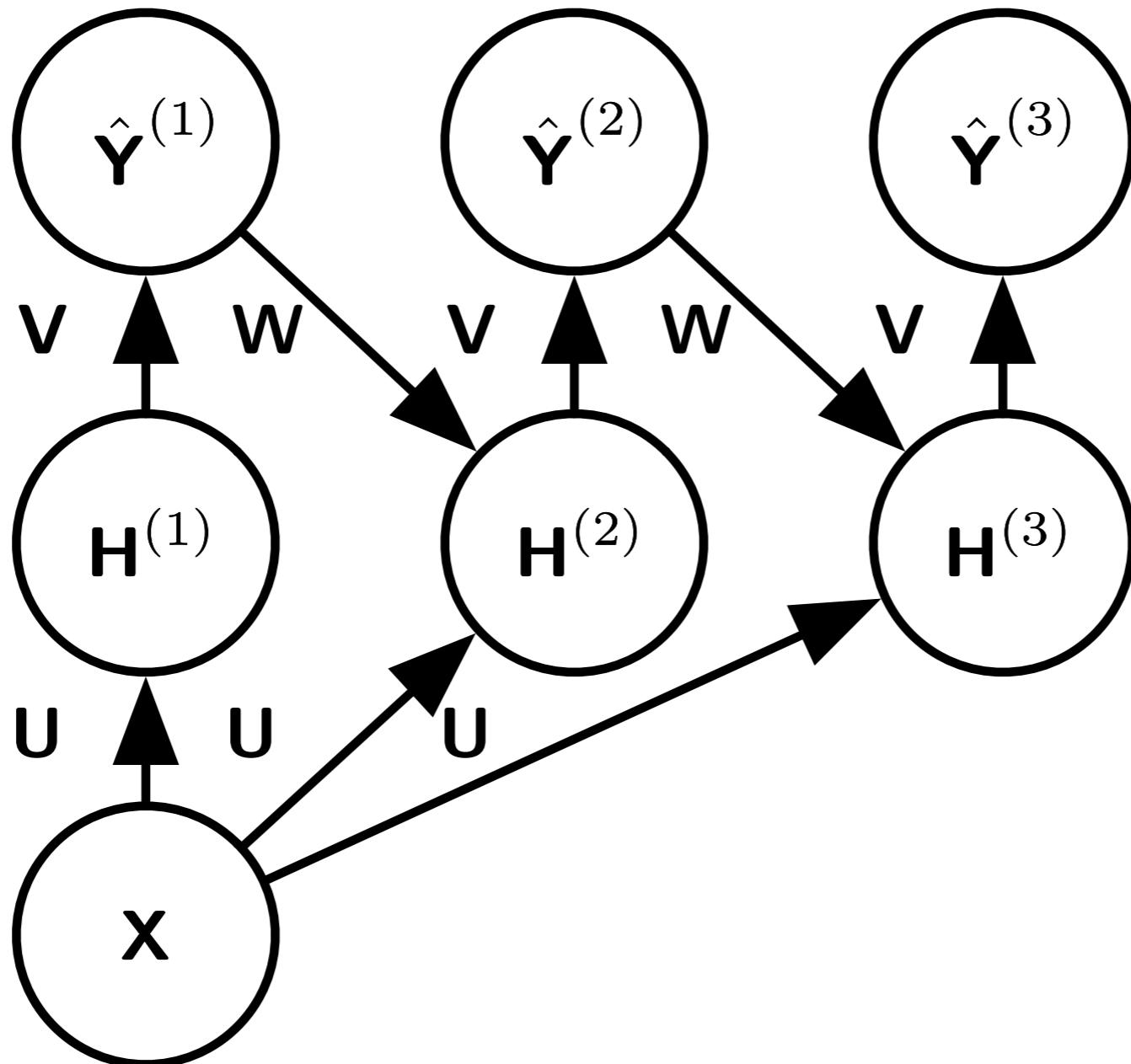


Figure 9.17

(Goodfellow 2016)

# Gabor Functions

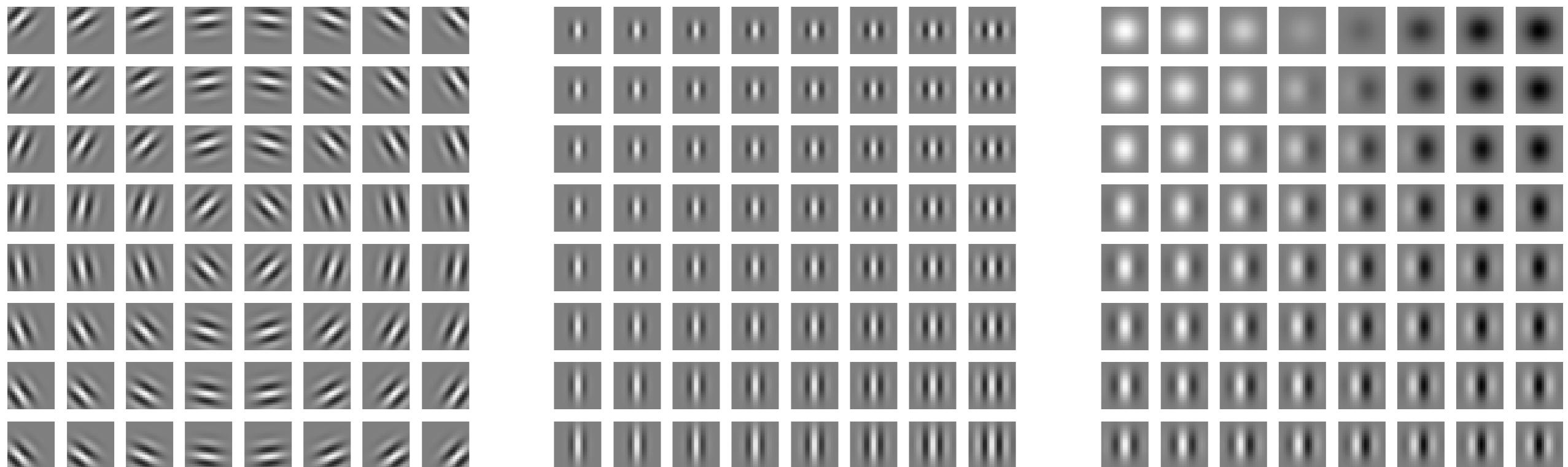


Figure 9.18

(Goodfellow 2016)

# Gabor-like Learned Kernels

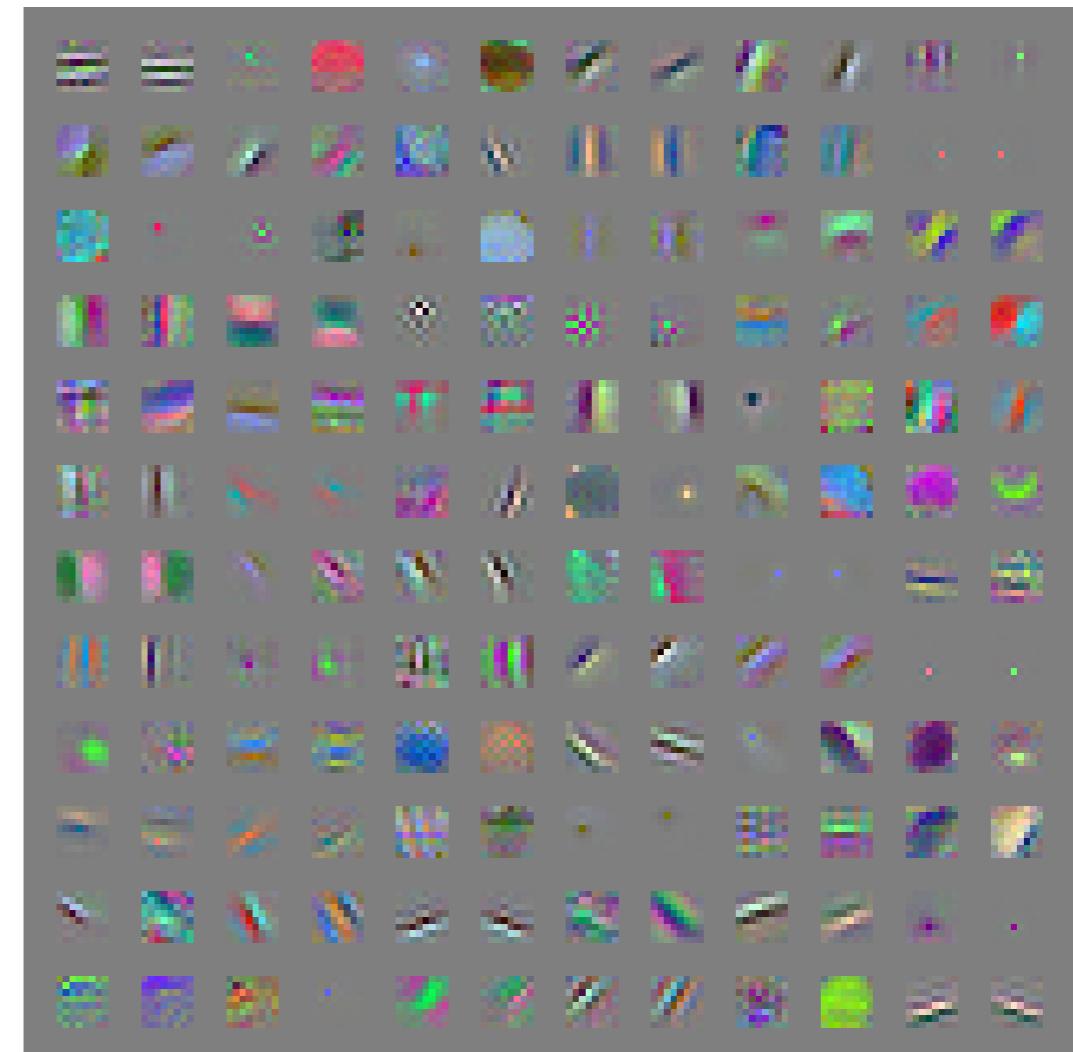
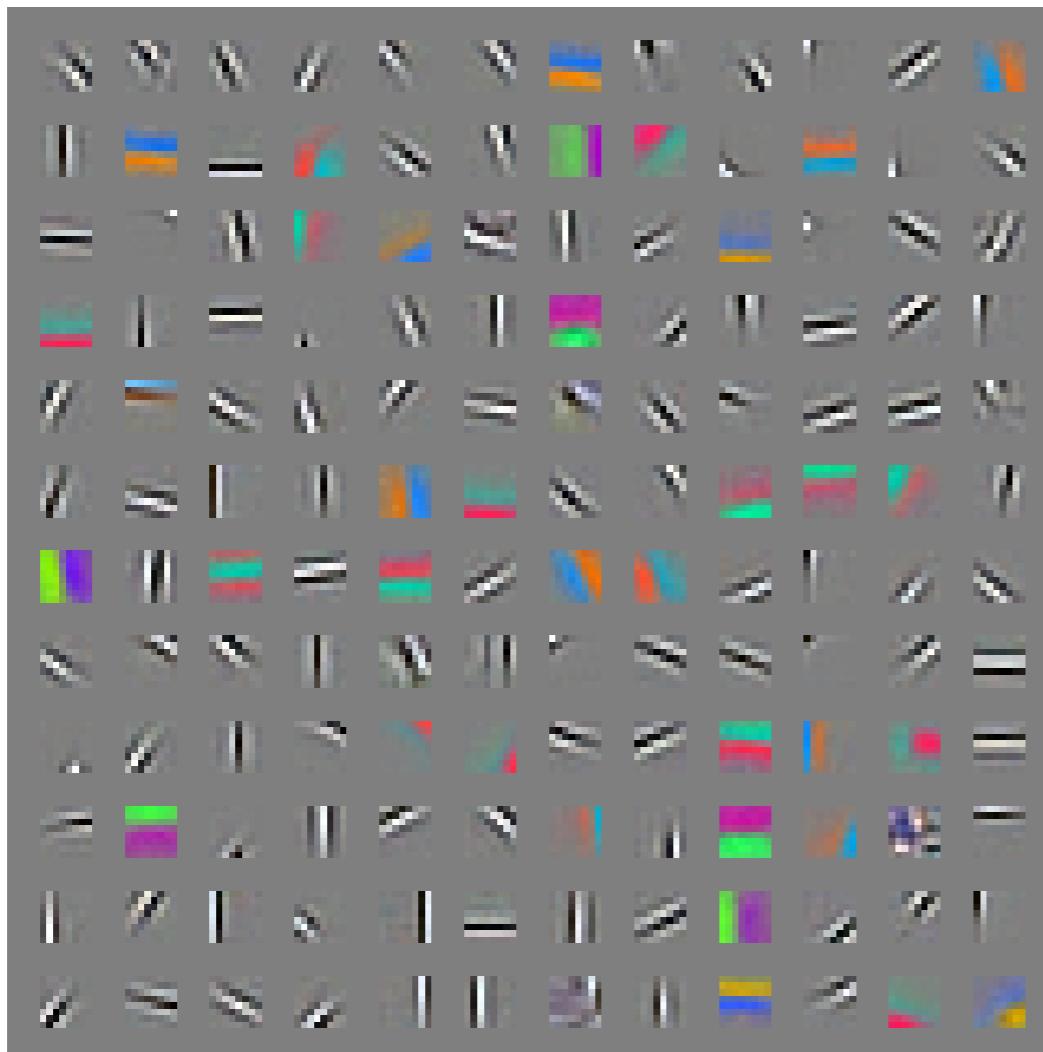


Figure 9.19

(Goodfellow 2016)

# Major Architectures

- Spatial Transducer Net: input size scales with output size, all layers are convolutional
- All Convolutional Net: no pooling layers, just use strided convolution to shrink representation size
- Inception: complicated architecture designed to achieve high accuracy with low computational cost
- ResNet: blocks of layers with same spatial size, with each layer's output added to the same buffer that is repeatedly updated. Very many updates = very deep net, but without vanishing gradient.