Machine Learning Systems

Lecture 9: Convolutional Neural Networks (CNNs, ConvNets)

Pooyan Jamshidi

Convolutional Neural Networks (CNNs, ConvNets)

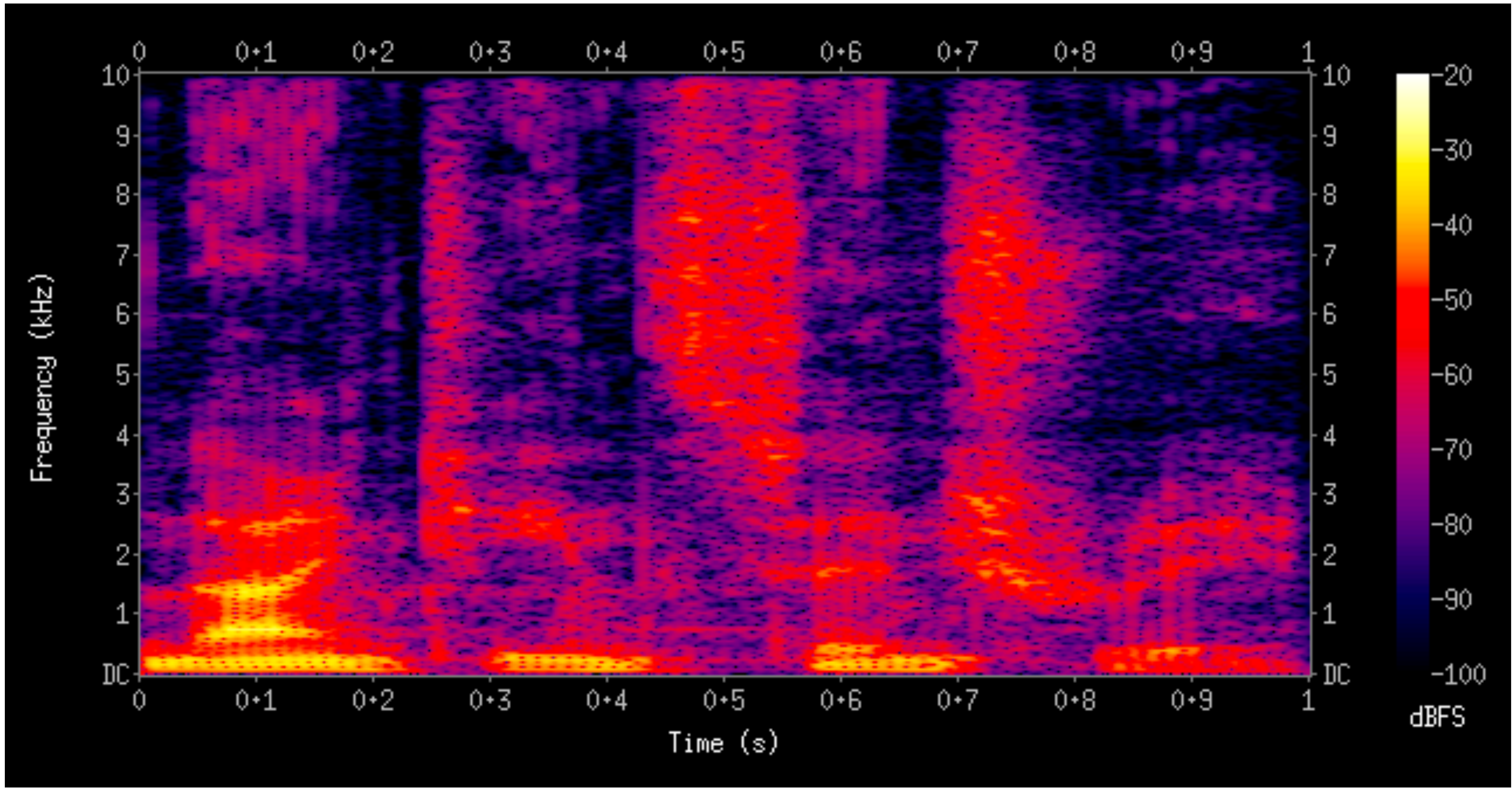
Pooyan Jamshidi
UofSC

Partially based on:
- Chapter 9 of the *Deep Learning Book:* https://www.deeplearningbook.org/
- CS231n Convolutional Neural Networks for Visual Recognition

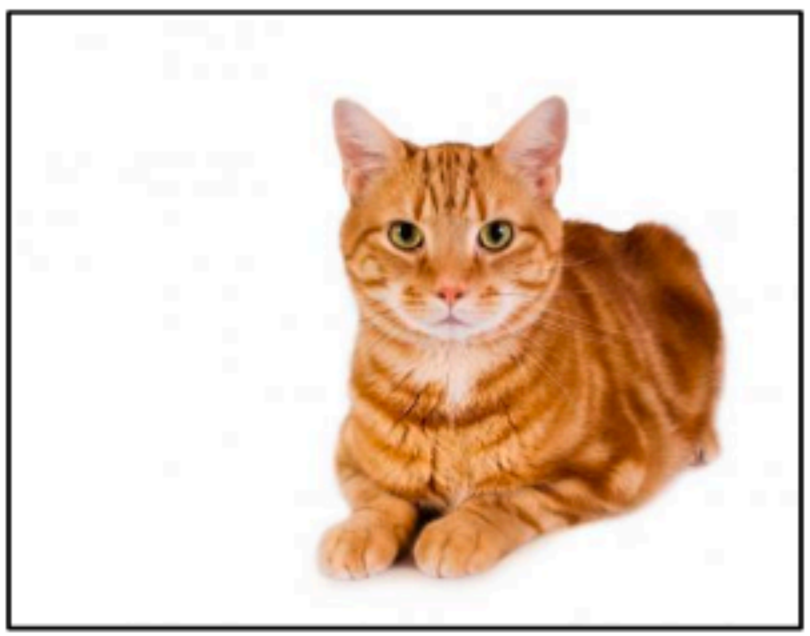
Convolutional Networks

- 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, …)

Examples of Grid Structures



Shift Invariance in Convolutional Neural Networks



Cat

Cat

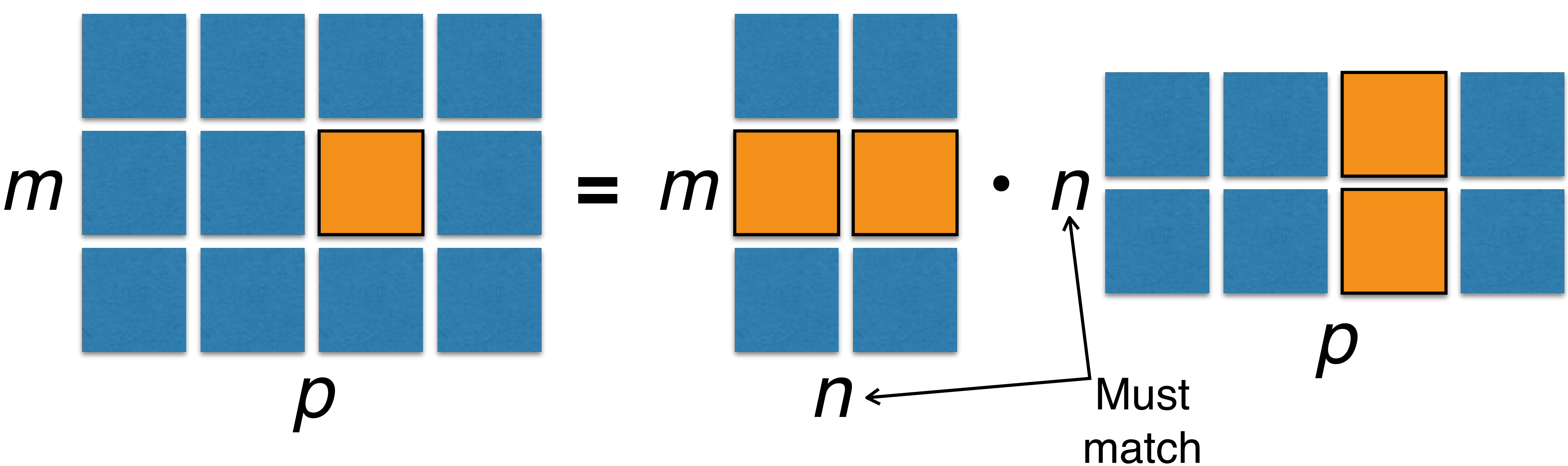
Key Idea

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

Matrix (Dot) Product

$$\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}. \tag{2.4}$$

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}. \tag{2.5}$$



Matrix Transpose

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

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

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

$$(\boldsymbol{A}\boldsymbol{B})^\top = \boldsymbol{B}^\top \boldsymbol{A}^\top. \tag{2.9}$$

2D Convolution

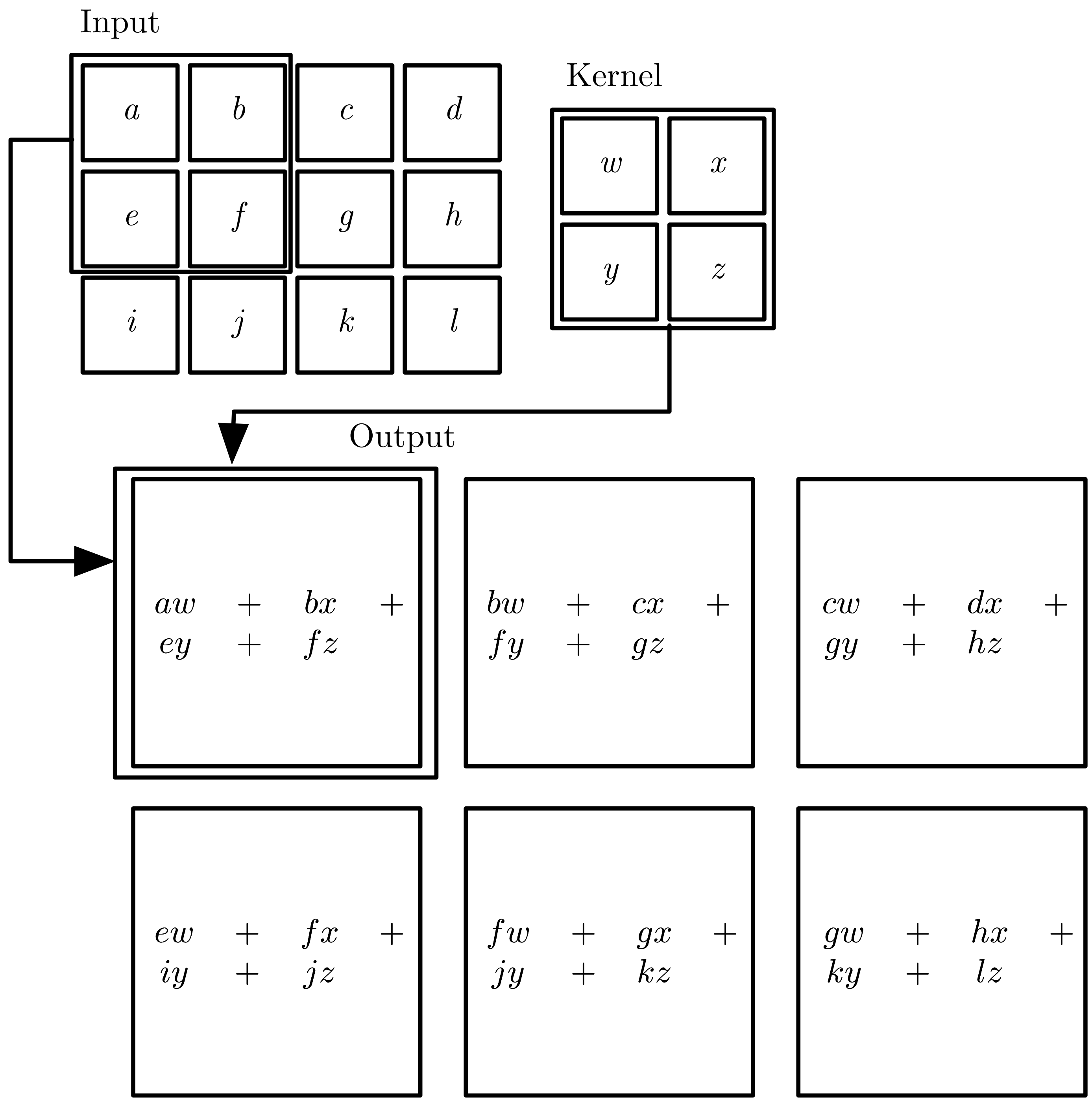


Figure 9.1

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

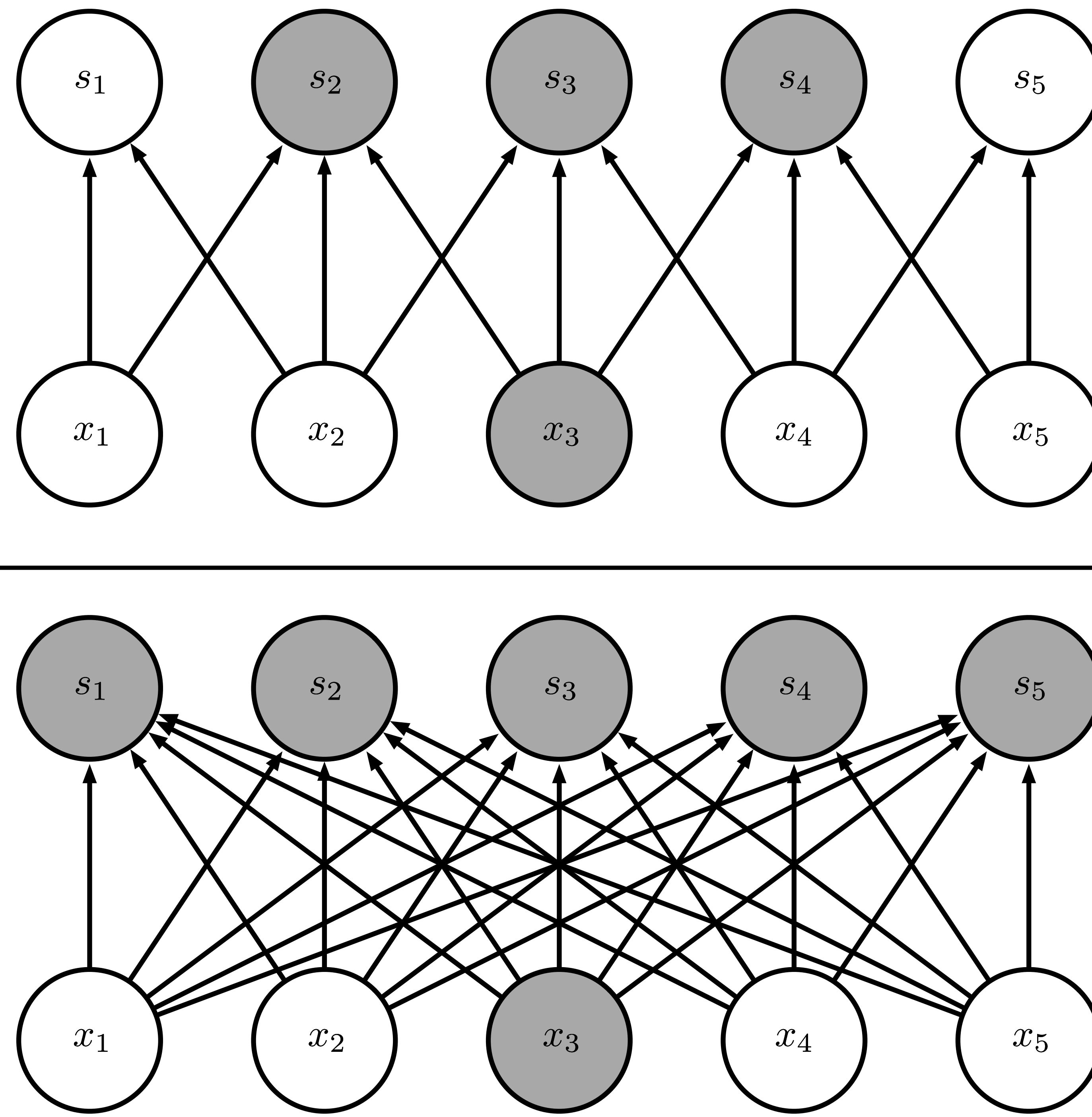
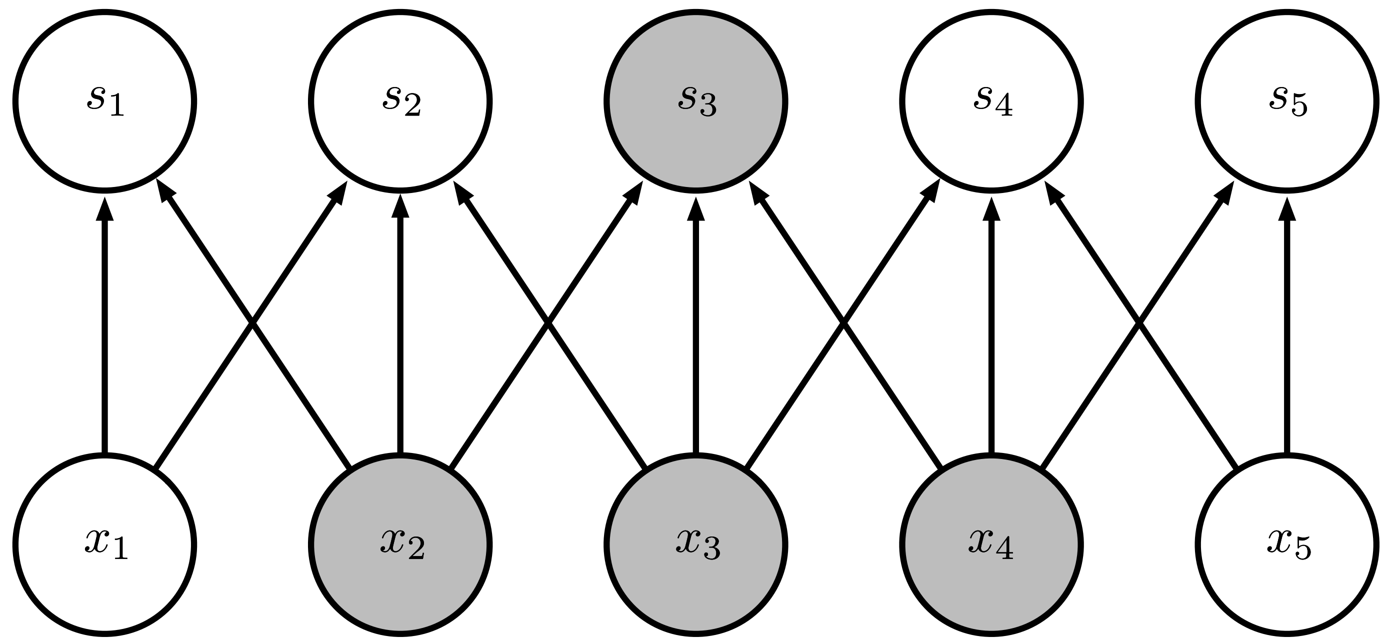


Figure 9.2

Sparse Connectivity

Sparse connections due to small convolution kernel



Dense connections

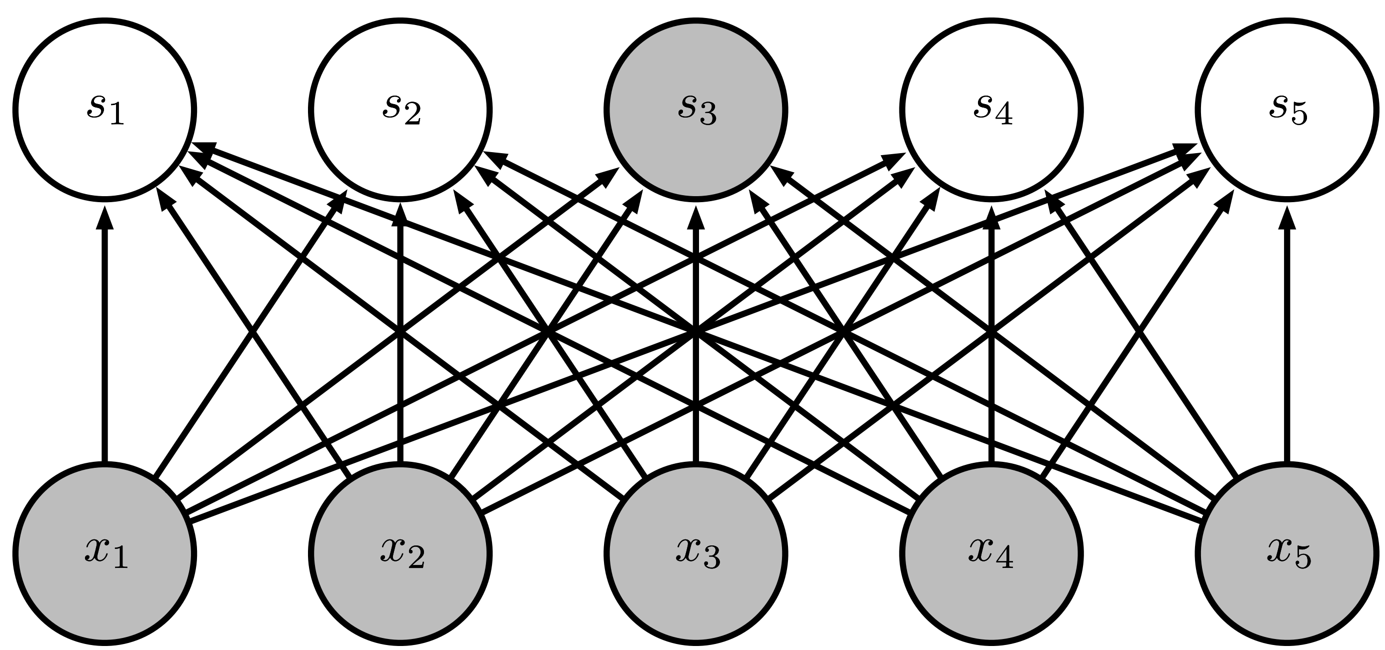


Figure 9.3

Growing Receptive Fields

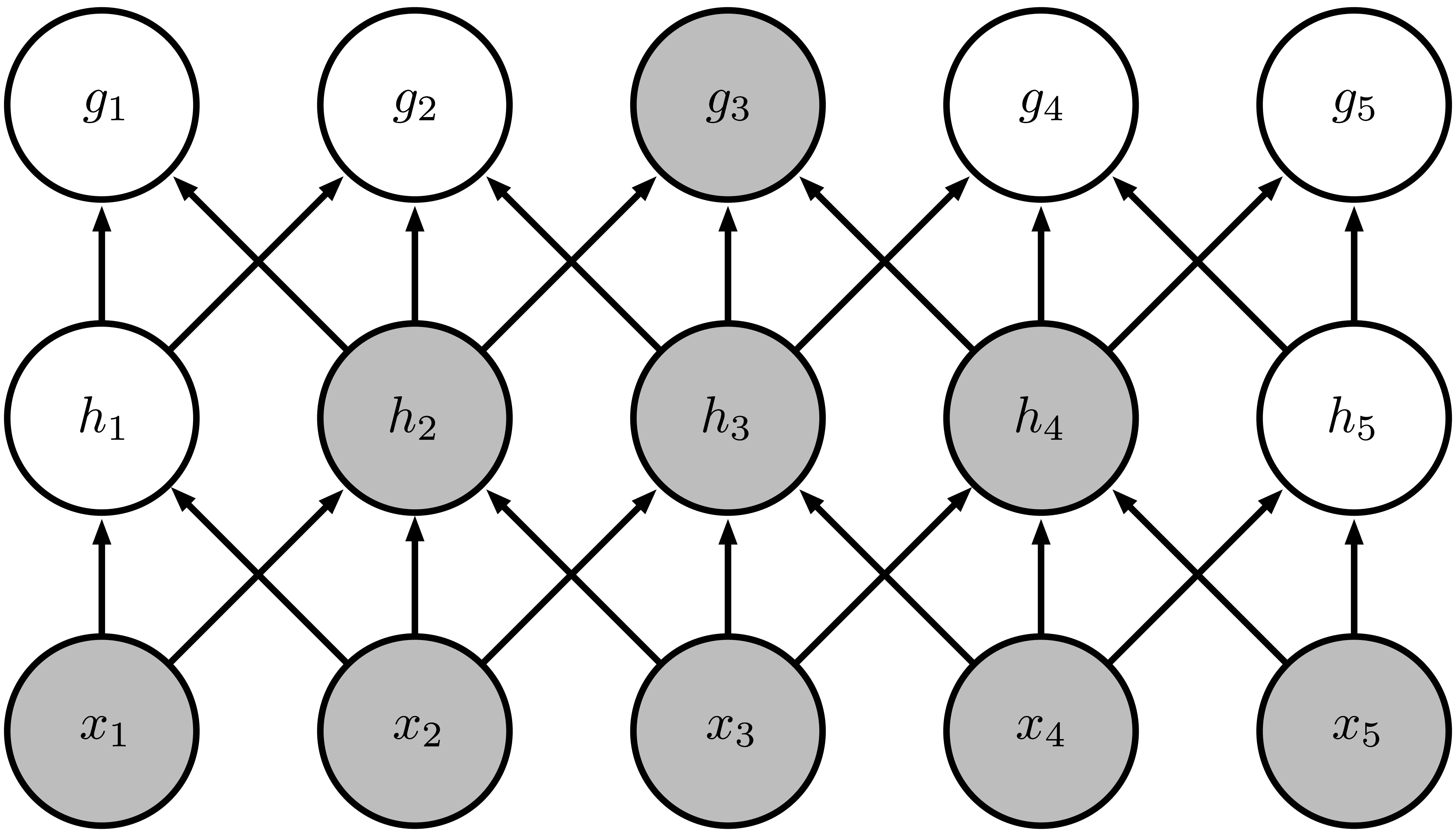
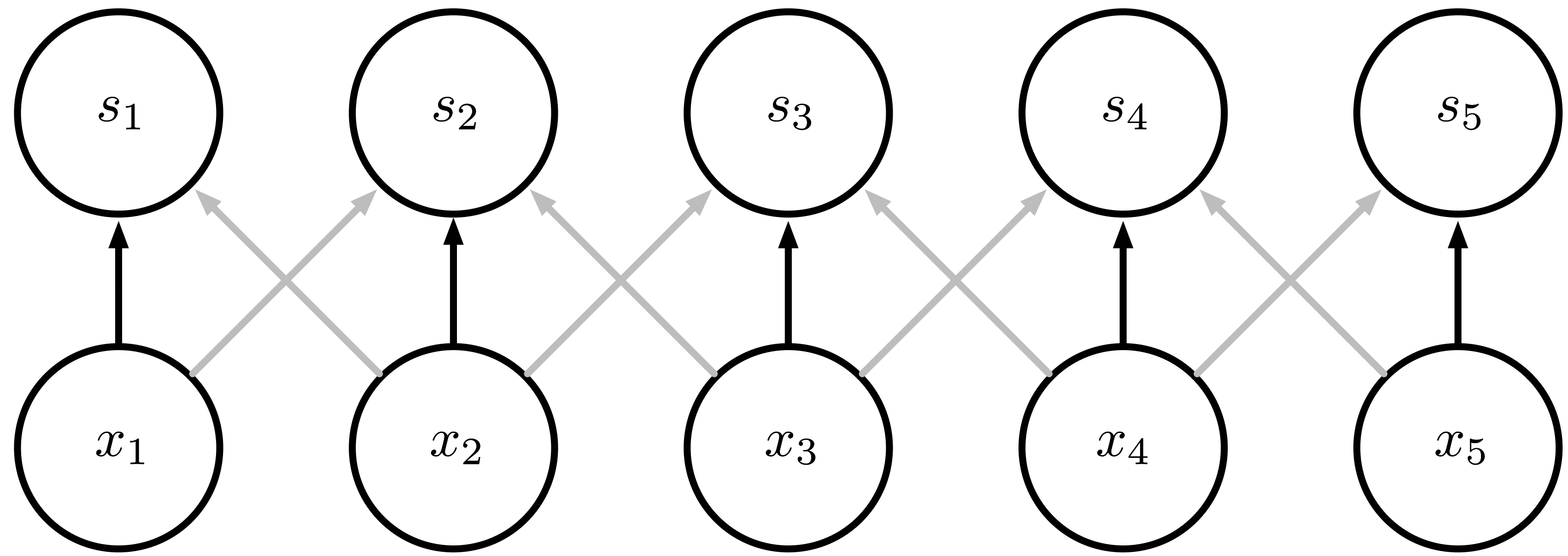


Figure 9.4

Parameter Sharing

Convolution shares the same parameters across all spatial locations



Traditional matrix multiplication does not share any parameters

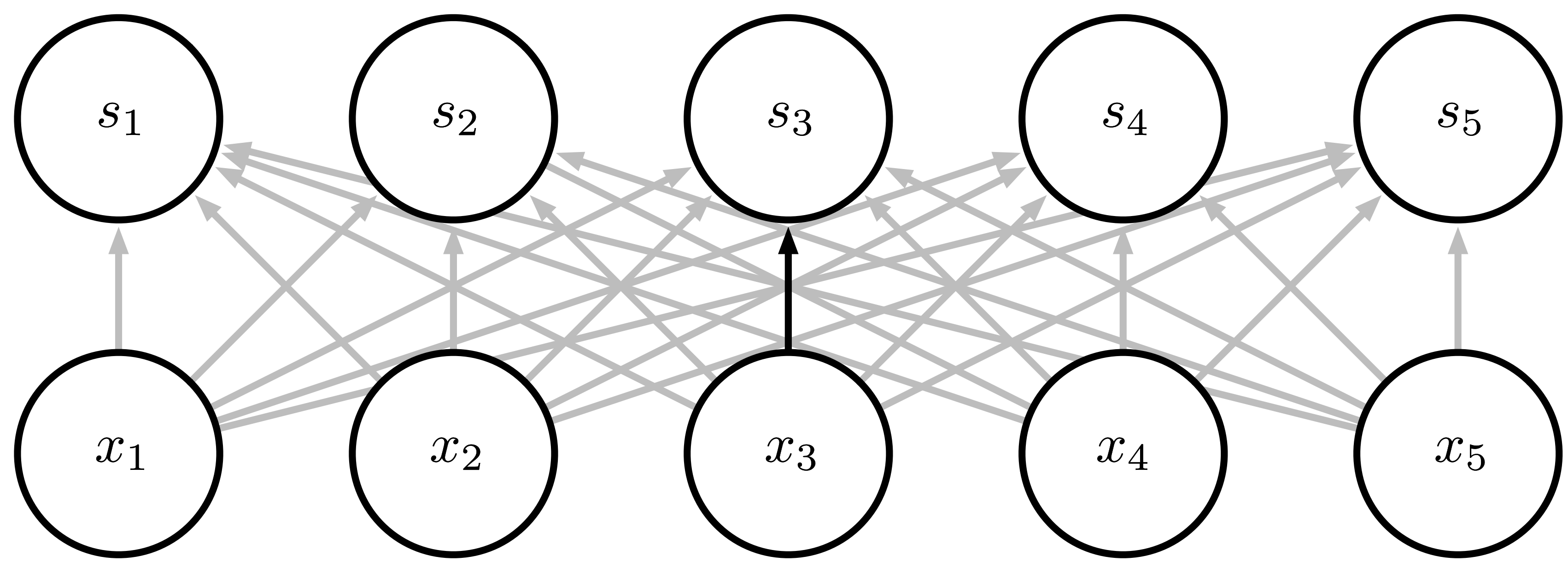


Figure 9.5

Edge Detection by Convolution

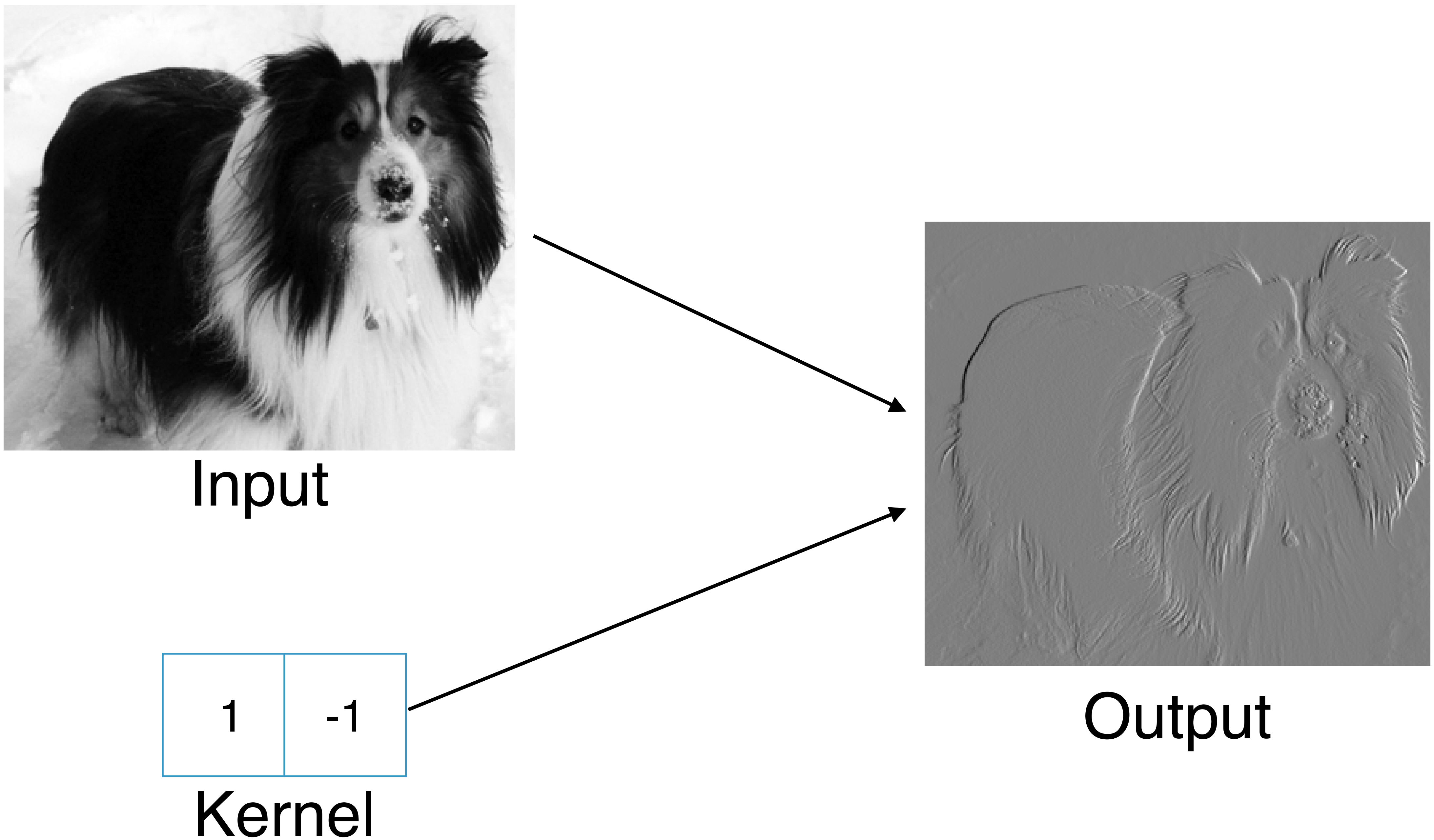


Figure 9.6

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			
Float muls or adds			

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	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

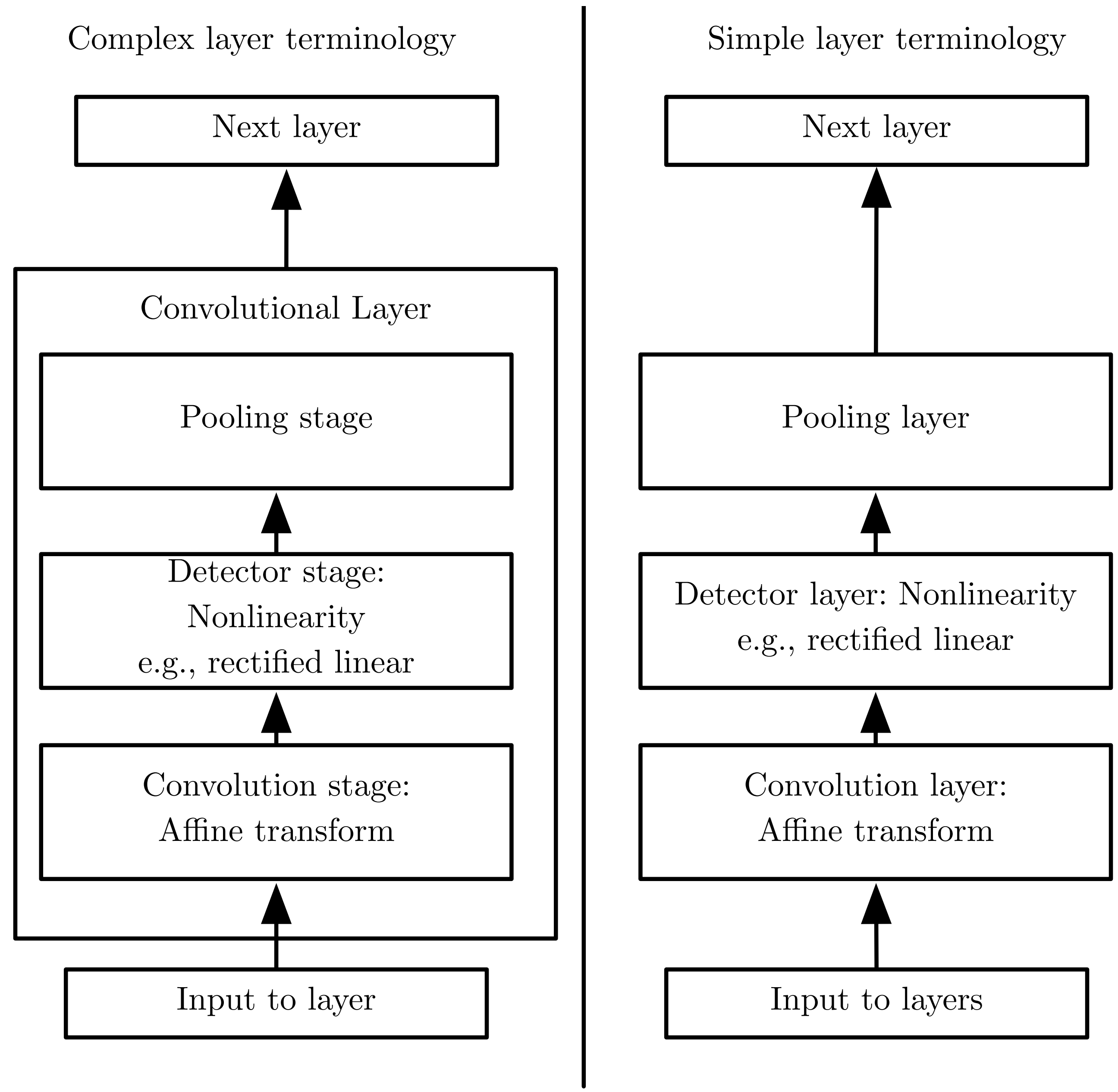
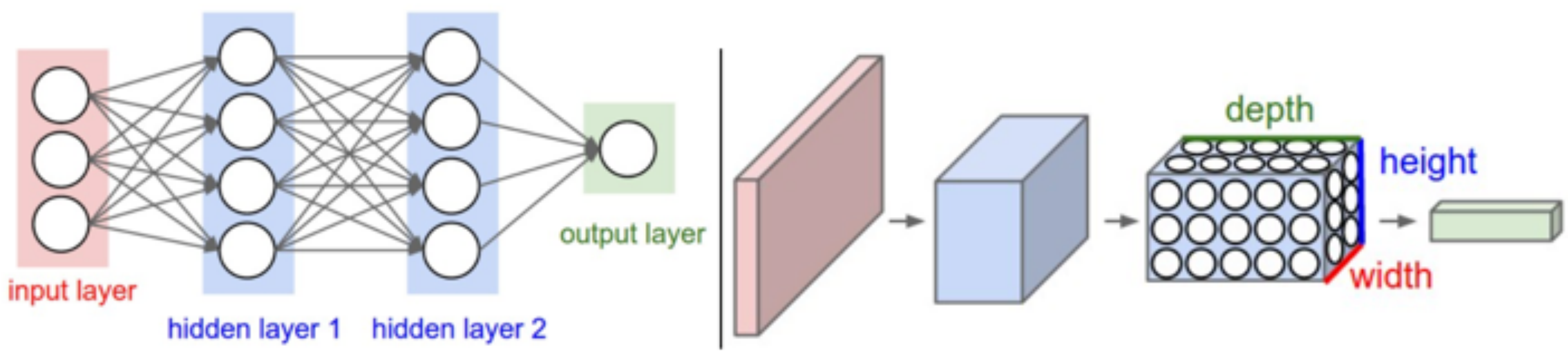


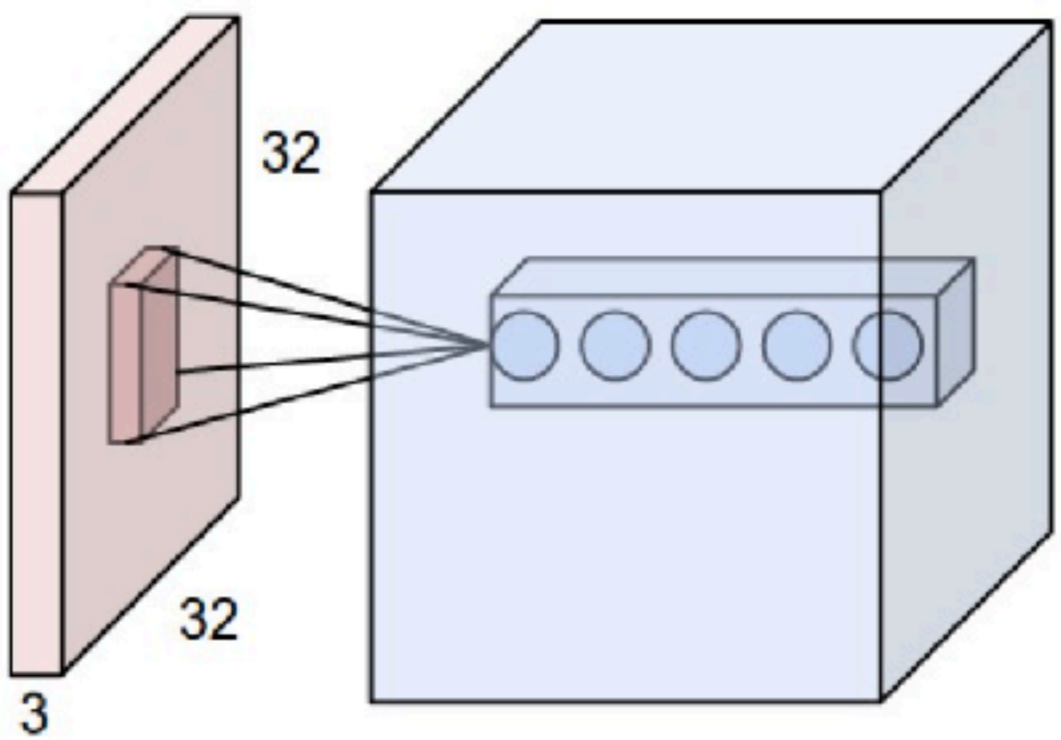
Figure 9.7

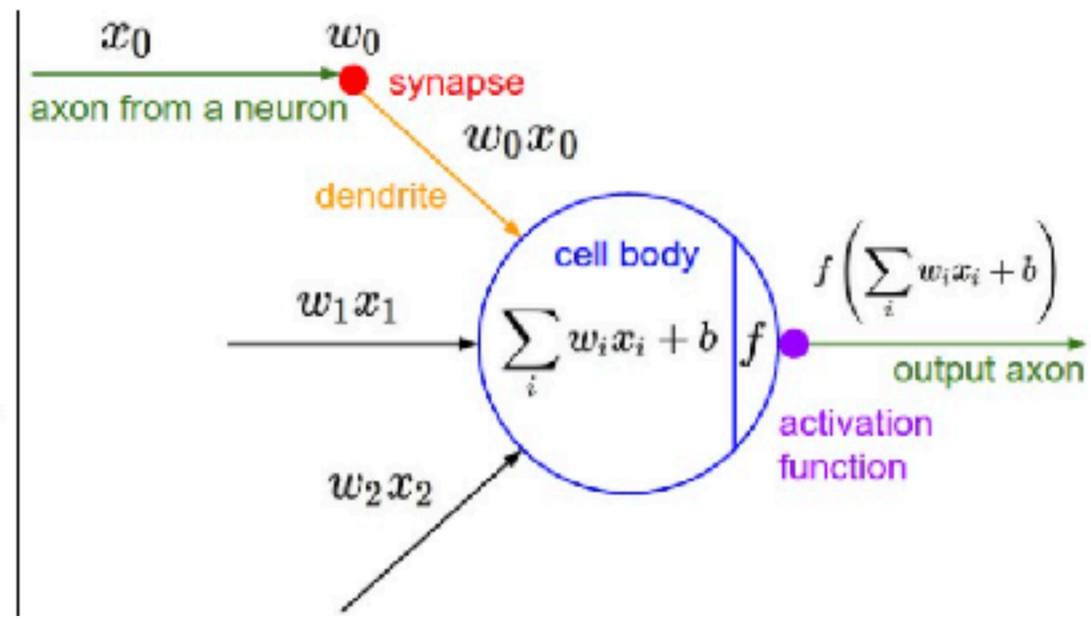
Regular fully-connected NN vs ConvNet



> *A ConvNet is made up of Layers. Every Layer has a simple API: It transforms an input 3D volume to an output 3D volume with some differentiable function that may or may not have parameters.*

Local Connectivity

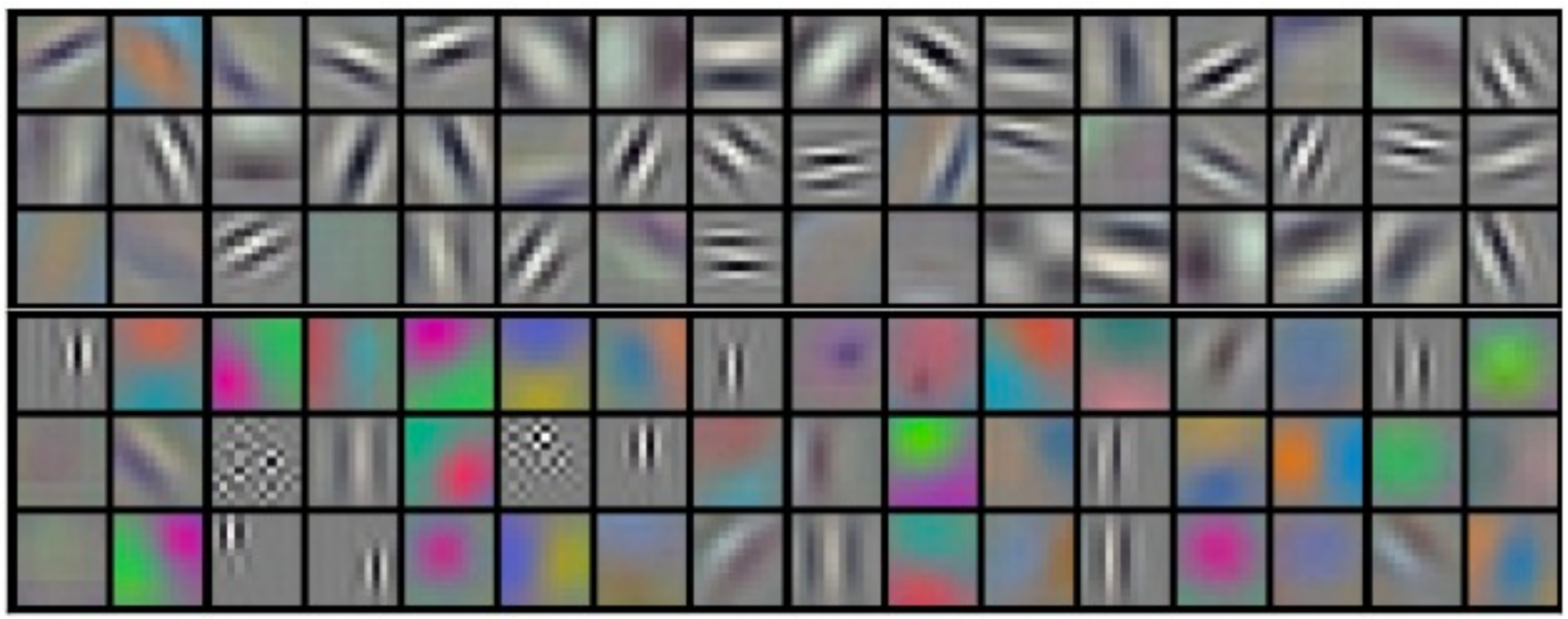




Example 1. For example, suppose that the input volume has size [32x32x3], (e.g. an RGB CIFAR-10 image). If the receptive field (or the filter size) is 5x5, then each neuron in the Conv Layer will have weights to a [5x5x3] region in the input volume, for a total of 5*5*3 = 75 weights (and +1 bias parameter). Notice that the extent of the connectivity along the depth axis must be 3, since this is the depth of the input volume.

Example 2. Suppose an input volume had size [16x16x20]. Then using an example receptive field size of 3x3, every neuron in the Conv Layer would now have a total of 3*3*20 = 180 connections to the input volume. Notice that, again, the connectivity is local in space (e.g. 3x3), but full along the input depth (20).

Example of learned kernels



Max Pooling and Invariance to Translation

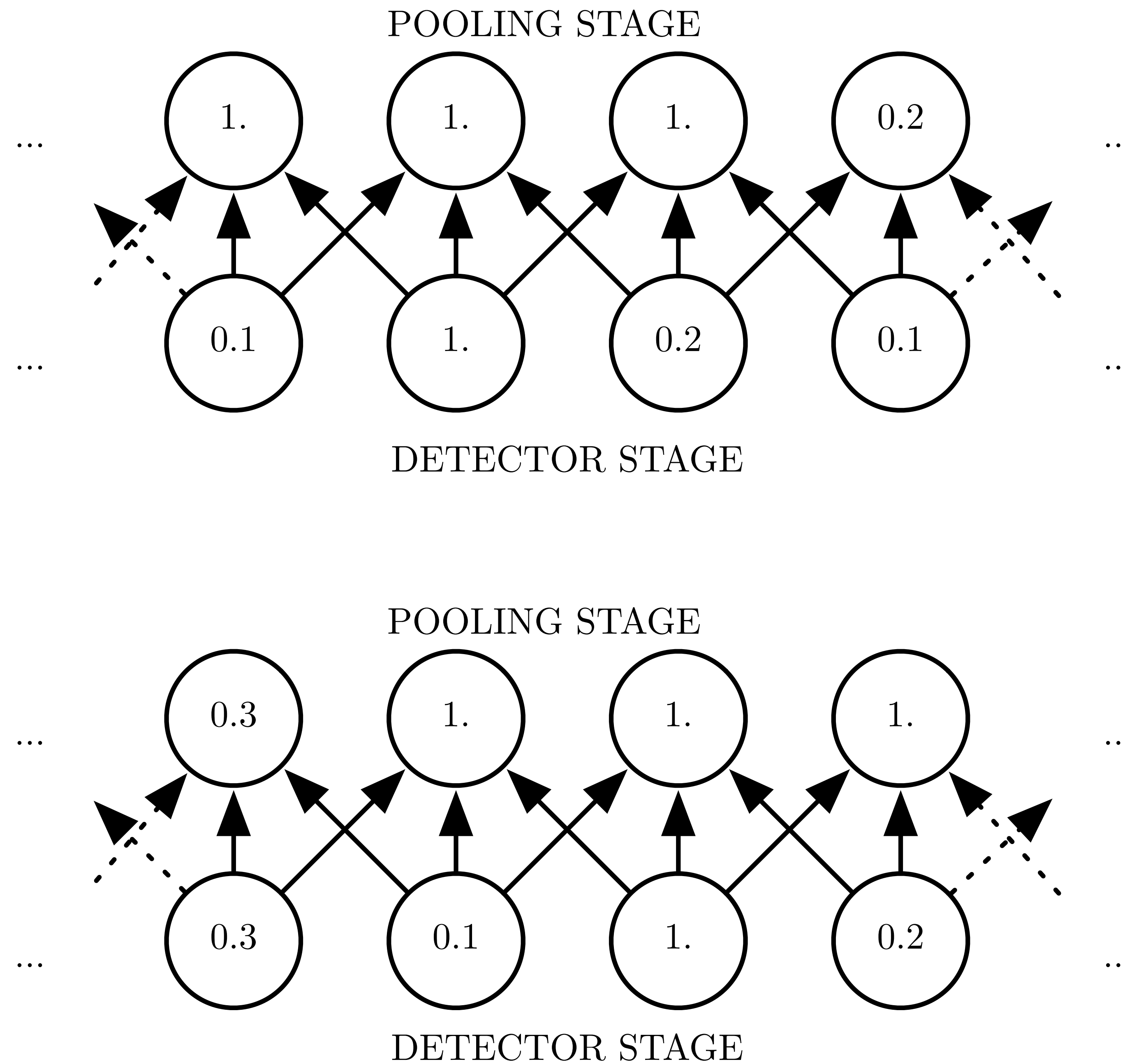


Figure 9.8

Cross-Channel Pooling and Invariance to Learned Transformations

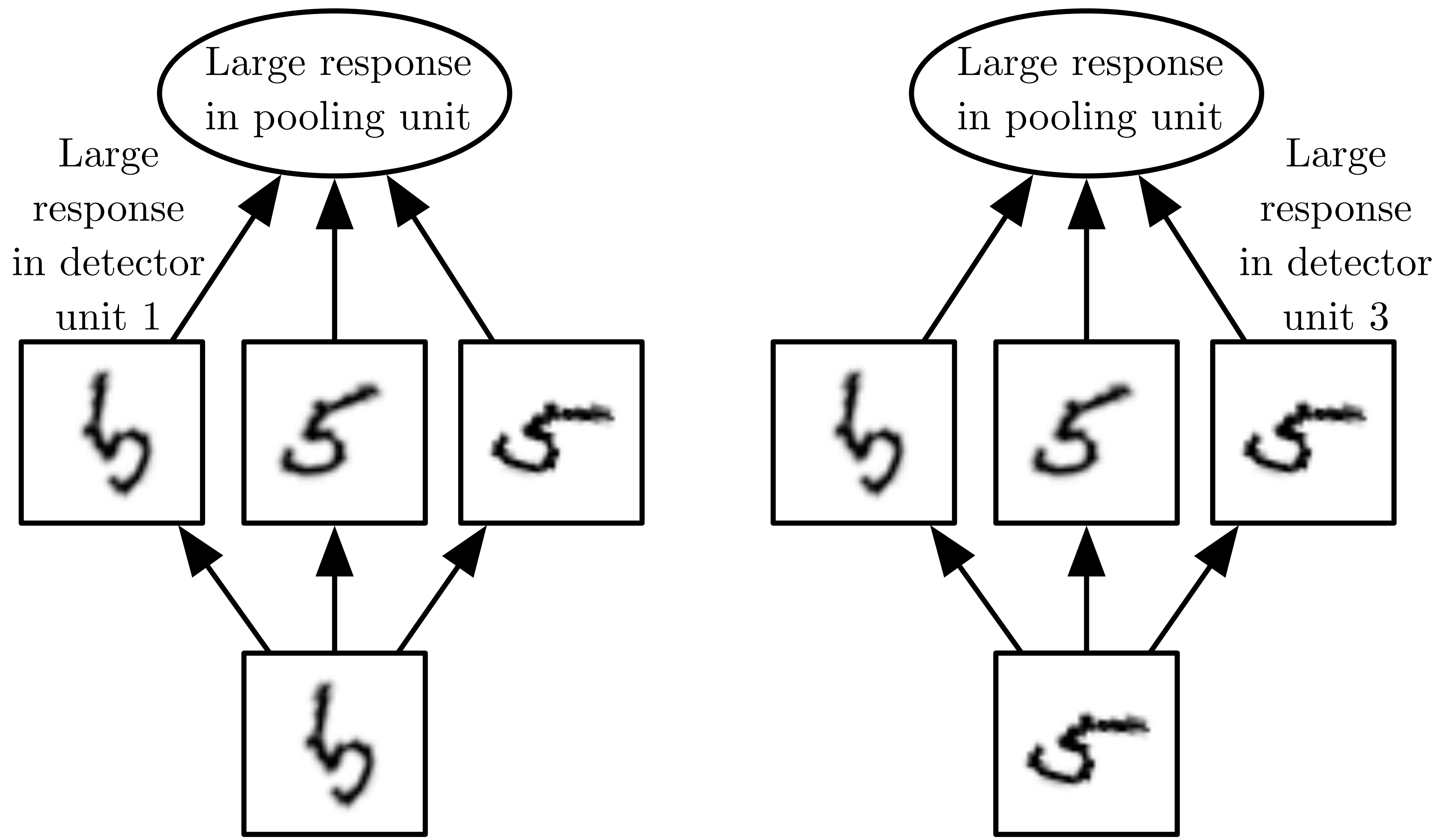


Figure 9.9

Pooling with Downsampling

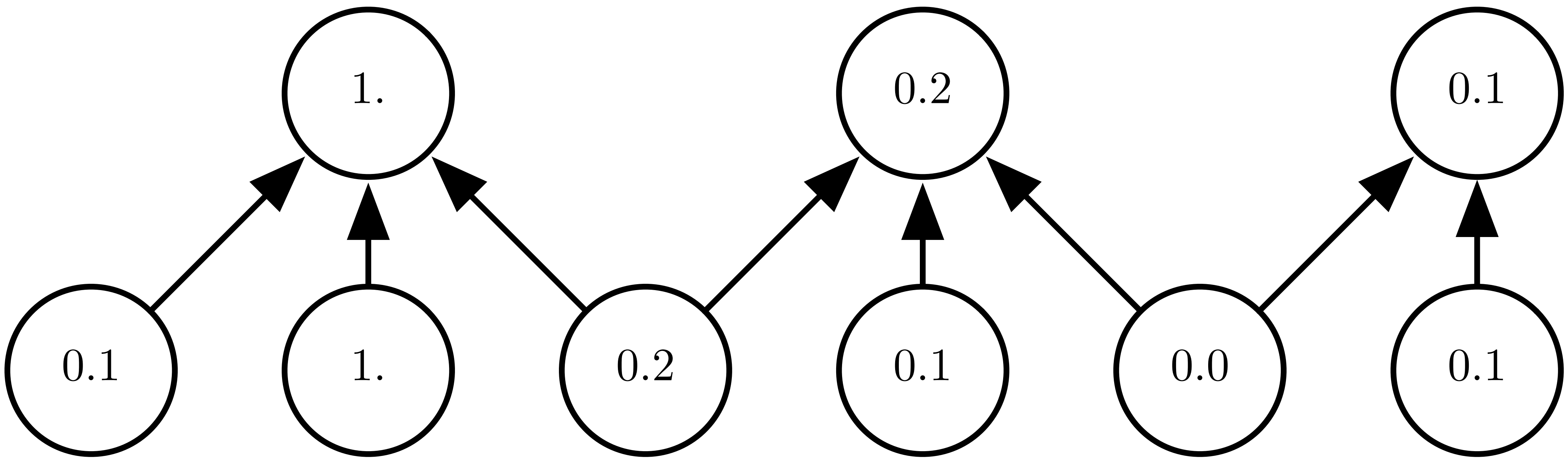
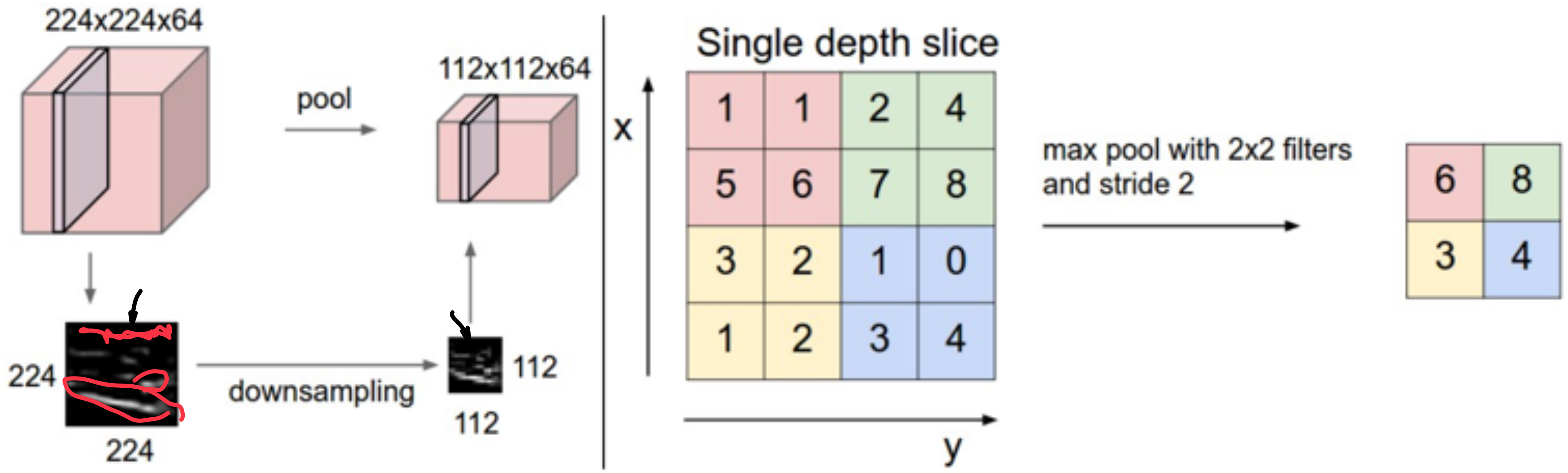


Figure 9.10

Pooling layer downsamples the volume spatially



Example Classification Architectures

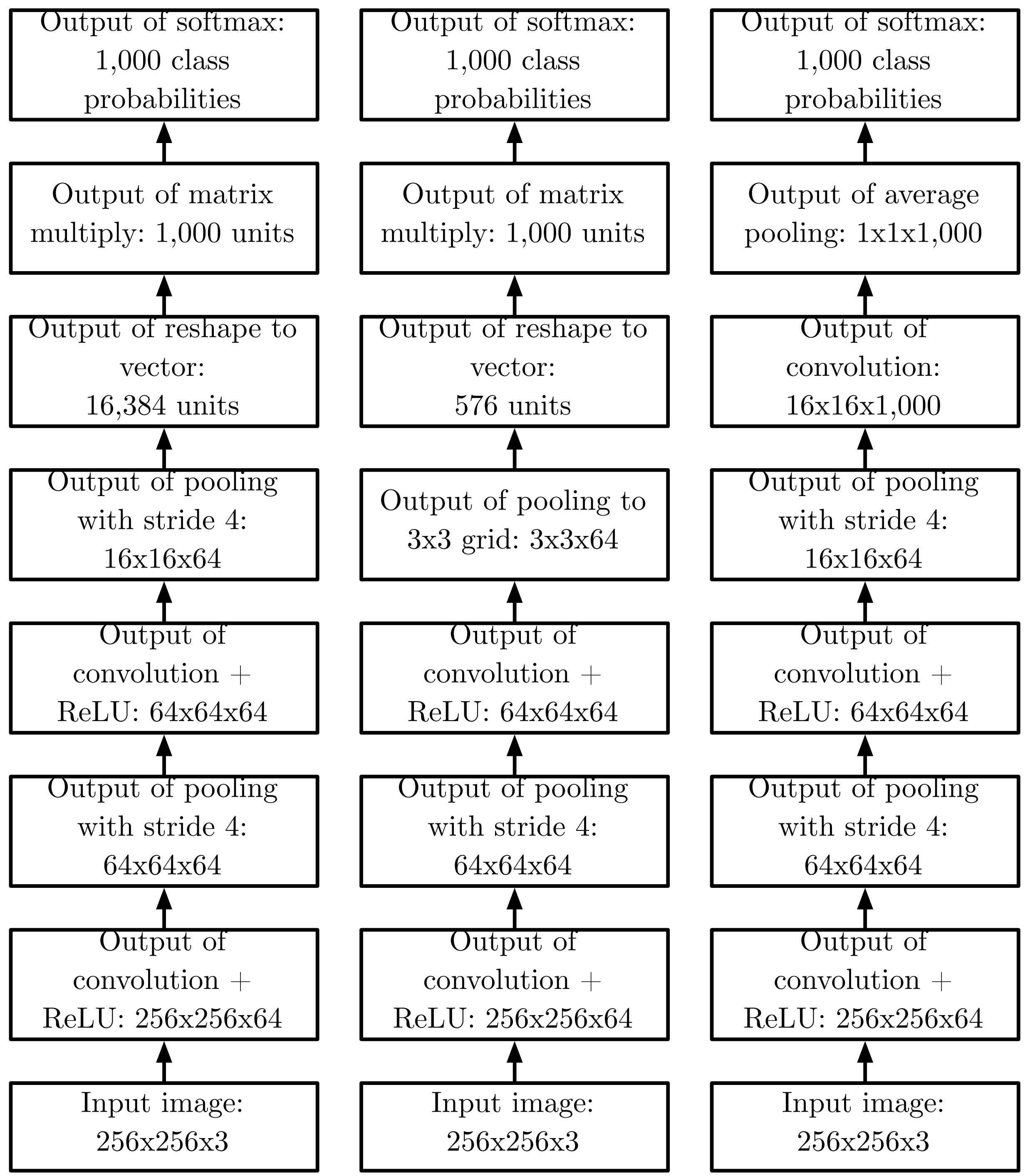
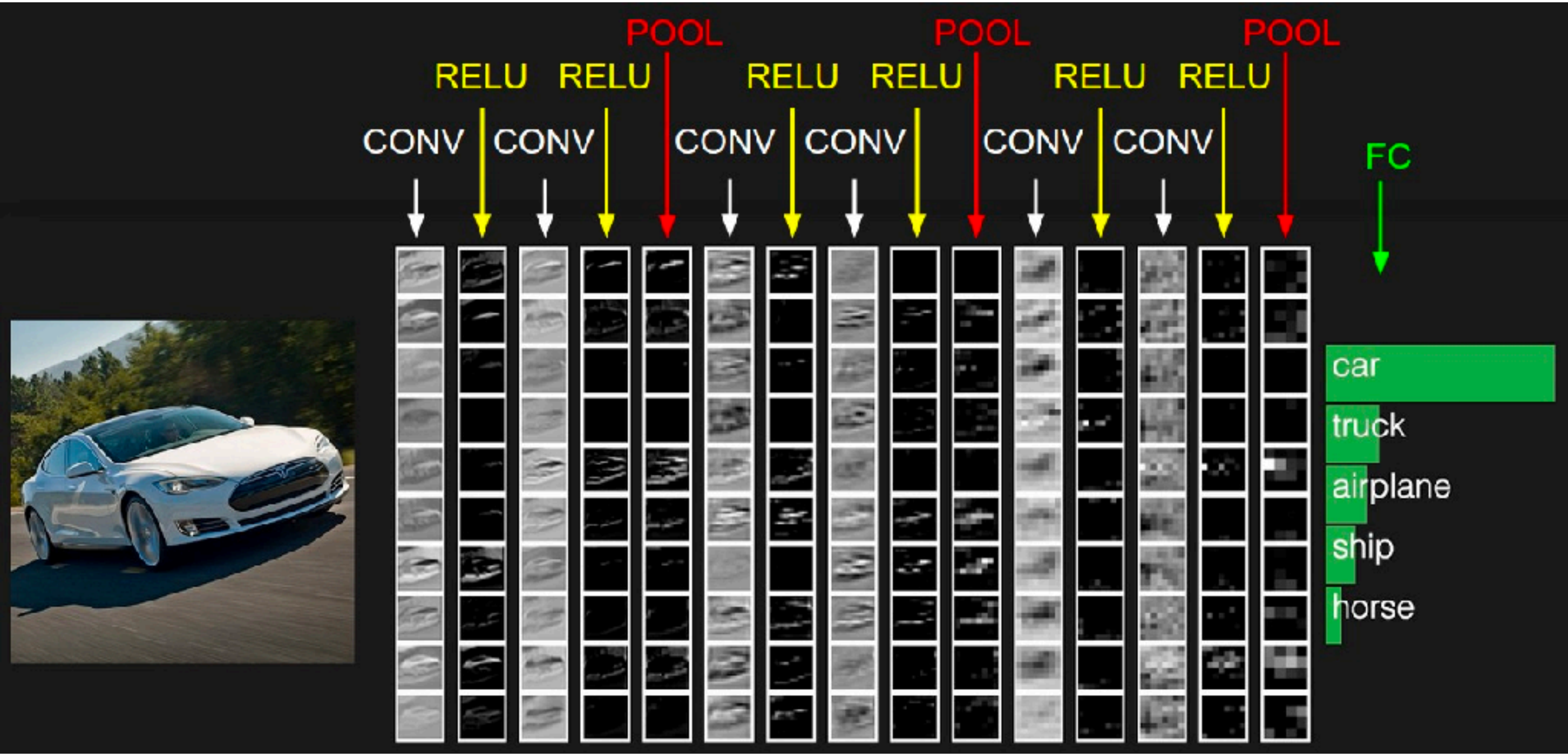


Figure 9.11

ConvNet architecture



Architecture Overview of ConvNets

[INPUT - CONV - RELU - POOL - FC]

- INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.
- RELU layer will apply an elementwise activation function, such as the $max(0, x)$ thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

Architecture Overview of ConvNets

- A ConvNet architecture is in the simplest case a list of Layers that transform the image volume into an output volume (e.g. holding the class scores)
- There are a few distinct types of Layers (e.g. CONV/FC/RELU/POOL are by far the most popular)
- Each Layer accepts an input 3D volume and transforms it to an output 3D volume through a differentiable function
- Each Layer may or may not have parameters (e.g. CONV/FC do, RELU/POOL don't)
- Each Layer may or may not have additional hyperparameters (e.g. CONV/FC/POOL do, RELU doesn't)

Convolution with Stride

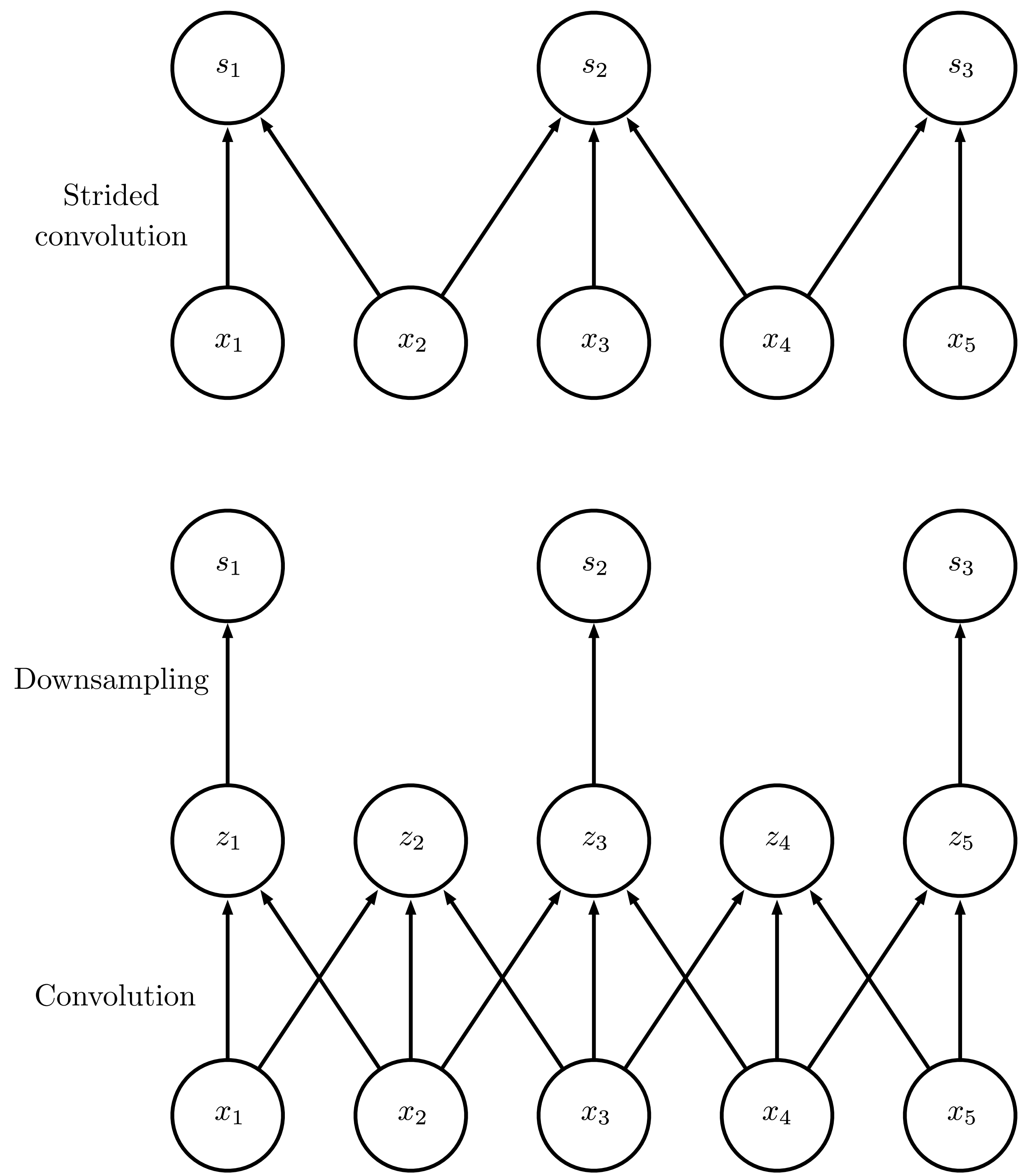


Figure 9.12

Zero Padding Controls Size

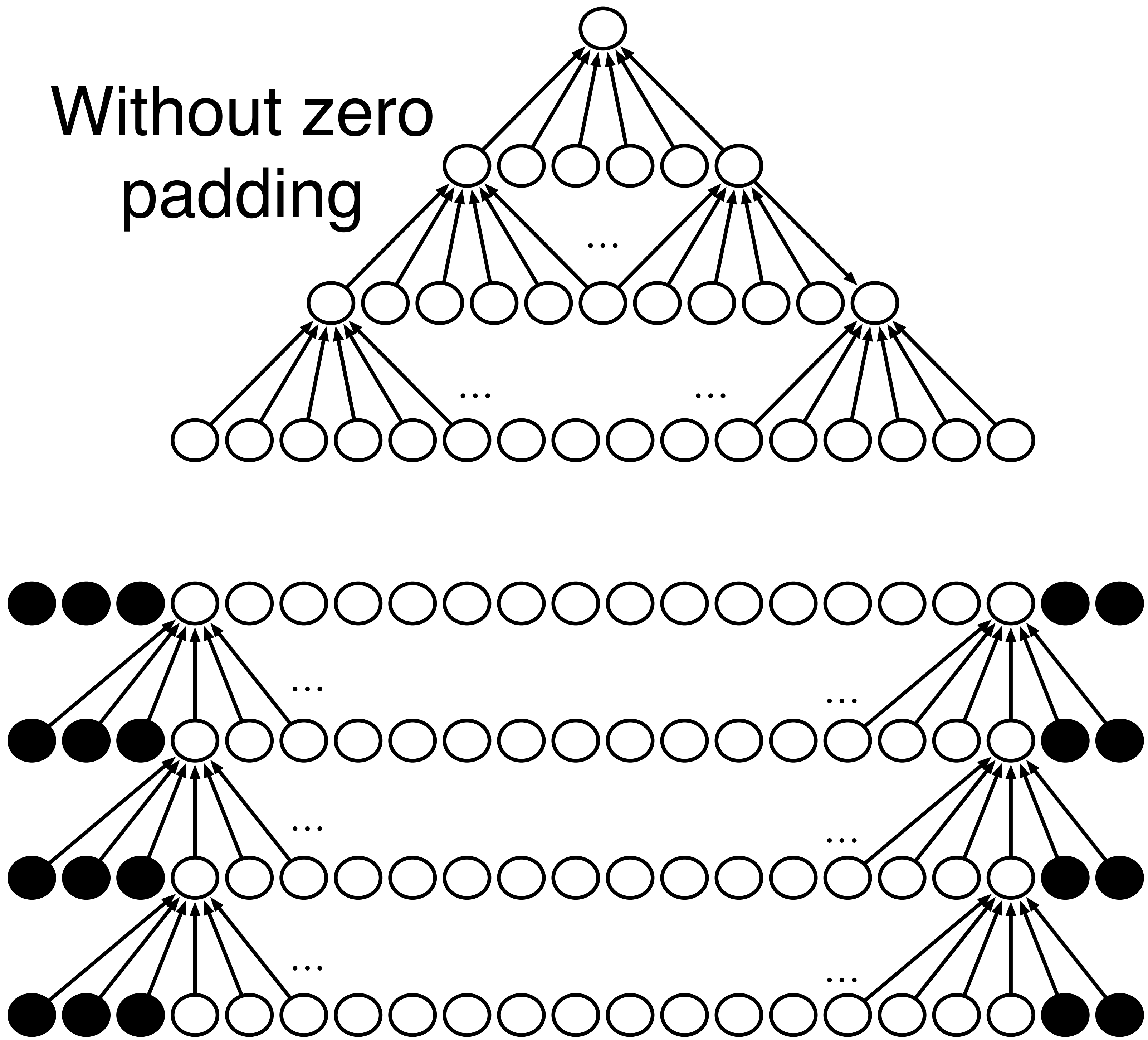
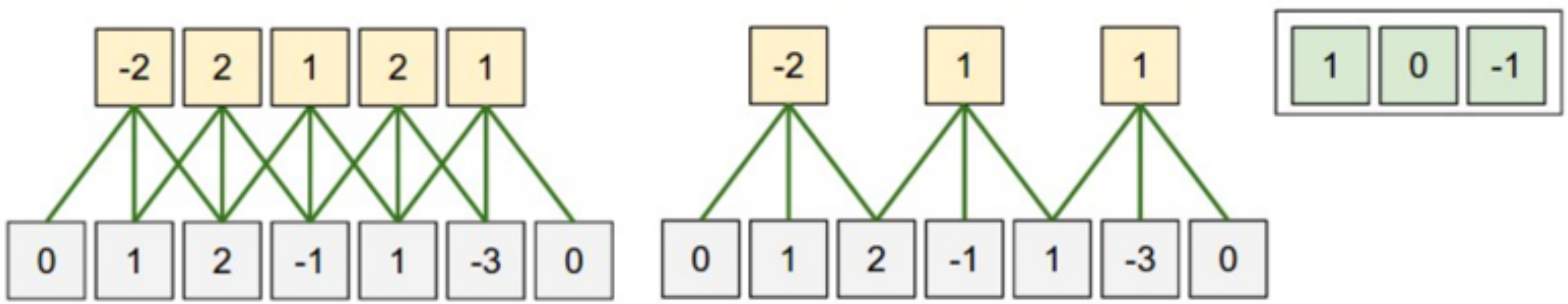
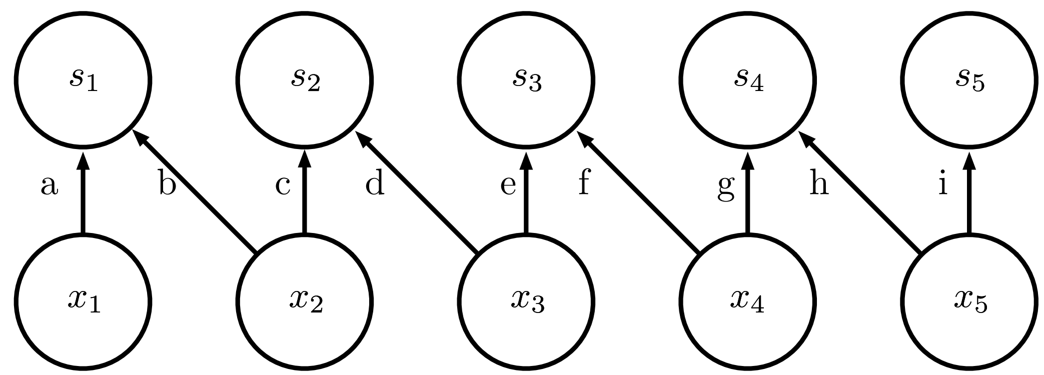


Figure 9.13

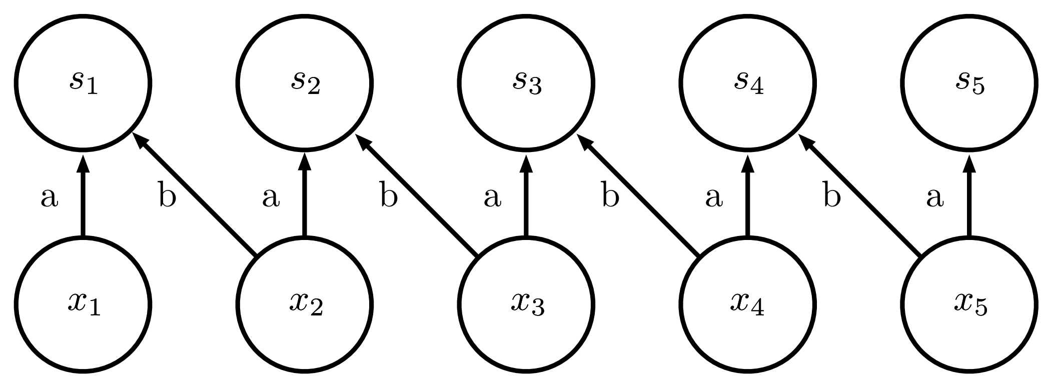
Output size with zero padding and stride



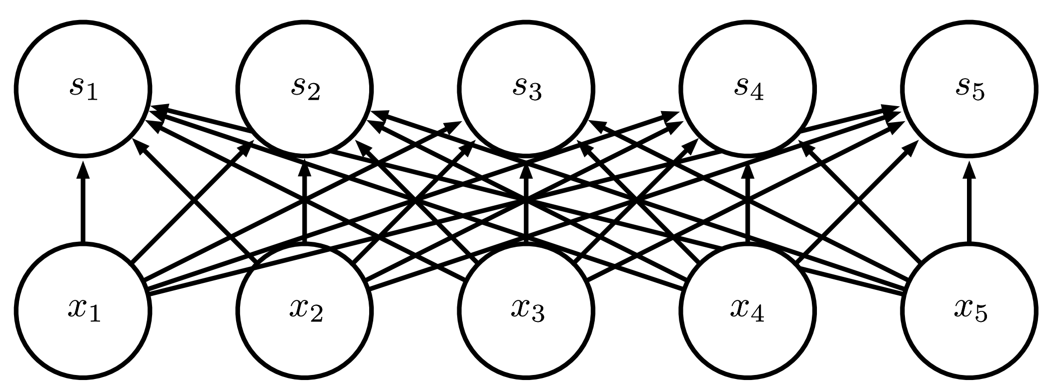
Kinds of Connectivity



Local connection: like convolution, but no sharing



Convolution



Fully connected

Figure 9.14

Partial Connectivity Between Channels

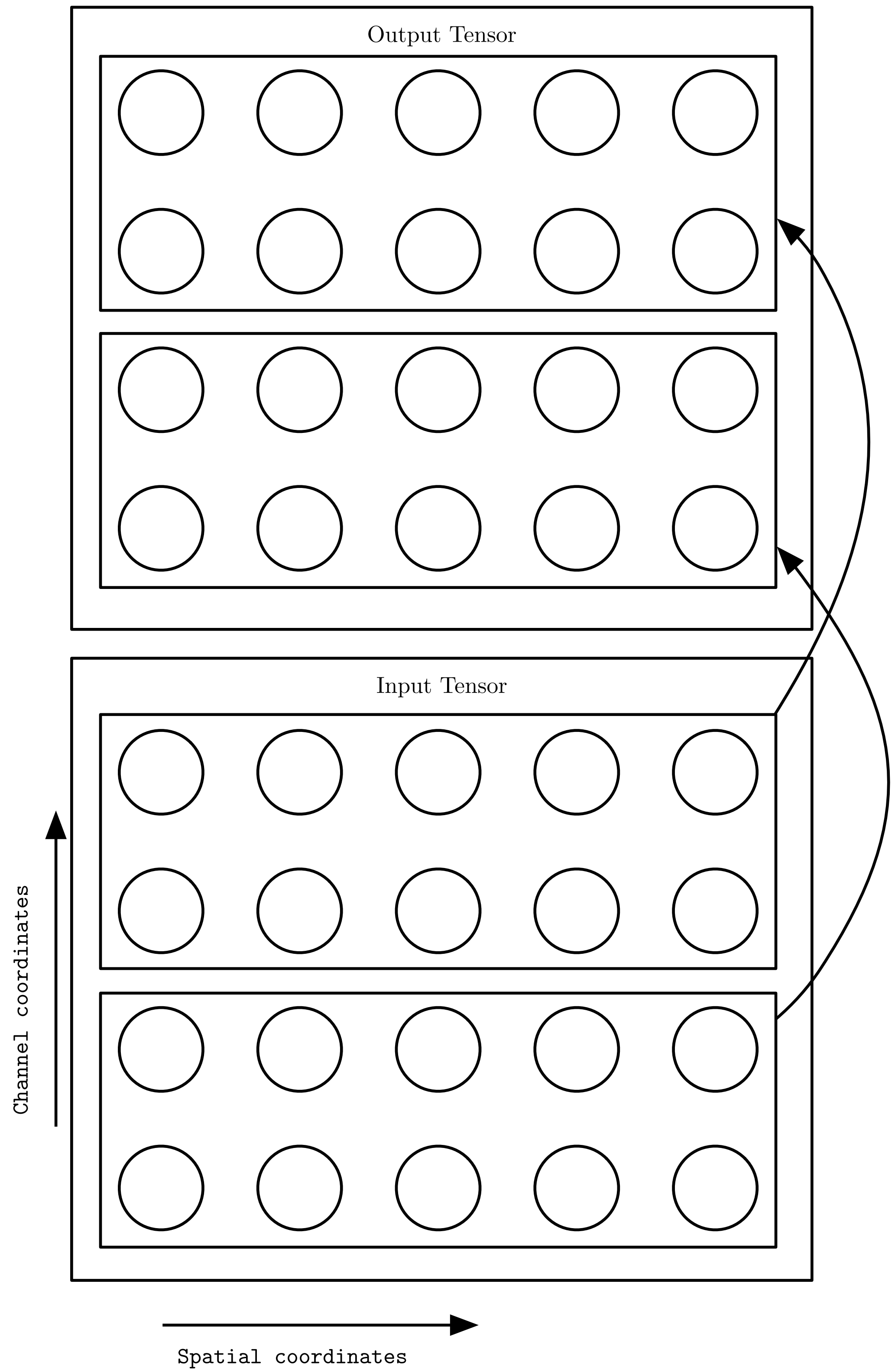
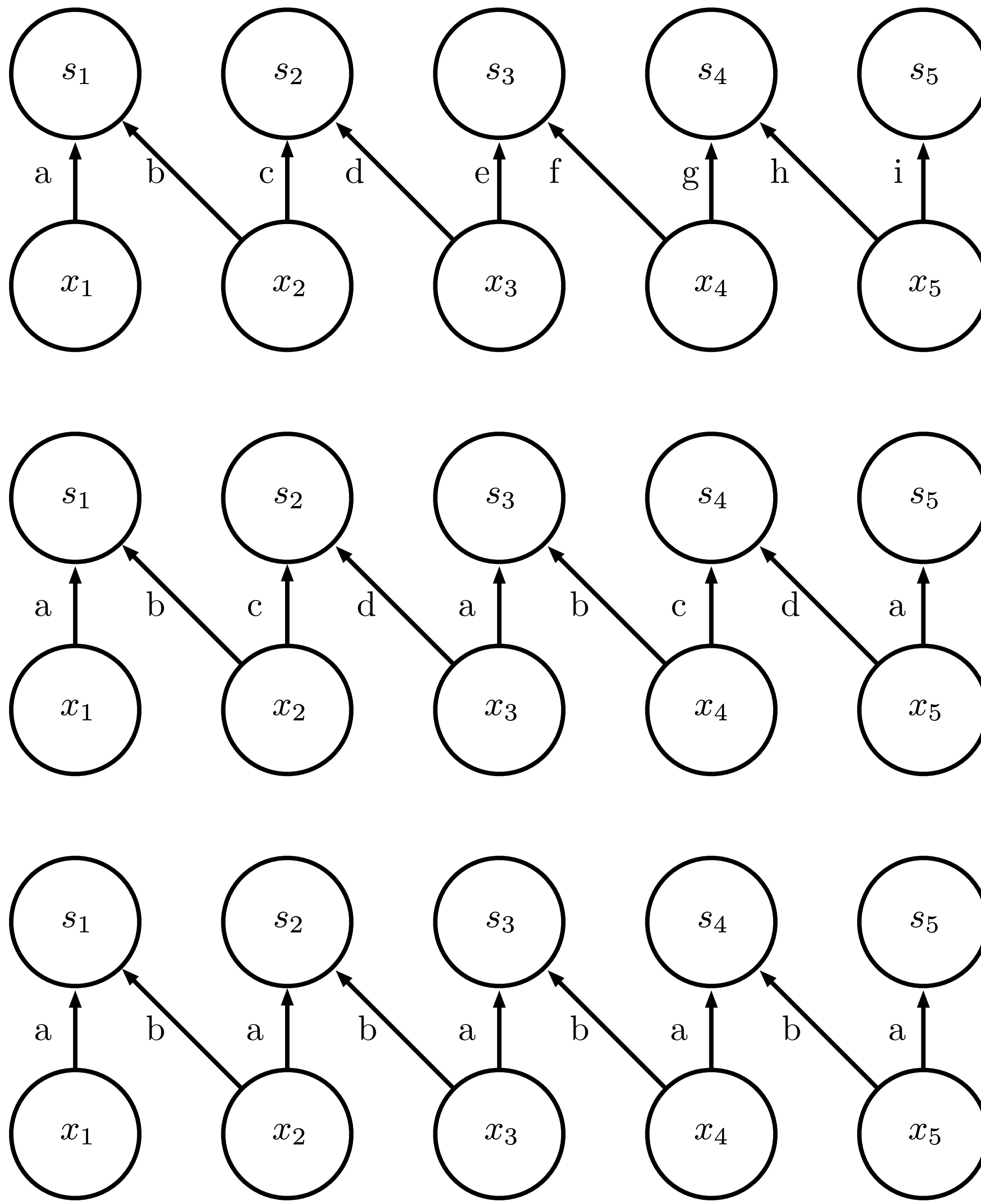


Figure 9.15

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

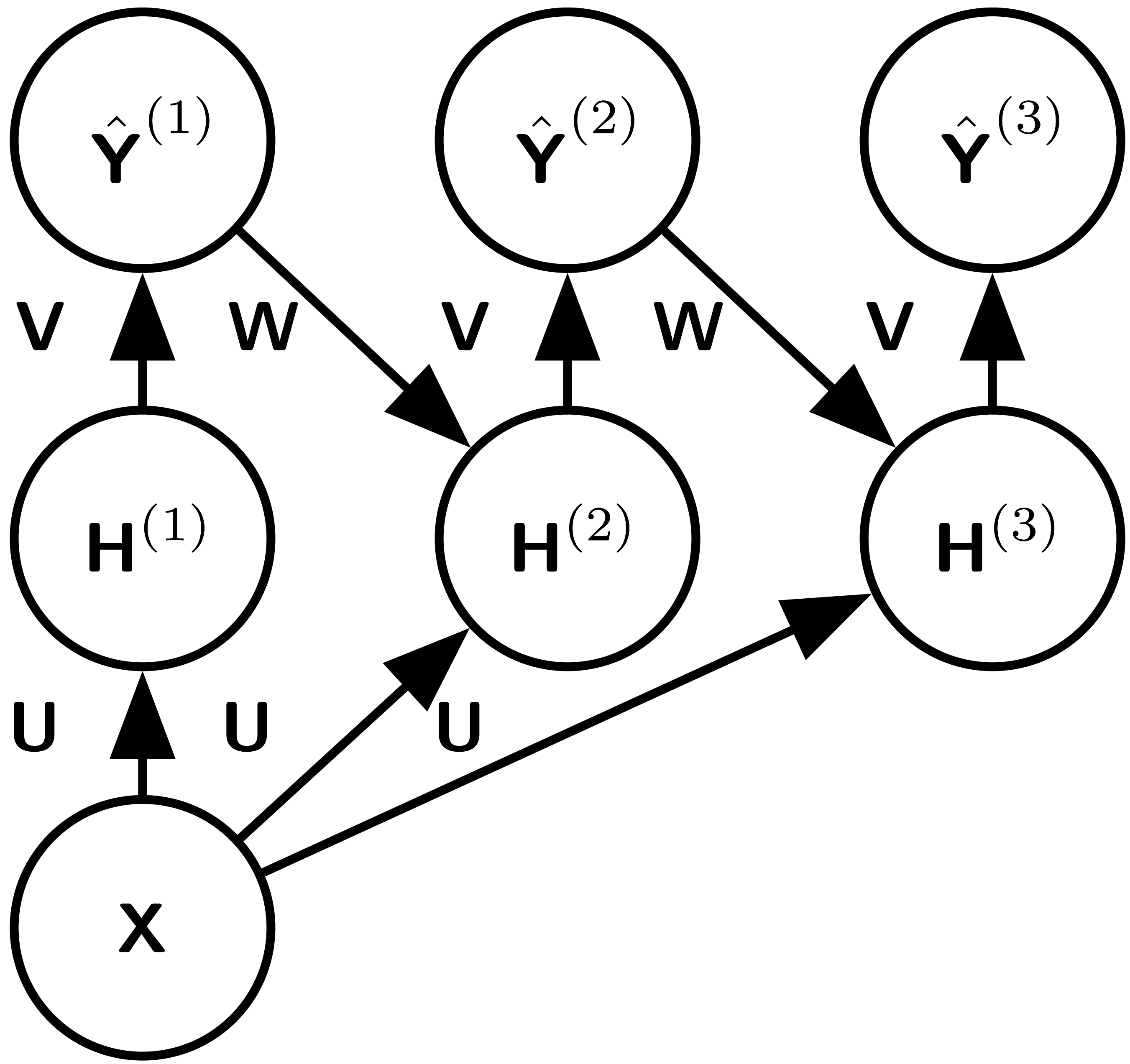


Figure 9.17

Gabor Functions

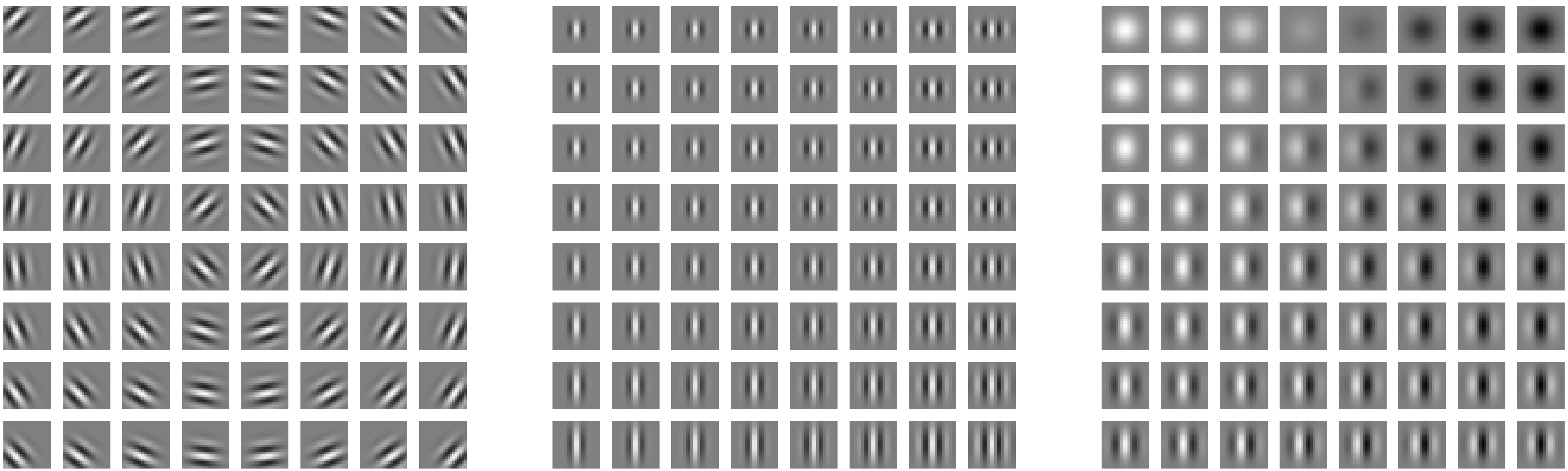


Figure 9.18

Gabor-like Learned Kernels

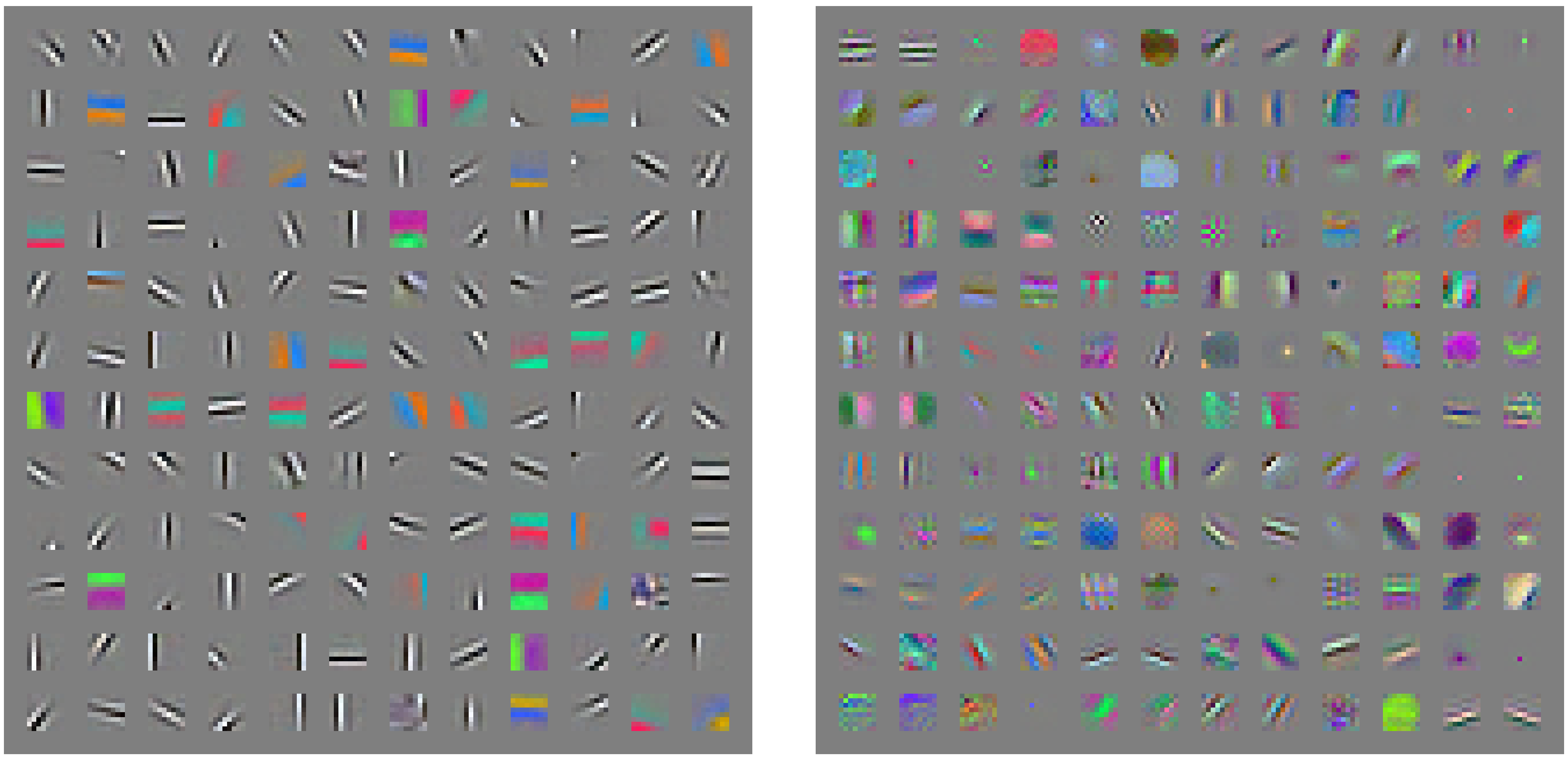


Figure 9.19

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.