Convolutional Networks

Lecture slides for Chapter 9 of *Deep Learning*

Ian Goodfellow

2016-09-12

Adapted by m.n. for CMPS 392

Convolutional networks

- Specialized kind of neural network for processing data that has a known, grid-like topology.
- Examples include
  - ❑ time-series data, which can be thought of as a 1D grid taking samples at regular time intervals,
  - ❑ and image data, which can be thought of as a 2D grid of pixels.
- Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

(Goodfellow 2016)

Key Idea

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

(Goodfellow 2016)

Convolution

- Convolution is just a weighted average:
  - $s(t) = \int x(a)w(t-a)$
  - $w(t)$ is a probability density function.
  - $w = 0$ for all negative arguments
- Denoted $s(t) = (x * w)(t)$
  - x is the input
  - w is the kernel
  - The output is the feature map
- Discrete convolution

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t-a)$$

(Goodfellow 2016)

2D convolution

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)\, K(i-m, j-n)$$

- ❑ $m = i,\ i-1,\ i-2, \ldots$
- ❑ $n = i,\ i-1,\ i-2, \ldots$
- ❑ $I(i,j)K(0,0) + I(i,j-1)K(0,1) + \cdots$

- Convolution is commutative:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n K(m,n) I(i-m, j-n)$$

- ❑ $m = 0, 1, 2, \ldots$
- ❑ $n = 0, 1\ , 2, \ldots$

- Cross correlation: (what we will really use / no kernel flipping)

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n K(m,n) I(i+m, j+n)$$

- Many machine learning libraries implement cross-correlation but call it convolution.

(Goodfellow 2016)

2D "valid" Convolution

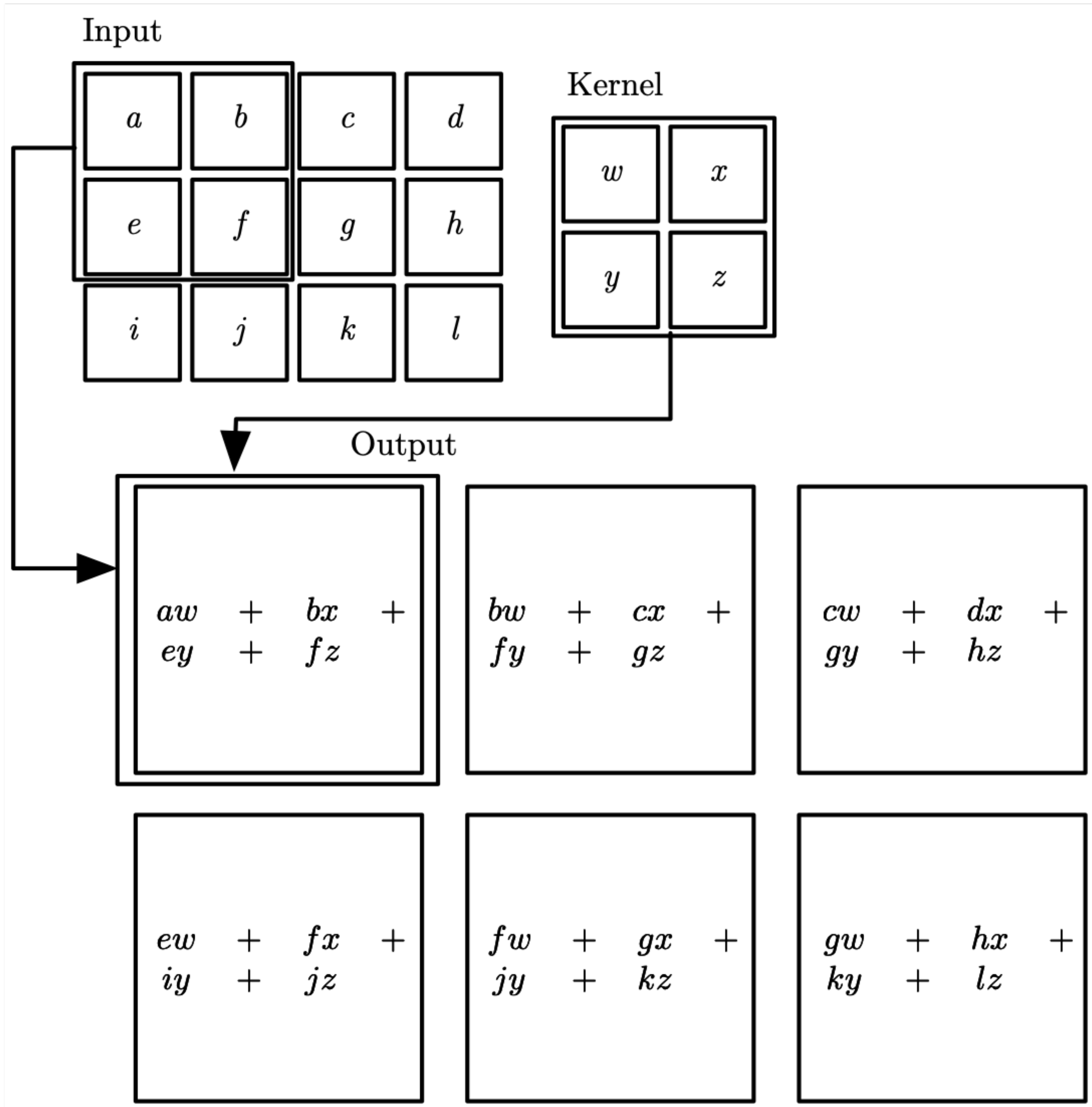


Figure 9.1

(Goodfellow 2016)

Motivation

- Scale up neural networks to process very large images / video sequences
  - ❑ Sparse connections
    - The kernel is smaller than the input: $O(m \times n) \rightarrow O(k \times n)$
  - ❑ Parameter sharing
    - The kernel is used at every position of the input
  - ❑ Equivariant representations
    - Automatically generalize across spatial translations of inputs
    - f is equivariant to g if $f\big(g(x)\big) = g(f(x))$
  - ❑ Works with inputs of variable size
- Applicable to any input that is laid out on a grid (1-D, 2-D, 3-D, …)

(Goodfellow 2016)

Sparse Connectivity (viewed from below)

Sparse connections due to small convolution kernel

Dense connections

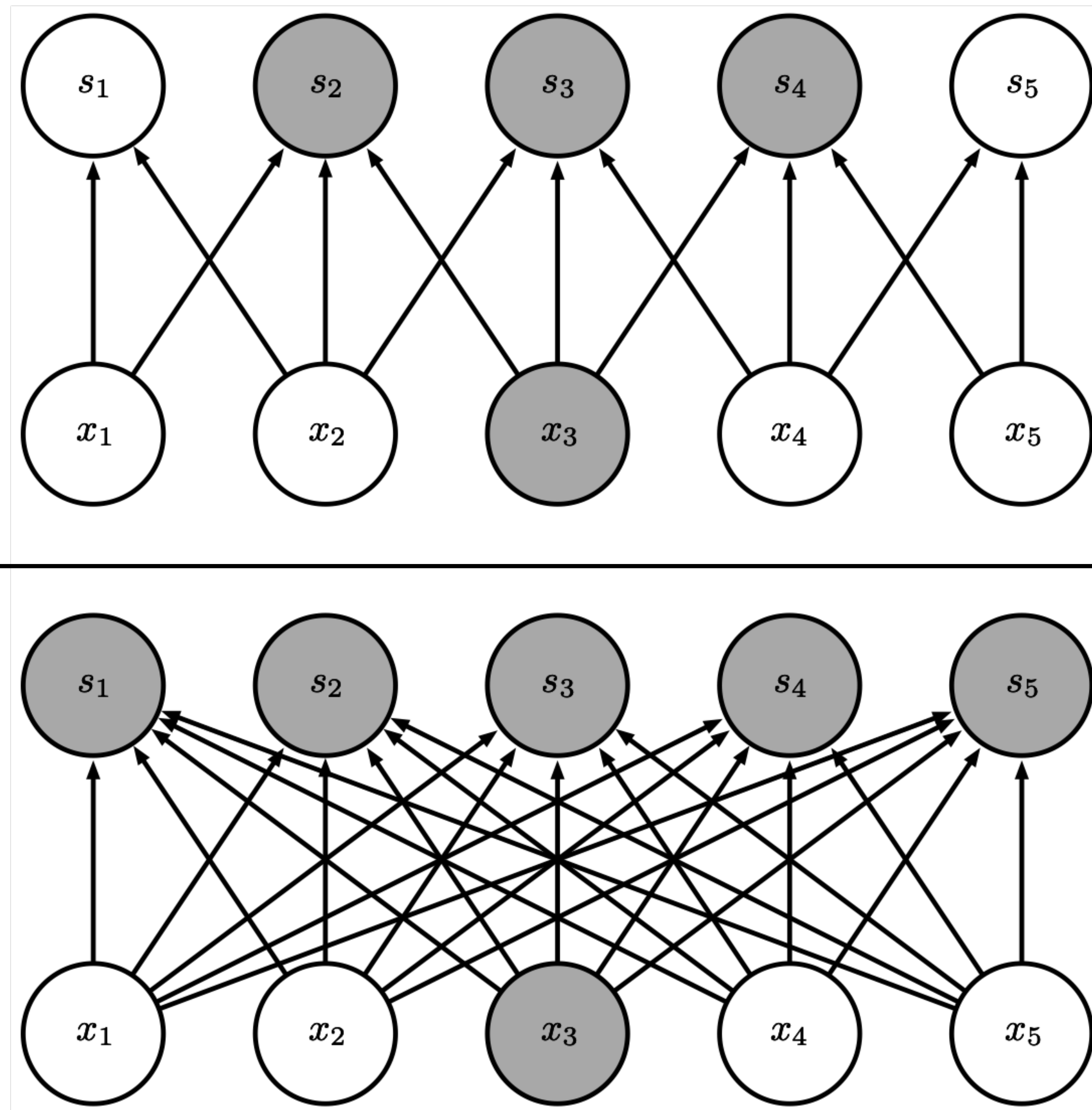


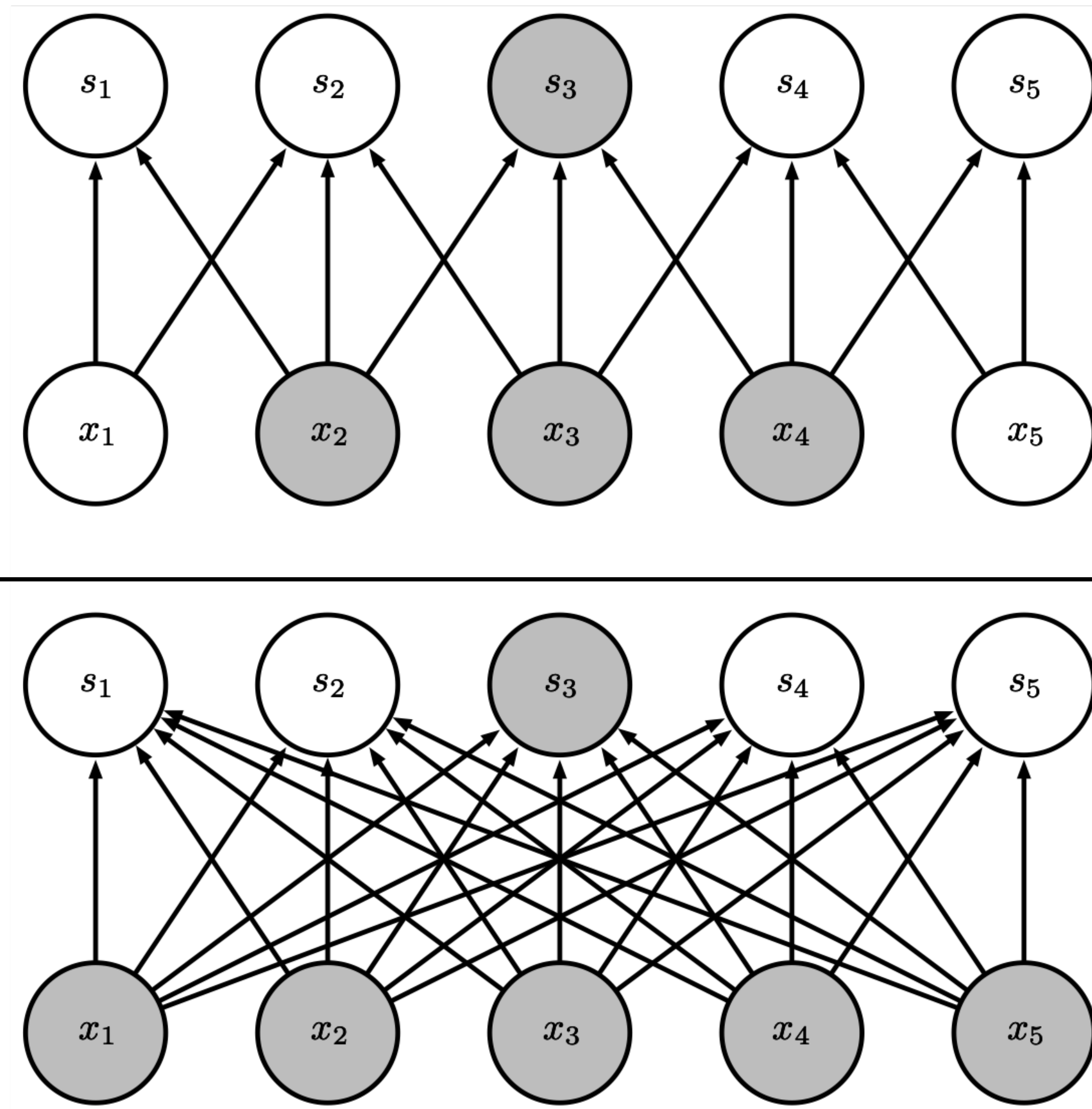
Figure 9.2

(Goodfellow 2016)

Sparse Connectivity (viewed from above)

Sparse connections due to small convolution kernel

Dense connections

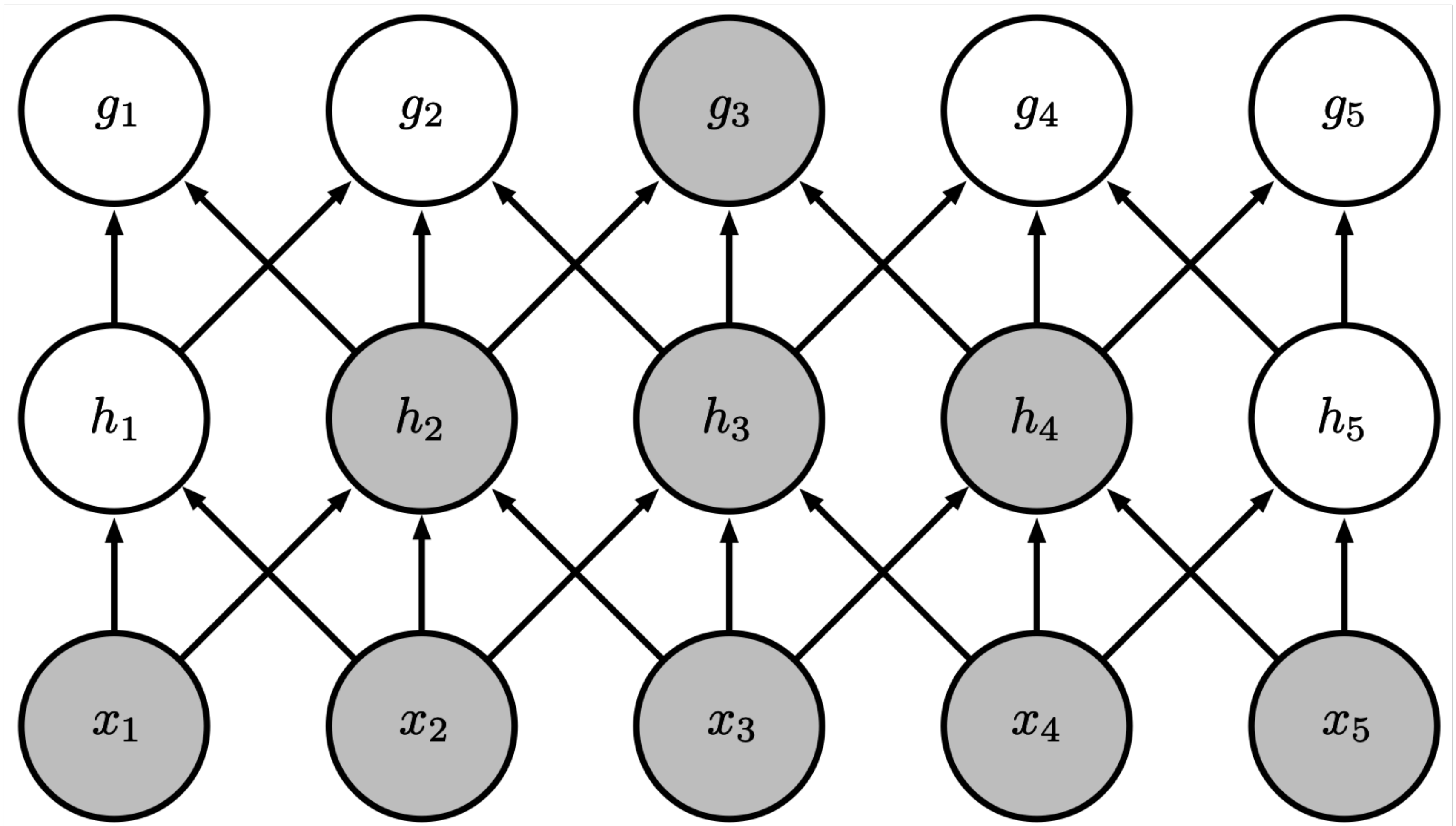


x_1, x_2, x_3 are the receptive field of s_3

Figure 9.3

(Goodfellow 2016)

Growing Receptive Fields



Even though direct connections in a convolutional net are very sparse, units in the deeper layers can be indirectly connected to all or most of the xinput image.

Figure 9.4

(Goodfellow 2016)

Parameter Sharing

Convolution shares the same parameters across all spatial locations

Traditional matrix multiplication does not share any parameters

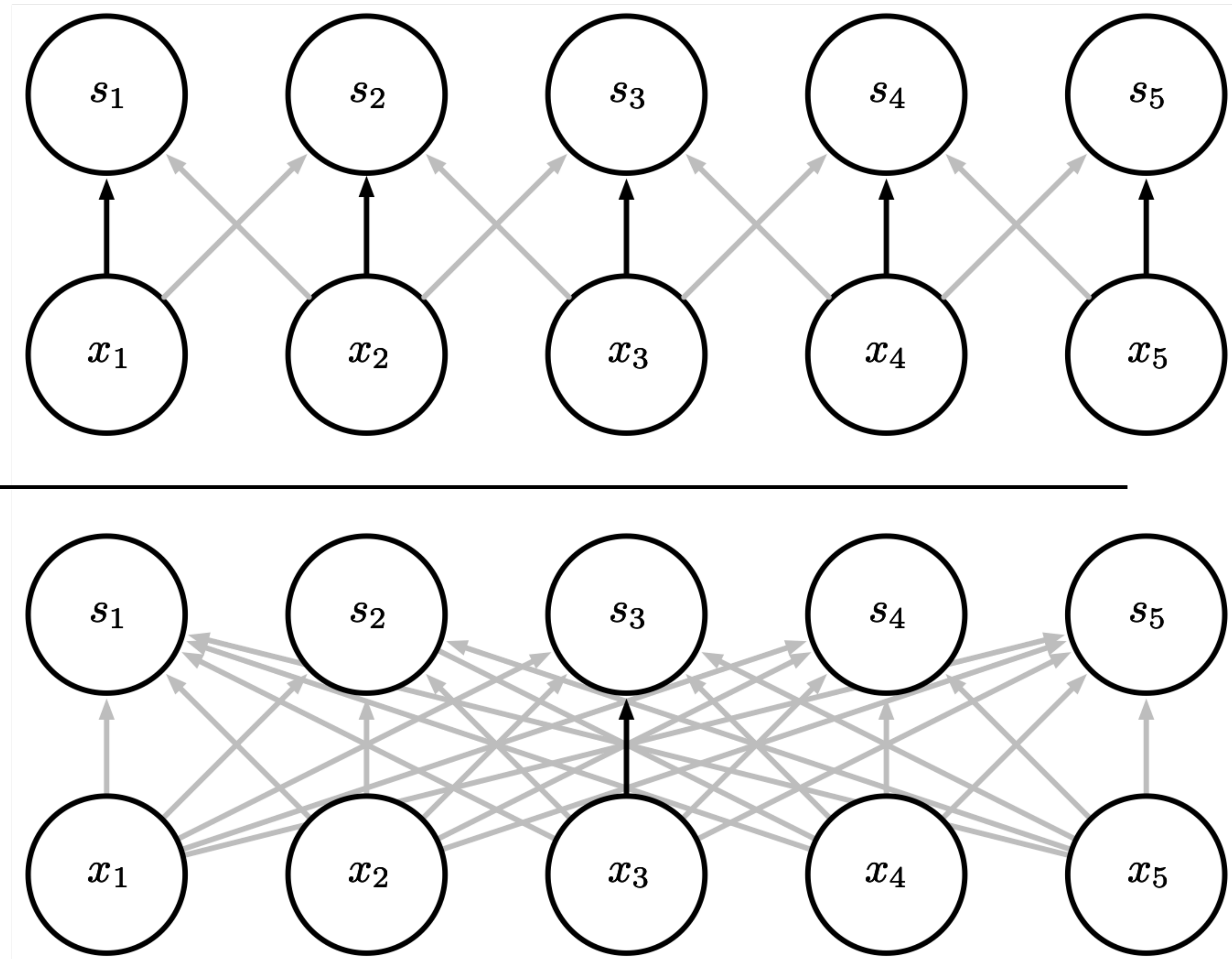


Figure 9.5

(Goodfellow 2016)

Edge Detection by Convolution

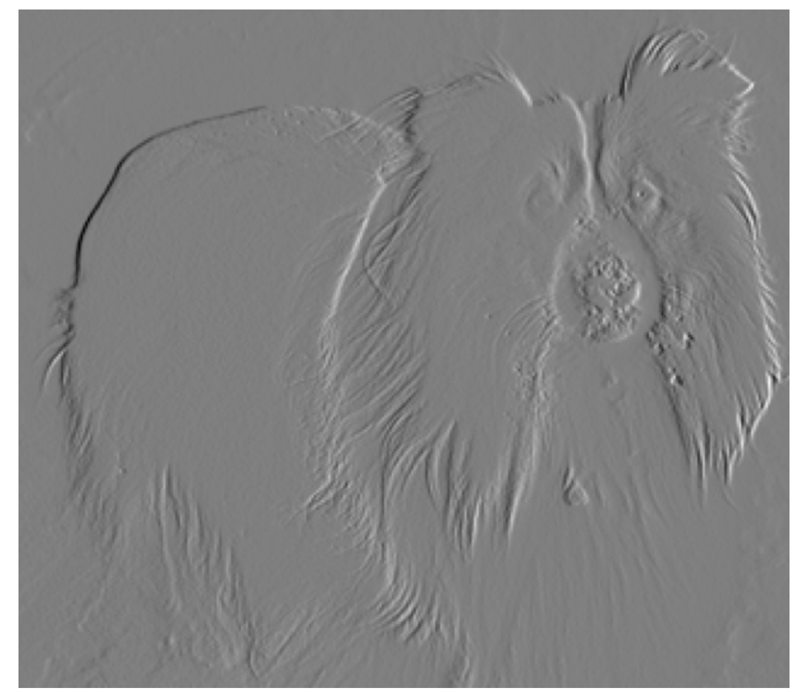
320×280

Input

1	-1

Kernel

319×280



Output

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	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)

(Goodfellow 2016)

Pooling

- A typical layer of a convolutional network consists of three stages
  - In the first stage, the layer performs several convolutions in parallel to produce a set of linear activations.
  - In the second stage, each linear activation is run through a nonlinear activation function, such as the rectified linear activation function. This stage is sometimes called the detector stage.
  - In the third stage, we use a pooling function to modify the output of the layer further.
- A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs:
  - The max pooling operation reports the maximum output within a rectangular neighborhood
  - Average pooling, weighted average pooling, L2 norm, etc.

(Goodfellow 2016)

Convolutional Network Components

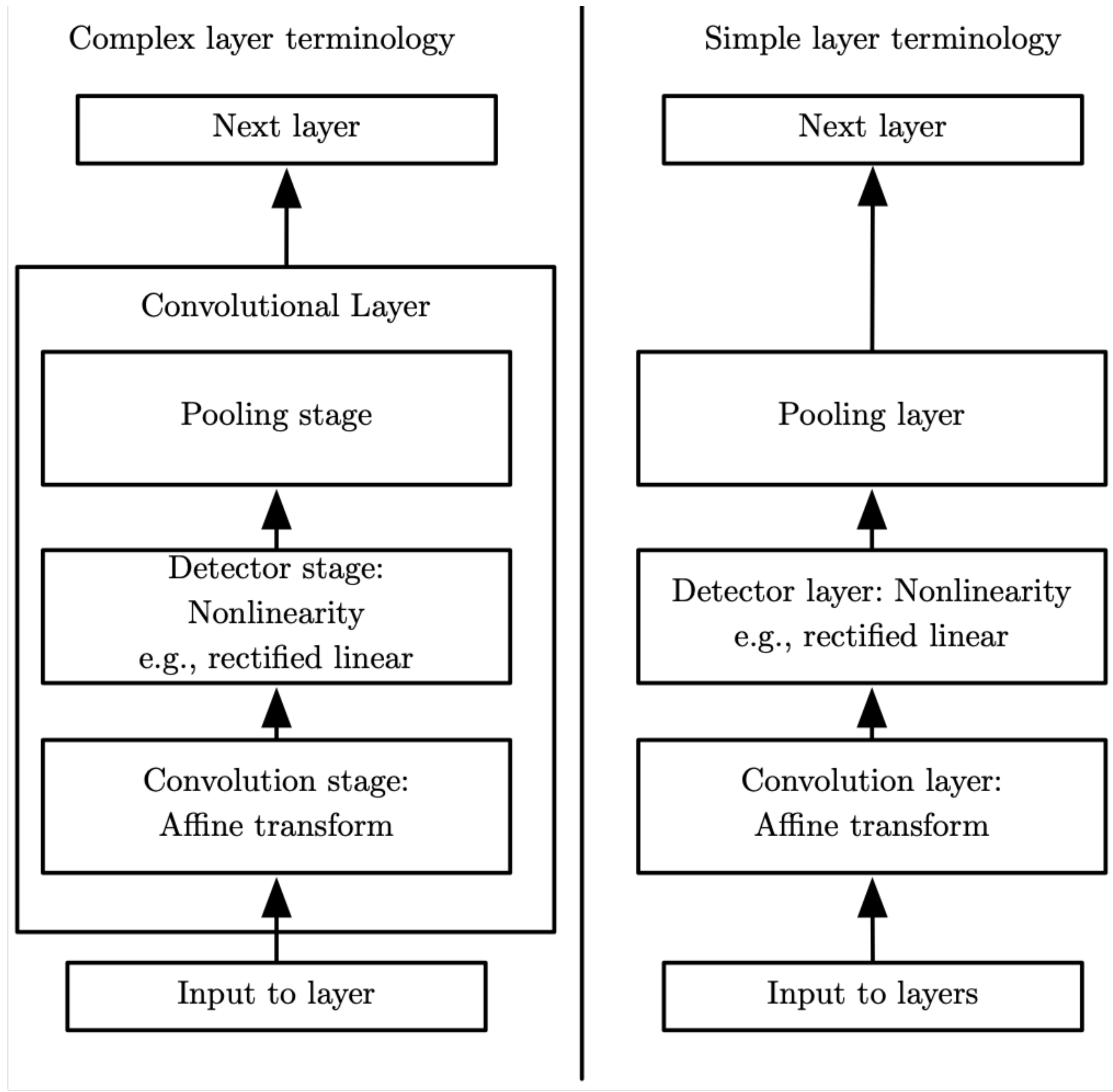


Figure 9.7

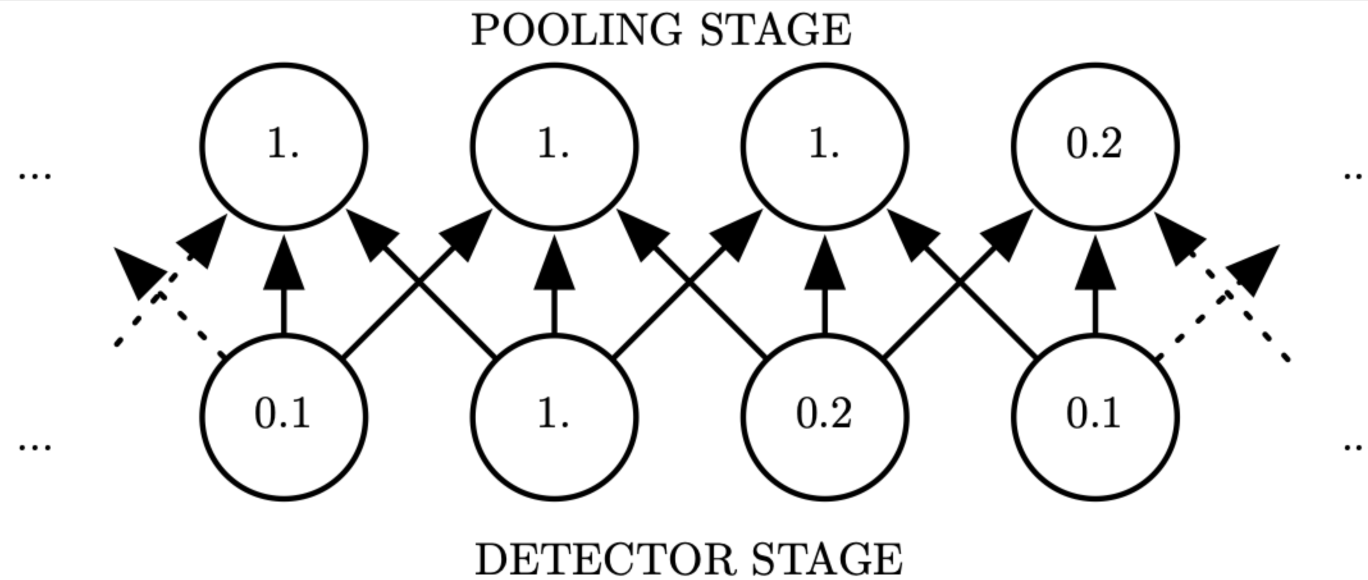
(Goodfellow 2016)

Why pooling?

- Invariance:
  - Pooling helps to make the representation become approximately invariant to small translations of the input.
  - Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change.
  - Invariance to local translation can be a very useful property if we care more about whether some feature is present than exactly where it is.
  - The use of pooling can be viewed as adding an infinitely strong prior that the function the layer learns must be invariant to small translations.
- Efficiency:
  - Pooling units summarize detector units by reporting summary statistics for pooling regions spaced k pixels apart rather than 1 pixel apart.
  - This improves the computational efficiency of the network
  - improved statistical efficiency and reduced memory requirements for storing the parameters.

(Goodfellow 2016)

Max Pooling and Invariance to Translation



Stride = 1
Width = 3

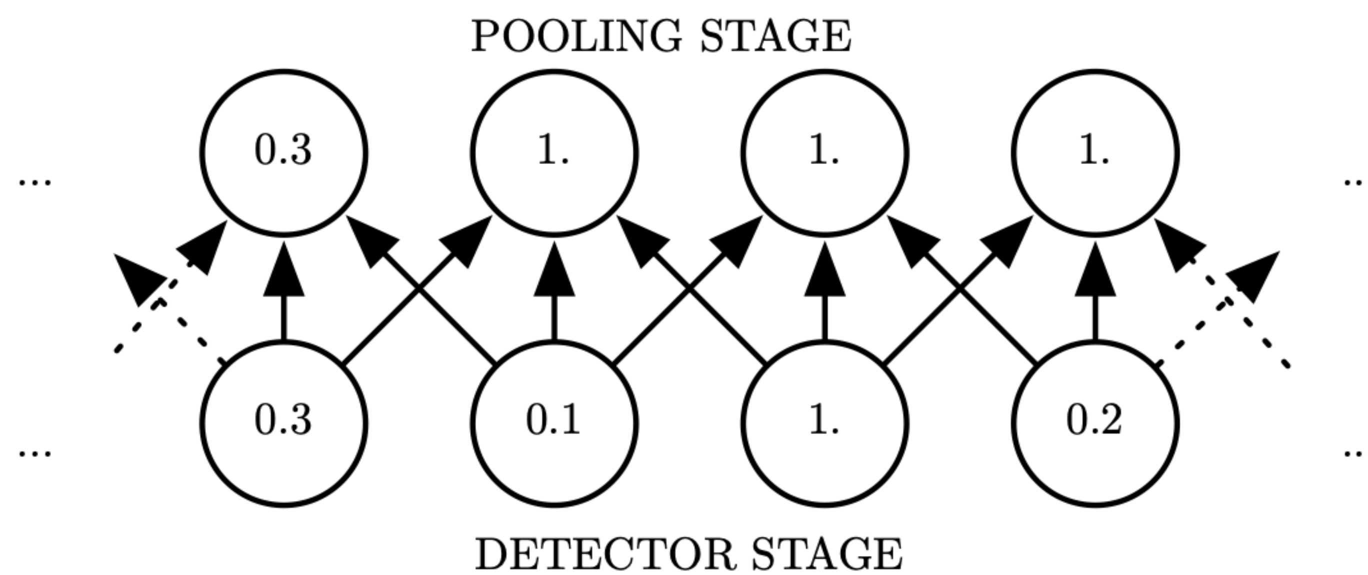


Figure 9.8

(Goodfellow 2016)

Cross-Channel Pooling and Invariance to Learned Transformations

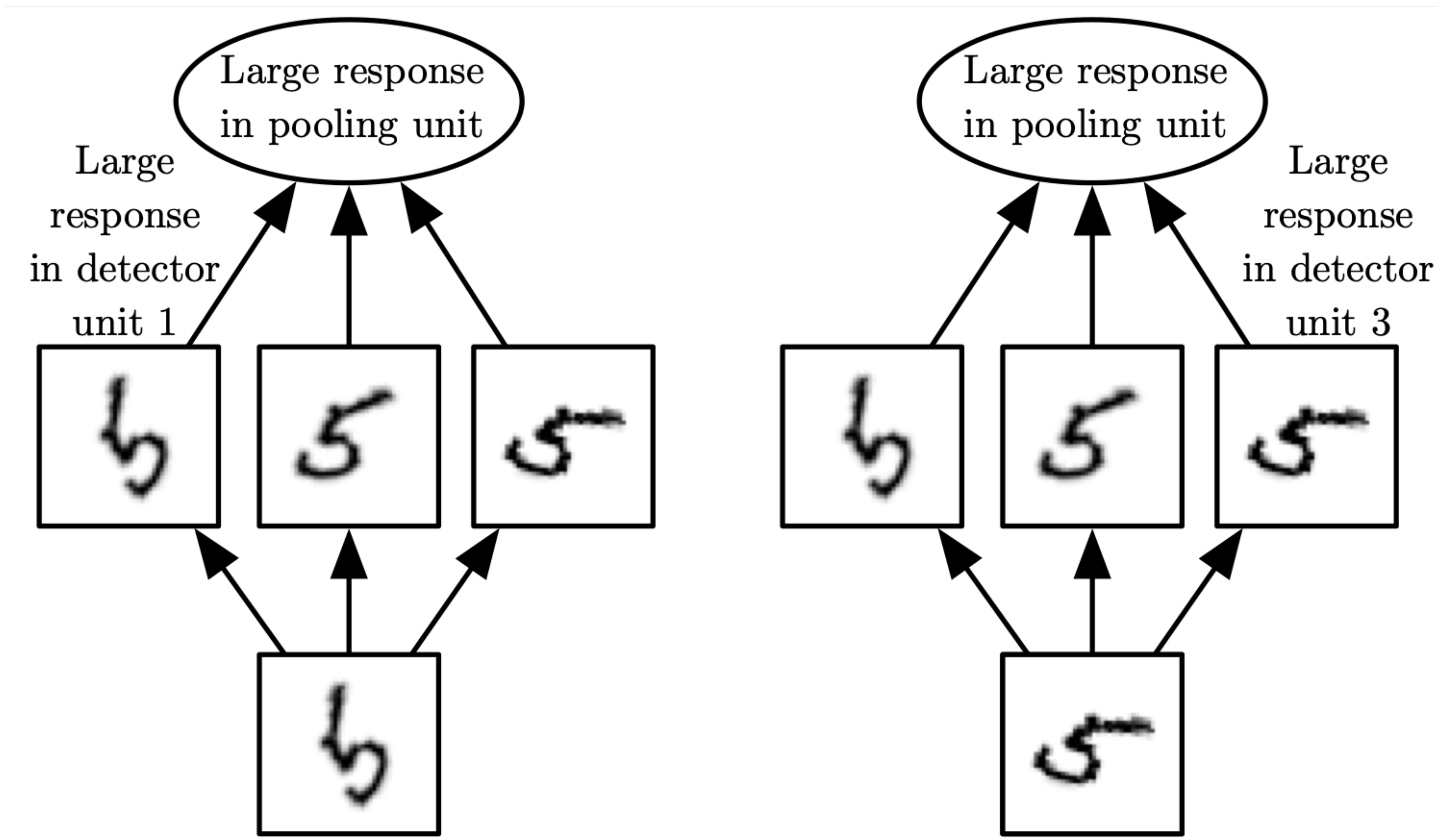


Figure 9.9

(Goodfellow 2016)

Pooling with Downsampling

Stride = 2
Width = 3

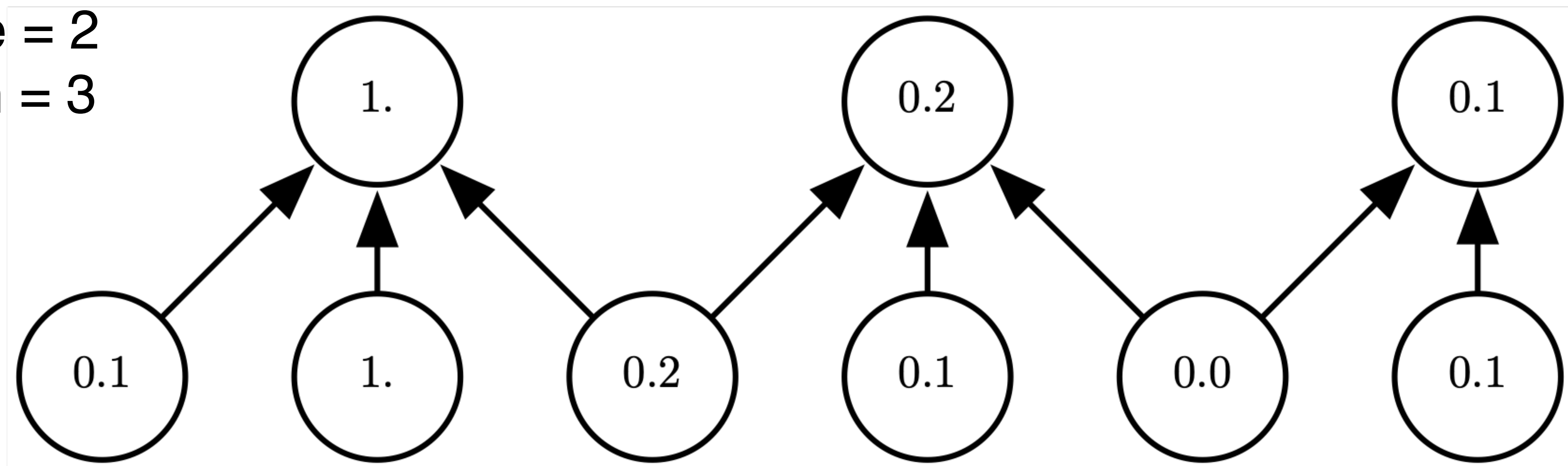
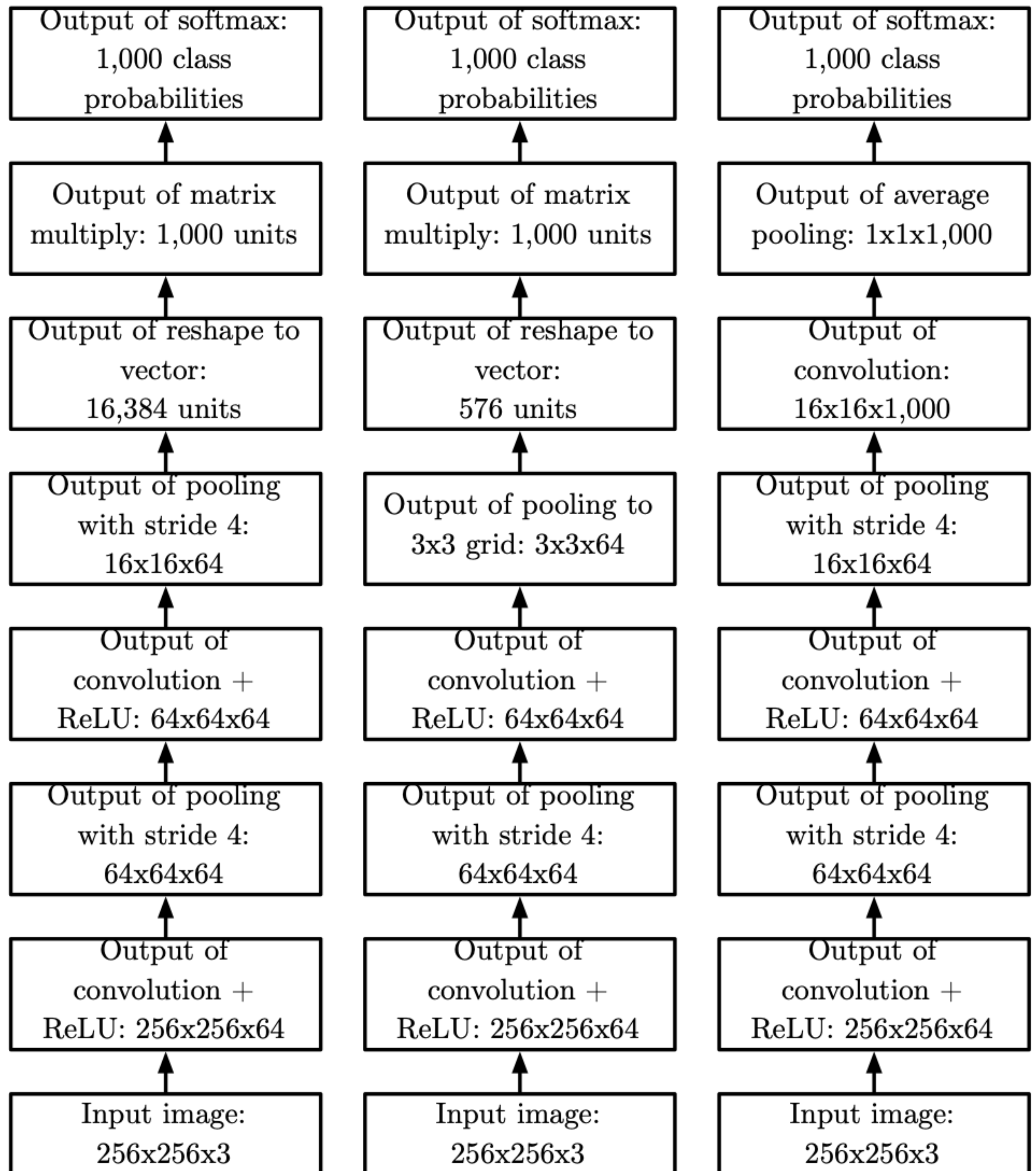


Figure 9.10

(Goodfellow 2016)

Example Classification Architectures



The specific strides and depths used in this figure are not advisable for real use

Figure 9.11

(Goodfellow 2016)

Variants of the convolution function

- We use many convolutions in parallel (multi-channel)
- The input is a 4-d tensor (batch, r, g, b). Let's ignore the batch index for the moment:

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n}$$

  - ❑ i: output channel
  - ❑ l: input channel (R,G,B)
  - ❑ m, n: offsets (row, column) $= 1,2,3, \ldots$
  - ❑ j, k: row, column
- Convolution with stride:

$$Z_{i,j,k} = c(K, V, s) = \sum_{l,m,n} V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n}$$

Stride	Output (row, column)	Input (row, column)
$s = 1$	$0,1,2, \ldots$	$0,1,2, \ldots$
$s = 2$	$0,1,2, \ldots$	$0,2,4, \ldots$
$s = 3$	$0,1,2, \ldots$	$0,3,6, \ldots$
$\ldots$		

(Goodfellow 2016)

Convolution with Stride

Stride of 2

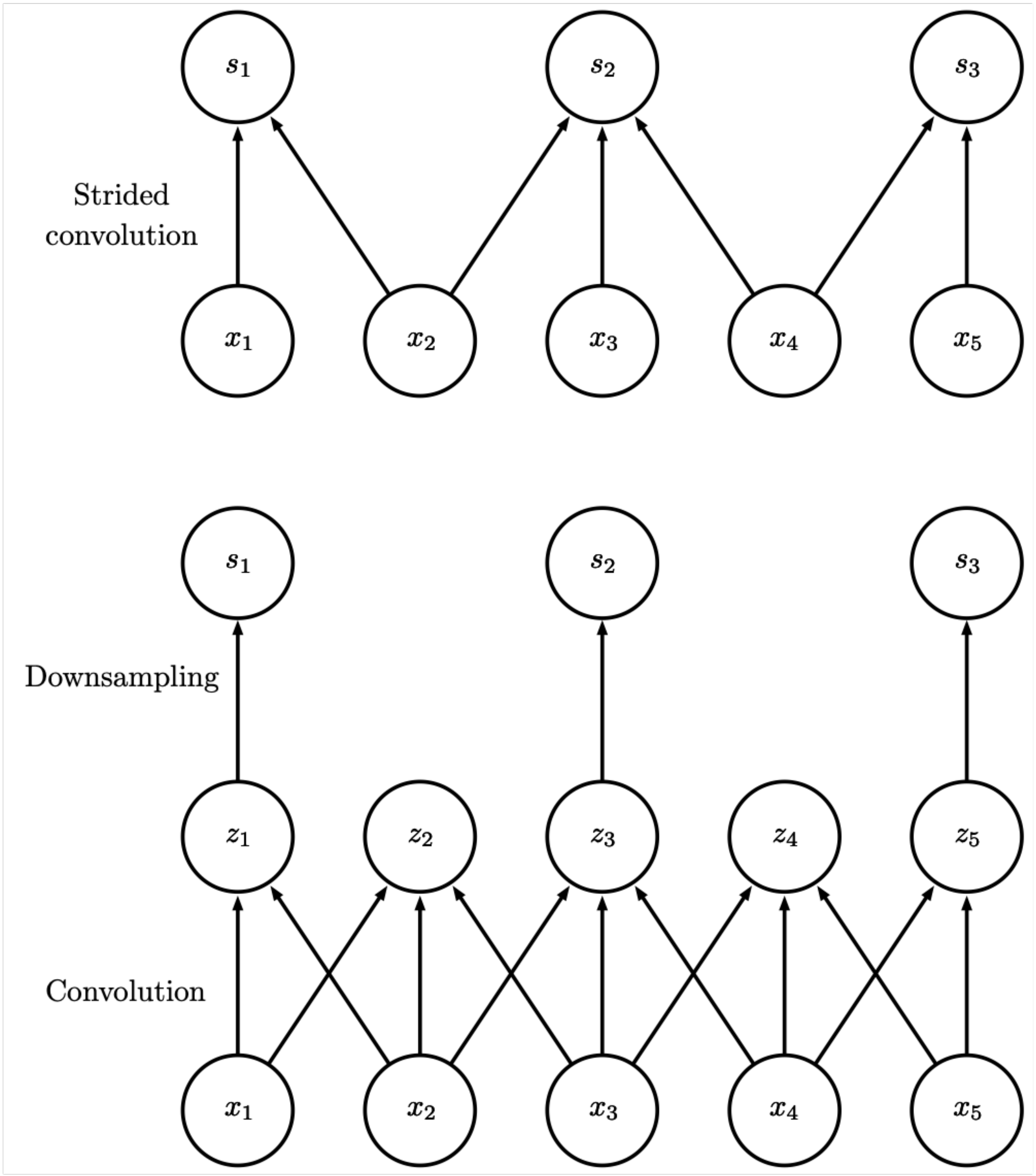


Figure 9.12

(Goodfellow 2016)

Zero padding

- Valid: no zero-padding, the convolution kernel is only allowed to visit positions where the kernel is contained entirely within the image.
  - ❑ input m, kernel k $\Rightarrow$ output $m - k + 1$
- Same: pad with enough zeroes to preserve the input dimension
  - ❑ input m $\Rightarrow$ output m
- Full: every input contributes to equal number of outputs
  - ❑ input m $\Rightarrow$ output $m + k - 1$

(Goodfellow 2016)

Zero Padding Controls Size

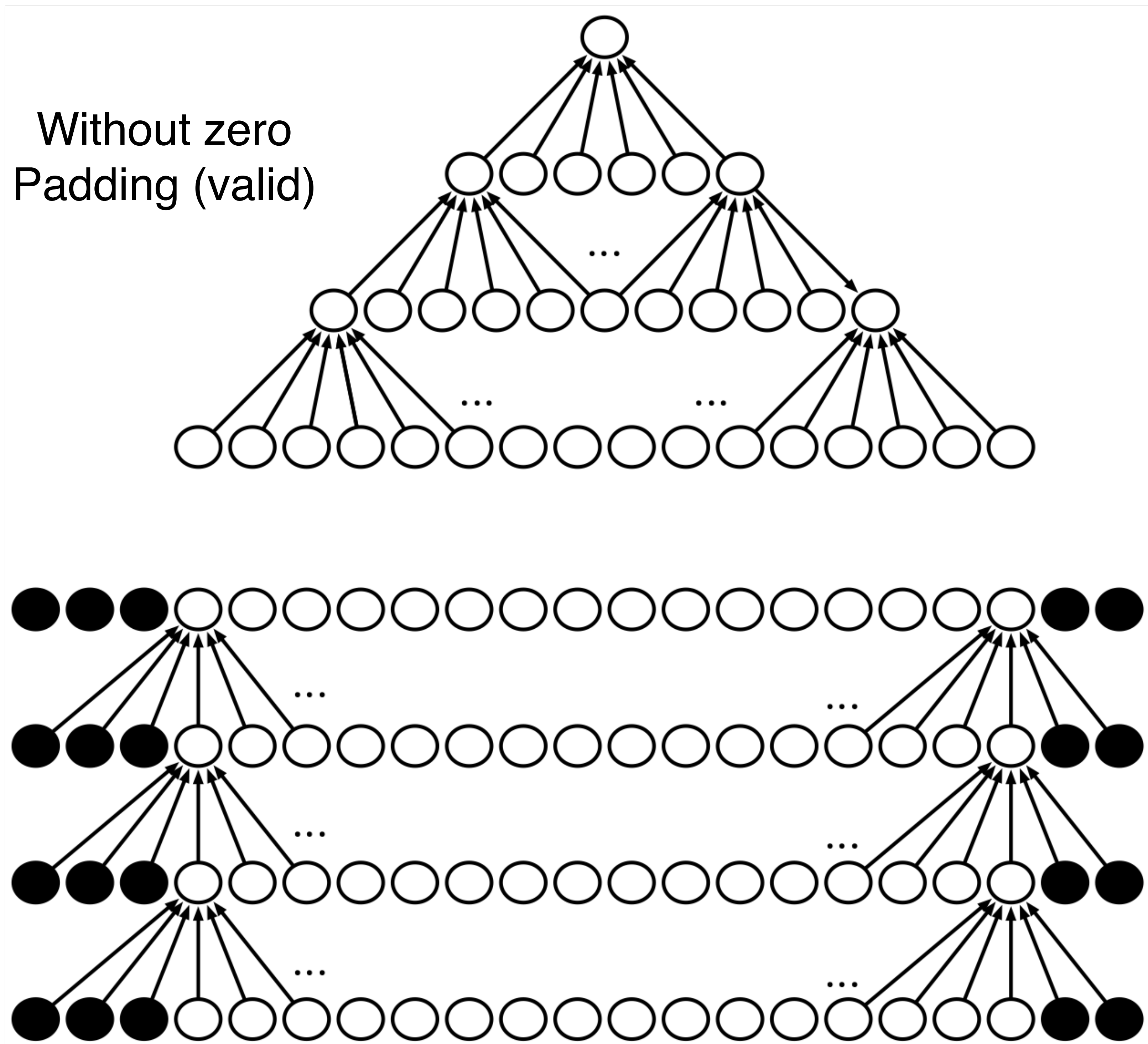
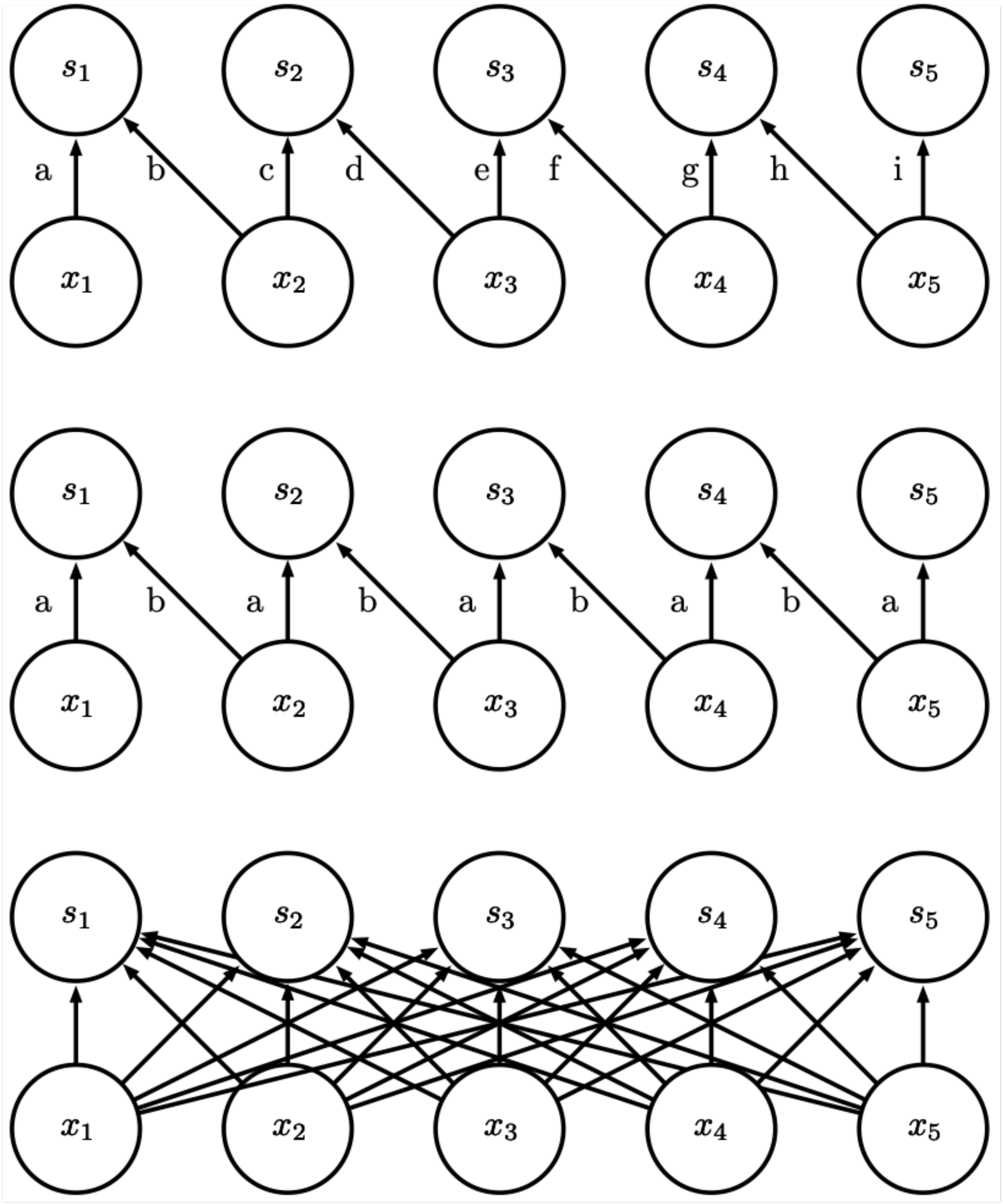


Figure 9.13

(Goodfellow 2016)

Kinds of Connectivity



Local connection: like convolution, but no sharing

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,j,k,l,m,n}$$

Convolution

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n}$$

Fully connected

Figure 9.14

(Goodfellow 2016)

Partial Connectivity Between Channels

A convolutional network with the first two output channels connected to only the first two input channels, and the second two output channels connected to only the second two input channels.

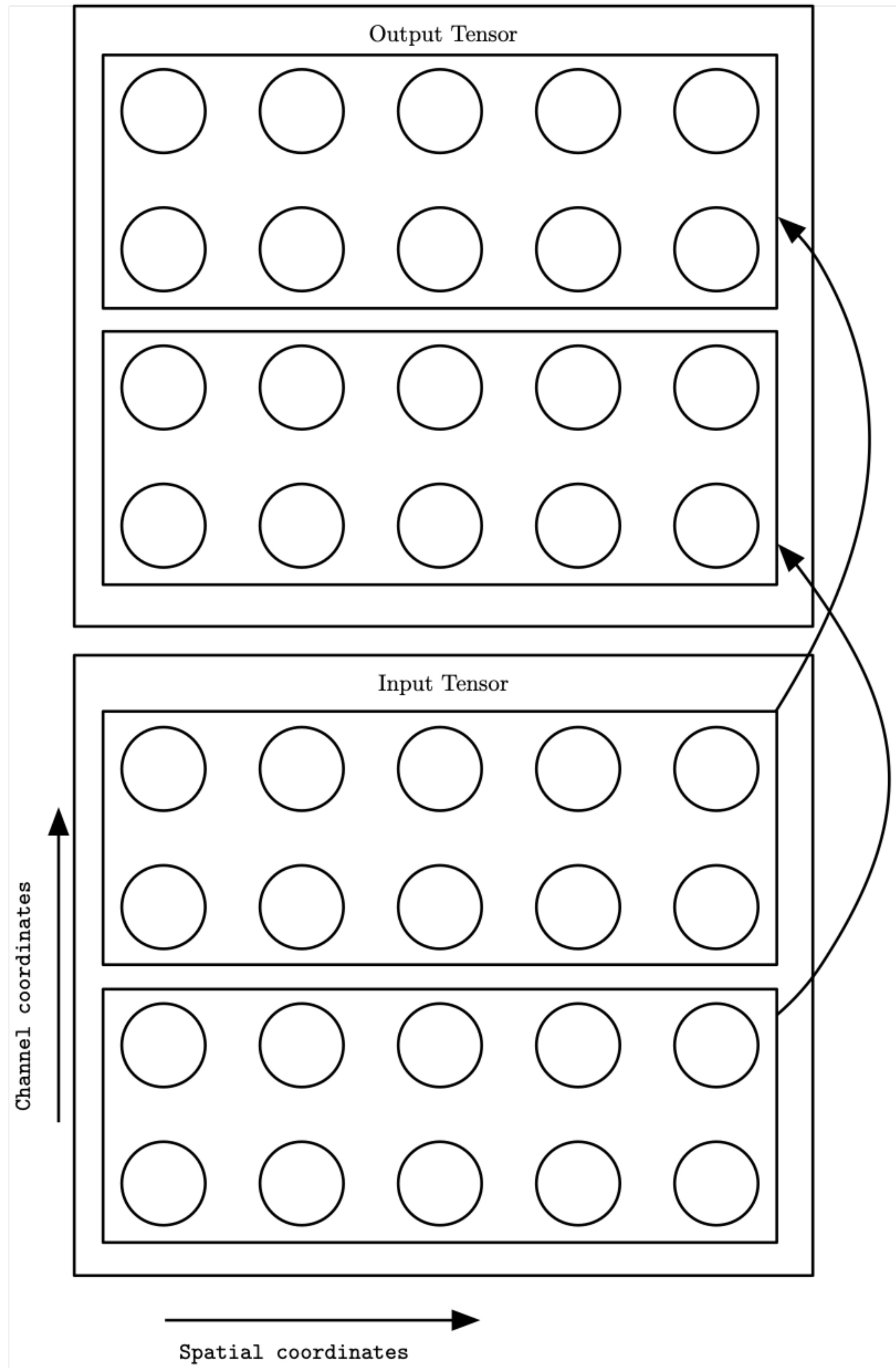
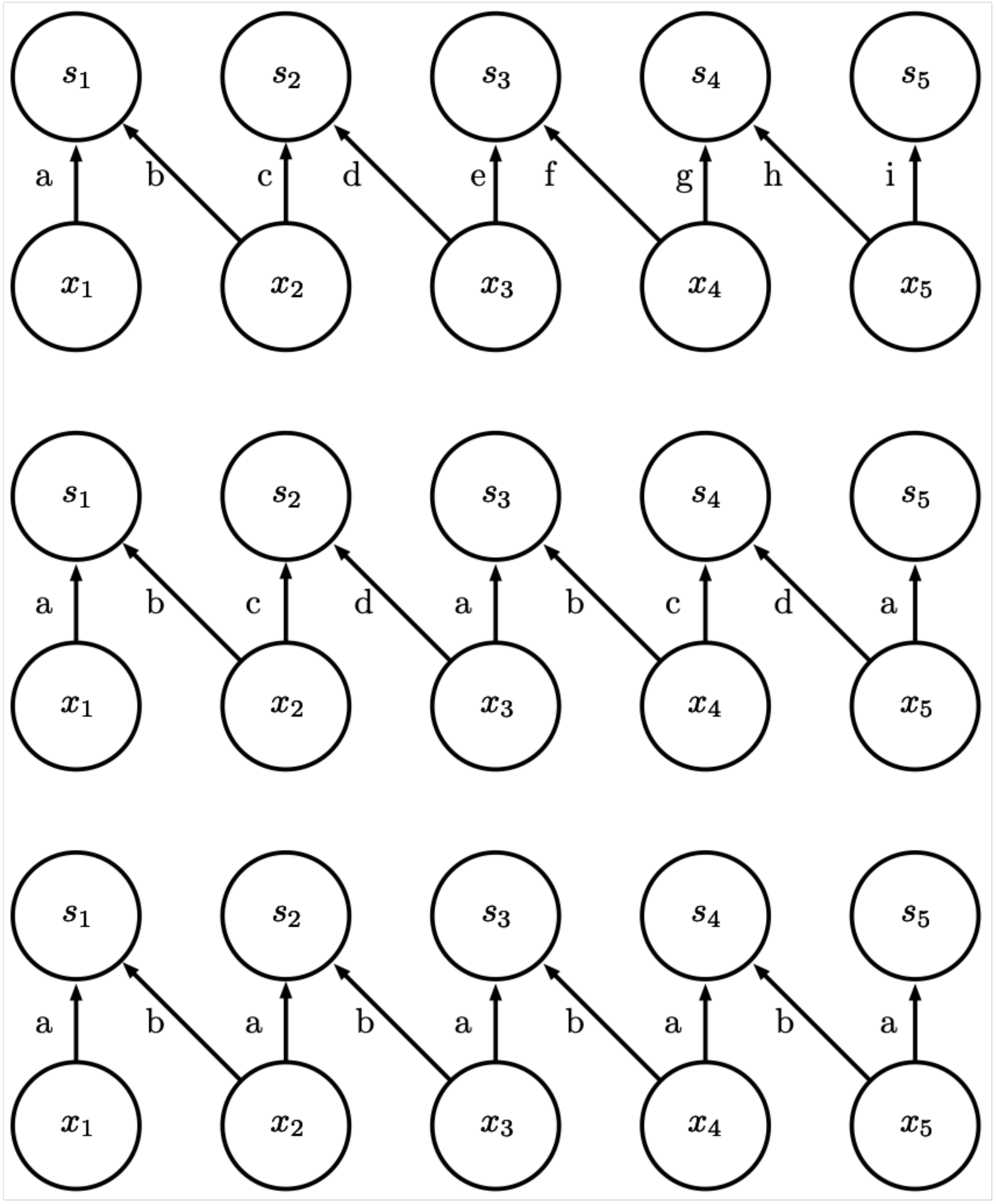


Figure 9.15

(Goodfellow 2016)

Tiled convolution

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n,\, j\%t+1,\, k\%t+1}$$



Local connection (no sharing)

Tiled convolution (cycle between groups of shared parameters)

Convolution (one group shared everywhere)

Figure 9.16

(Goodfellow 2016)

Three Operations

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

(Goodfellow 2016)

Gradient computation

- Image/input V, kernel K, conv. output $Z = c(K, V, s)$, Cost function $J(V, K)$

$$Z_{i,j,k} = c(K, V, s) = \sum_{l,m,n} V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n}$$

- We receive G, $G_{i,j,k} = \frac{\partial J(V,K)}{\partial z_{i,j,k}}$
  - Compute the gradient wrt weights of the kernel:

$$g(G, V, s)_{i,l,m,n} = \frac{\partial J(V, K)}{\partial K_{i,l,m,n}} = \sum_{j,k} \frac{\partial J(V, K)}{\partial z_{i,j,k}} \frac{\partial z_{i,j,k}}{\partial K_{i,l,m,n}}$$

$$= \sum_{j,k} G_{i,j,k} V_{l,(j-1)\times s+m,(k-1)\times s+n}$$

  - Compute the gradient wrt V:

$$h(K, G, s)_{l,u,v} = \frac{\partial J(V, K)}{\partial V_{l,u,v}} = \sum_{i} \sum_{\substack{j,m\ s.t.\\ (j-1)\times s+m=u}} \sum_{\substack{k,n\ s.t.\\ (k-1)\times s+n=v}} \frac{\partial J(V, K)}{\partial z_{i,j,k}} \frac{\partial z_{i,j,k}}{\partial V_{l,u,v}}$$

$$= \sum_{i} \sum_{\substack{j,m\ s.t.\\ (j-1)\times s+m=u}} \sum_{\substack{k,n\ s.t.\\ (k-1)\times s+n=v}} G_{i,j,k} K_{i,l,m,n}$$

(Goodfellow 2016)

Deconvolution

- More generally, the function $h(K, H, s)$ is called deconvolution
- Can be used for reconstruction: $R = h(K, H, s)$ in an autoencoder (similar to PCA)
- To train:
  - ❑ Receive a gradient E (w.r.t. R)
  - ❑ Compute the gradient w.r.t. K:
    - This is given by $g(H, E, s)$
  - ❑ Compute the gradient wrt H:
    - This is given by $c(K, E, s)$

(Goodfellow 2016)

Data types

	Single channel	Multichannel
1-D	Audio waveform (amplitude over time) Channel: Amplitude Dimension: T	Skeleton animation data Each channel in the data represents the angle about one axis of one joint of a character's skeleton. Channels: Angles Dimesion: T
2-D	Audio data that has been transformed with a Fourier transform. • Rows correpsond to frequencies. (equivariance to a shift in octaves) • Columns correspond to different points in time. (equivariance to shifts in time) Channel: Amplitude Dimensions: F,T	Color Image Data Channels: R,G,B Dimensions: X,Y
3-D	Volumetric data such as provening from medical imaging technology Channel: GrayScale Dimensions: X,Y,Z	Color video data Channels: R,G,B Dimesions: X,Y,T

(Goodfellow 2016)

Structures output
Recurrent Pixel Labeling

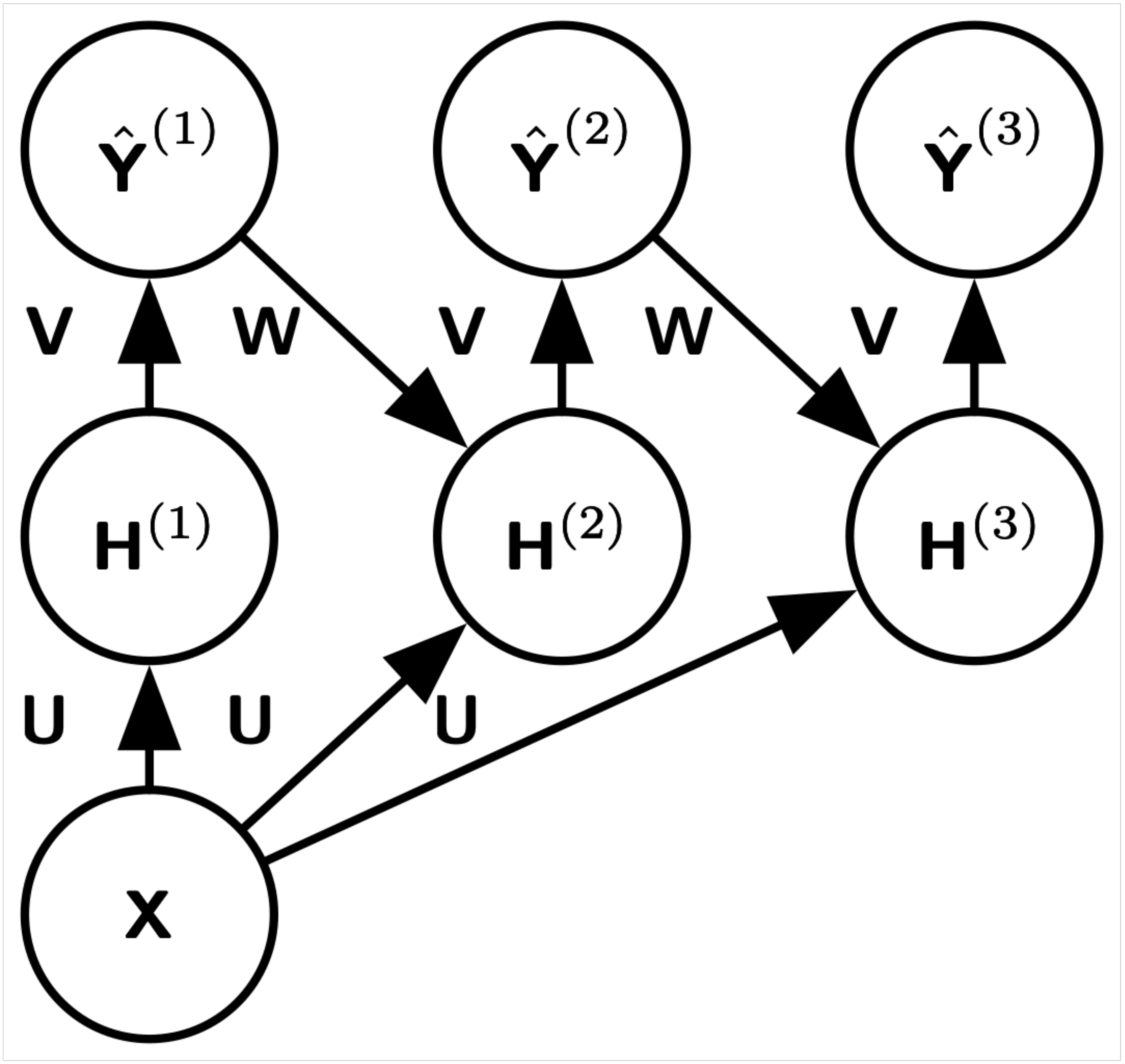


Figure 9.17

(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.

Watch

- https://www.youtube.com/watch?v=Xogn6veSyxA

(Goodfellow 2016)