UNIVERSITÀ PARTHENOPE
DiST
Dipartimento di Scienze e Tecnologie

Machine Learning (part II)

Convolutional Neural Network

Angelo Ciaramella

Convolutional Neural 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, …)

Introduction

- Convolutional Neural Networks (CNN)
  - processing data that has a known grid-like topology
    - e.g., time series and image data
  - use convolution in place of general matrix multiplication in at least one of their layers

- Everything else stays the same
  - Maximum likelihood
  - Back-propagation
  - etc.

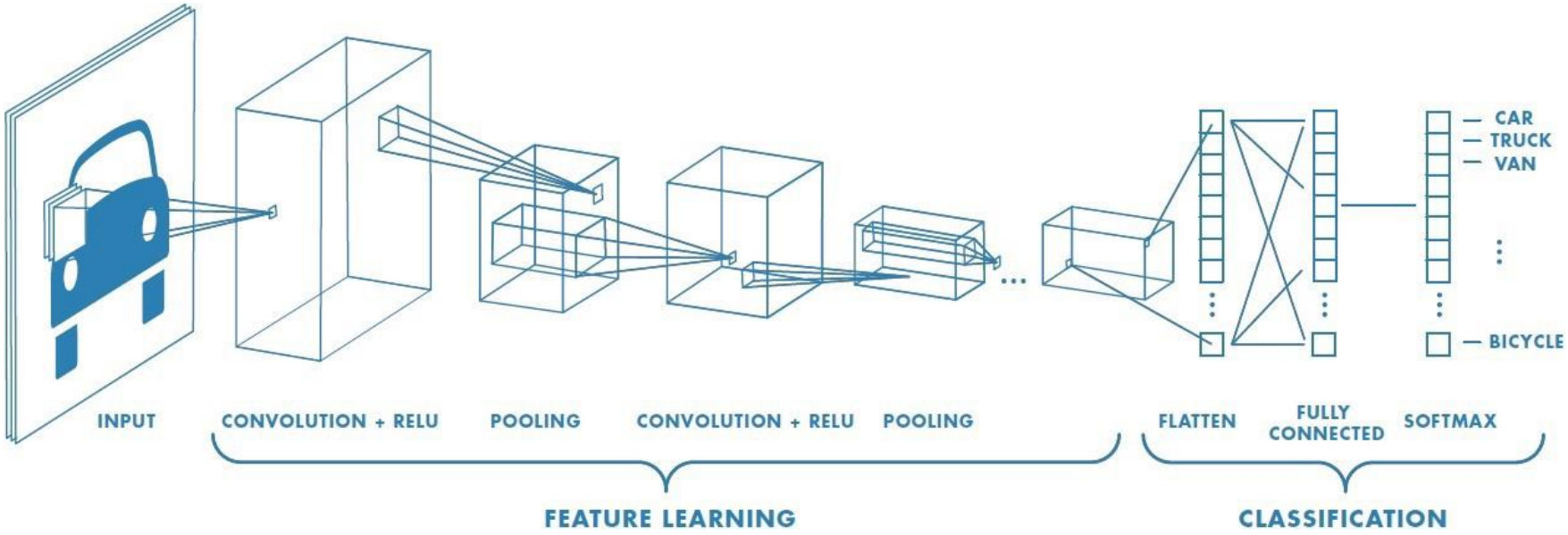
Computer Vision

- Applications for ML in CV
  - classification of images
  - detection
  - segmentation
  - caption generation
  - synthesis
  - inpainting
  - style transfer
  - super-resolution
  - depth prediction
  - scene reconstruction

Image data

- image
  - rectangular array of pixels
  - each pixel has either a grey-scale intensity or more commonly a triplet of red, green, and blue channels each with its own intensity value
  - e.g., 8-bit numbers represented as integers in the range 0, . . ., 255
- video
  - three dimensional structures in which successive frames are stacked through time

CNN


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

Feature detectors

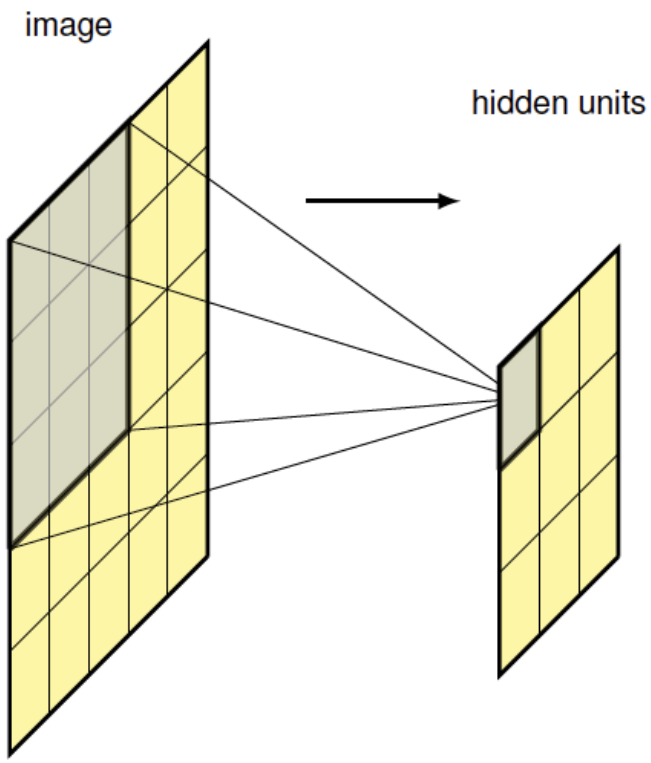
- Receptive field
  - captures the notion of locality

$$z = \mathrm{ReLU}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

  - the weights form a small two-dimensional grid known as a filter (also called kernel)
  - the ReLU generates a non-zero output only when $w^T x$ exceeds a threshold of $-w_0$
  - the unit acts as a feature detector that signals when it finds a sufficiently good match to its kernel

Feature detectors

Figure 10.1 (a) Illustration of a receptive field, showing a unit in a hidden layer of a network that receives input from pixels in a 3×3 patch of the image. Pixels in this patch form the receptive field for this unit. (b) The weight values associated with this hidden unit can be visualized as a small 3×3 matrix, known as a kernel. There is also an additional bias parameter that is not shown here.



(a)

0.4	1.7	0.9
2.3	−2.1	4.0
−1.4	0.7	2.1

(b)

Convolution

- Convolution operation

$$s(t) = \int x(a)w(t-a)da \qquad s(t) = (x * w)(t)$$

- Discrete convolution

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

Convolution

- 2D convolution operation

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

- Commutative

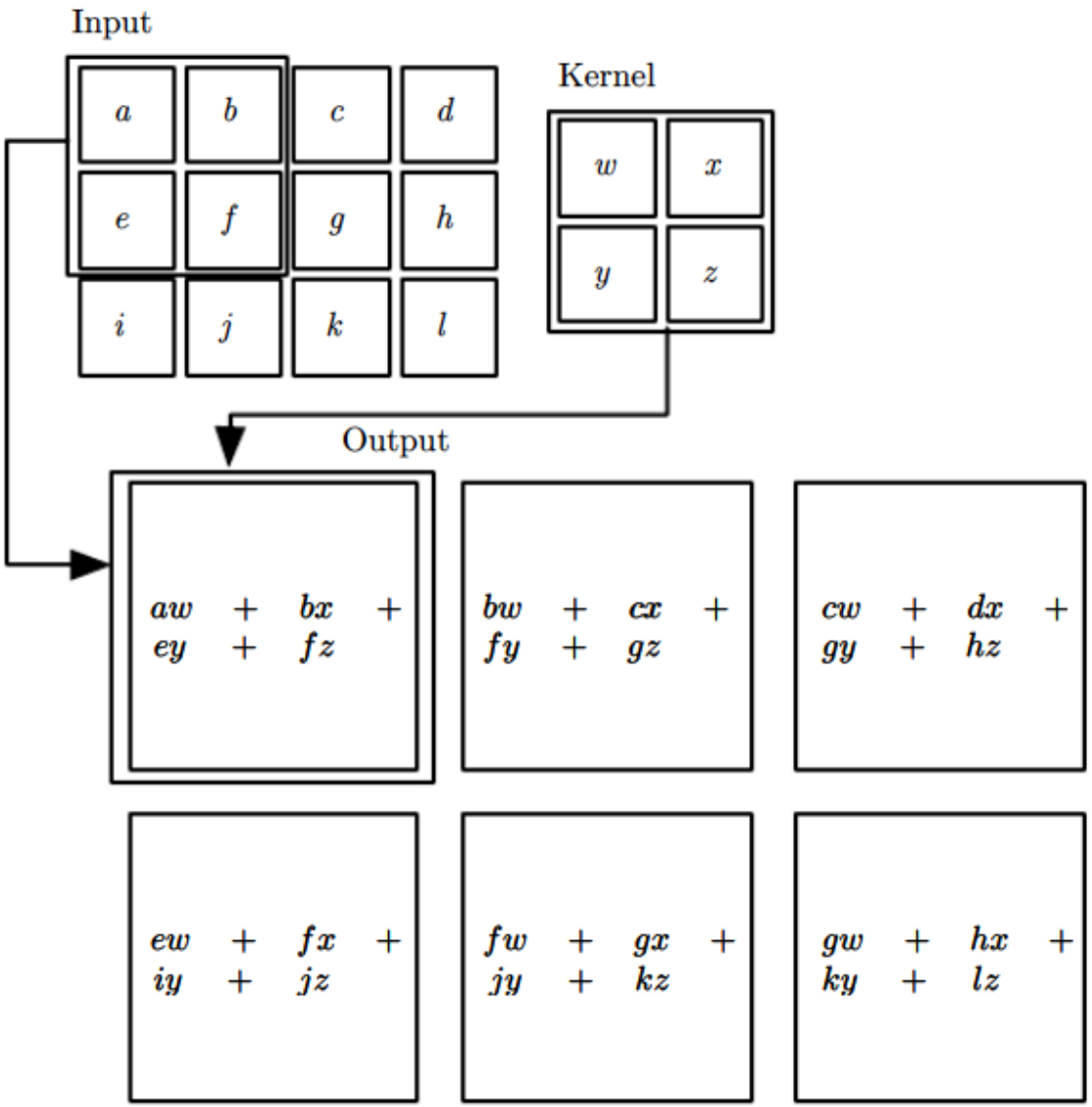
flipping the kernel

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

- Cross-correlation

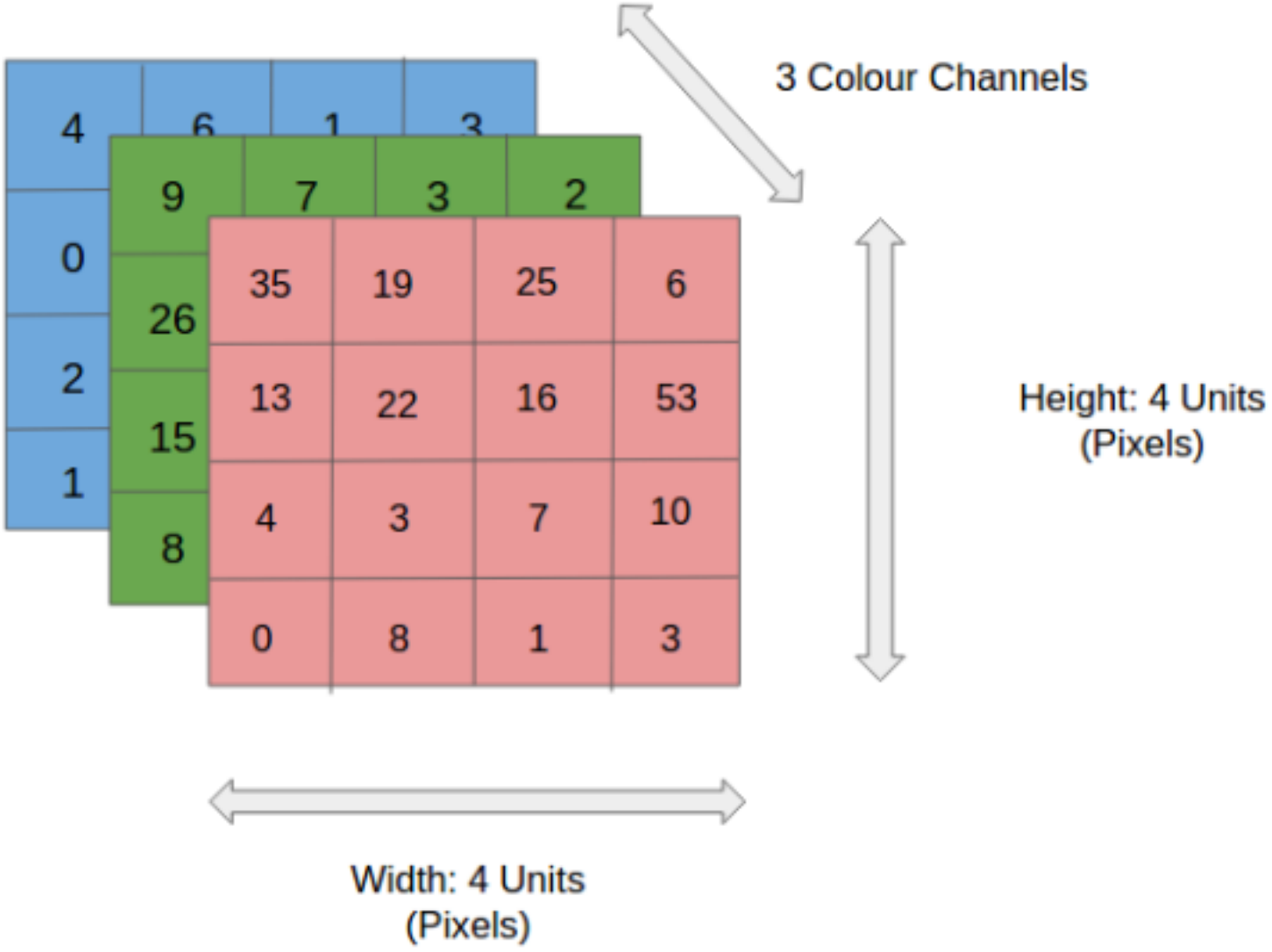
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

Convolution



2D convolution example

Images


4
6
1
3
9
7
3
2
0
26
35
19
25
6
2
15
13
22
16
53
1
8
4
3
7
10
0
8
1
3
3 Colour Channels
Height: 4 Units
(Pixels)
Width: 4 Units
(Pixels)

Images

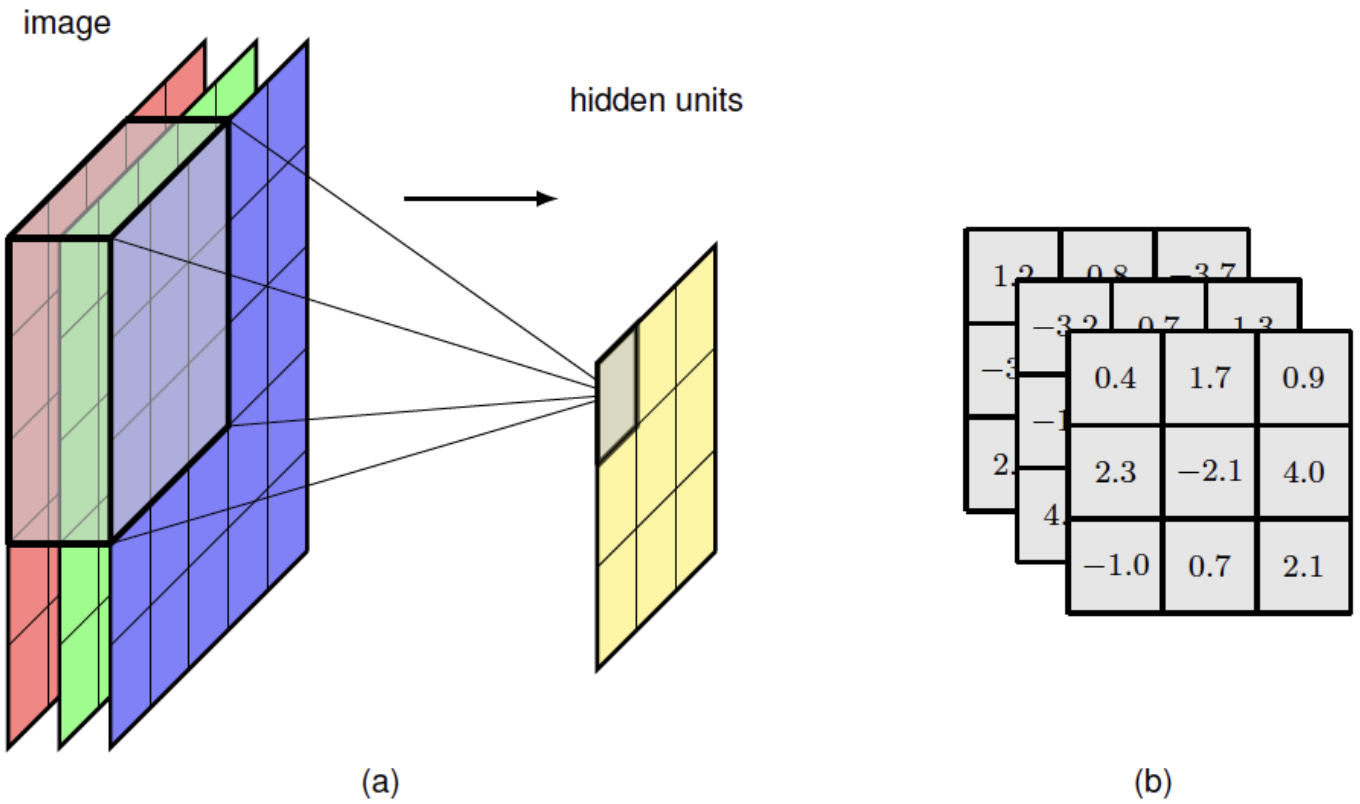
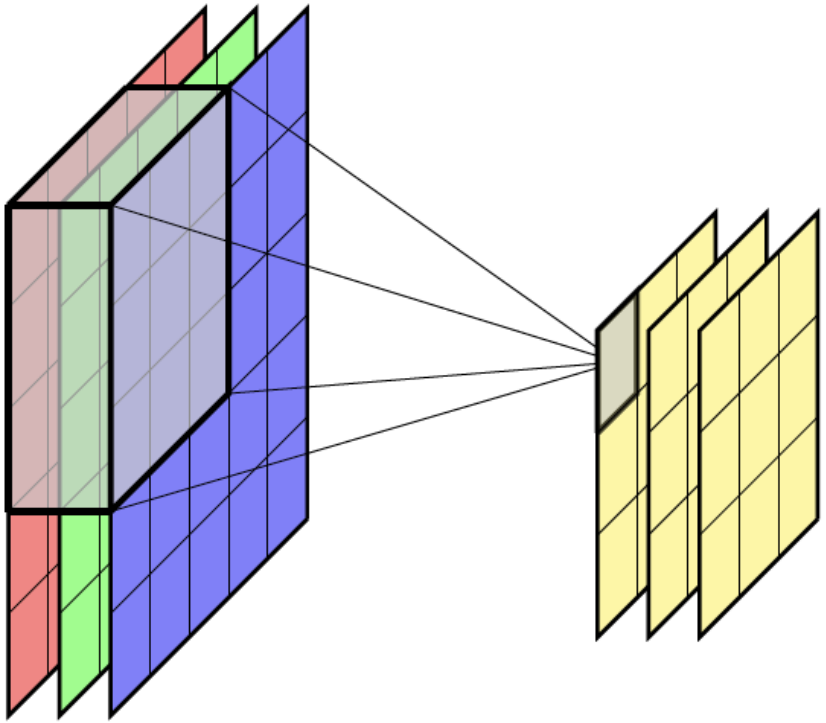


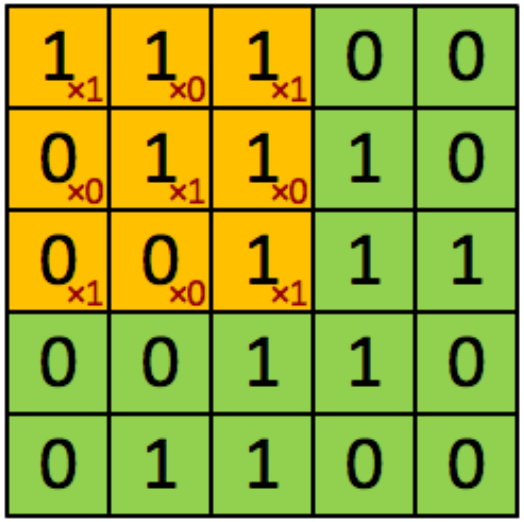
Figure 10.6 (a) Illustration of a multi-dimensional filter that takes input from across the R, G, and B channels. (b) The kernel here has 27 weights (plus a bias parameter not shown) and can be visualized as a $3 \times 3 \times 3$ tensor.

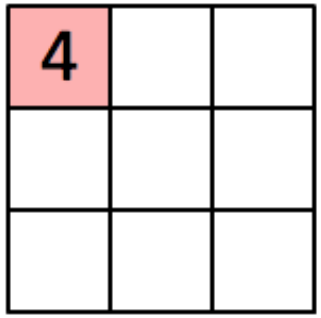
Images

Figure 10.7 The multi-dimensional convolutional filter layer shown in Figure 10.6 can be extended to include multiple independent filter channels.



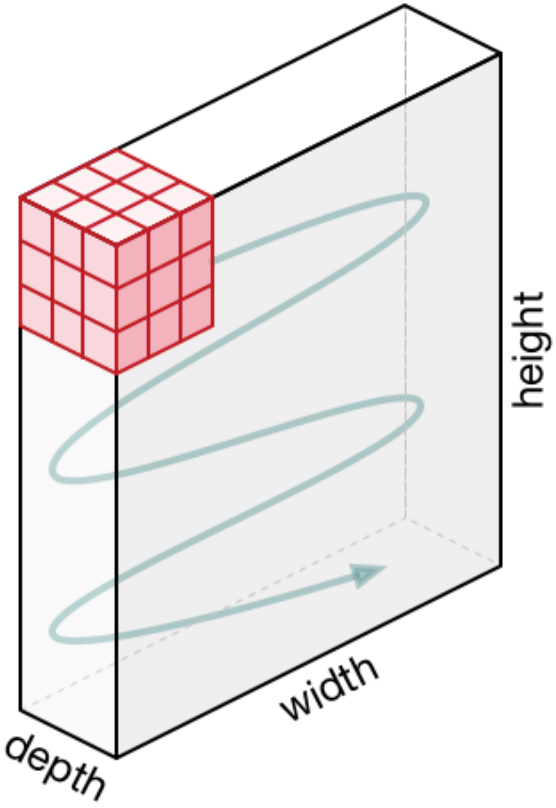
Convolution





Convolution example

Movement of the kernel



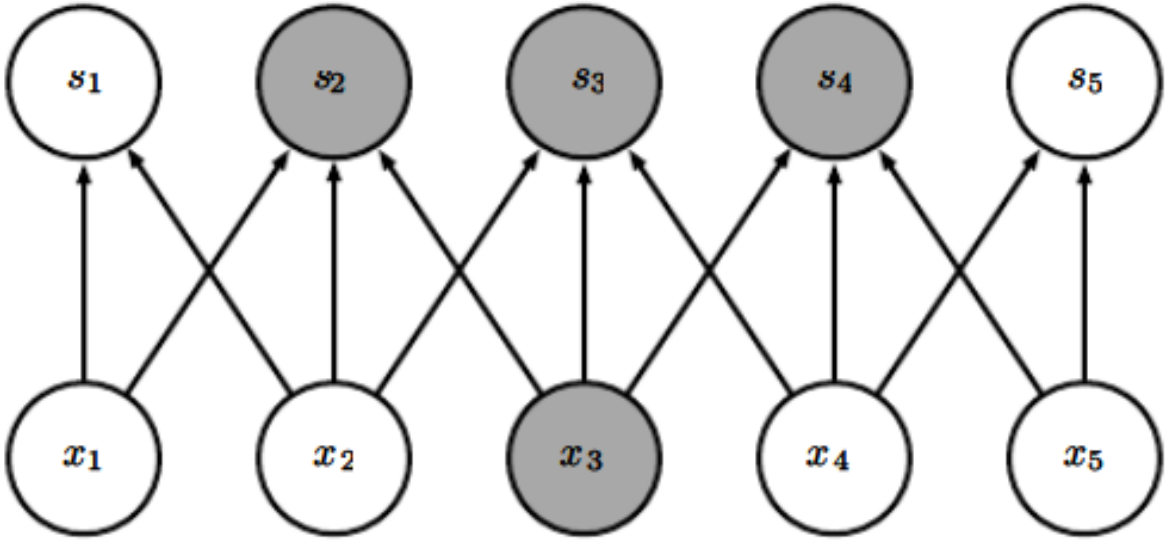
Operations

- 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

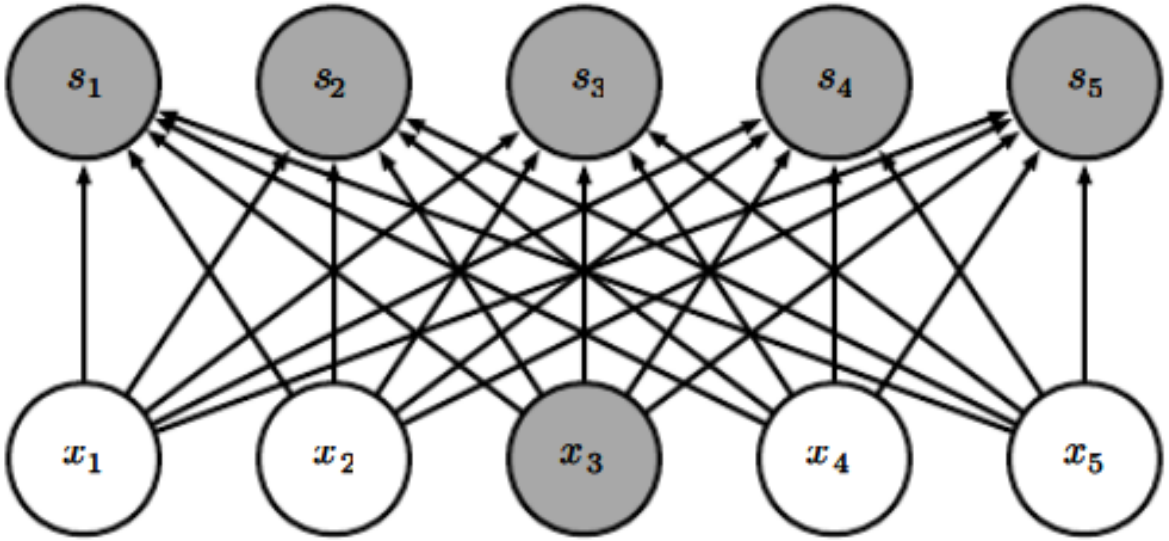
Motivation

- Three ideas
  - Sparse interactions
  - Parameter sharing
  - Equivariant representations

Sparse interactions

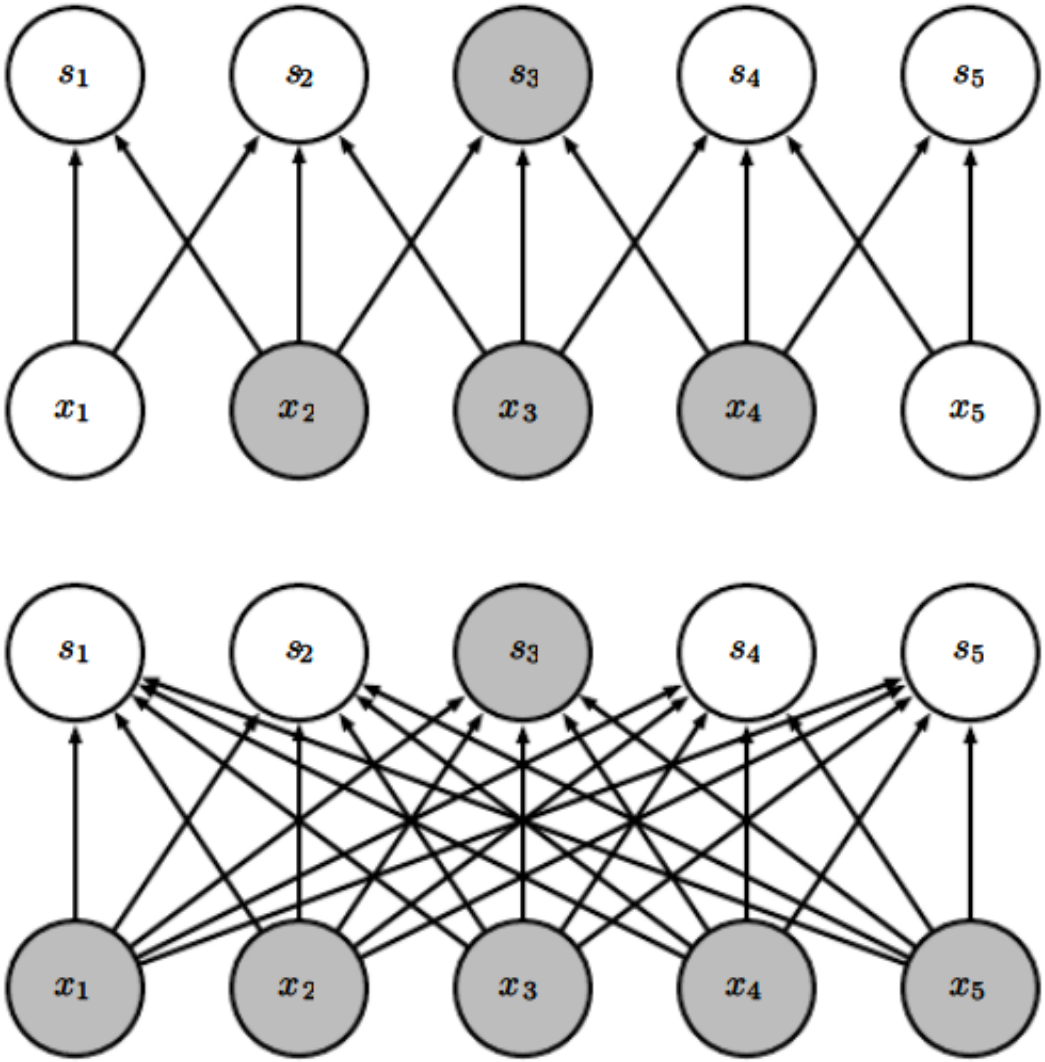


sparse
(kernel of width 3)



Dense connections
no longer sparse

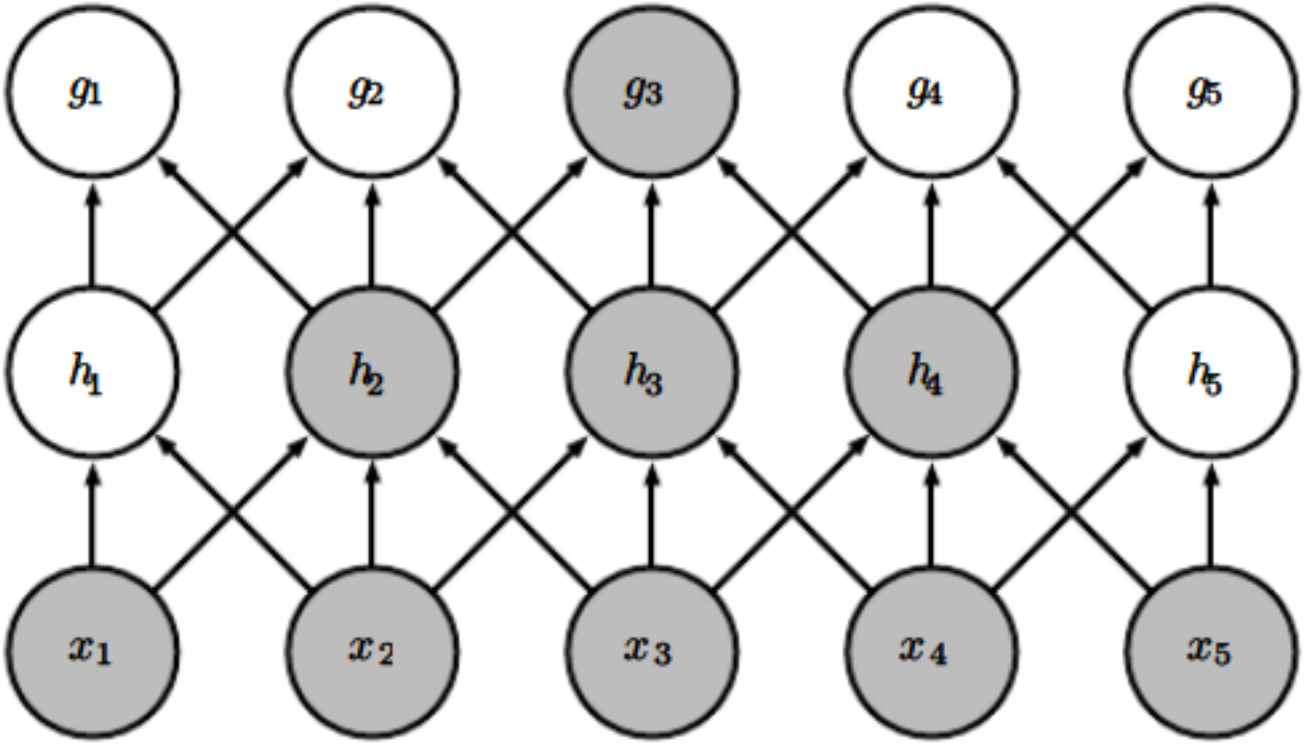
Sparse interactions



sparse
(receptive field)

Dense
connections
no longer sparse

Sparse interactions

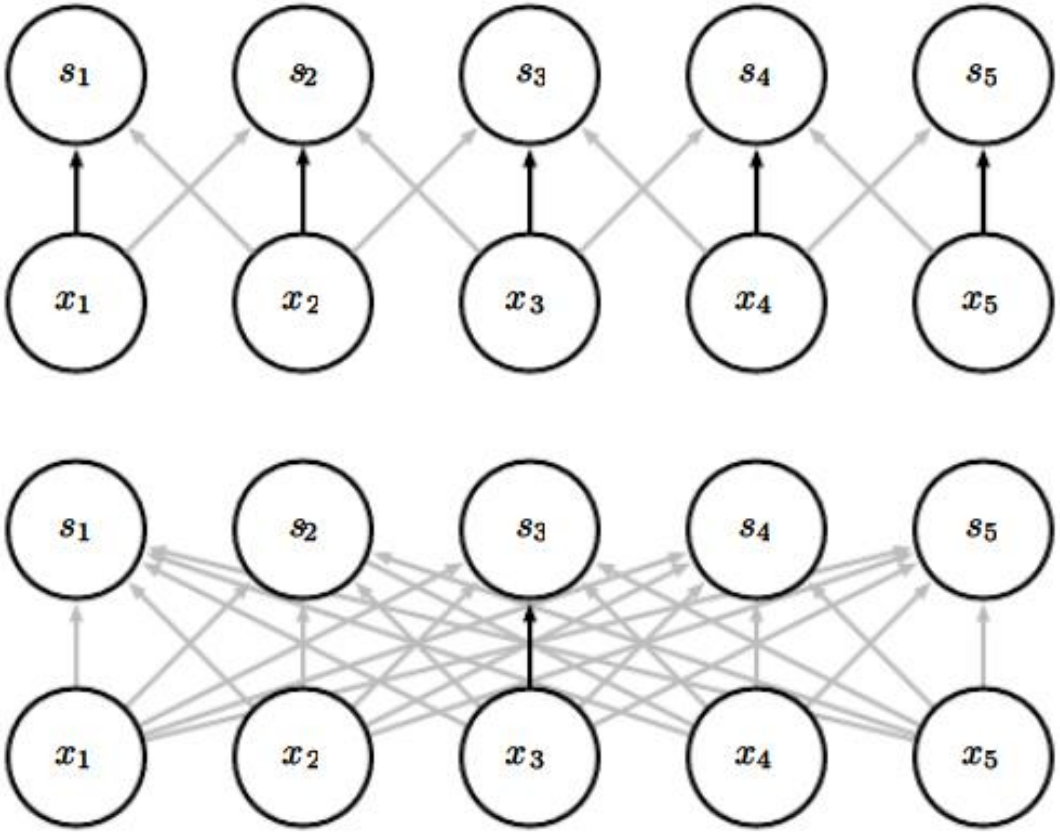


even though direct connections in a convolutional net are very sparse, units in the deeper layers can be indirectly connected to all or most of the input image

Parameter sharing

- Goal
  - same parmeter for more that one function in a model
  - Full connected
    - Tied weights – each element of the weight matrix is used exactly once when computing the oputput of a layer
  - CNN
    - each member of the kernel is used at at every position of the input
    - Parameter sharing means that rather than learning a separate set of parameters for every location, we learn only one set
    - Convolution is dramatically more efficient than dense matrix multiplication in terms of memory requirements and statistical efficiency

Parameter sharing



The black arrows indicate uses of the central element of a 3-element kernel in a convolutional model. Due to parameter sharing, this single parameter is used at all input locations.

Equivariance to translation

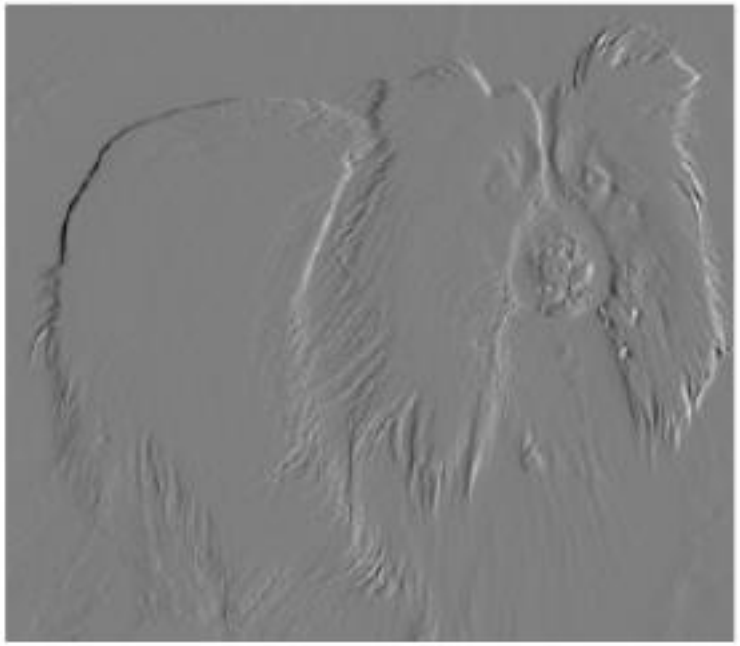
- Goal
  - layers have a property called equivariance to translation
    - if the input changes, the output changes in the same way
    - f(x) is equivariant to a function g if f(g(x)) = g(f(x)) if we let g be any function that translates the input
      - i.e., shifts it, then the convolution function is equivariant to g
    - I - image brightness
    - g - function mapping one image function to another such that I' = g(I), I'(x,y) = I(x -1, y) (shift every pixel of I one unit to right)

Equivariance to translation

- Goal
  - If we apply this transformation to I then apply convolution the result will be the same as if we pplied convolution to I' then applied the transformation g to the output
  - If we move the object in the input its representation will move the same amount in the output
  - Convolution is not naturally equivariant to some other transformations
    - Changes in the scale or rotation of an image

Equivariance to translation

Input



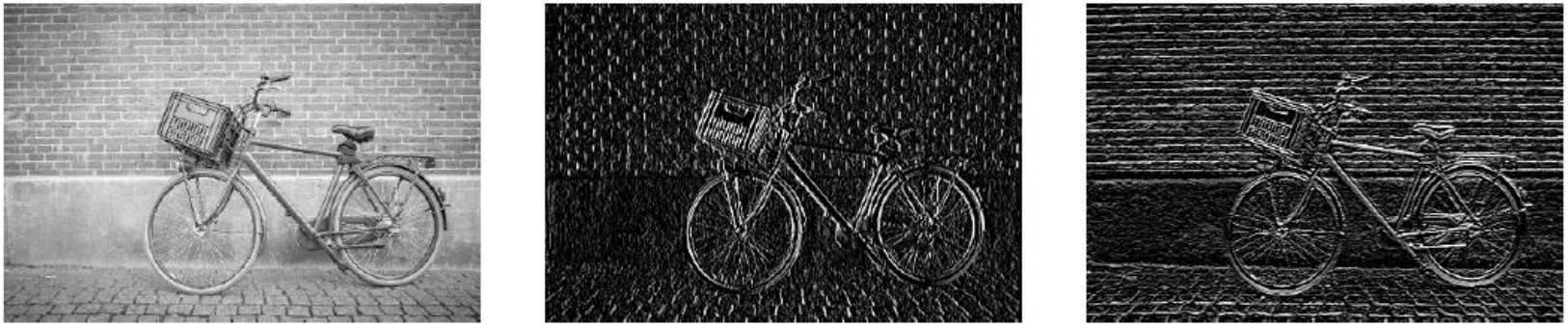
Output

1	-1

Kernel

Efficiency of edge detection

Equivariance to translation



(a) (b) (c)

Figure 10.4 Illustration of edge detection using convolutional filters showing (a) the original image, (b) the result of convolving with the filter (10.3) that detects vertical edges, and (c) the result of convolving with the filter (10.4) that detects horizontal edges.

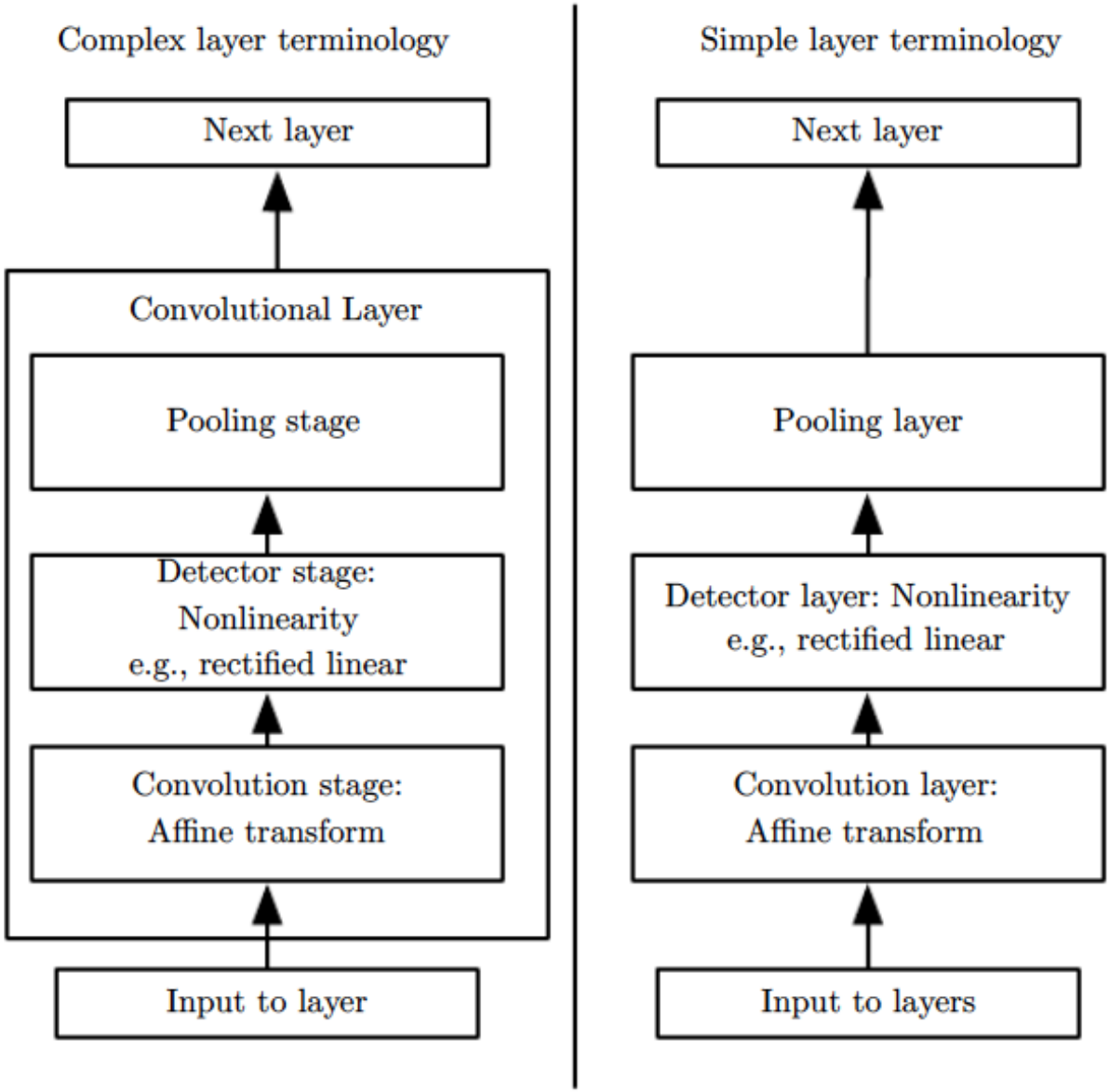
−1	0	1
−1	0	1
−1	0	1

10.3 filter

−1	−1	−1
0	0	0
1	1	1

10.4 filter

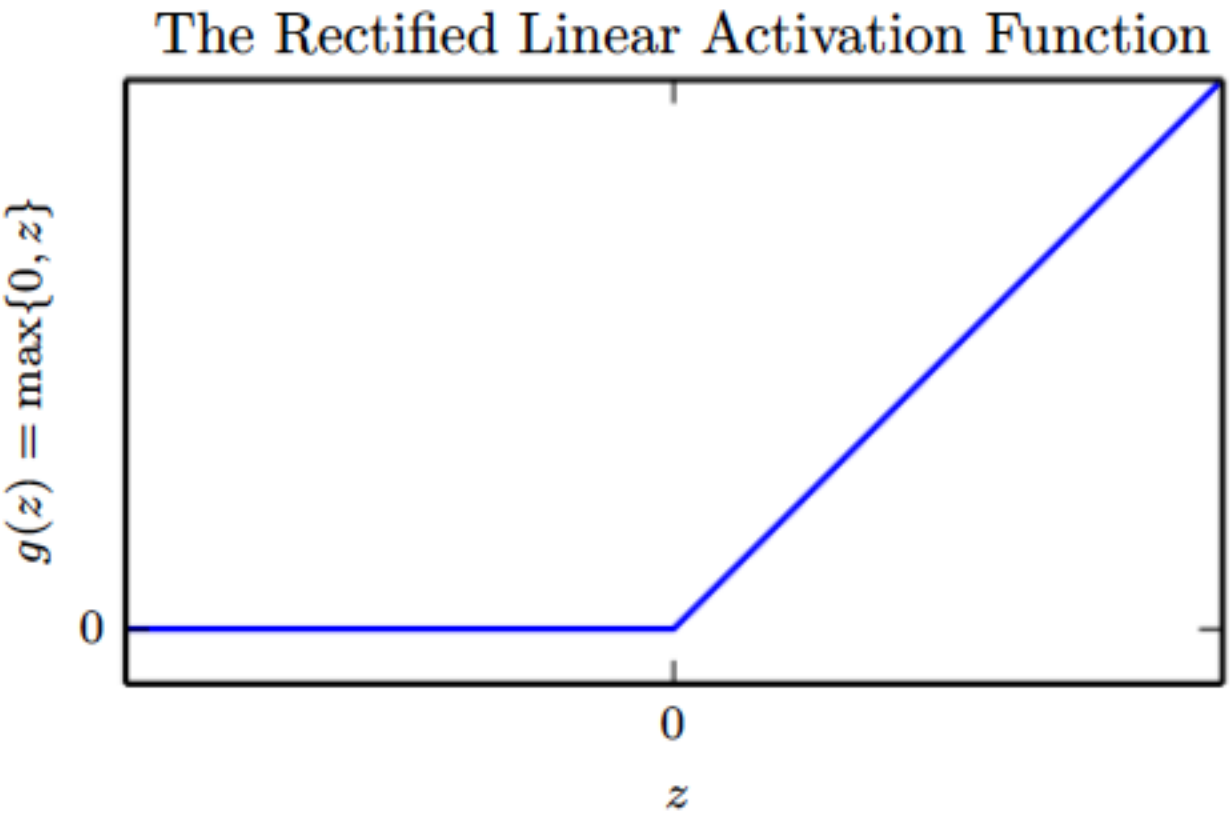
Stages


Complex layer terminology
Next layer
Convolutional Layer
Pooling stage
Detector stage:
Nonlinearity
e.g., rectified linear
Convolution stage:
Affine transform
Input to layer
Simple layer terminology
Next layer
Pooling layer
Detector layer: Nonlinearity
e.g., rectified linear
Convolution layer:
Affine transform
Input to layers

Rectified Linear Units

- ReLU activation function

$$g(z) = \max\{0, z\}$$



Rectified Linear Units

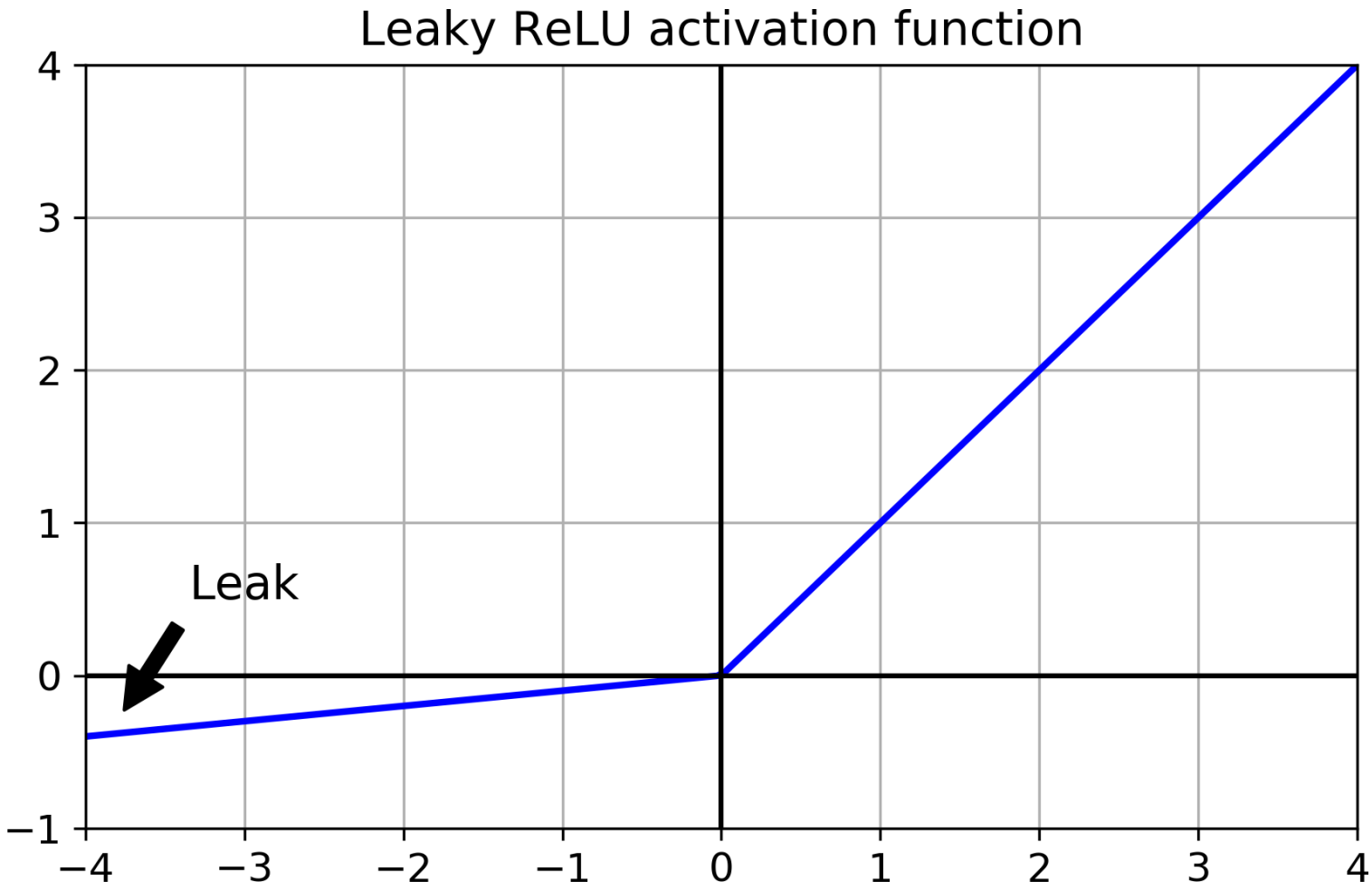
- ReLU generalizations
  - Slope

$$h_i = g(\boldsymbol{z}, \boldsymbol{\alpha})_i = \max(0, z_i) + \alpha_i \min(0, z_i)$$

  - Absolute value rectification

$$\alpha_i = -1 \qquad g(z) = |z|$$

Leaky ReLU



$\max(\alpha z, z)$

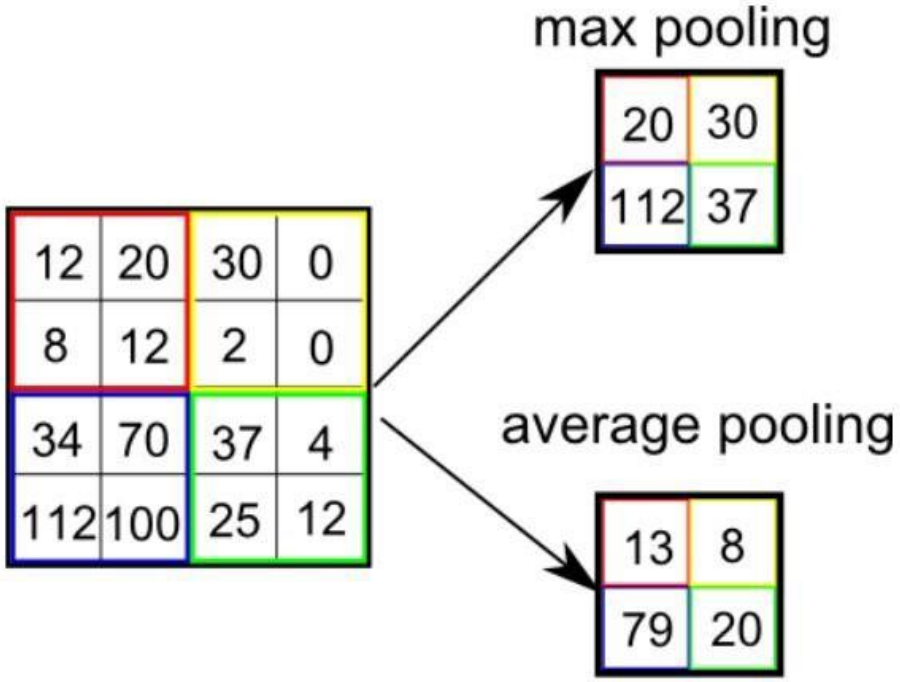
Pooling

- Pooling function
  - replaces the output of the net at a certain location with a summary statistic of the nearby outputs
  - helps to make the representation become approximately invariant to small translations of the input
  - Max pooling
    - maximum output within a rectangular neighborhood
  - Average of a rectangular neighborhood
  - L^2 norm of a rectangular neighborhood

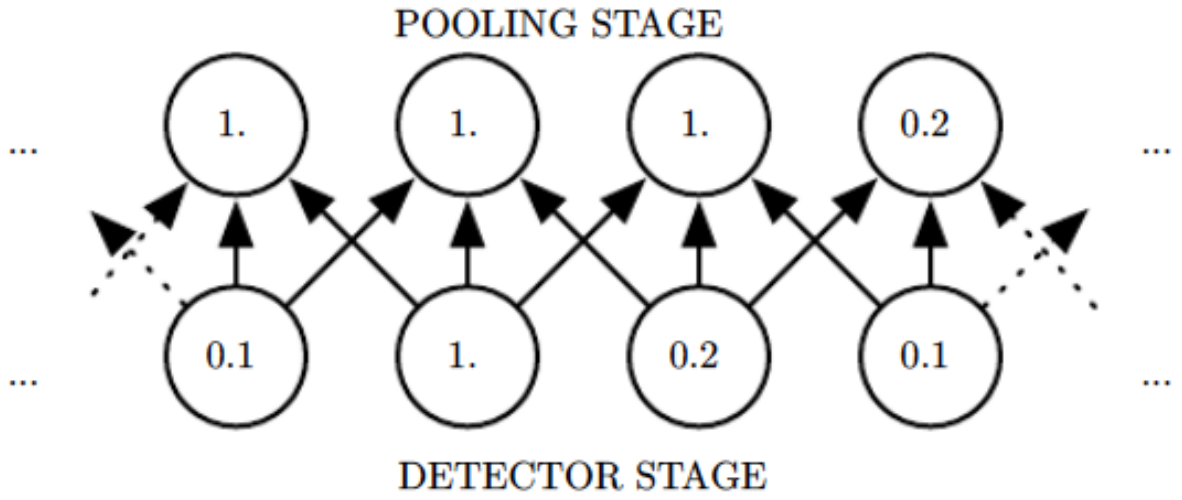
Pooling

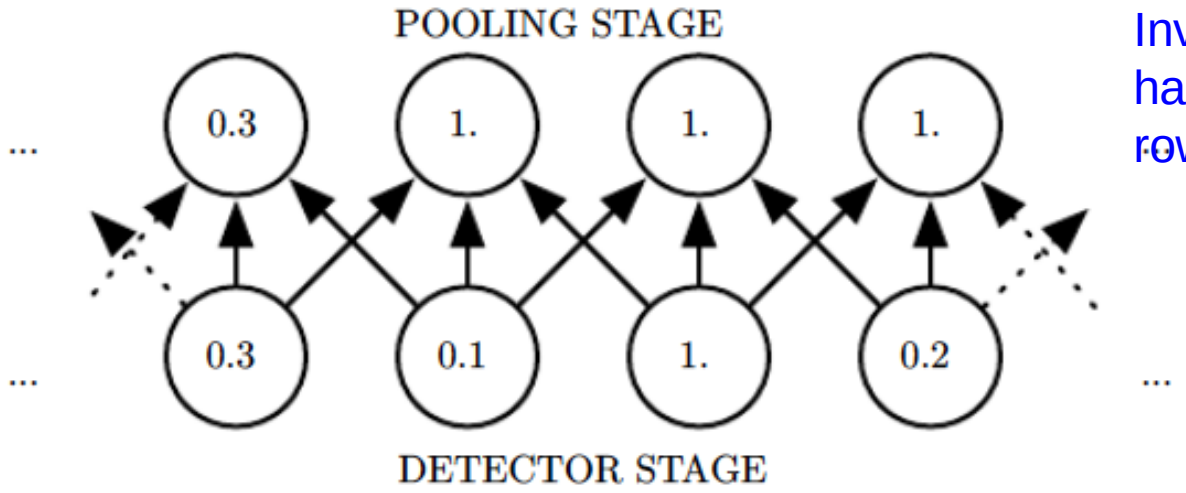
3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



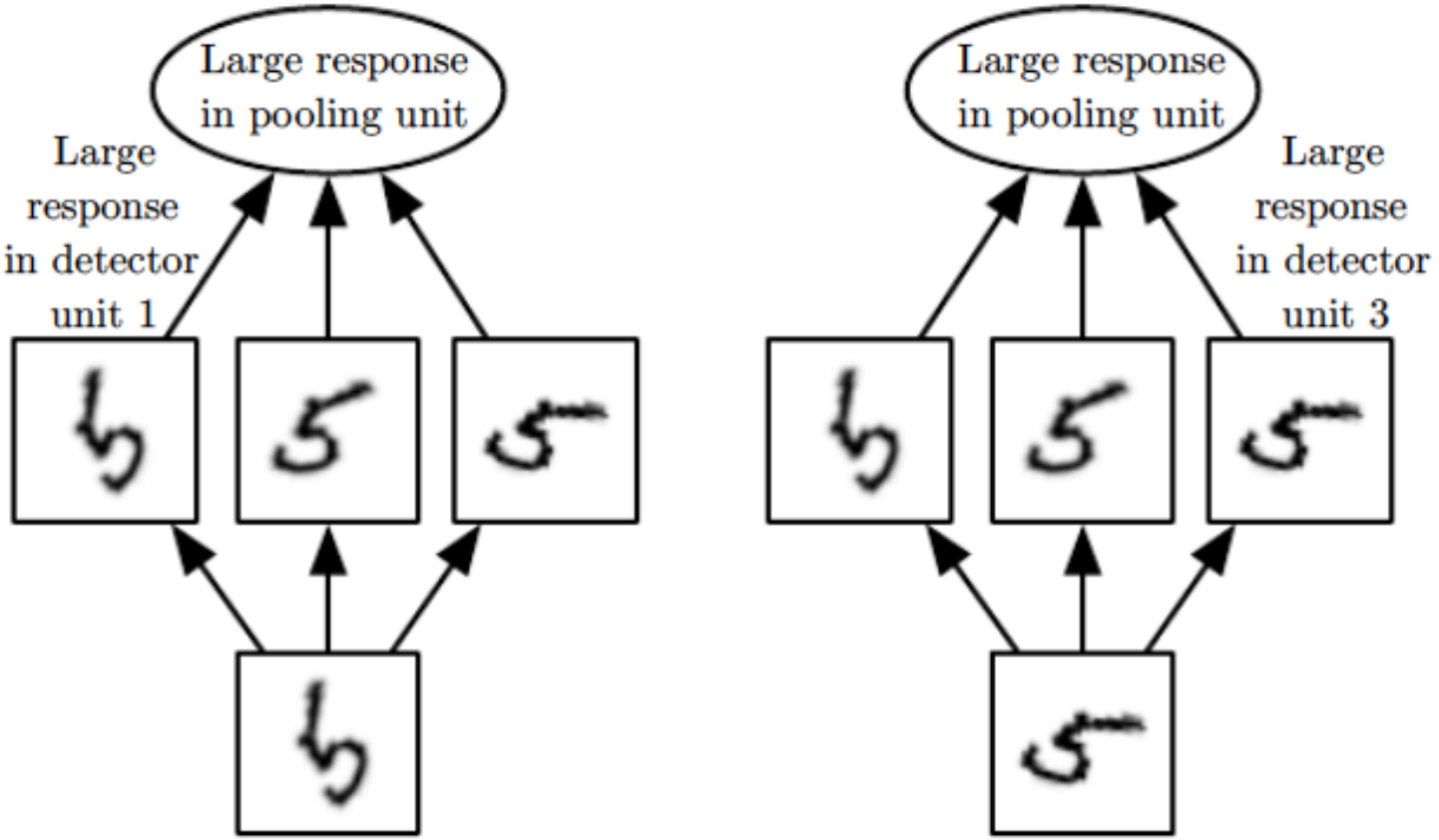
Pooling





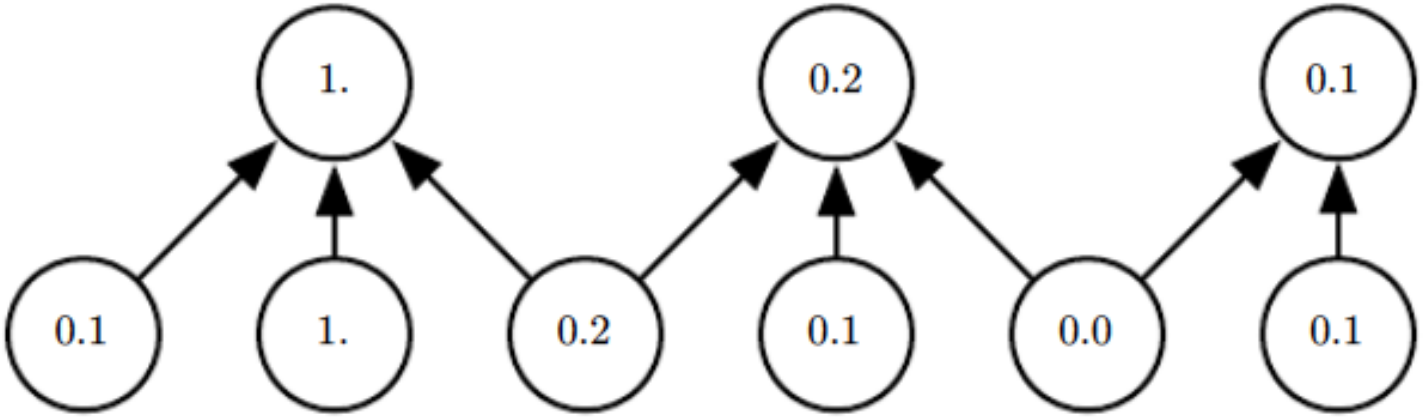
Invariance –
half of the values in the top row have changed

Pooling



Invariance -invariant to transformations of the input

Pooling



Pooling with downsampling. Max-pooling with a pool width of three and a stride between pools of two. It reduces the representation size by a factor of two, which reduces the computational and statistical burden on the next layer

CNN

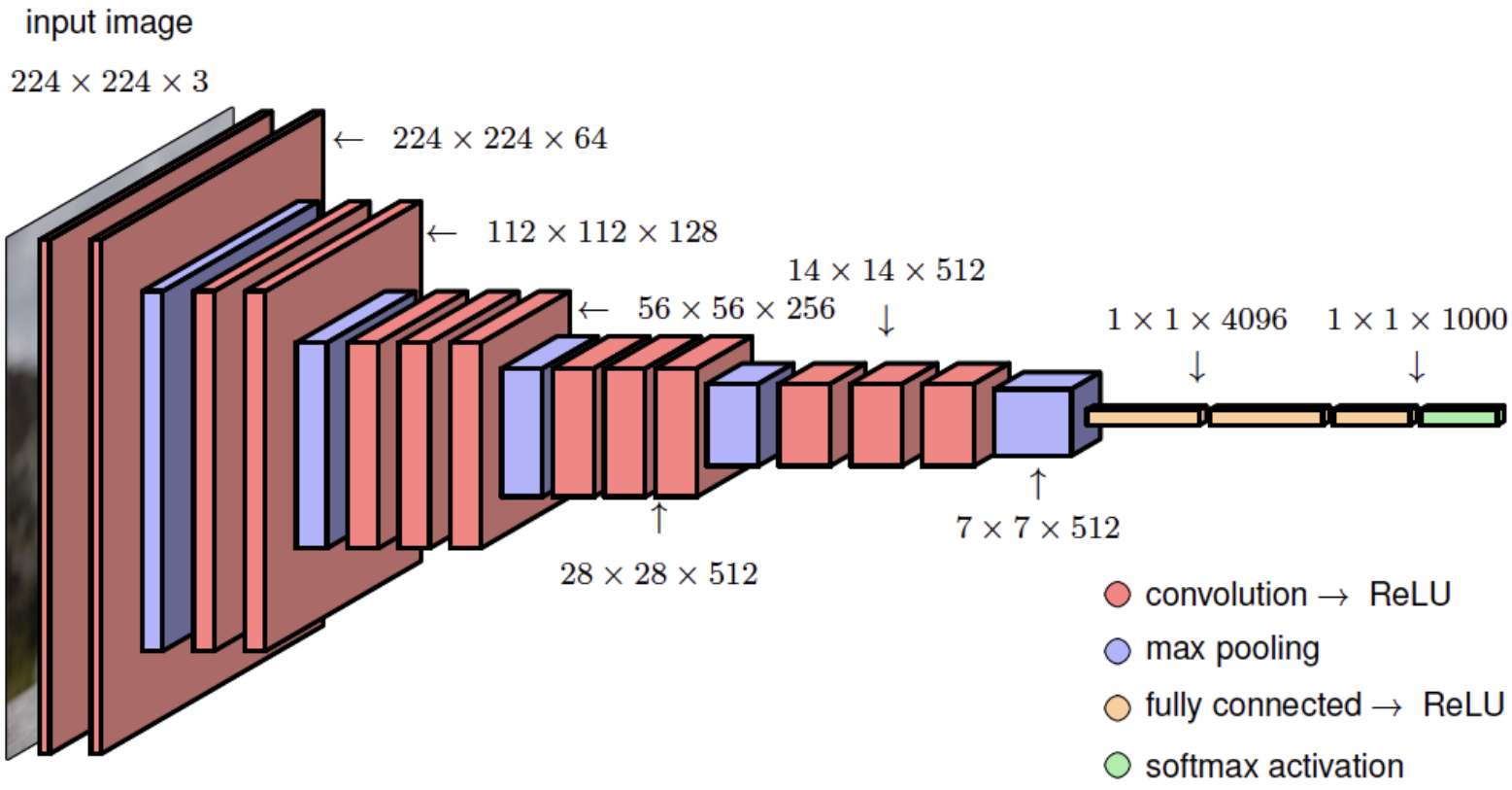


Figure 10.10 The architecture of a typical convolutional network, in this case a model called VGG-16.

Variants of the CNN

- Input observed data

$\mathbf{V}$ $V_{i,j,k}$

input unit within channel i at row j and column k

- Convolution

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

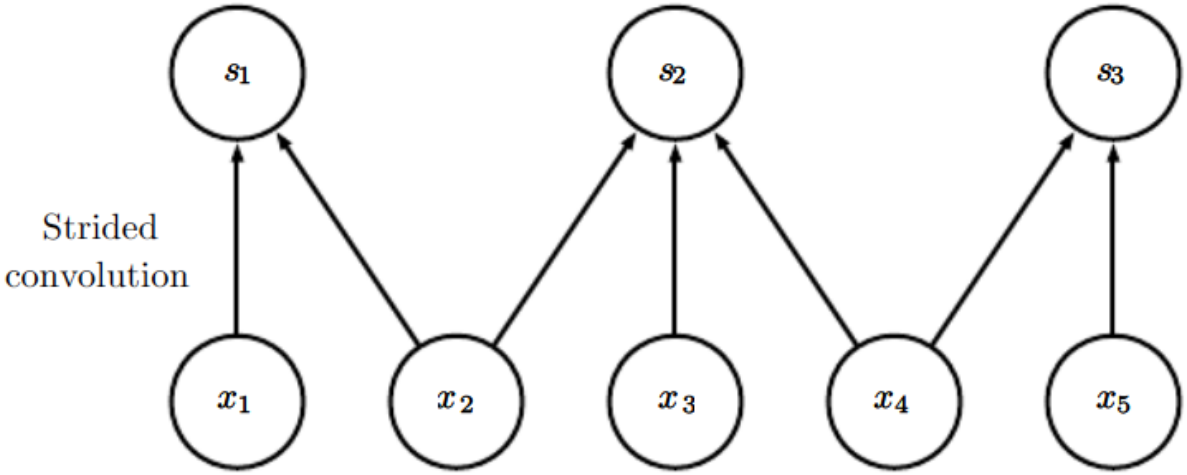
Variants of the CNN

- downsampled convolution function c such that

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

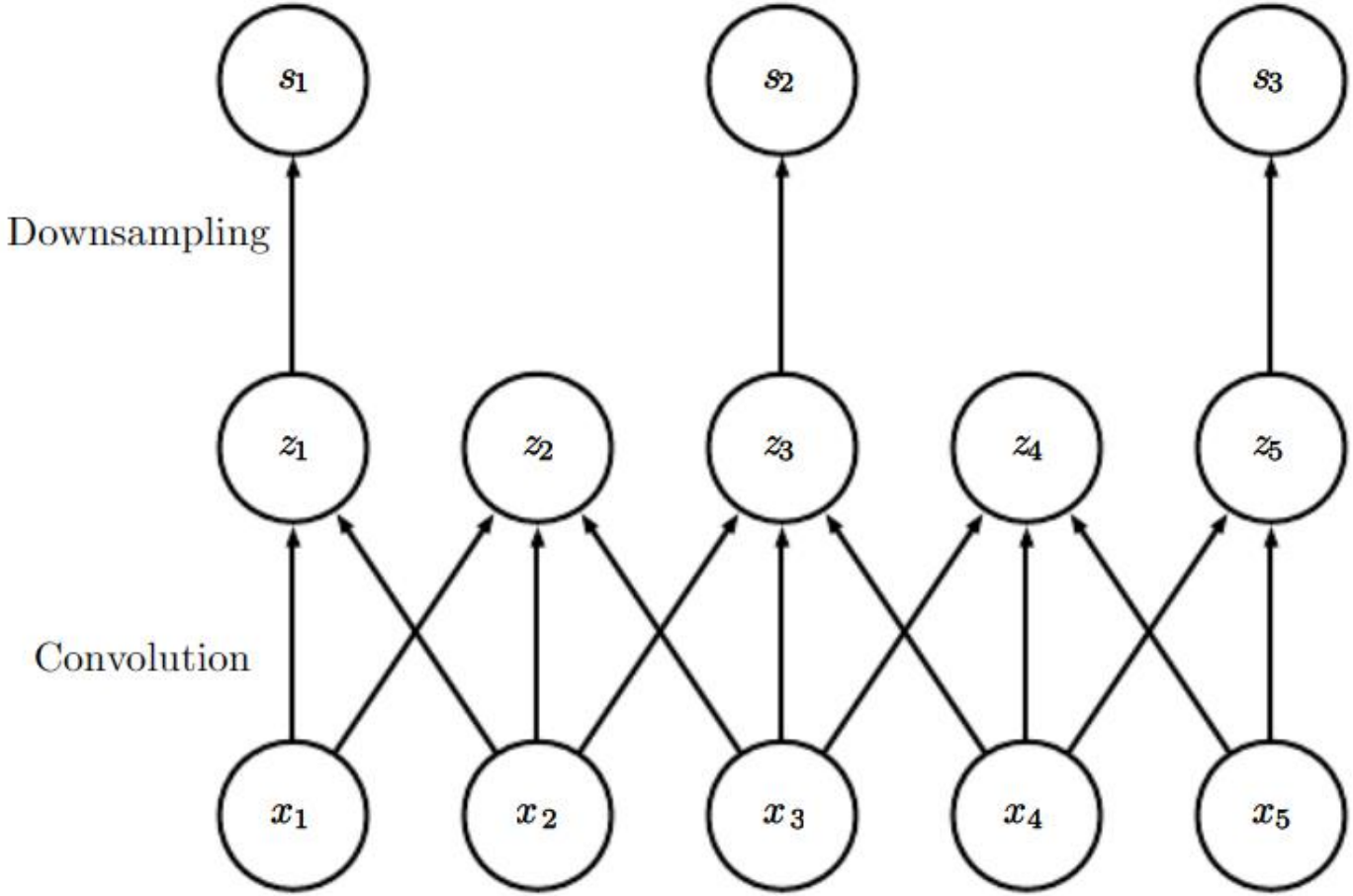
- s is the stride of the downsampled convolution
  - It is possible to define a separate stride for each direction of motion

Variants of the CNN



Stride of two. Convolution with a stride length of two implemented in a single operation.

Variants of the CNN



Convolution with a stride greater than one pixel is mathematically equivalent to convolution with unit stride followed by downsampling.

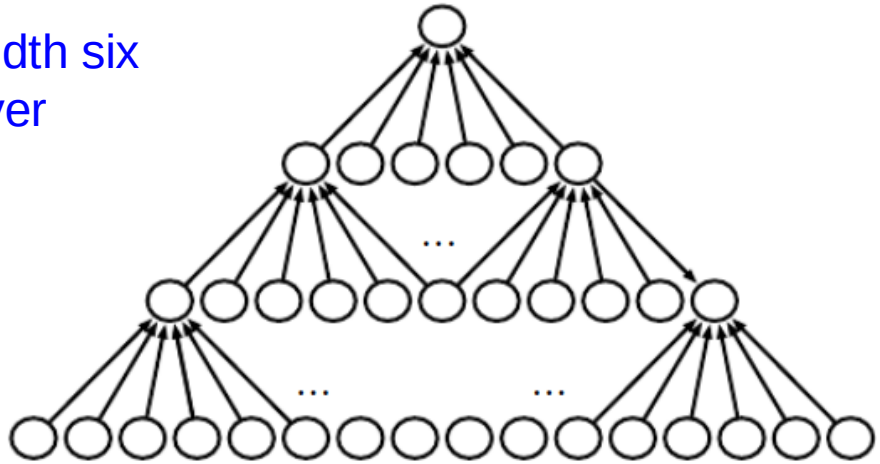
Variants of the CNN

- Essential feature zero-pad V
  - to make it wider

- Without zero-padding
  - the width of the representation shrinks by one pixel less than the kernel width at each layer
  - shrinking the spatial extent of the network rapidly or using small kernels

Variants of the CNN

kernel of width six
at every layer



representation
shrinks by five pixels
at each layer

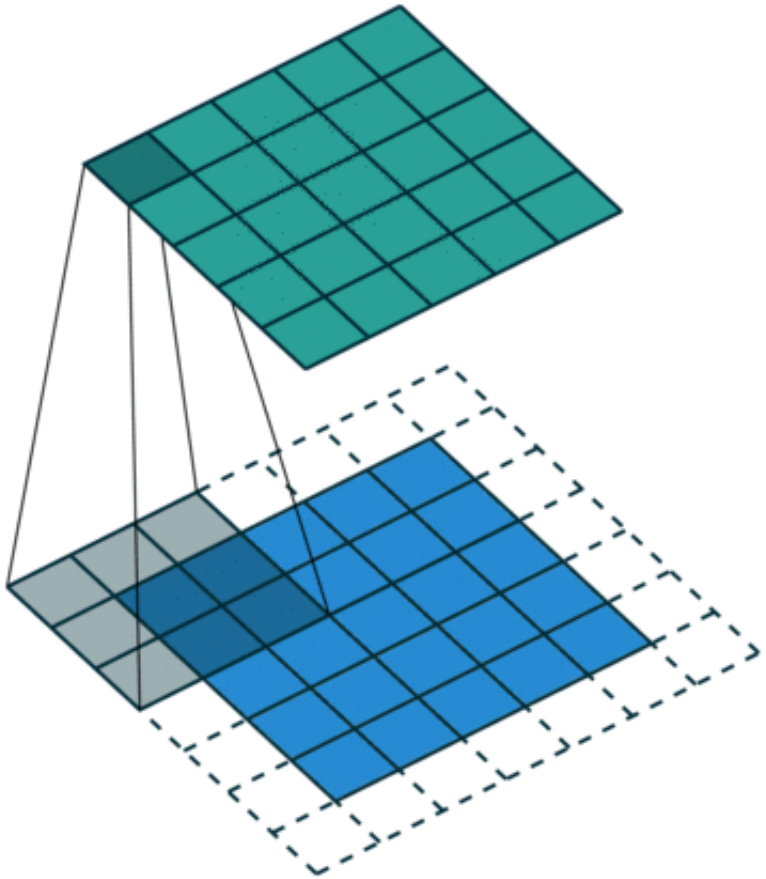
Sixteen pixels

The effect of zero
padding on network
size

Adding five implicit zeroes

Variants of the CNN

0	0	0	0	0	0
0	X_{11}	X_{12}	X_{13}	X_{14}	0
0	X_{21}	X_{22}	X_{23}	X_{24}	0
0	X_{31}	X_{32}	X_{33}	X_{34}	0
0	X_{41}	X_{42}	X_{43}	X_{44}	0
0	0	0	0	0	0



Same convolution

Zero-padding

- Three special cases
  - valid convolution
    - no zero-padding
    - all pixels in the output are a function of the same number of pixels in the input
    - the output shrinks at each layer
  - same convolution
    - zero-padding is added to keep the size of the output equal to the size of the input
  - the optimal amount of zero padding (in terms of test set classification accuracy) lies somewhere between "valid" and "same" convolution

Unshared convolution

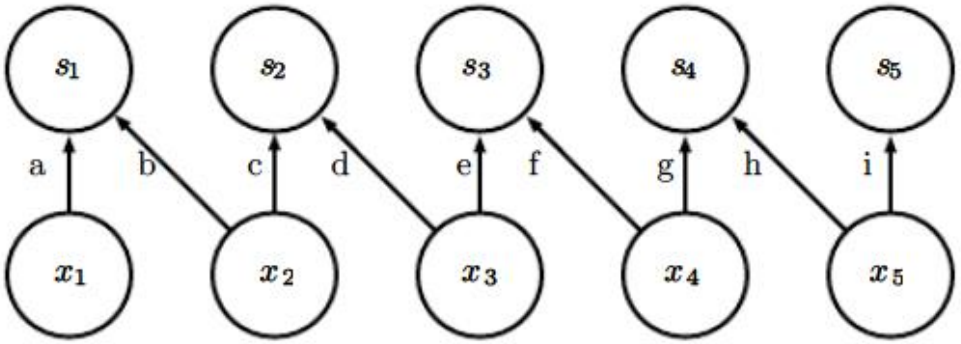
- adjacency matrix (no convolution) in the graph of our MLP is the same
  - Weights W by a 6-D tensor

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

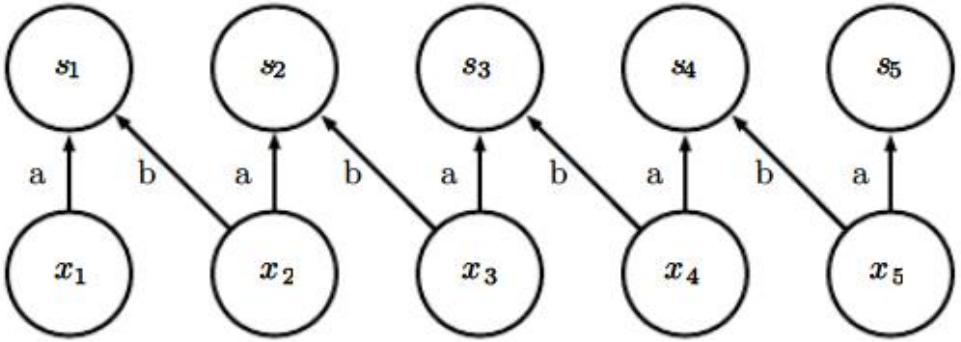
i, the output channel, j, the output row, k, the output column, l, the input channel, m, the row offset within the input, and n, the column offset within the input

Unshared convolution

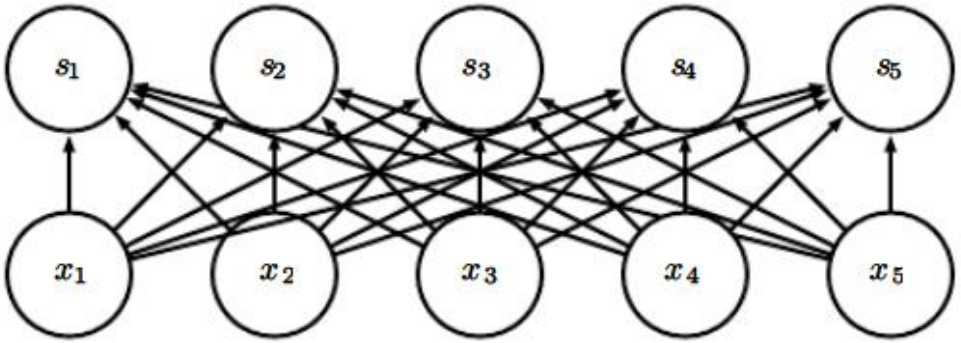
patch size of two pixels



Local connections

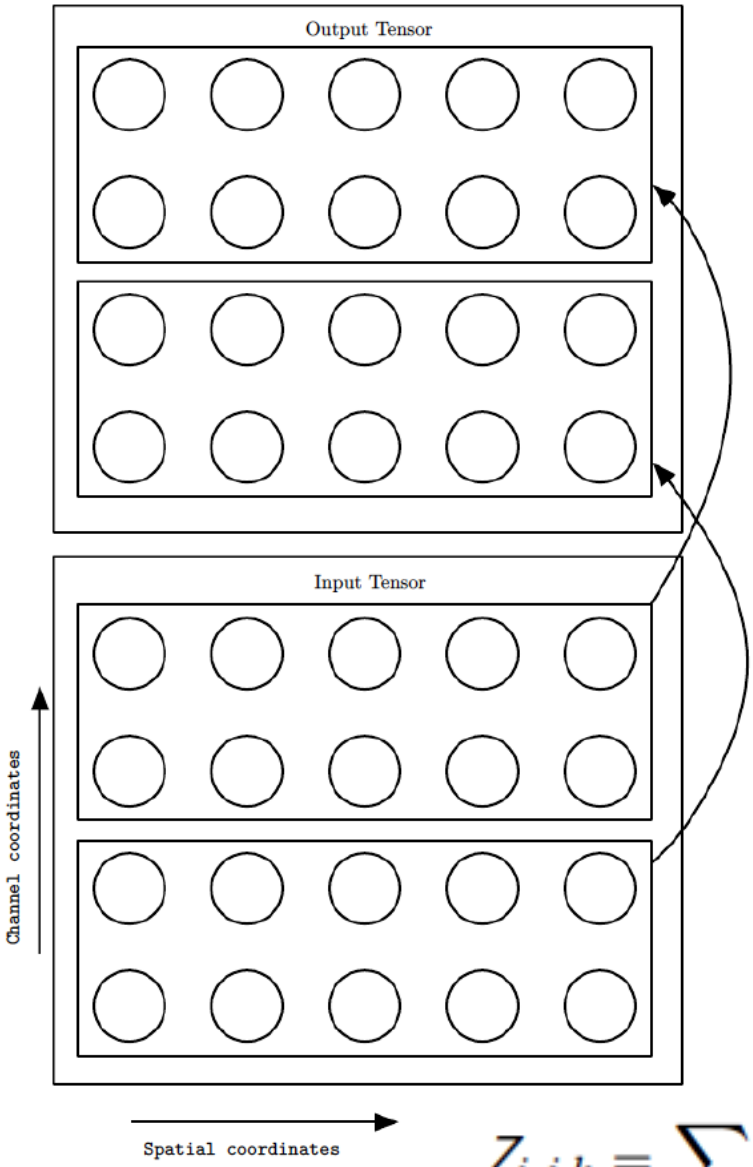


Convolution



Full connections

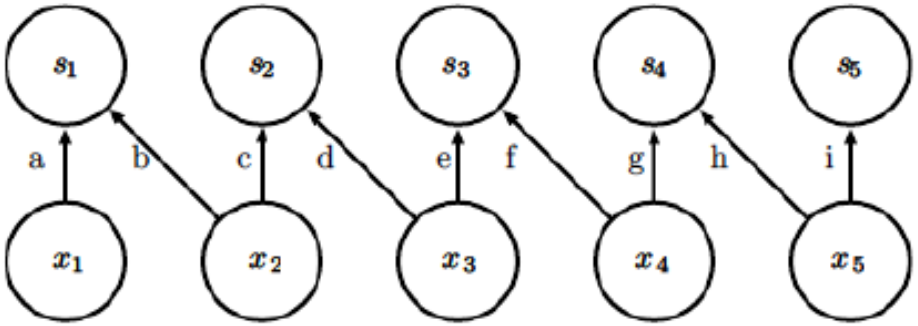
Tiled convolution



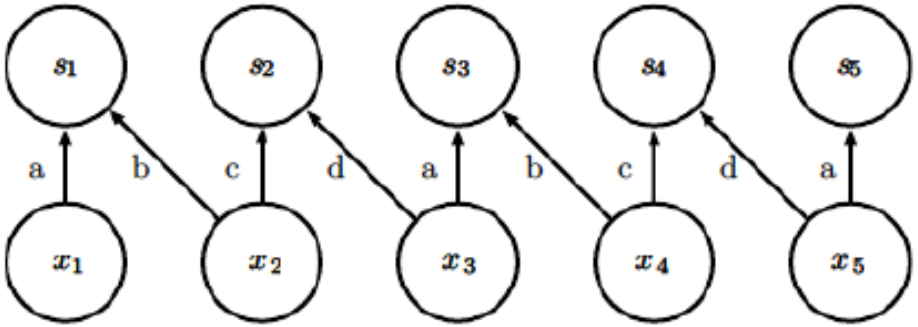
t different choices of kernel stack in each direction

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

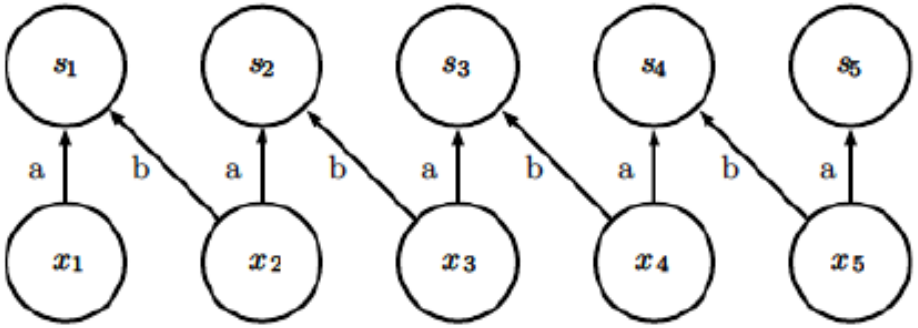
Tiled convolution



Locally connected layer
(no sharing)



Tiled convolution has a set of
t (=2) different kernels



Traditional convolution (tiled
Convolution with t = 1)

Learning

- Three operations for training any depth of feedforward convolutional network
  - convolution
  - backprop from output to weights
  - backprop from output to inputs

- We consider strided CNN

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

Learning

- Minimize the loss function

$$J(\mathbf{V}, \mathbf{K})$$

- Derivatives w.r.t. the weights in the kernel

$$g(\mathbf{G}, \mathbf{V}, s)_{i,j,k,l} = \frac{\partial}{\partial K_{i,j,k,l}} J(\mathbf{V}, \mathbf{K}) = \sum_{m,n} G_{i,m,n} V_{j,(m-1)\times s+k,(n-1)\times s+l}$$

$$G_{i,j,k} = \frac{\partial}{\partial Z_{i,j,k}} J(\mathbf{V}, \mathbf{K})$$

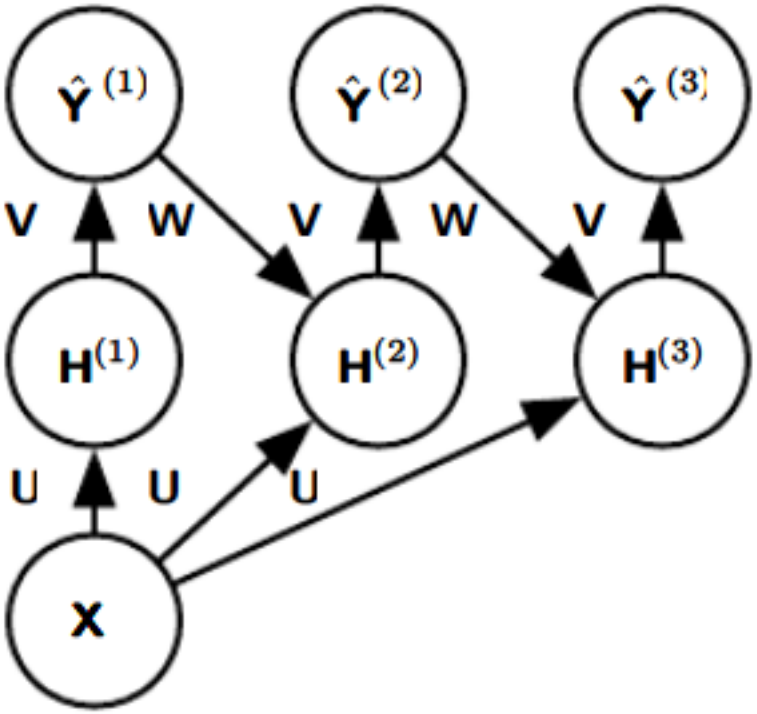
Learning

- If the layer is not the bottom layer

$$h(\mathbf{K}, \mathbf{G}, s)_{i,j,k} = \frac{\partial}{\partial V_{i,j,k}} J(\mathbf{V}, \mathbf{K})$$

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

Structured output



recurrent convolutional network for pixel labeling

pooling operator with unit stride

Data types

	Single channel	Multi-channel
1-D	Audio waveform: The axis we convolve over corresponds to time. We discretize time and measure the amplitude of the waveform once per time step.	Skeleton animation data: Animations of 3-D computer-rendered characters are generated by altering the pose of a "skeleton" over time. At each point in time, the pose of the character is described by a specification of the angles of each of the joints in the character's skeleton. Each channel in the data we feed to the convolutional model represents the angle about one axis of one joint.
2-D	Audio data that has been preprocessed with a Fourier transform: We can transform the audio waveform into a 2D tensor with different rows corresponding to different frequencies and different columns corresponding to different points in time. Using convolution in the time makes the model equivariant to shifts in time. Using convolution across the frequency axis makes the model equivariant to frequency, so that the same melody played in a different octave produces the same representation but at a different height in the network's output.	Color image data: One channel contains the red pixels, one the green pixels, and one the blue pixels. The convolution kernel moves over both the horizontal and vertical axes of the image, conferring translation equivariance in both directions.
3-D	Volumetric data: A common source of this kind of data is medical imaging technology, such as CT scans.	Color video data: One axis corresponds to time, one to the height of the video frame, and one to the width of the video frame.

Data types

Features

- reduce the cost of CNN training
  - use features that are not trained in a supervised fashion
    - simply initialize them randomly
    - design them by hand
    - one can learn the kernels with an unsupervised criterion (clustering)
    - random filters

Neuroscientific basis

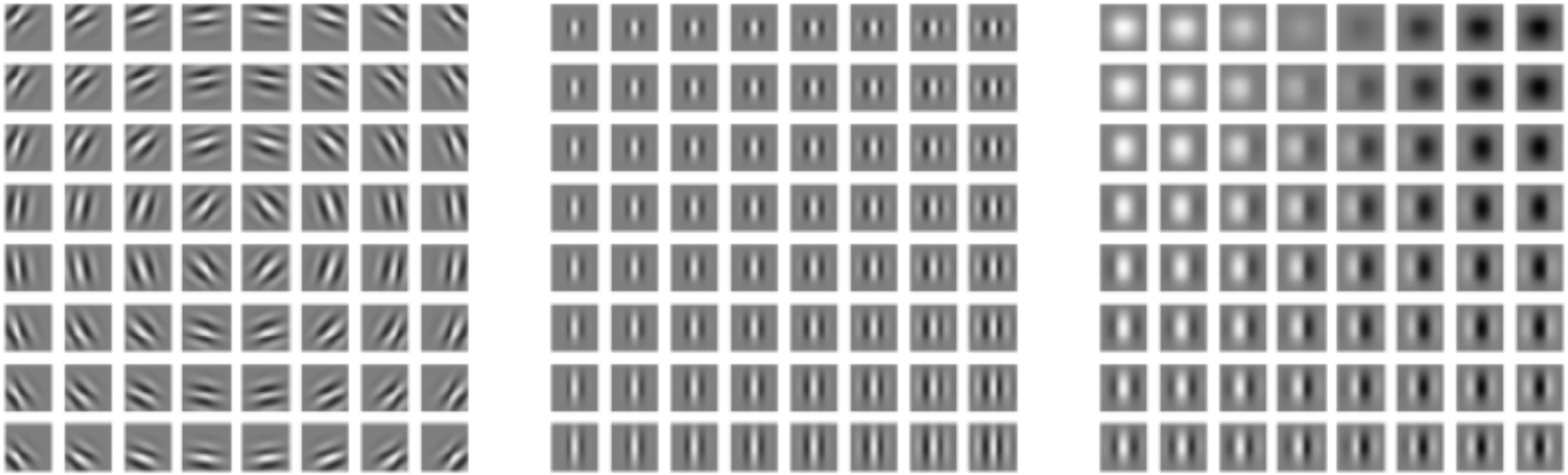
- Some of the key design principles of neural networks were drawn from neuroscience
  - mammalian vision system
  - David Hubel and Torsten Wiesel (Nobel Prize)
    - neurons in the early visual system responded most strongly to very specific patterns of light, such as precisely oriented bars, but responded hardly at all to other patterns

- Primary visual cortex (V1)
  - Spatial map
  - Simple cells
  - Complex cells

Neuroscientific basis

- Grandmother cells – human brain
  - medial temporal lobe
  - cells that respond to some specific concept and are invariant to many transformations of the input
  - The closest analog to a CNN's last layer of features is the brain area called the inferotemporal cortex

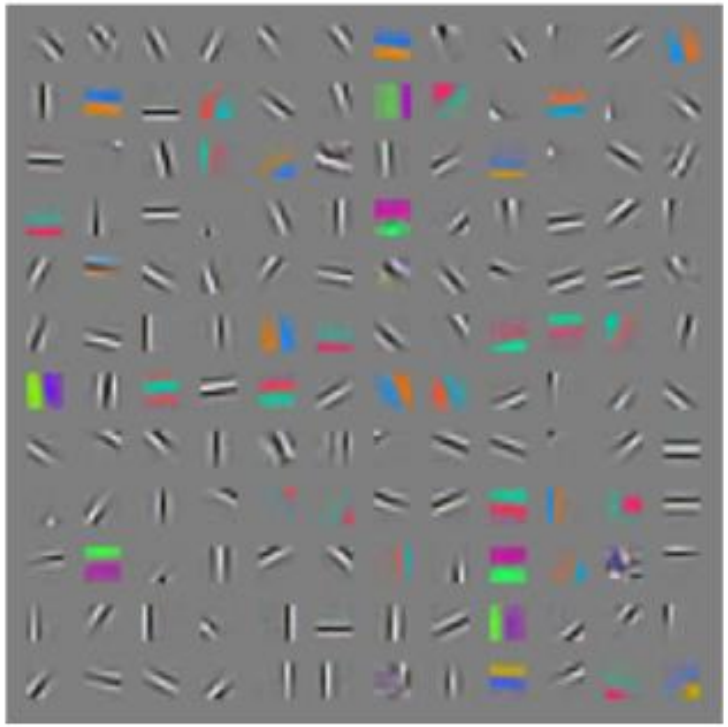
Gabor functions



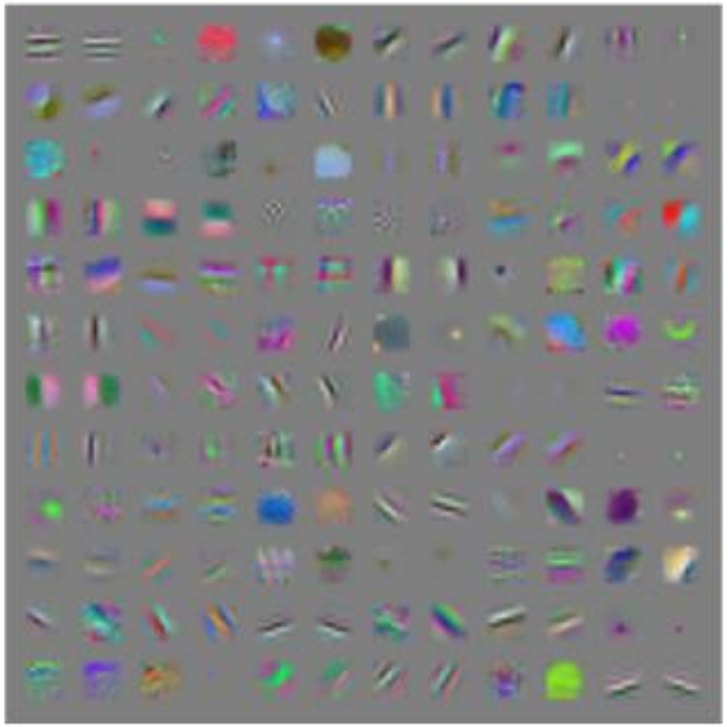
Reverse correlation shows us that most V1 cells have weights that are described by Gabor functions

Gabor functions

Unsupervised learning



First layer convolution kernels



Many machine learning algorithms (e.g, Independent Component Analysis) learn features that detect edges or specific
colors of edges when applied to natural images. These feature detectors are reminiscent of the Gabor functions known to be present in primary visual cortex.

Saliency maps

- Saliency maps
  - Interpretability
  - aim to identify those regions of an image that are most significant in determining the class label

- Grad-CAM (gradient class activation mapping)
  - derivatives of the output-unit pre-activation $a^{(c)}$ for a given class c, before the softmax, with respect to the pre-activations $a_{ij}^{(k)}$ of all the units in the final convolutional layer for channel *k*

$$\alpha_k = \frac{1}{M_k} \sum_i \sum_j \frac{\partial a^{(c)}}{\partial a_{ij}^{(k)}} \quad \Rightarrow \quad \mathbf{L} = \sum_k \alpha_k \mathbf{A}^{(k)}$$

average of derivatives

Saliency maps

Original image | Saliency map for 'dog' | Saliency map for 'cat'

Figure 10.15 Saliency maps for the VGG-16 network with respect to the 'dog' and 'cat' categories. [From Selvaraju *et al.* (2016) with permission.]

Object detection

Bounding box

$(b_x, b_y, b_{\mathrm{W}}, b_{\mathrm{H}})$

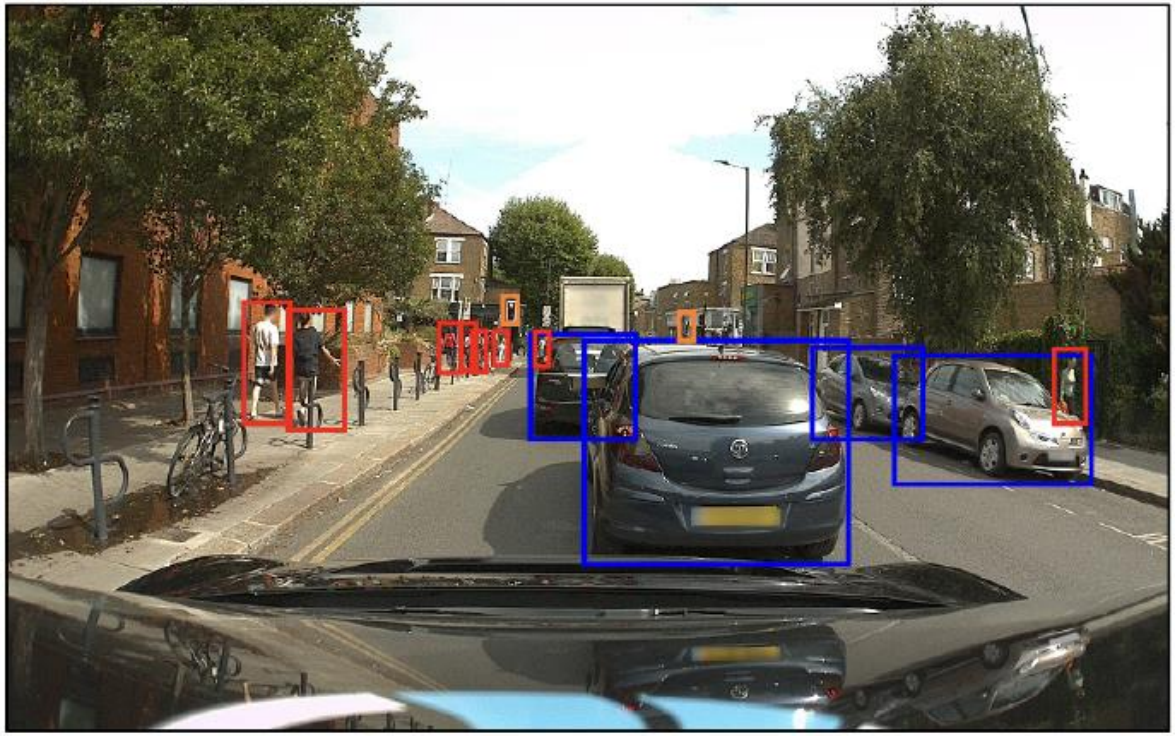
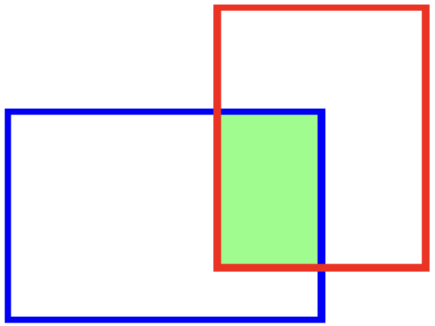


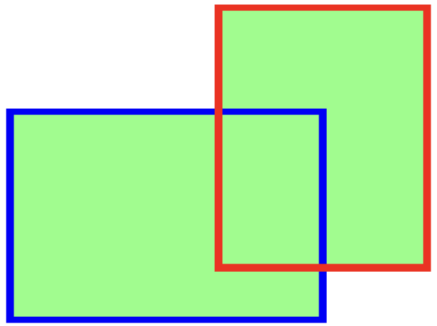
Figure 10.19 An image containing several objects from different classes in which the location of each object is labelled by a close-fitting rectangle known as a bounding box. Here blue boxes correspond to the class 'car', red boxes to the class 'pedestrian', and orange boxes to the class 'traffic light'. [Original image courtesy of Wayve Technologies Ltd.]

Object detection

Figure 10.20 Illustration of the intersection-over-union metric for quantifying the accuracy of a bounding box prediction. If the predicted bounding box is shown by the blue rectangle and the ground truth by the red rectangle, then the intersection-over-union is defined as the ratio of the area of the intersection of the boxes, shown in green on the left, divided by the area of their union, shown in green on the right.



area of intersection



area of union

Faster R-CNN

Figure 10.25 Schematic illustration of multiple detections of the same object at nearby locations, along with their associated probabilities. The red bounding box corresponds to the highest overall probability. Non-max suppression eliminates the other overlapping candidate bounding boxes shown in blue, while preserving the detection of another instance of the same object class shown by the bounding box in green.

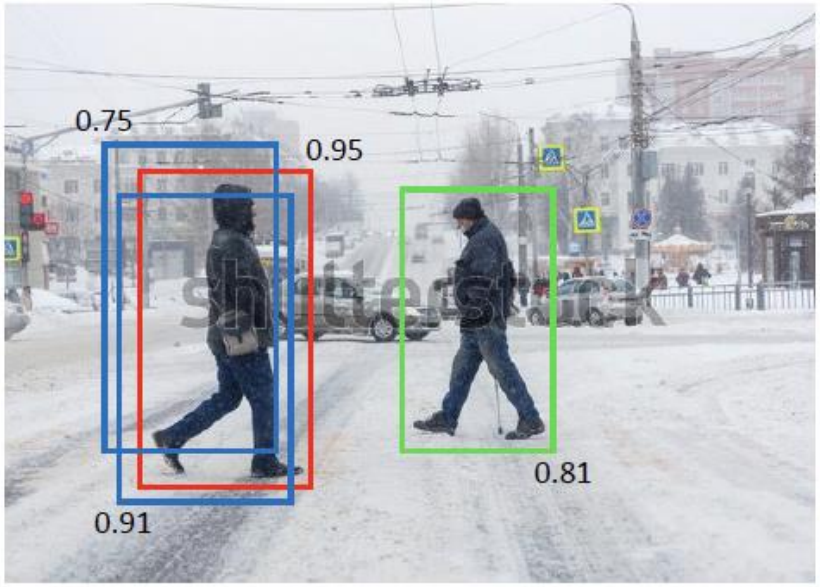


Image segmentation

Figure 10.26 Example of an image and its corresponding semantic segmentation in which each pixel is coloured according to its class. For example, blue pixels correspond to the class 'car', red pixels to the class 'pedestrian', and orange pixels to the class 'traffic light'. [Courtesy of Wayve Technologies Ltd.]

U-net

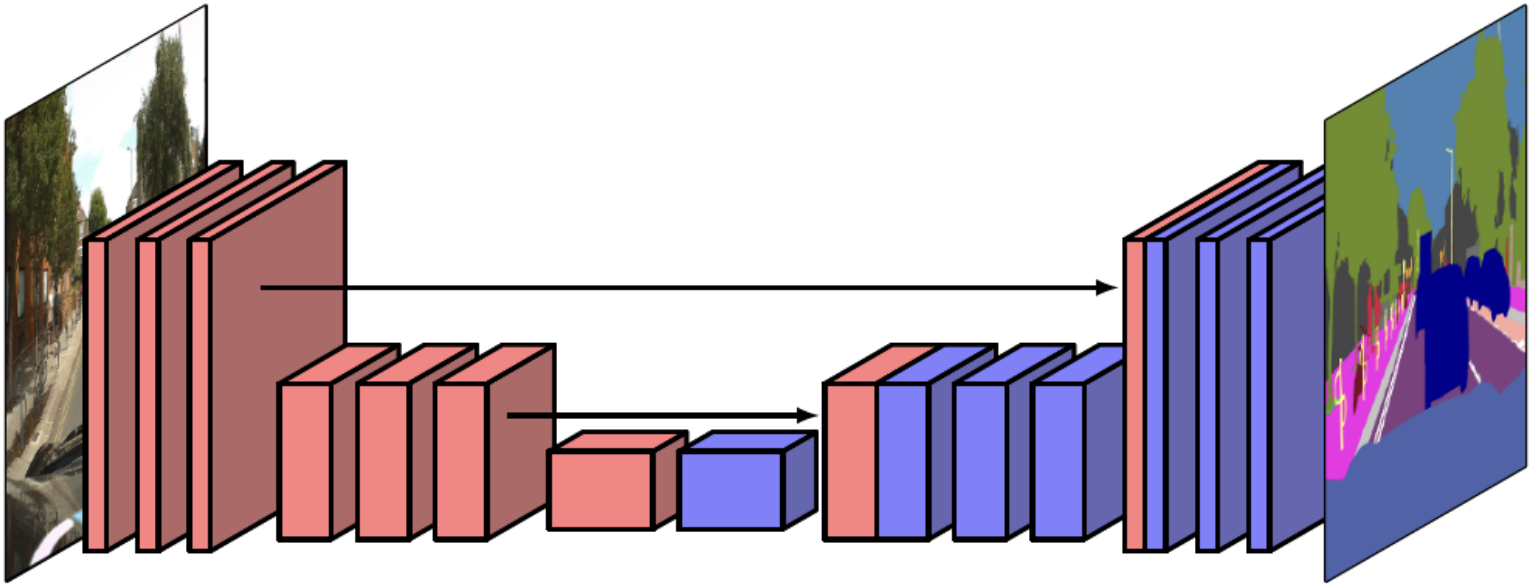


Figure 10.31 The U-net architecture has a symmetrical arrangement of down-sampling and up-sampling layers, and the output from each down-sampling layer is concatenated with the corresponding up-sampling layer.

Style transfer

Figure 10.32 An example of neural style transfer showing a photograph of a canal scene (left) that has been rendered in the style of *The Wreck of a Transport Ship* by J. M. W. Turner (centre) and in the style of *The Starry Night* by Vincent van Gogh (right). In each case the image used to provide the style is shown in the inset. [From Gatys, Ecker, and Bethge (2015) with permission.]

CNN models

- Recent models
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
  - ResNet
  - ZFNet
  - …