Machine Learning (part II)

Convolutional Neural Network

Angelo Ciaramella

Convolutional Neural Networks

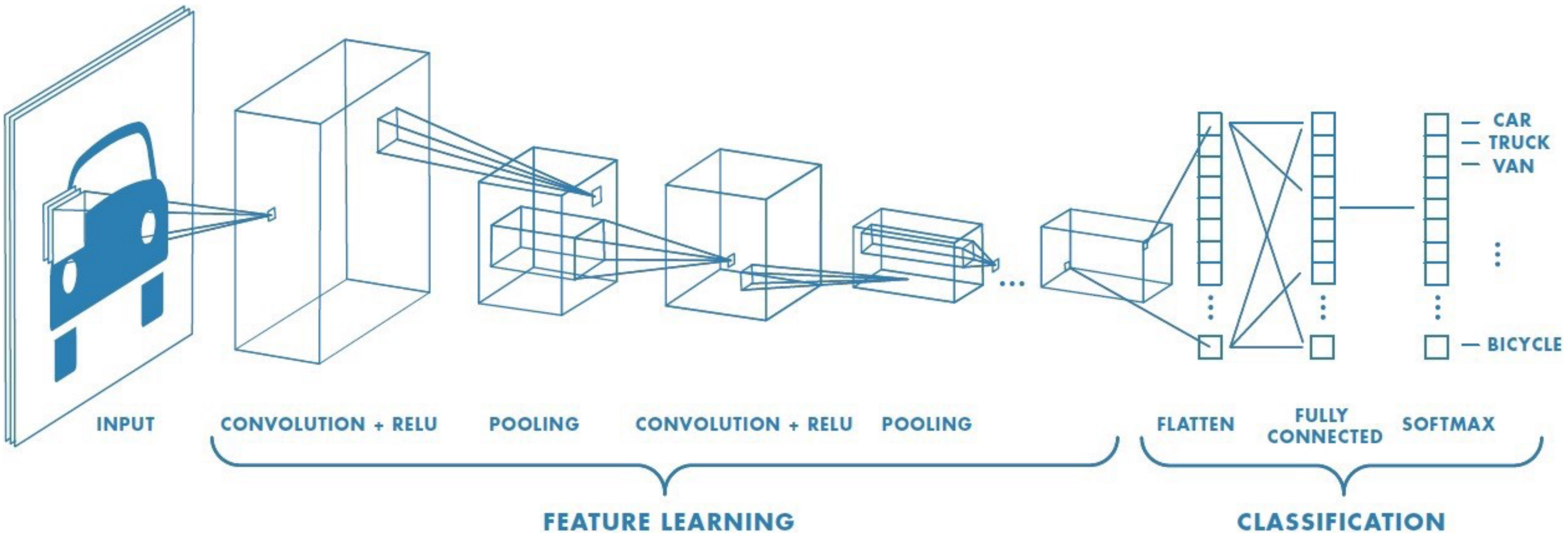
- Scale up neural networks to process very large images / video sequences
  - Sparse connections
  - Parameter sharing
- Automatically generalize across spatial translations of inputs
- Applicable to any input that is laid out on a grid (1-D, 2-D, 3-D, …)

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.

CNN



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

Convolution

- Convolution operation

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

$$s(t) = (x * w)(t)$$

- 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

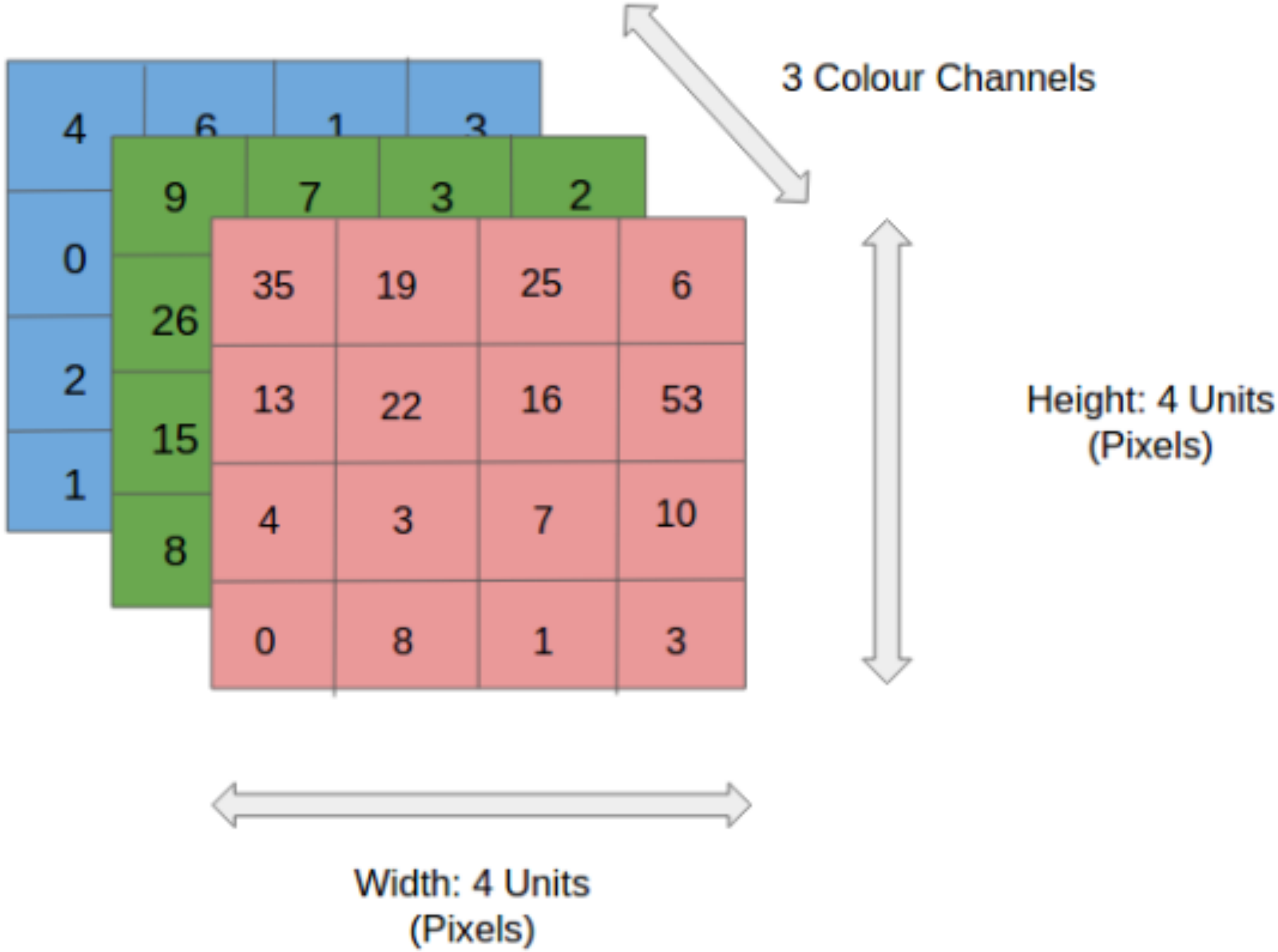
flipping the kernel

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

- Cross-correlation

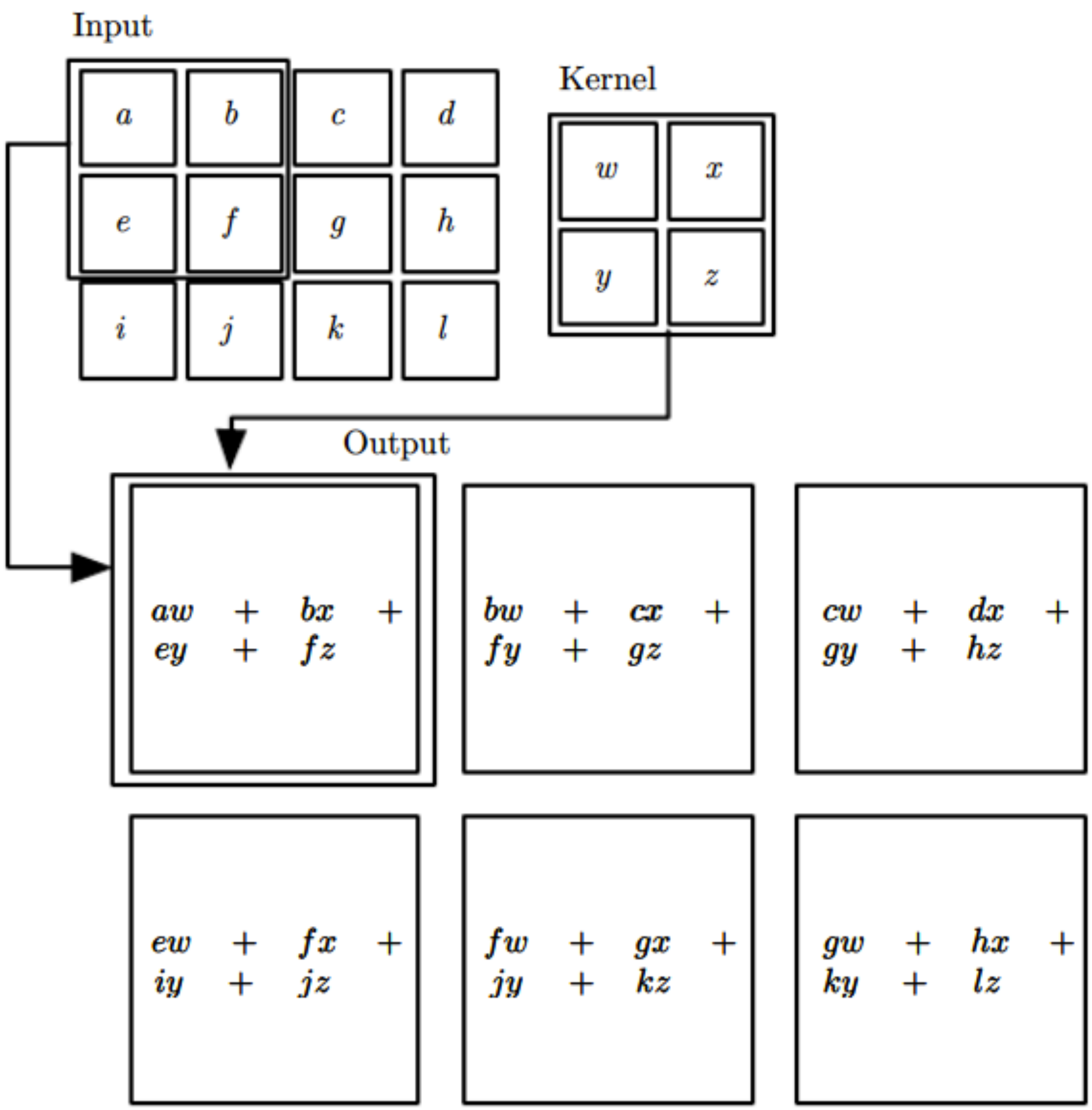
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

Images



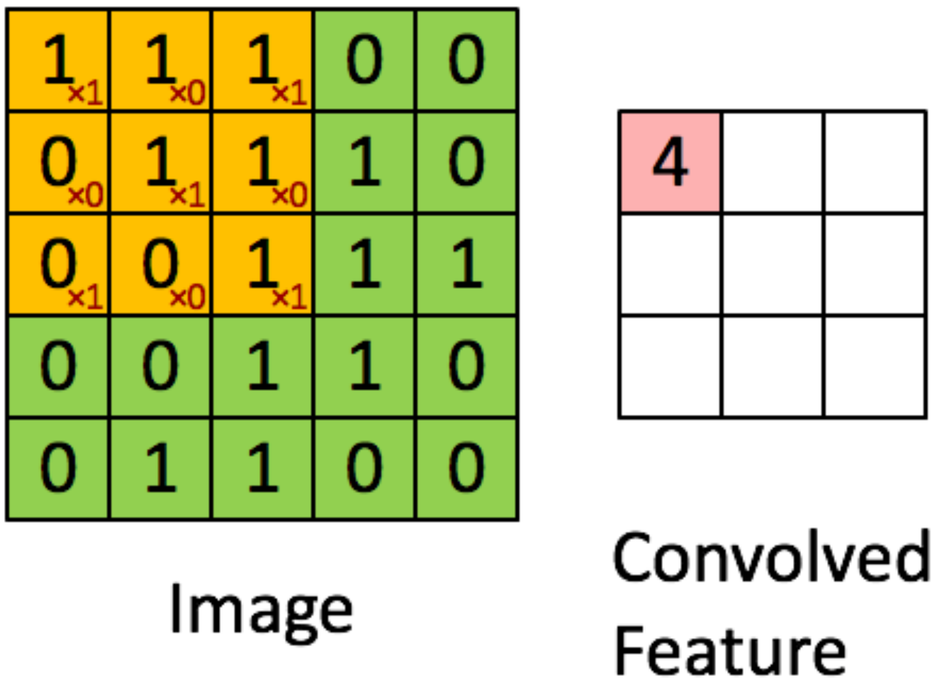
3 Colour Channels
4 6 1 3
0
2
1
9 7 3 2
26
15
8
35 19 25 6
13 22 16 53
4 3 7 10
0 8 1 3
Height: 4 Units
(Pixels)
Width: 4 Units
(Pixels)

Convolution



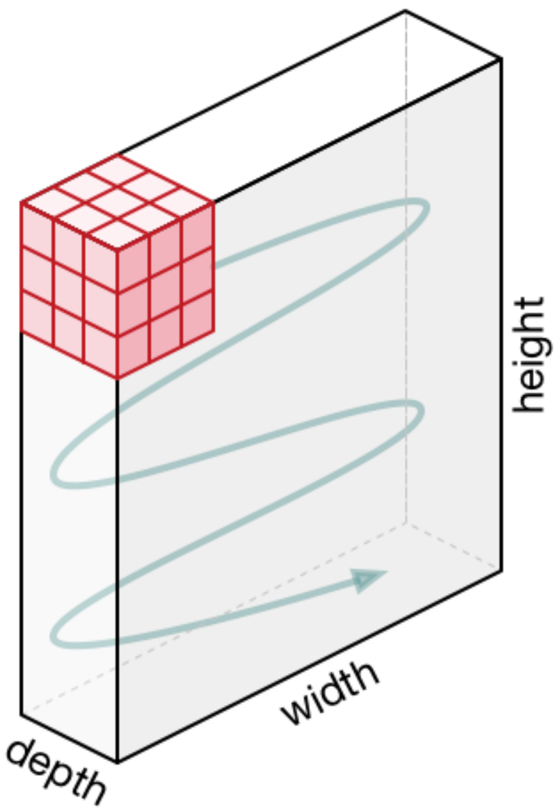
2D convolution example

Convolution



Convolution example

Movement of the kernel



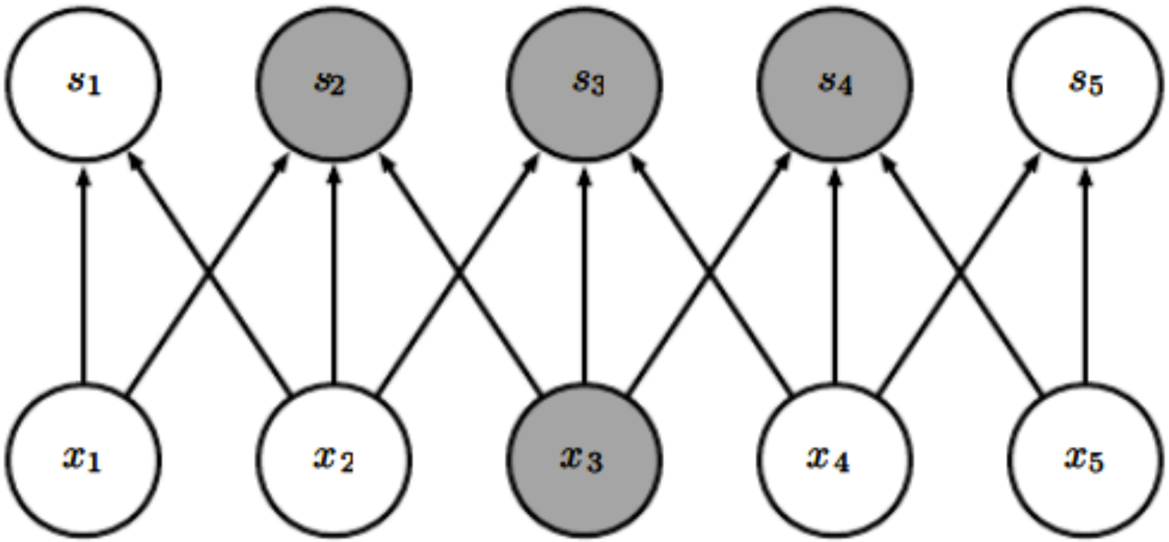
Motivation

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

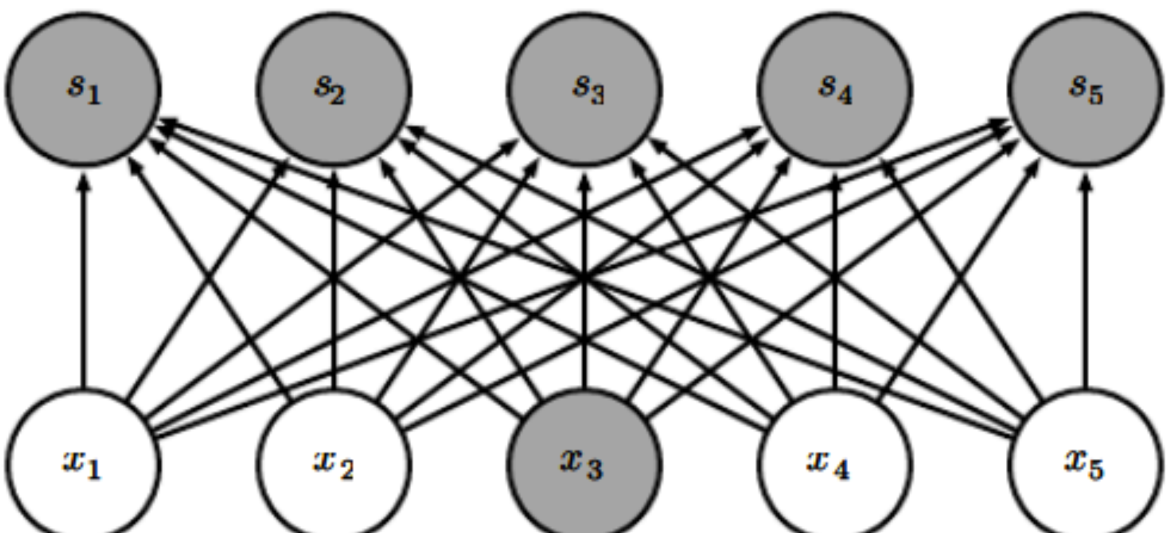
Operations

- 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

Sparse interactions

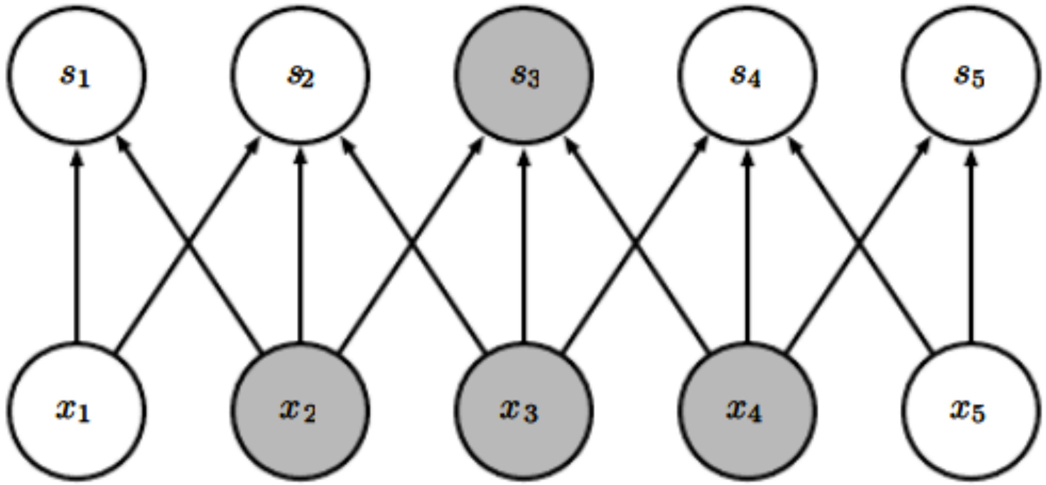


sparse
(kernel of width 3)

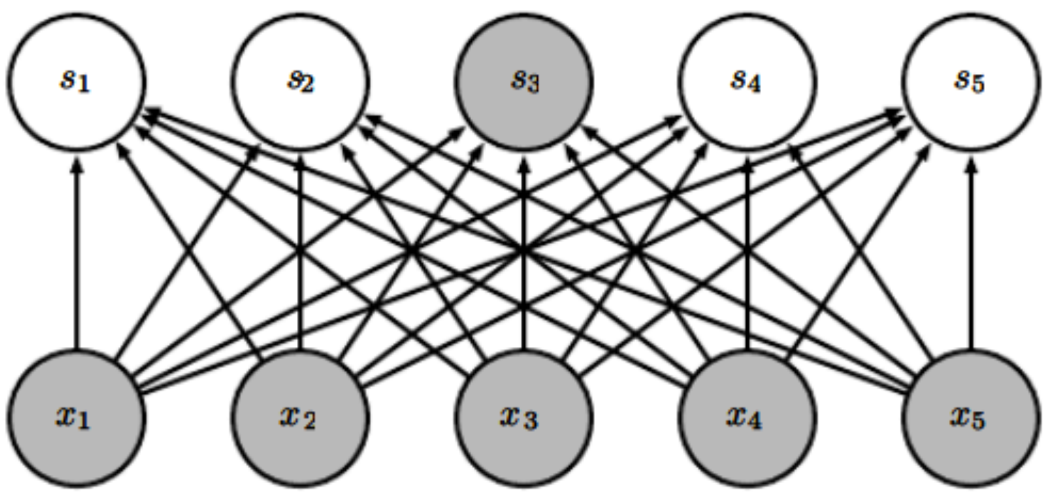


Dense connections
no longer sparse

Sparse interactions

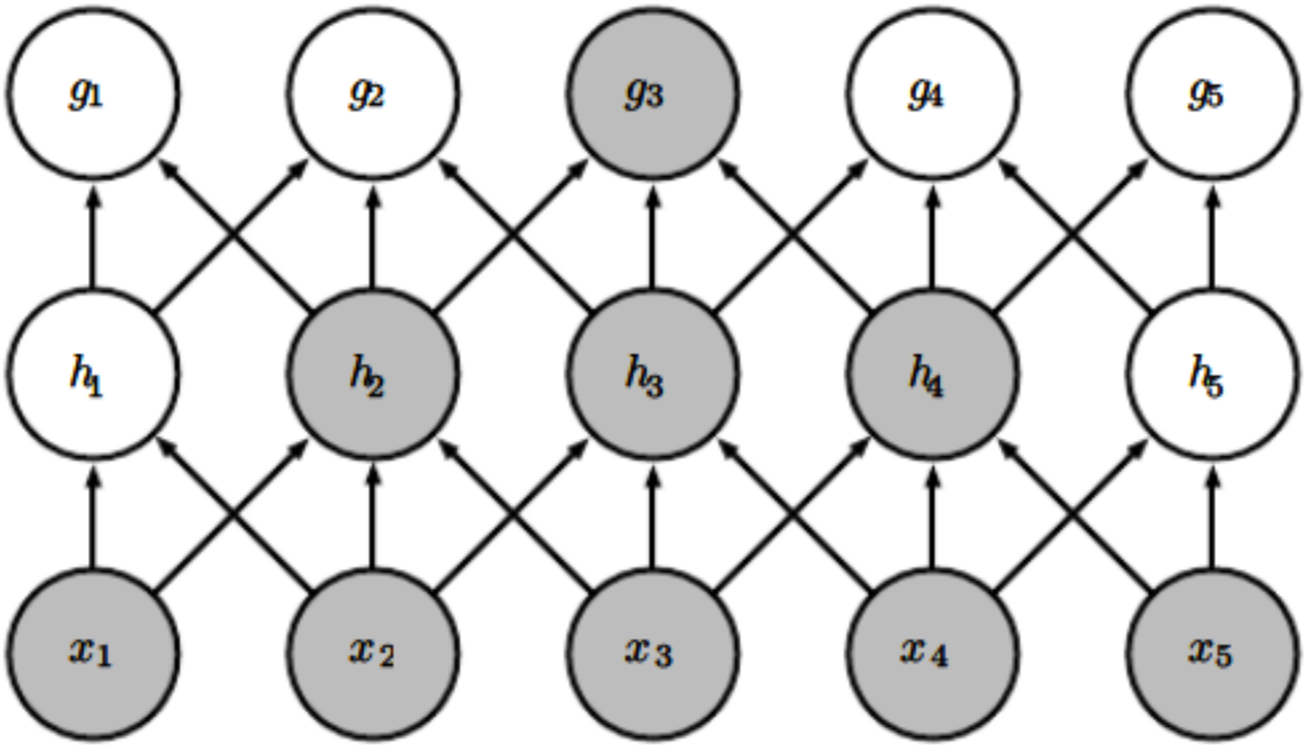


sparse



Dense
connections
no longer sparse

Sparse interactions



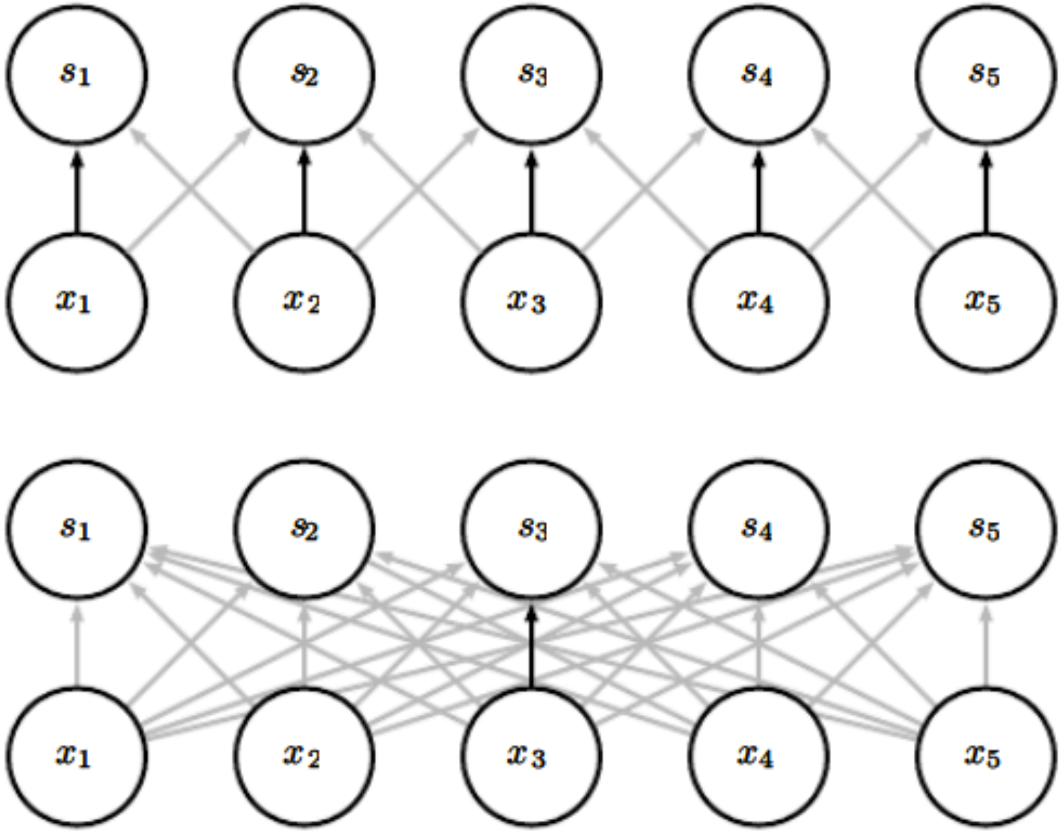
even though direct connections in a convolutional net are very sparse, units in the deeper layers can be indirectly connected to all or most of the input image

Parameter sharing

- Goal
  - rather than learning a separate set of parameters for every location, we learn only one set
  - 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

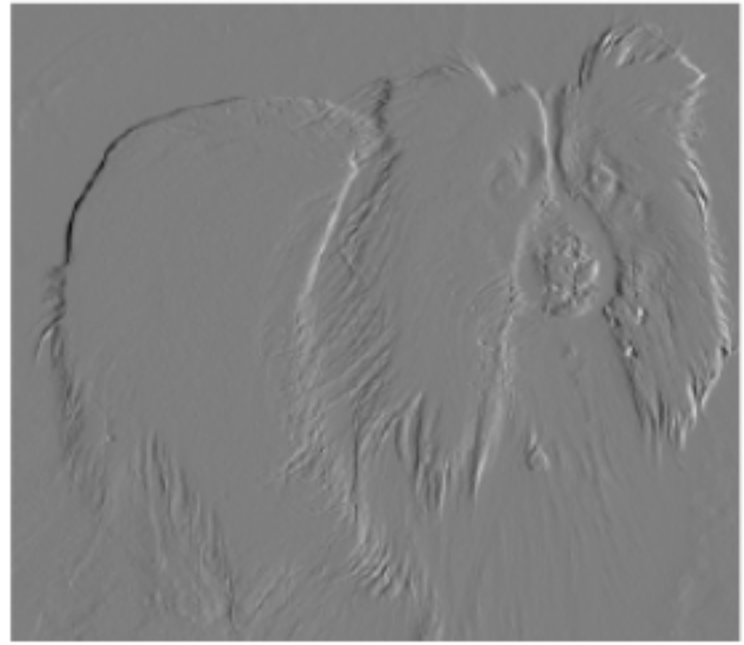
Parameter sharing

- Goal
  - rather than learning a separate set of parameters for every location, we learn only one set



Parameter sharing

Input



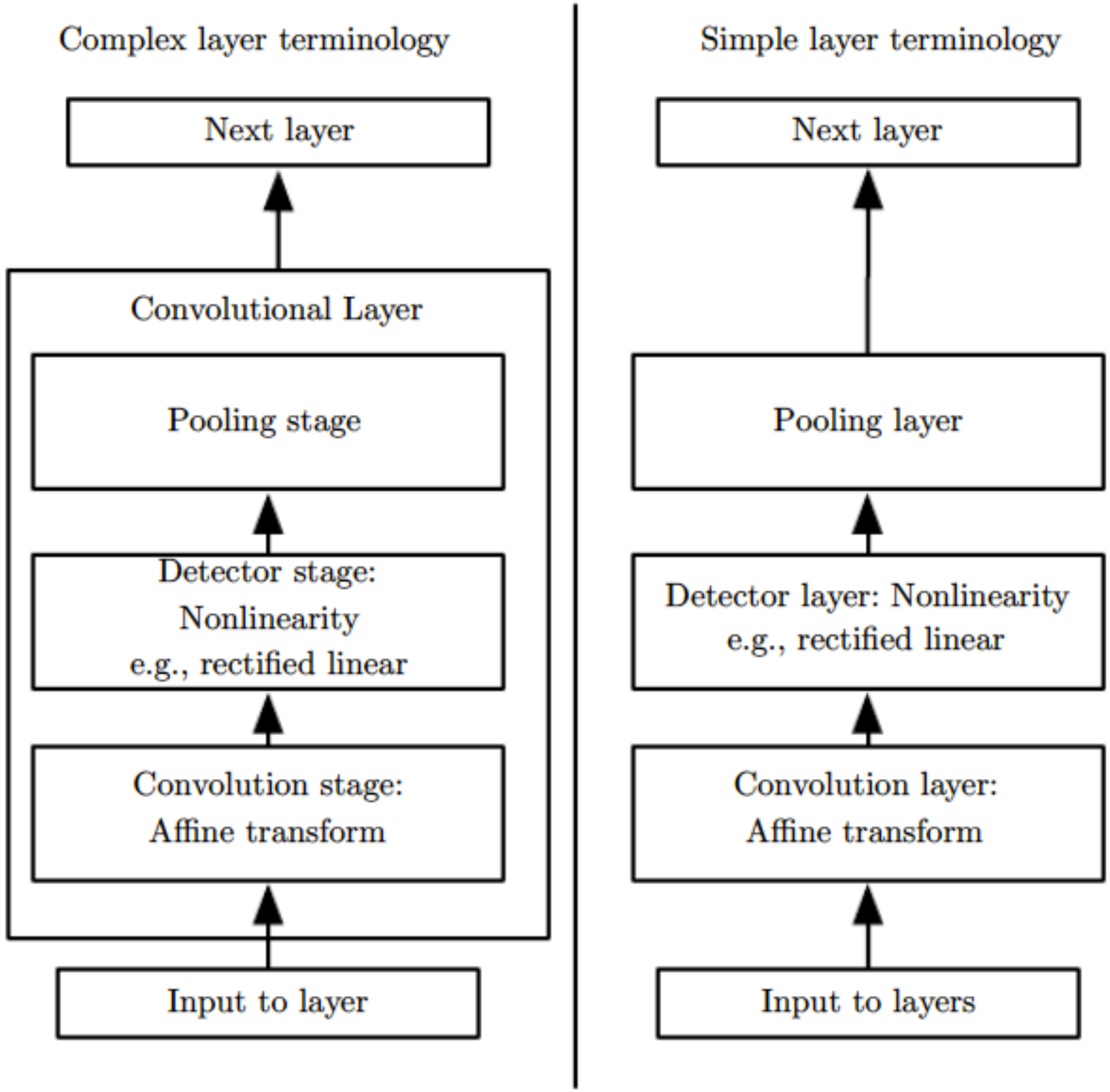
Output

1	-1

Kernel

Efficiency of edge detection

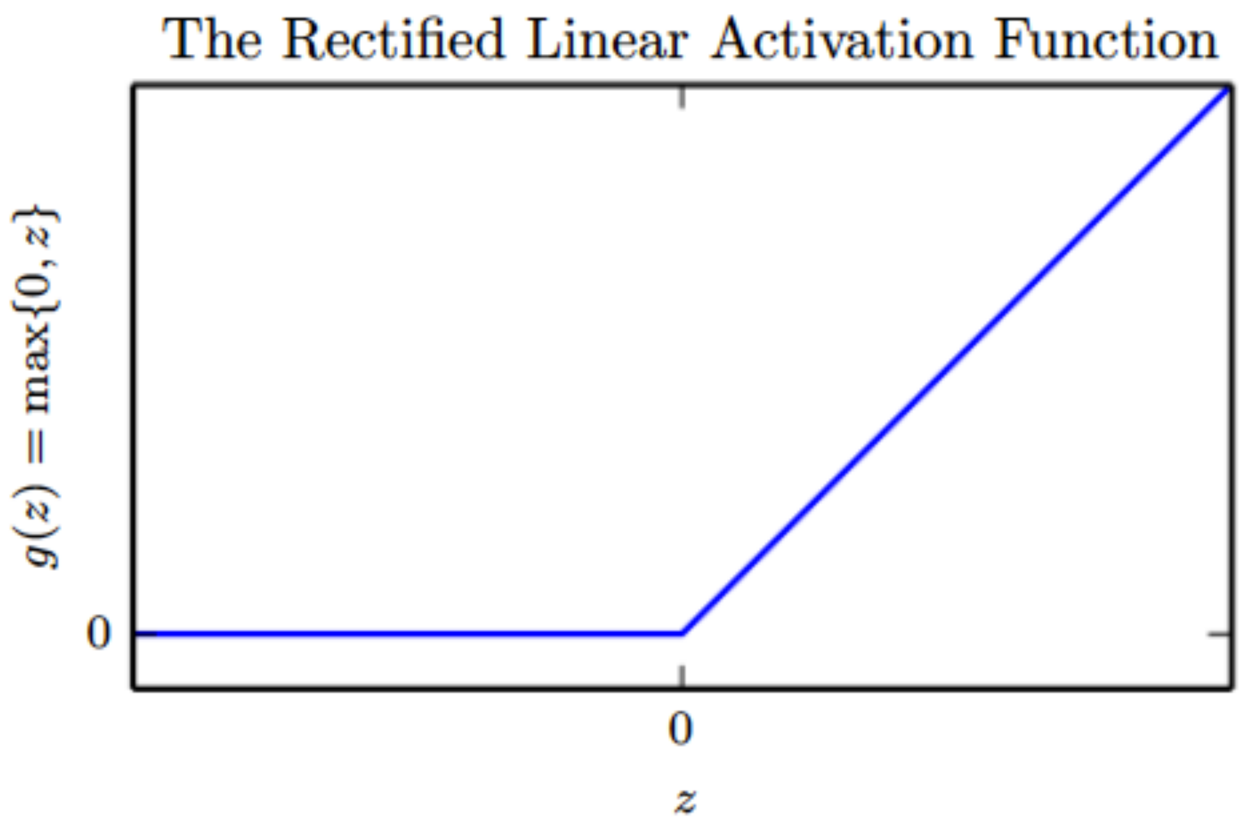
Stages


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

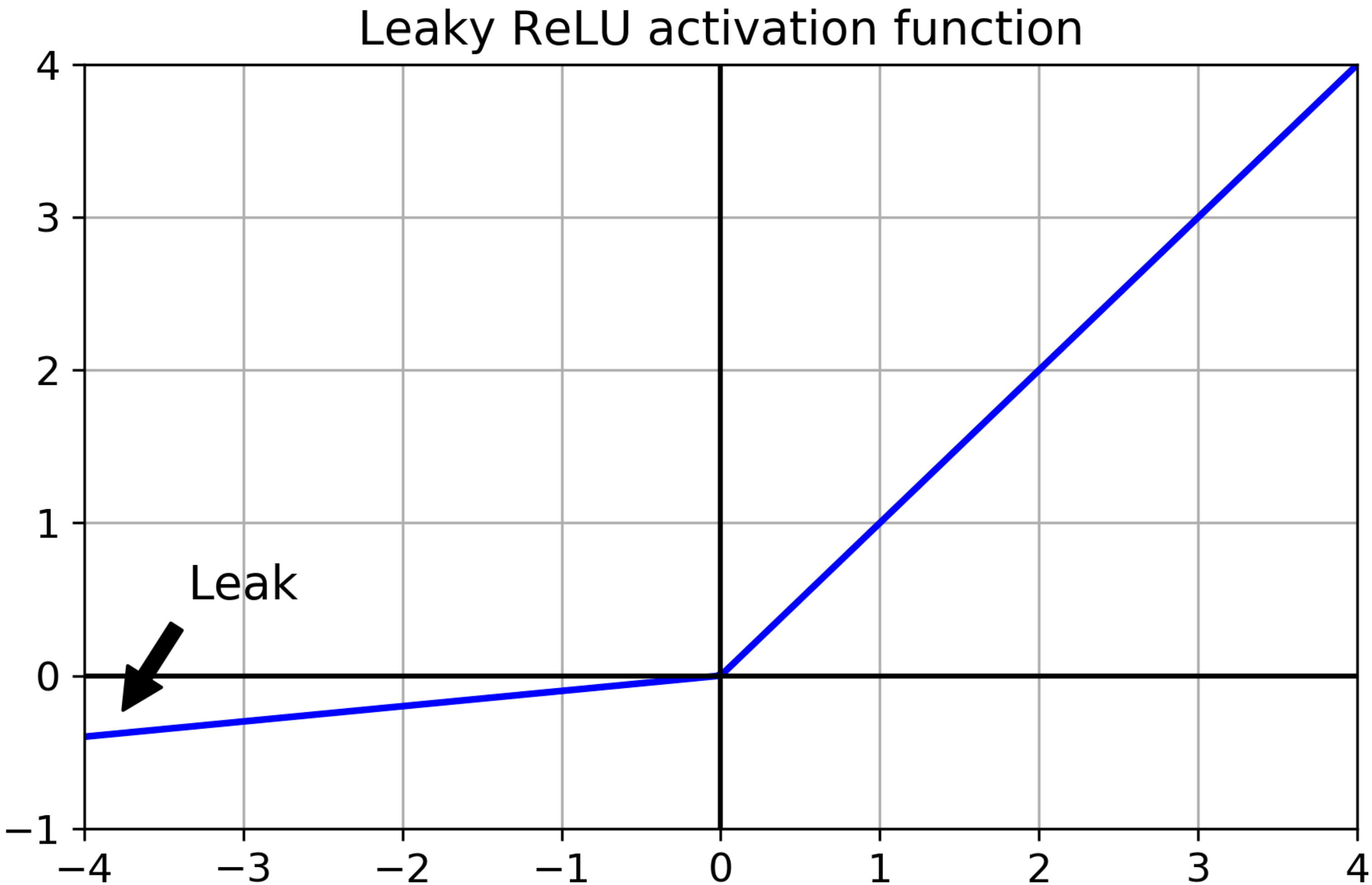
- ReLU generalizations
  - Slope

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

  - Absolute value rectification

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

Leaky ReLU

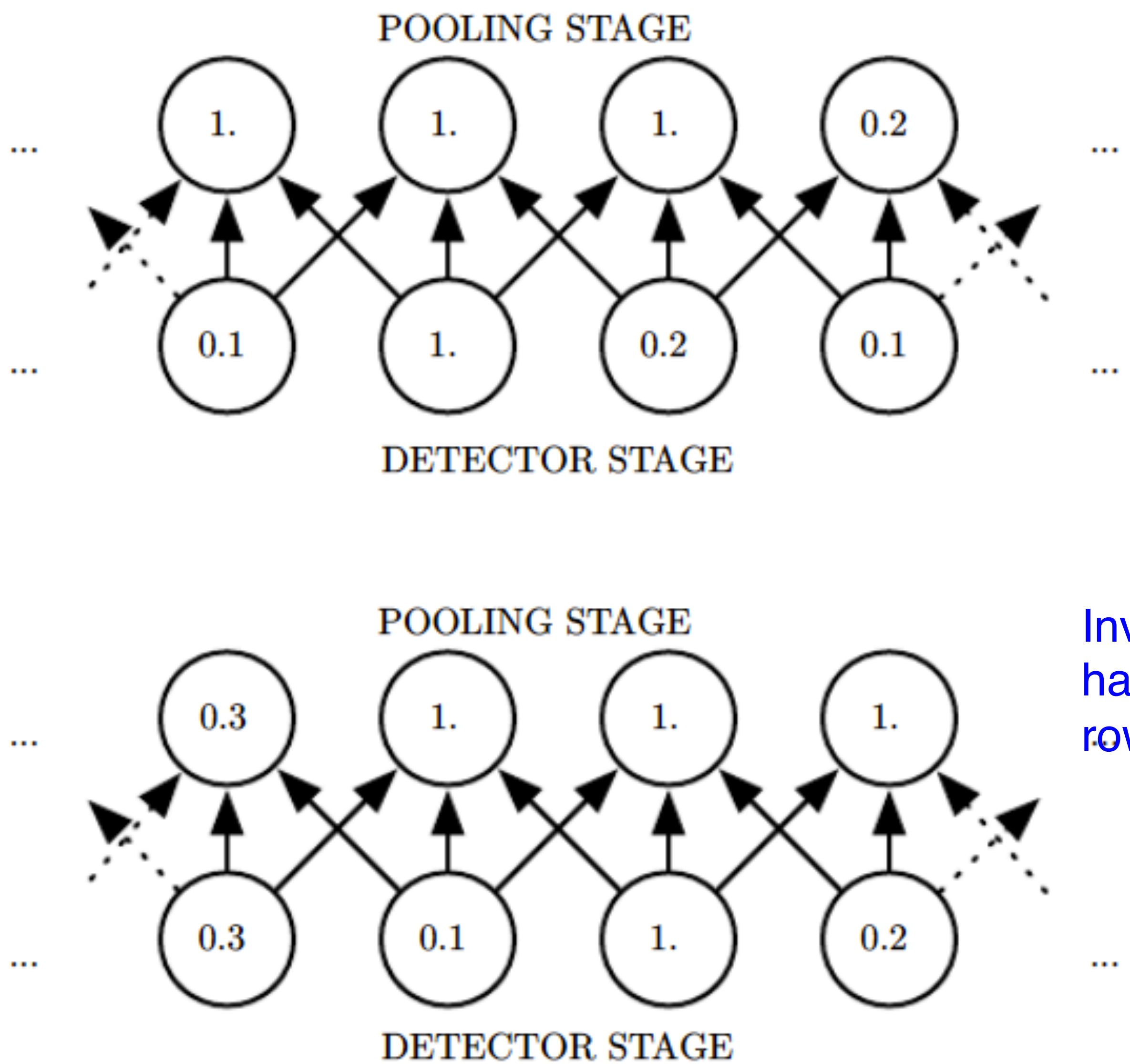


max(αz, z)

Pooling

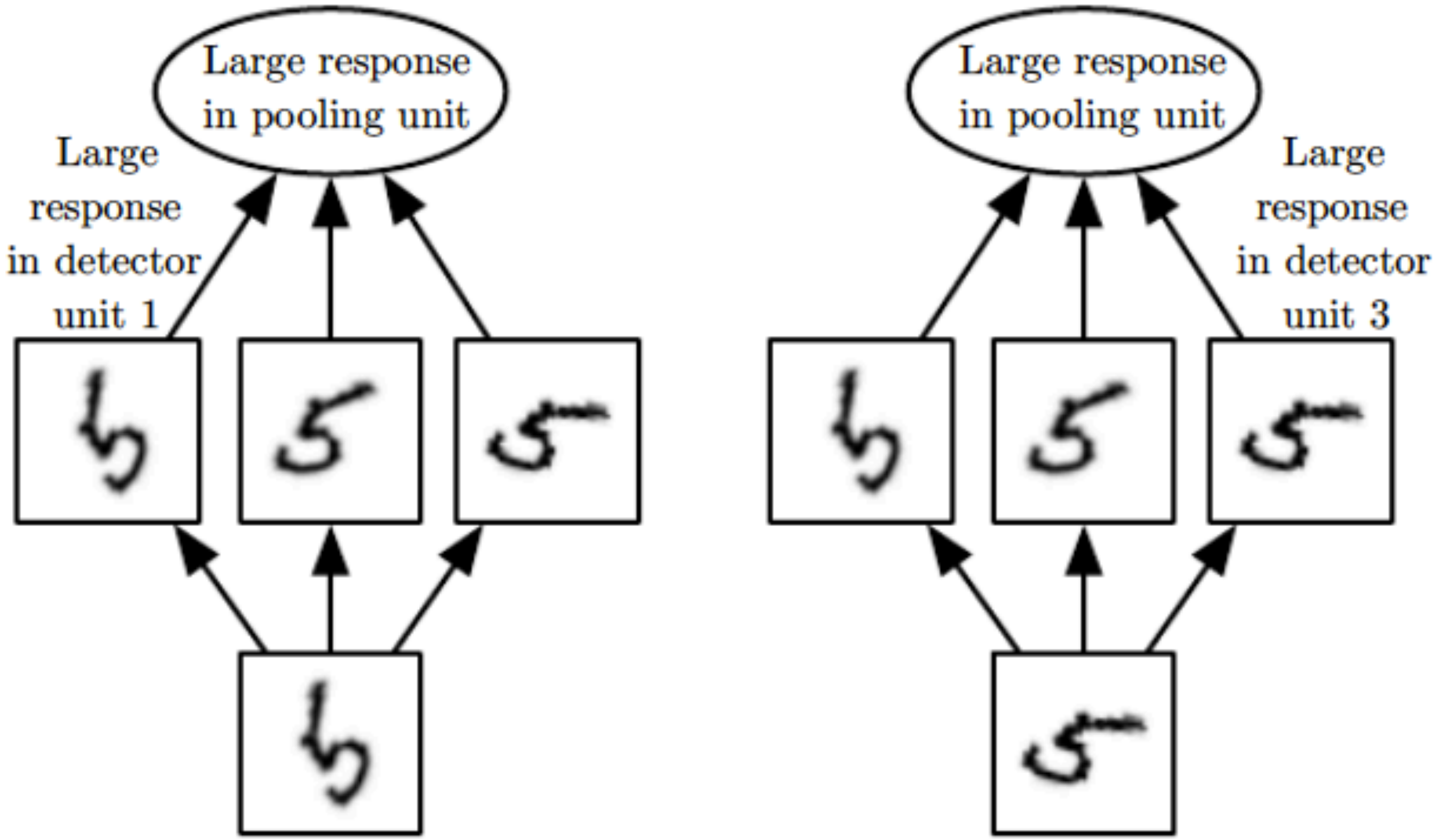
- Pooling function
  - replaces the output of the net at a certain location with a summary statistic of the nearby outputs
  - helps to make the representation become approximately invariant to small translations of the input
  - Max pooling
    - maximum output within a rectangular neighborhood
  -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

Pooling



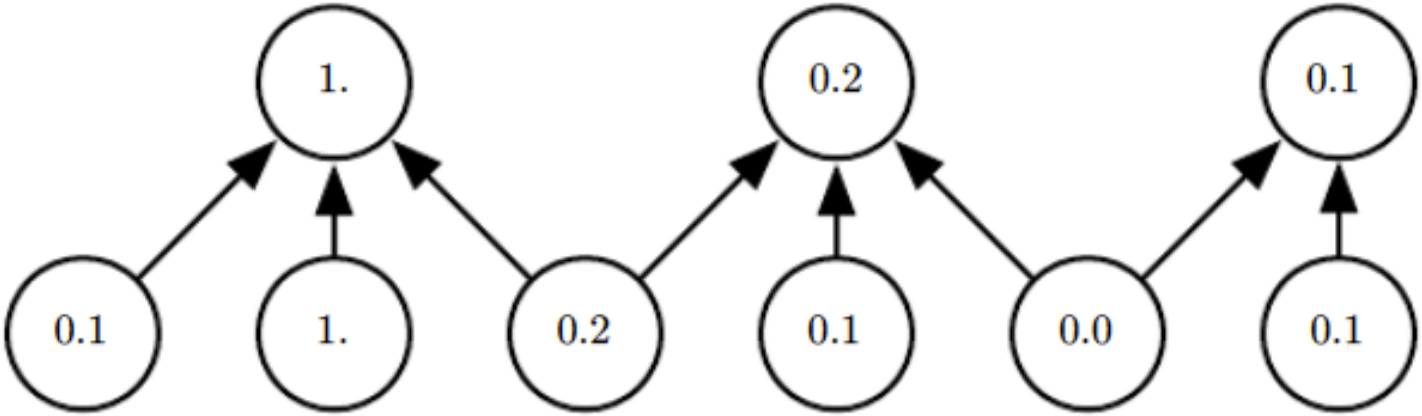
Invariance –
half of the values in the top row have changed

Pooling



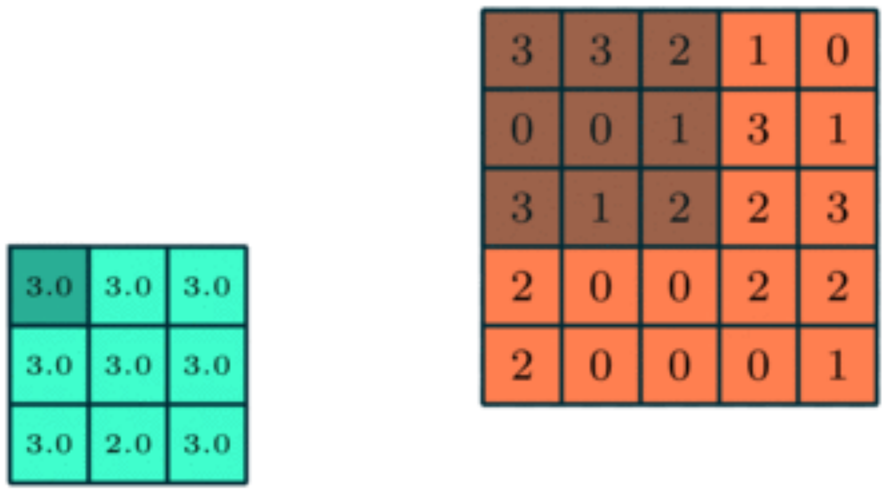
Invariance -invariant to transformations of the input

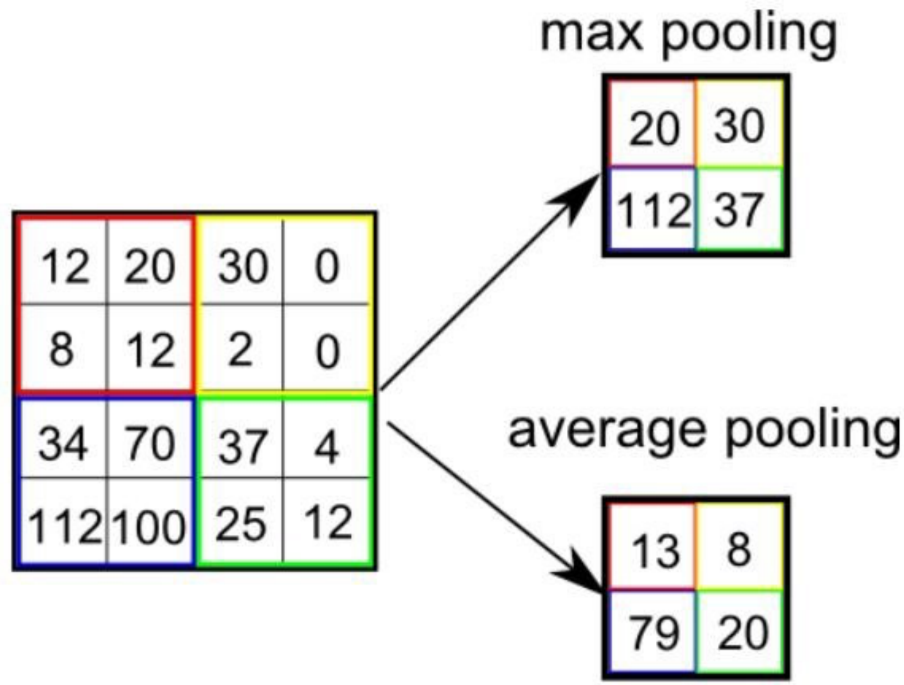
Pooling



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

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


max pooling
average pooling
12 20 30 0
8 12 2 0
34 70 37 4
112 100 25 12
20 30
112 37
13 8
79 20

Variants of the CNN

- Basic Convolution Function
  - The functions used in practice differ slightly

- Neural Network context
  - operation that consists of many applications of convolution in parallel
    - each layer of our network to extract many kinds of features
  - the input is usually not just a grid of real values
    - color image has a red, green and blue intensity at each pixel
    - working with images usually the input and output of the convolution as being 3-D tensors
    - usually work in batch mode using 4-D tensors

Variants of the CNN

- 4-D kernel tensor

K	$K_{i,j,k,l}$
tensor	elements

- Tensors
  - Array with more that two axes

- $K_{i,j,k,l}$
  - connection strength between a unit in channel *i* of the output and a unit in channel *j* of the input, with an offset of *k* rows and *l* columns between the output unit and the input unit

Variants of the CNN

- Input observed data

$$\mathsf{V} \qquad\qquad V_{i,j,k}$$

input unit within channel i at row j and column k

- Convolution

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

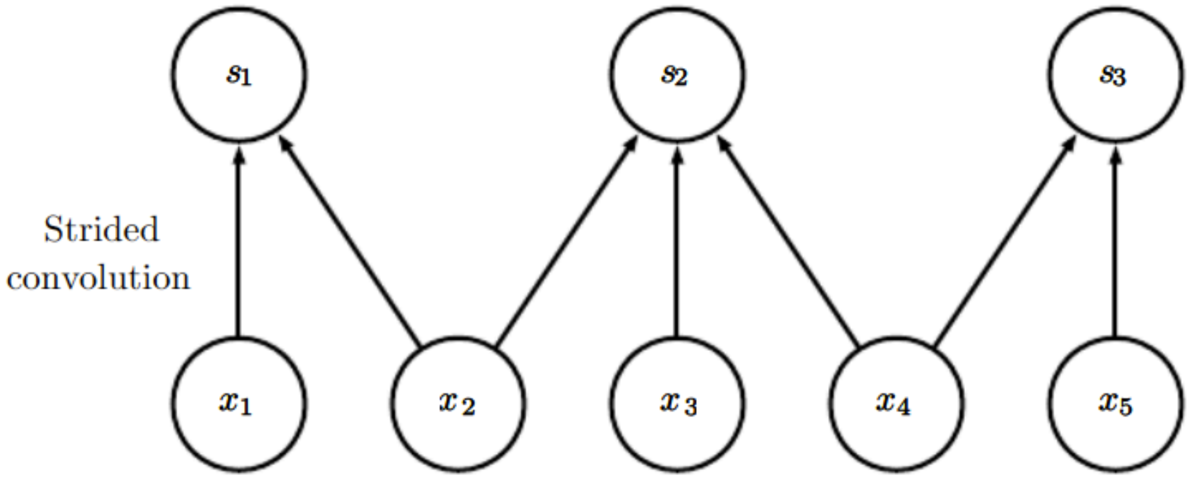
Variants of the CNN

- downsampled convolution function c such that

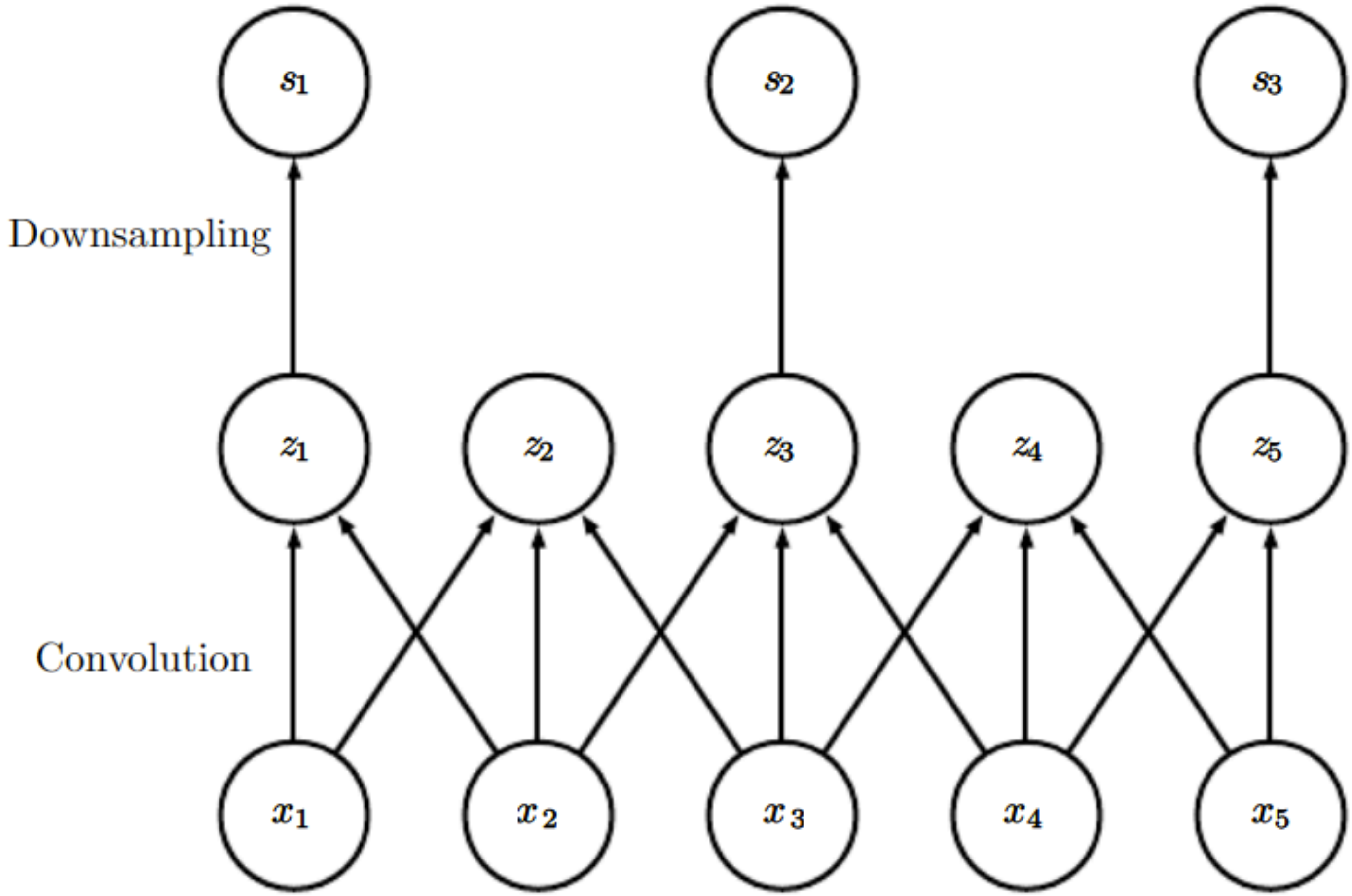
$$Z_{i,j,k} = c(\mathsf{K}, \mathsf{V}, s)_{i,j,k} = \sum_{l,m,n} \left[V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n} \right]$$

- 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



Variants of the CNN



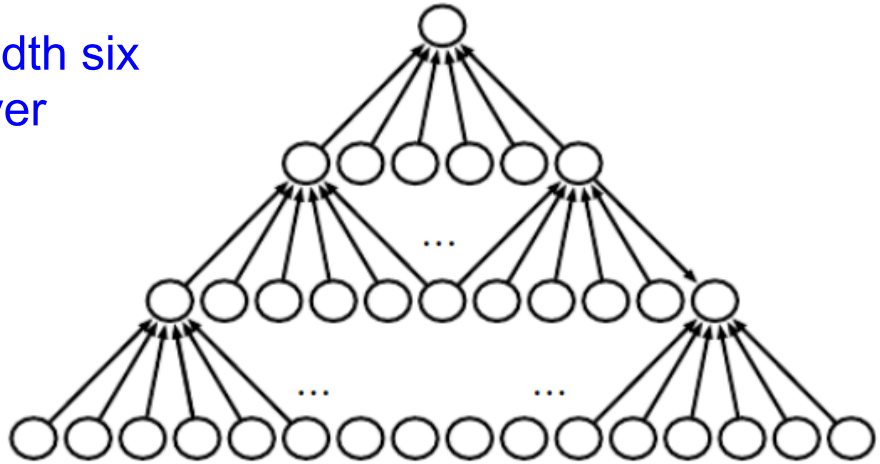
Variants of the CNN

- 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

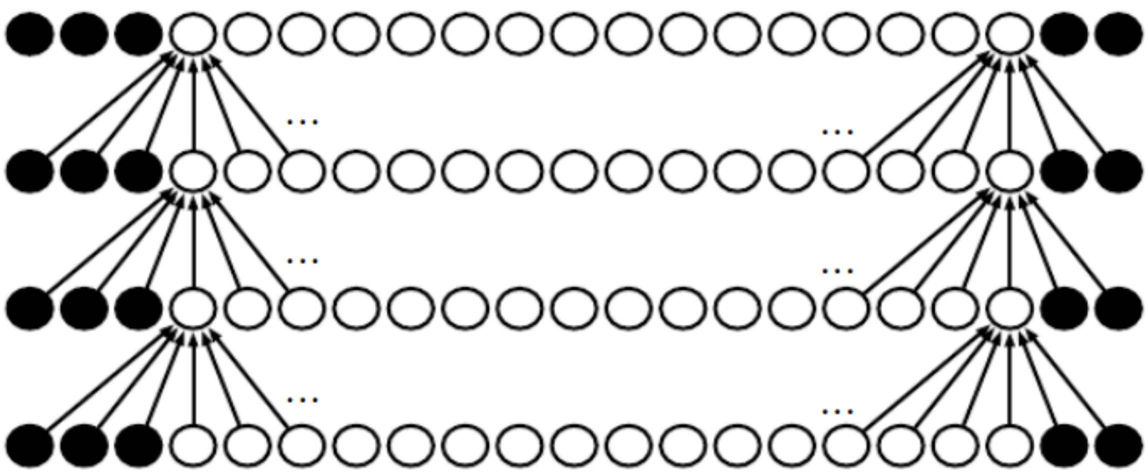
Variants of the CNN

kernel of width six
at every layer



representation
shrinks by five pixels
at each layer

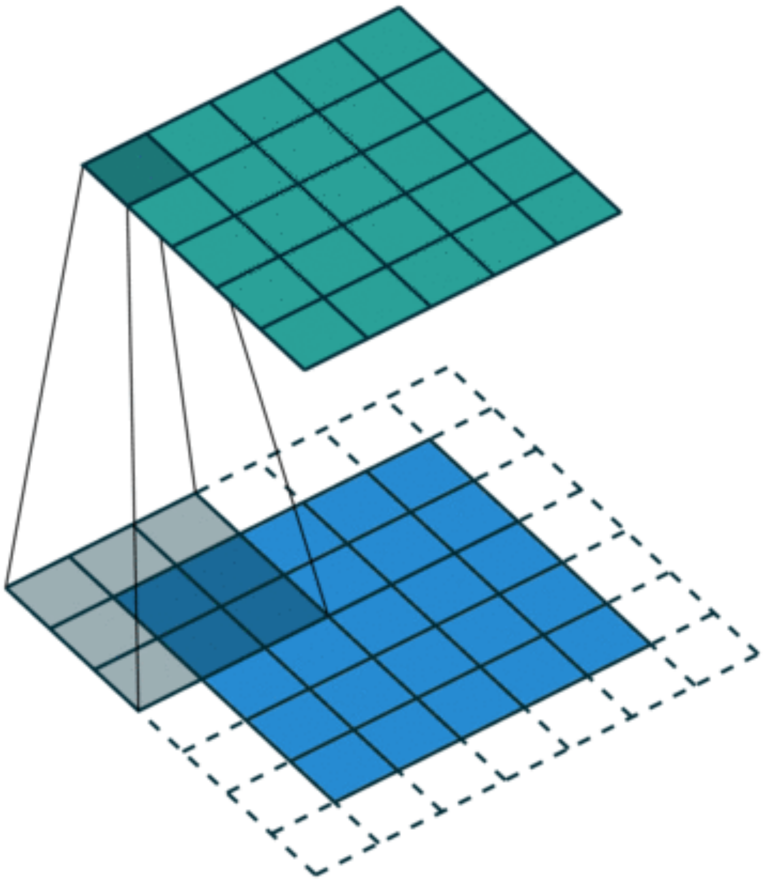
Sixteen pixels



The effect of zero
padding on network
size

Adding five implicit zeroes

Variants of the CNN



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

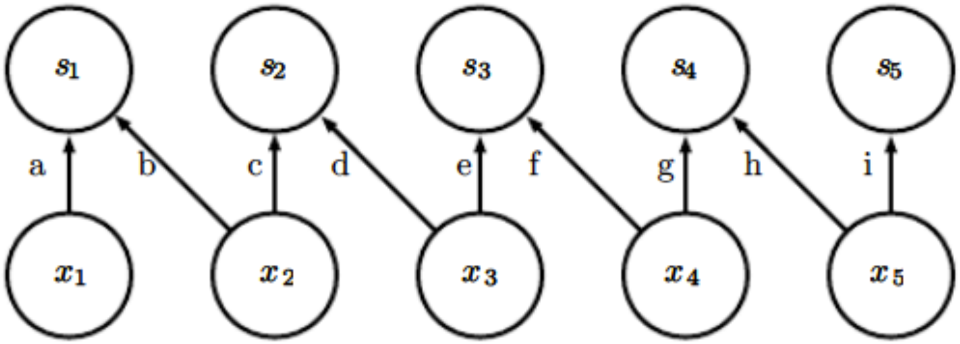
Unshared convolution

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

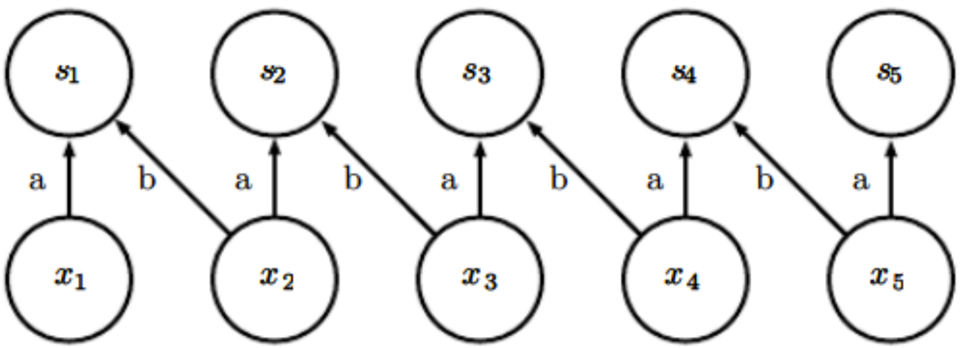
$$Z_{i,j,k} = \sum_{l,m,n} [V_{l,j+m-1,k+n-1} w_{i,j,k,l,m,n}]$$

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

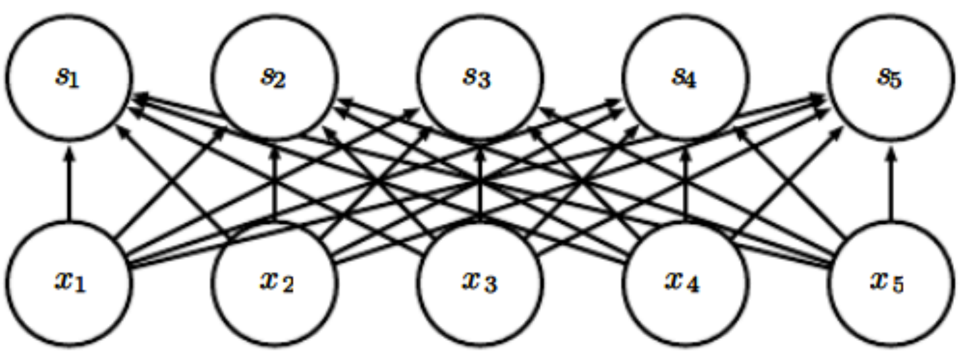
Unshared convolution



Local connections

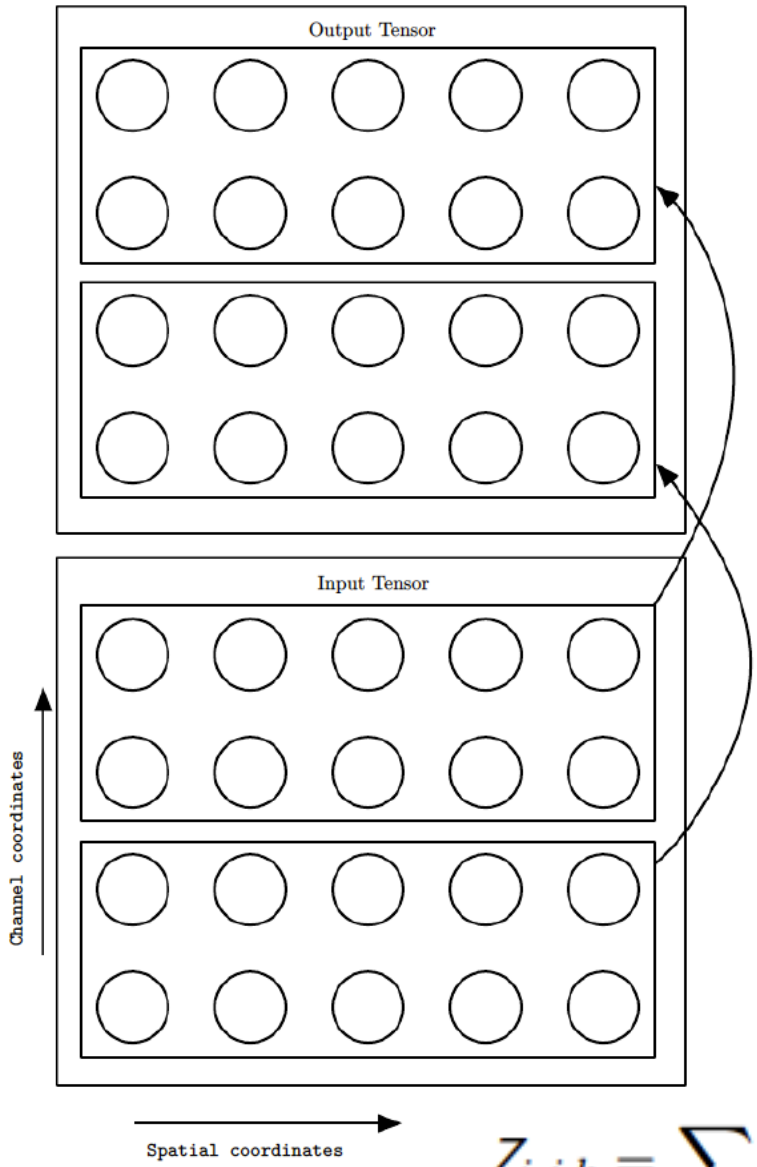


Convolution



Full connections

Tiled convolution



t different choices of kernel stack in each direction

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

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

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

Learning

- Minimize the loss function

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

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

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

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

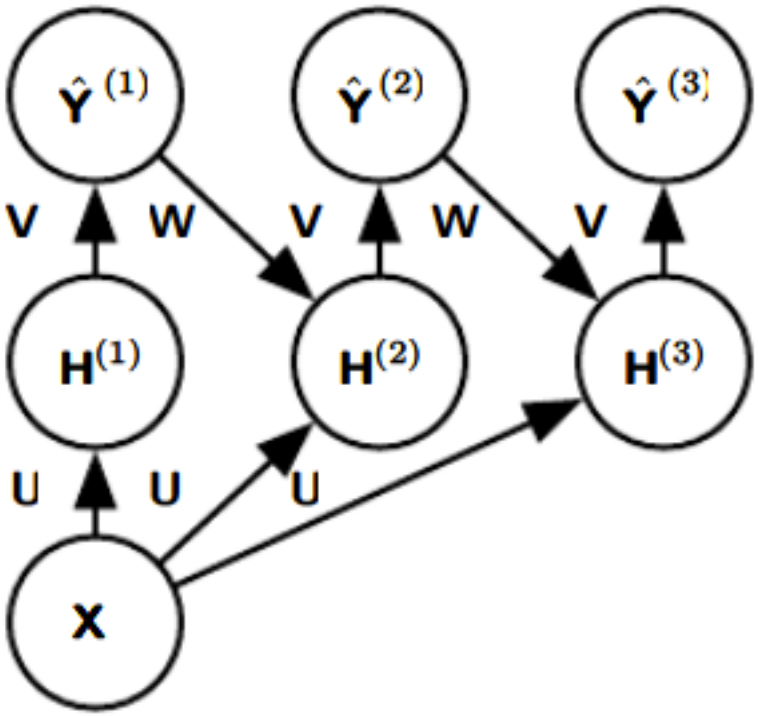
Learning

- If the layer is not the bottom layer

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

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

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

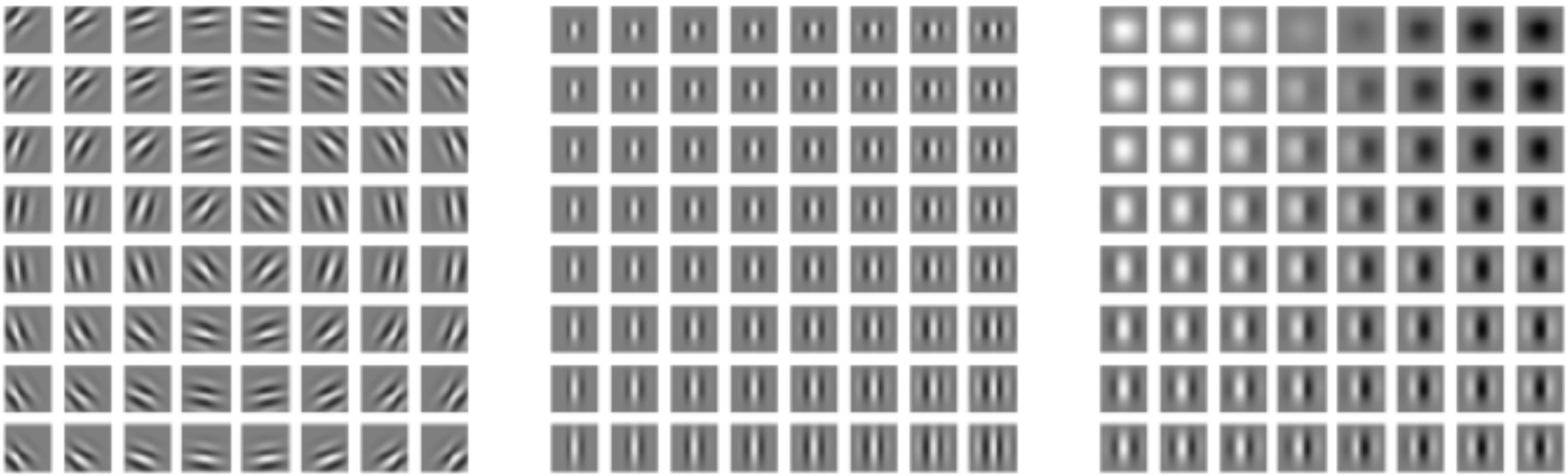
Neuroscientific basis

- Some of the key design principles of neural networks were drawn from neuroscience
  - mammalian vision system works
  - 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
  - Spatial map
  - Simple cells
  - Complex cells

Neuroscientific basis

- Grandmother cells
  - cells that respond to some specific concept and are invariant to many transformations of the input
  - medial temporal lobe

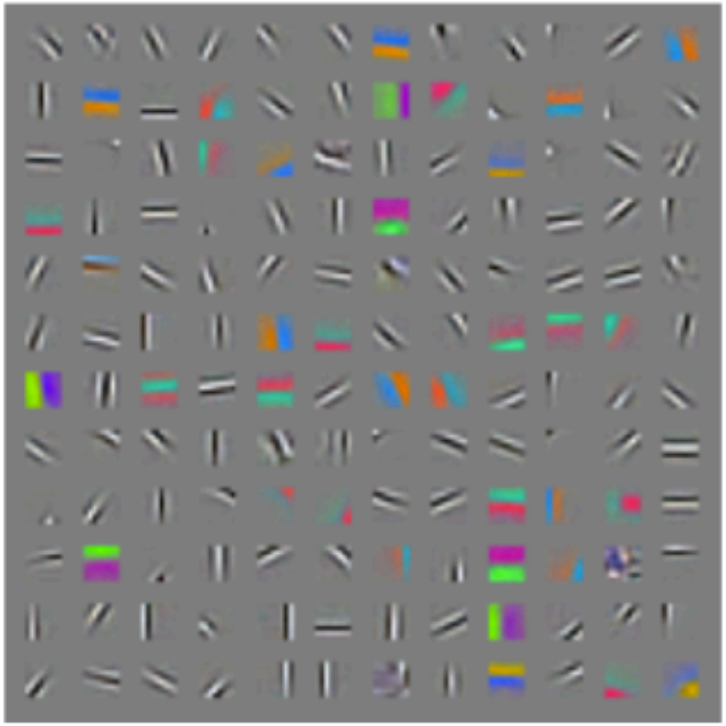
Gabor functions



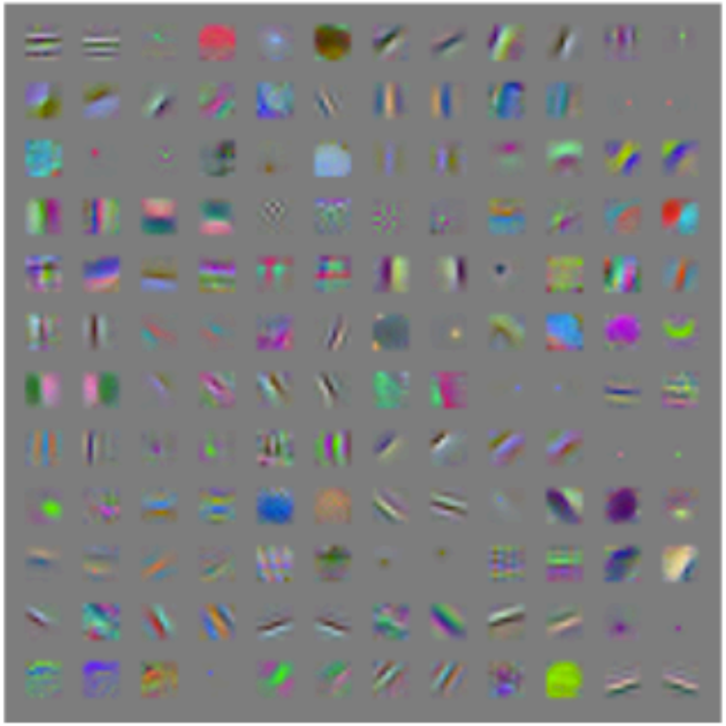
Reverse correlation shows us that most V1 cells have weights that are described by Gabor functions

Gabor functions

Unsupervised learning



First layer convolution kernels



Many machine learning algorithms 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.

CNN models

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