TBMI26
Neural Networks and Learning Systems
Lecture 5
Convolutional Neural Networks

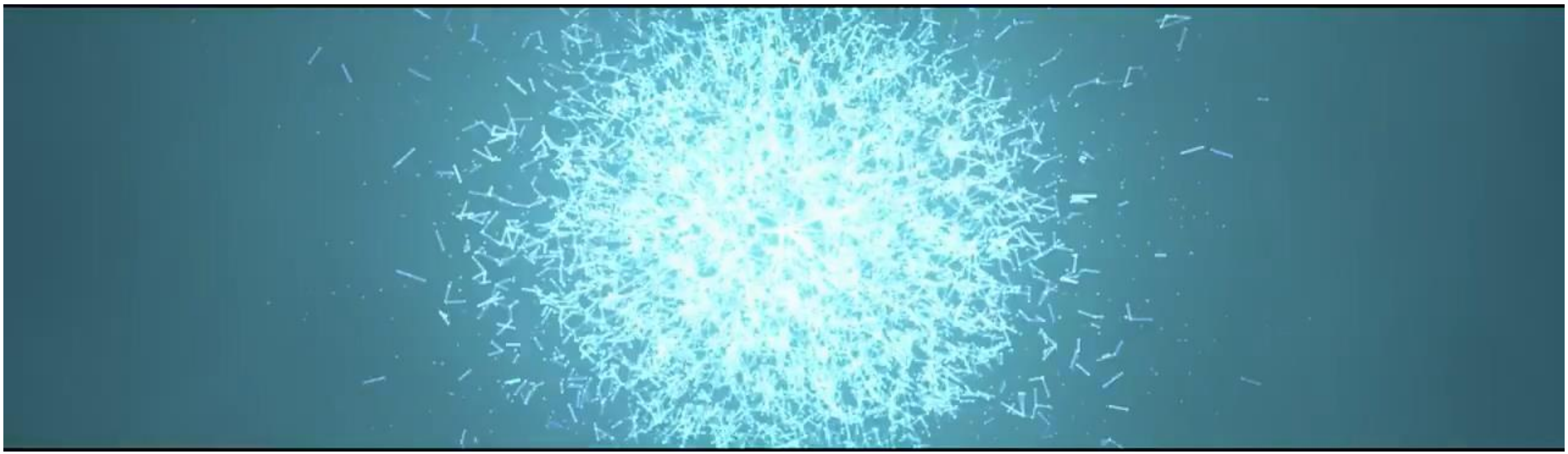
Michael Felsberg
Computer Vision Laboratory
Department of Electrical Engineering
Linköping University, Sweden

LINKÖPINGS UNIVERSITET

Introduction

LINKÖPINGS
UNIVERSITET

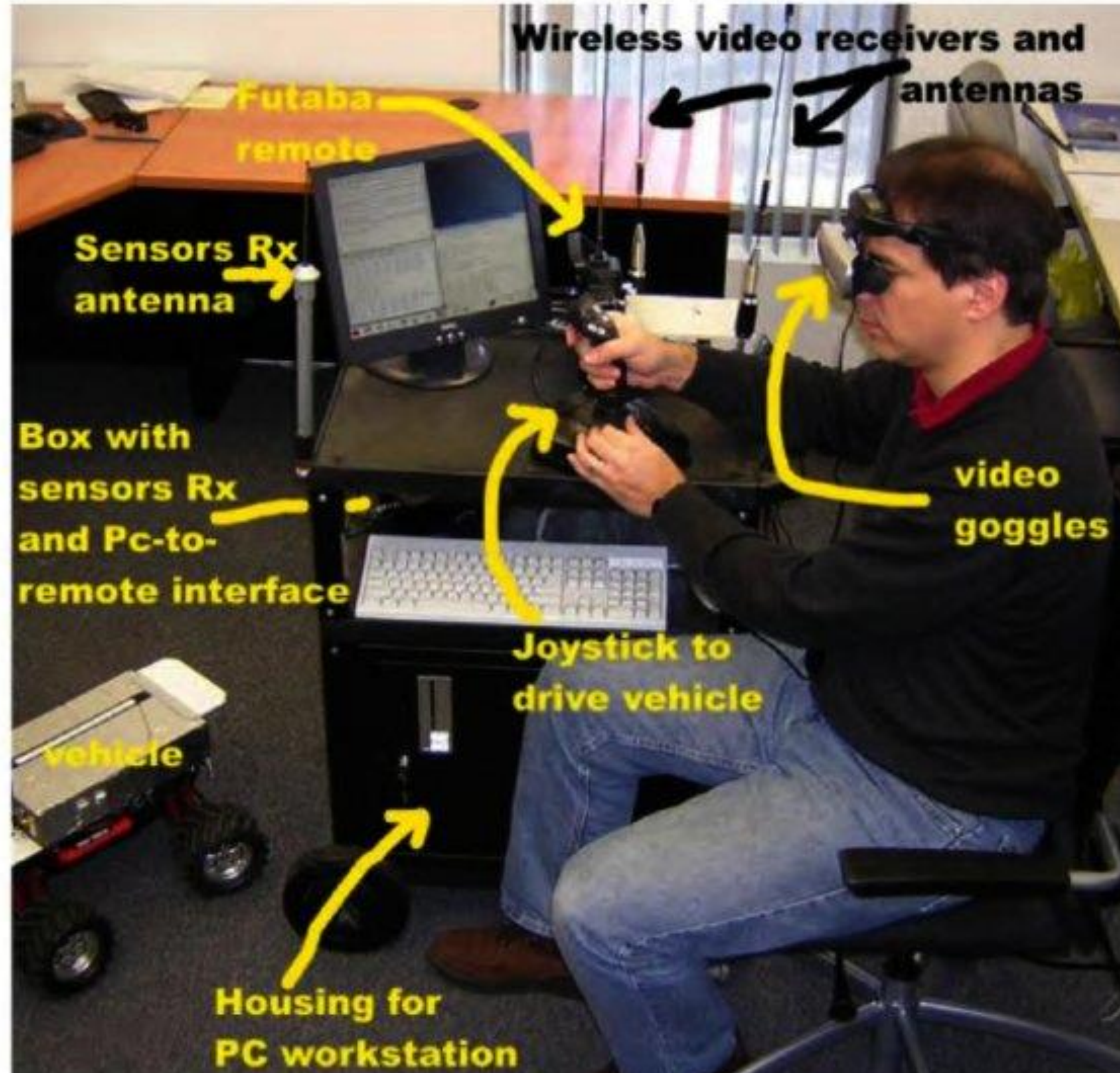
The Deep Learning Revolution



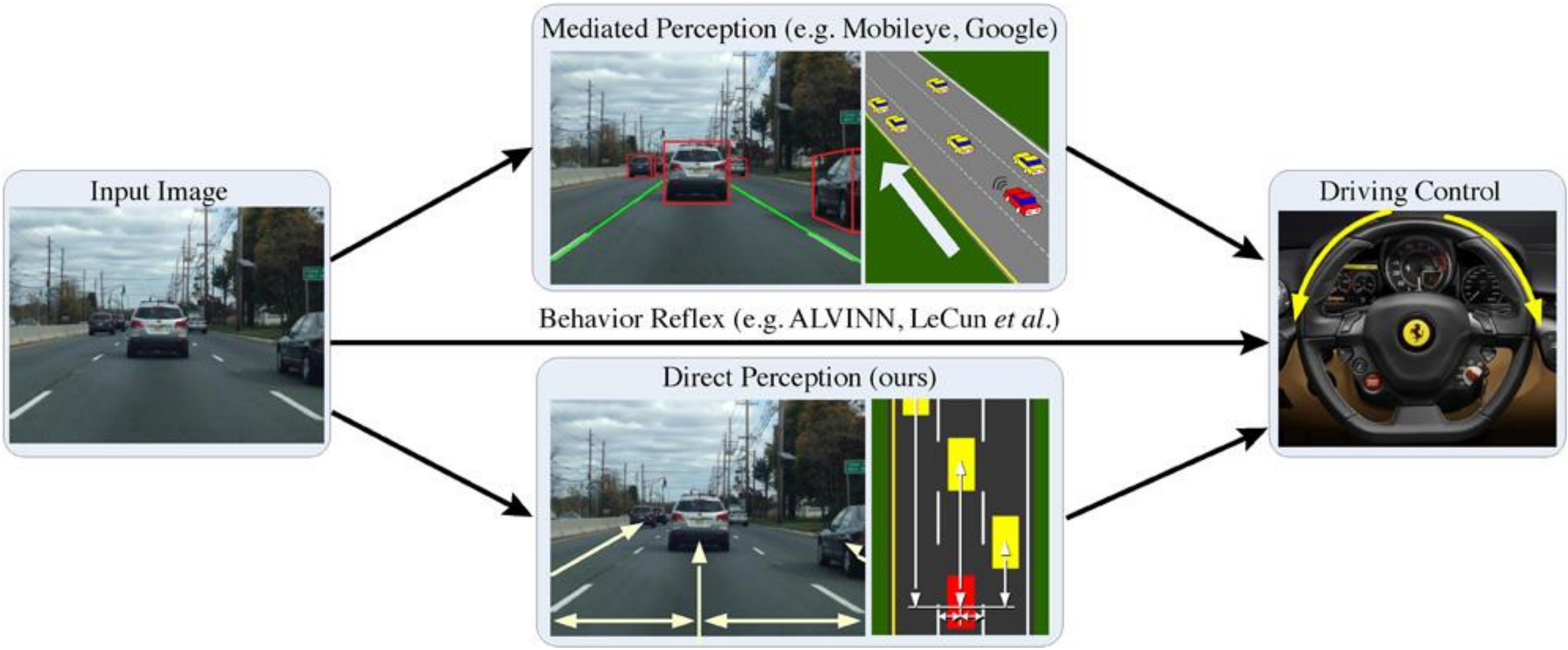
Step 1: Convolutional (Neural) Networks

DAVE [LeCun et al. 2005]

http://www.cs.nyu.edu/~yann/research/dave/



Regression learning for driving



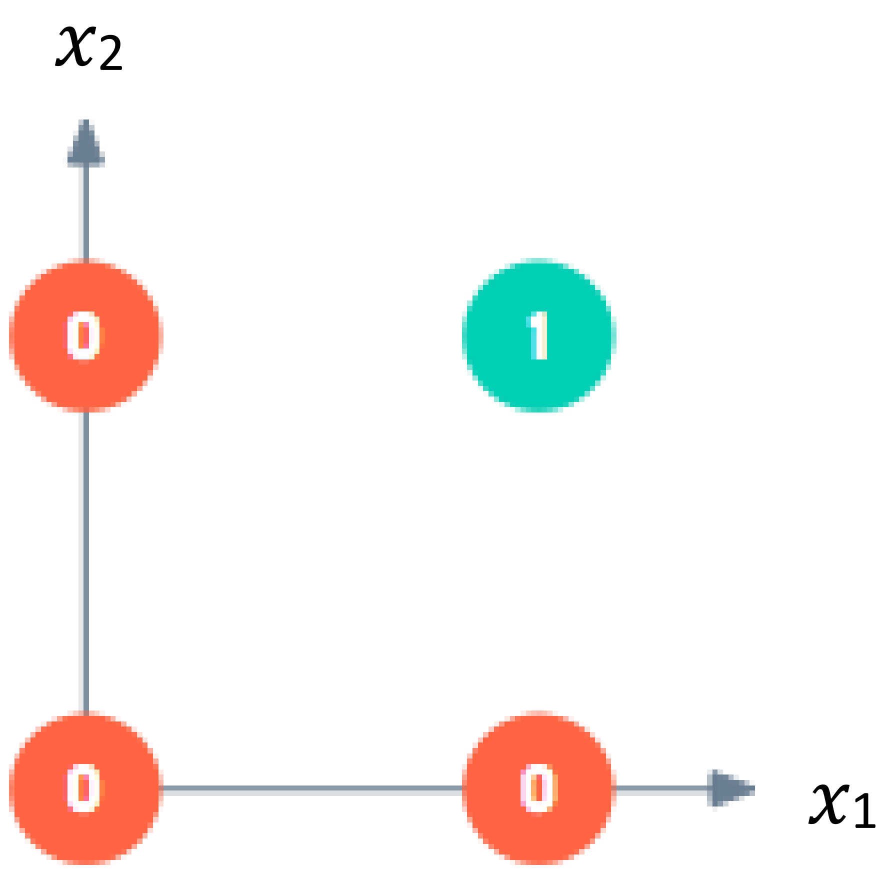
http://deepdriving.cs.princeton.edu/

qHebb driving [Öfjäll et al. 2014]

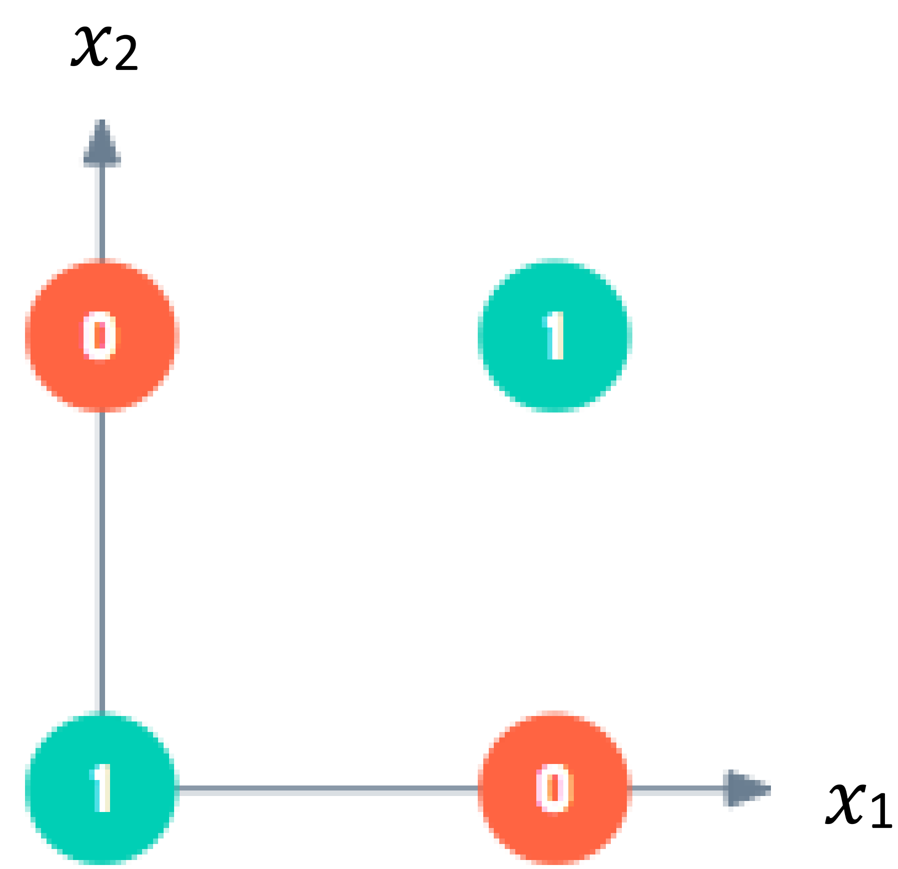
Feedforward networks

LINKÖPINGS UNIVERSITET

Linear separability

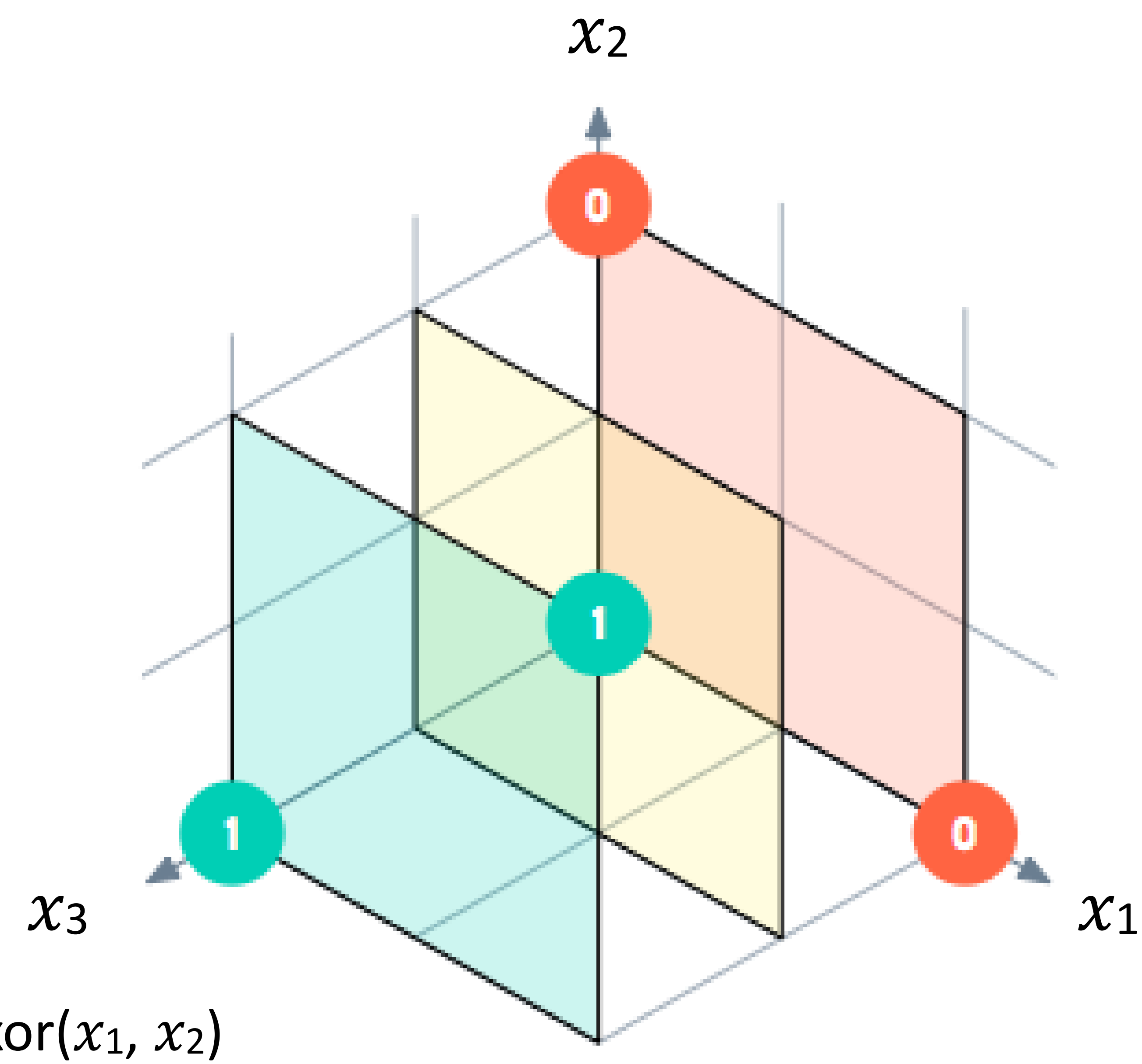


linearly separable



not linearly separable

New features to the rescue!



x_3 = xor(x_1, x_2)

How do we get new features?

We want to apply the linear model not to $\boldsymbol{x}$ directly but to a representation $\phi(\boldsymbol{x})$ of $\boldsymbol{x}$. How do we get this representation?

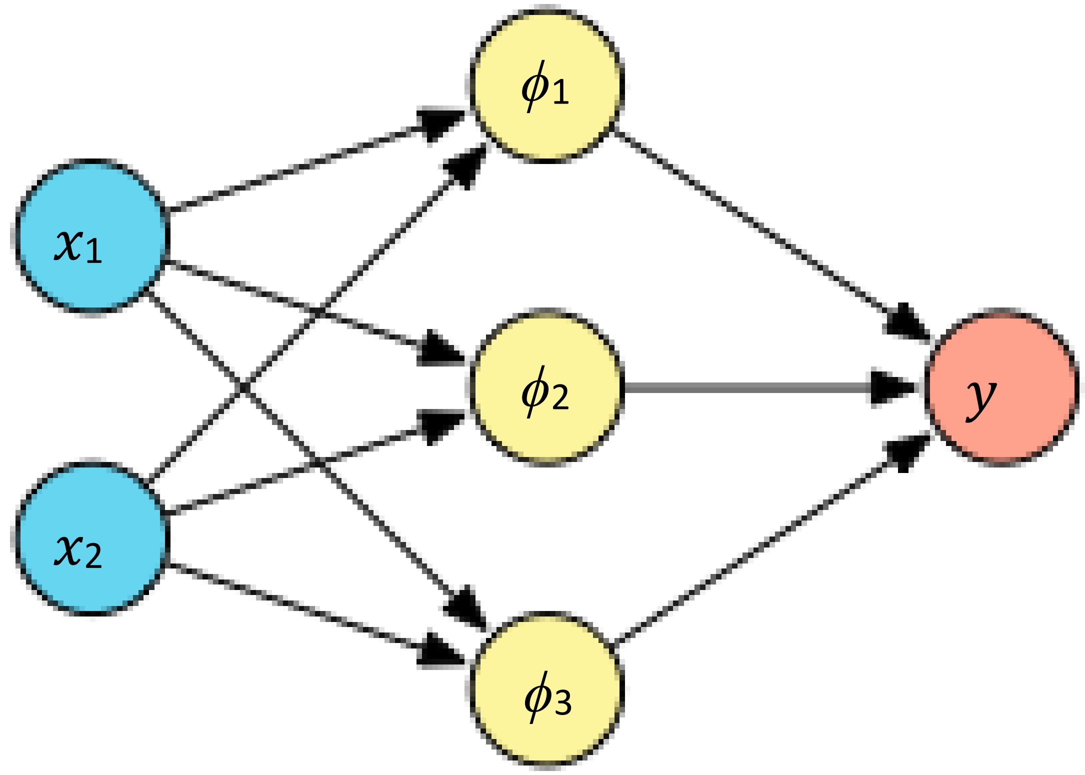
- Option 1. Manually engineer ϕ using expert knowledge.

  feature engineering

- Option 2. Make the model sensitive to parameters such that learning these parameters identifies a good representation ϕ.

  feature learning

Shapes of the parameter matrices



input layer　　hidden layer　　output layer

$\bar{\boldsymbol{\theta}}$: (2, 3)　　$\boldsymbol{\theta}$: (3, 1)

Convolutional networks

LINKÖPINGS UNIVERSITET

Apply networks to images

- What happens with $\boldsymbol{\theta}$ for image-sized input ?
- What happens with $\boldsymbol{\theta}$ and $\bar{\boldsymbol{\theta}}$ for about same order of magnitude hidden units?
- Computational effort
- Overfitting

$$\mathbf{z} = \boldsymbol{\theta}\mathbf{x} =$$

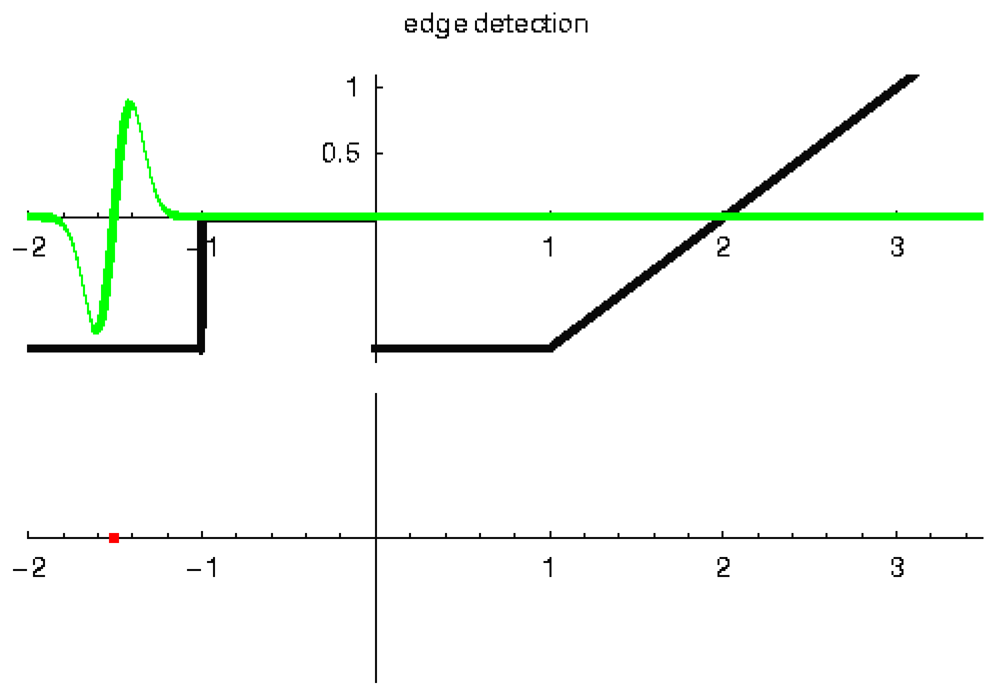
$$\begin{bmatrix} \theta_{1,1} & \theta_{1,2} & \theta_{1,3} & \dots & \theta_{1,n} \\ \theta_{2,1} & \theta_{2,2} & \theta_{2,3} & \dots & \theta_{2,n} \\ \theta_{3,1} & \theta_{3,2} & \theta_{3,3} & \dots & \theta_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{m,1} & \theta_{m,2} & \theta_{m,3} & \dots & \theta_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Convolutional (neural) networks

- CNN [LeCun, 1989]
- suitable for data with known, grid-like topology
  - Time series
  - Images “tensors”
  - Medical data
- “Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.”

Convolution

- Separate lesson
- CNNs use correlation
- flipping irrelevant for learned coefficients
- 1D convolution: Toeplitz matrix
- Circulant if periodic boundary conditions
- Often: sparse



http://bmia.bmt.tue.nl/education/courses/fev/course/notebooks/Convolution.html

Black:	input
Green:	kernel
Red:	output

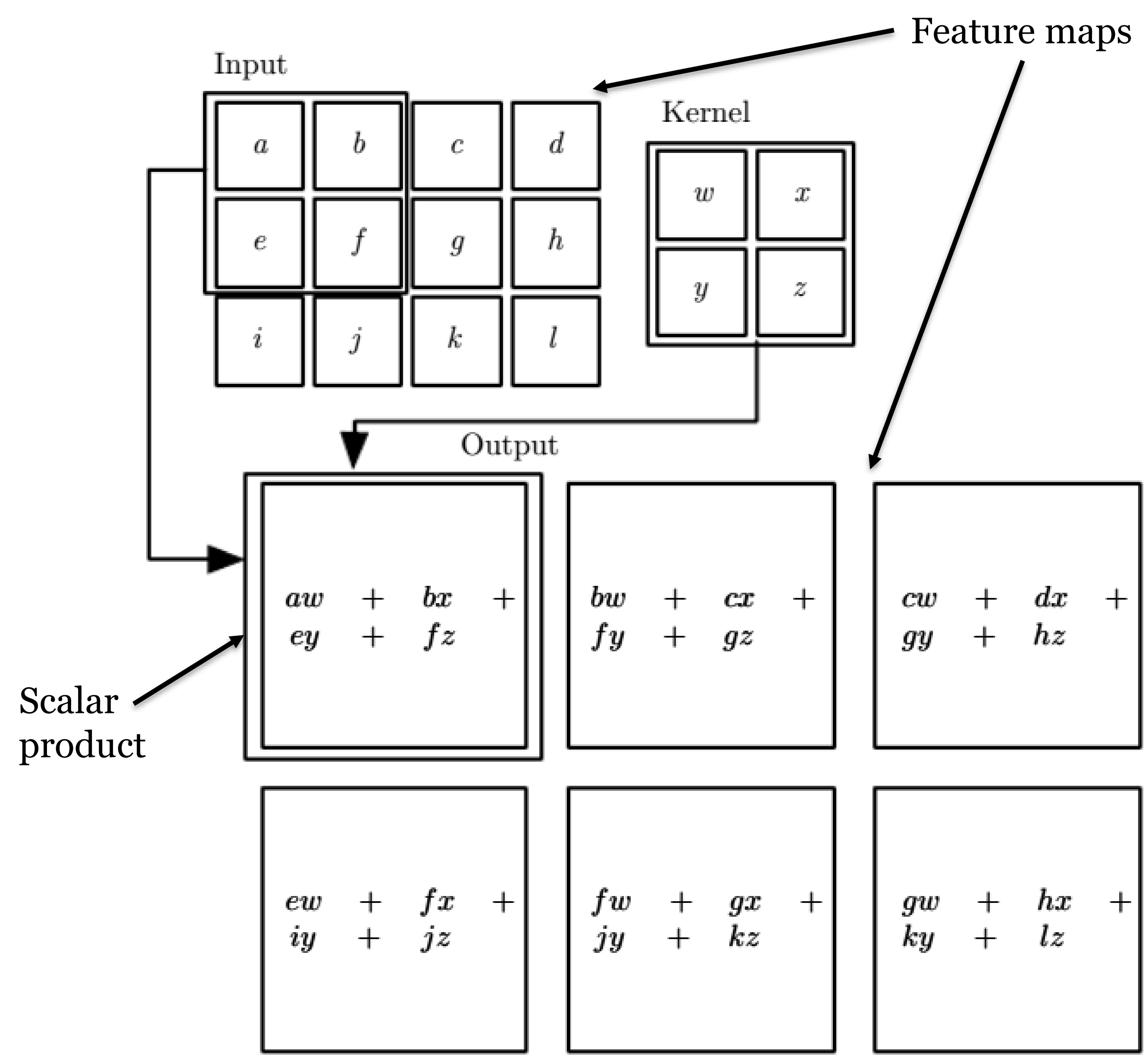
Convolution

- Separate lesson
- CNNs use correlation
- flipping irrelevant for learned coefficients
- 1D convolution: Toeplitz matrix
- Circulant if periodic boundary conditions
- Often: sparse

$$z = \theta * x = \begin{bmatrix} \theta_0 & \theta_{-1} & \theta_{-2} & 0 & 0 & \dots & 0 \\ \theta_1 & \theta_0 & \theta_{-1} & \theta_{-2} & 0 & \ddots & 0 \\ \theta_2 & \theta_1 & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \theta_2 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \theta_{-2} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \theta_{-1} \\ 0 & 0 & \dots & 0 & \theta_2 & \theta_1 & \theta_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

2D Convolution - algorithmic

- images become feature maps
- local operation
- boundary conditions (valid, reflective, periodic, zeros)
- doubly block circulant doubly block Toeplitz
- sparse



http://www.deeplearningbook.org/

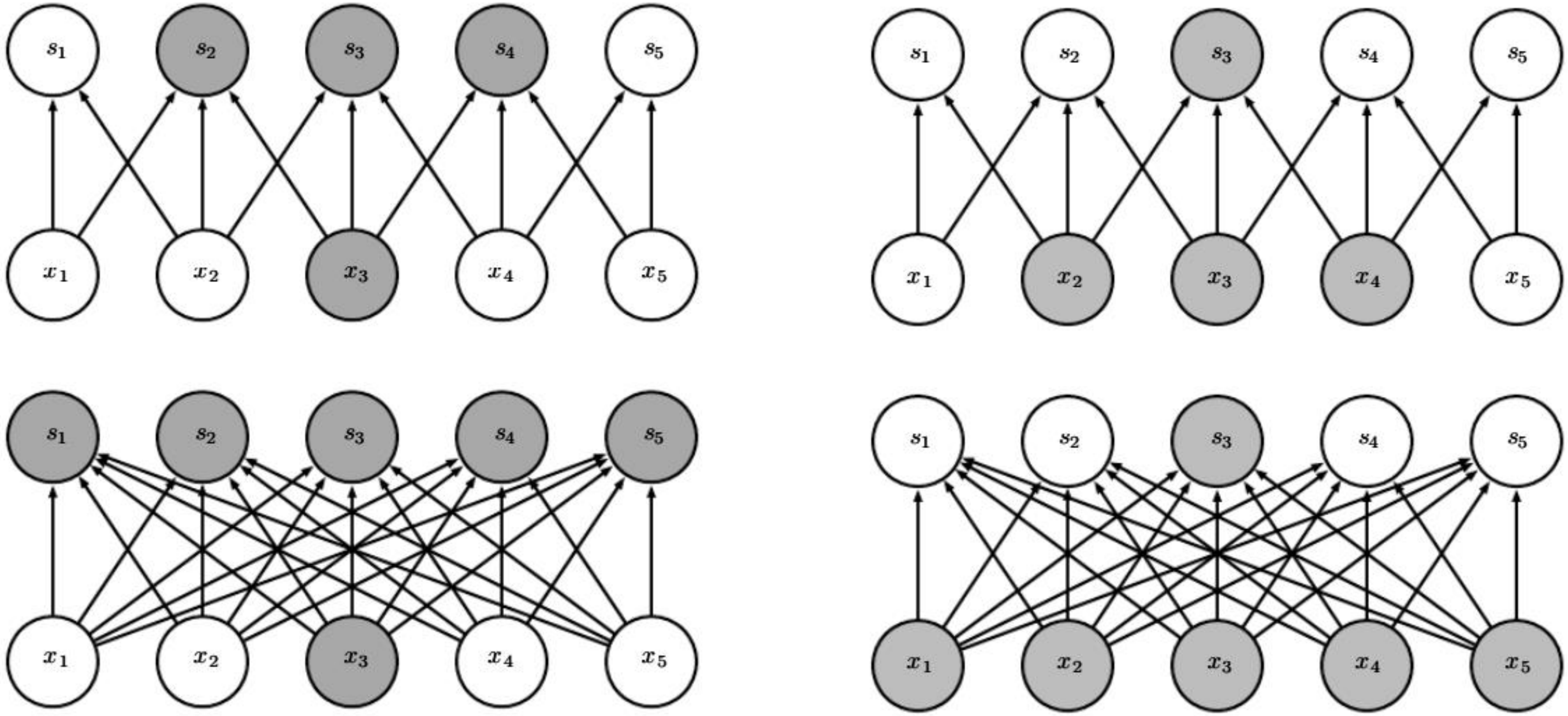
Convolutional Neural Networks

Motivation CNNs

1. sparse (and local) interaction
2. parameter sharing
3. equivariant representations

Sparse (and local) interaction

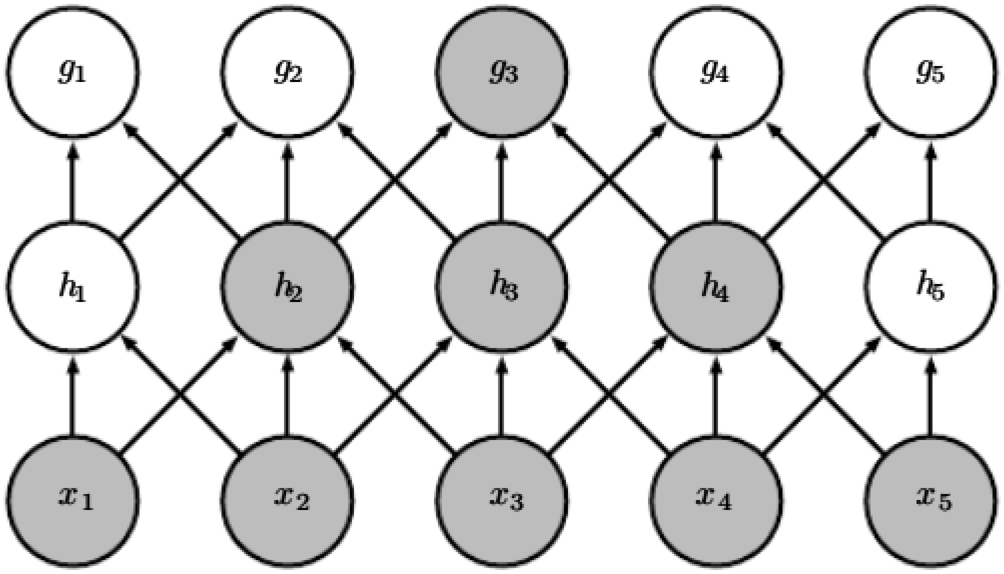
- kernel smaller than the input (topology, s=z/y)



http://www.deeplearningbook.org/

Sparse (and local) interaction

- kernel smaller than the input (topology, $h/g=z/y$)
  - fewer parameters
  - lower memory requirements
  - better statistical efficiency
  - fewer operations

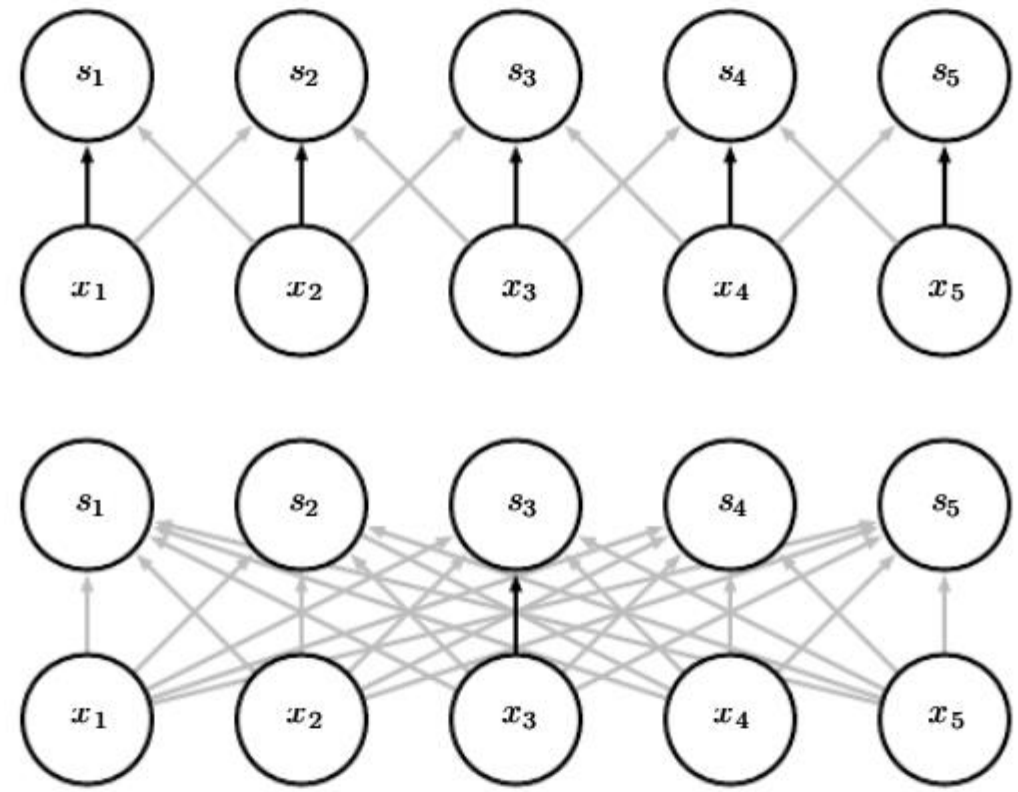


http://www.deeplearningbook.org/

- by increased depth indirectly connected to all input

Parameter sharing

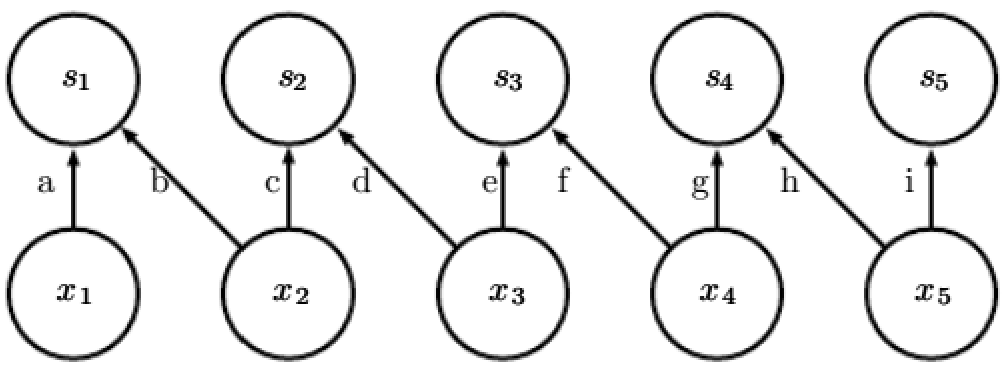
- tied weights (topology, $s=z/y$)
- reduced storage requirements
- but same time complexity
- sometimes sharing should be limited, e.g. cropped images



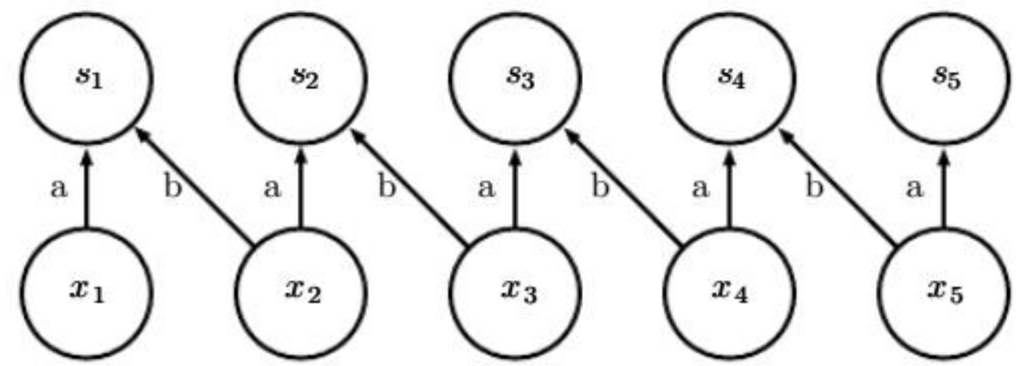
http://www.deeplearningbook.org/

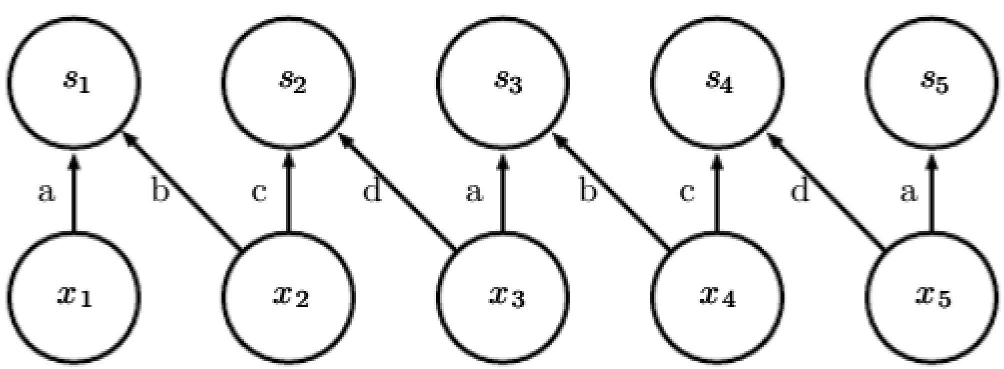
Overview of options / convolution

local connections unshared



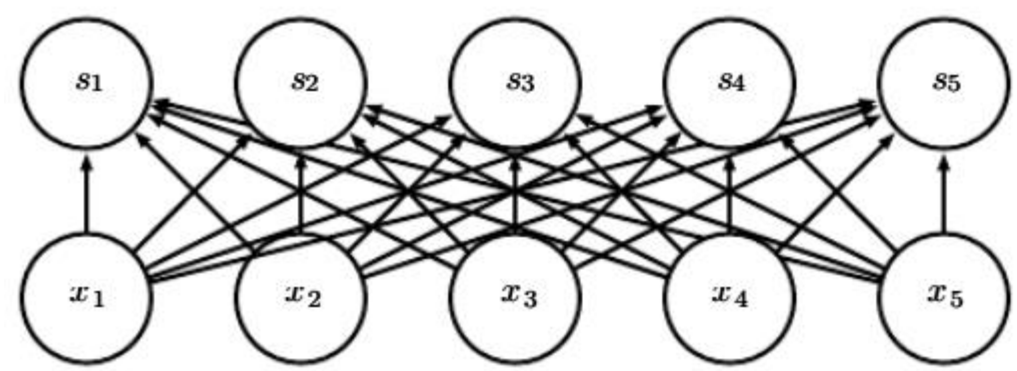
local connections shared





local connections tiled

(topology, s=z/y)



full connections

http://www.deeplearningbook.org/

Network Layers

LINKÖPINGS UNIVERSITET

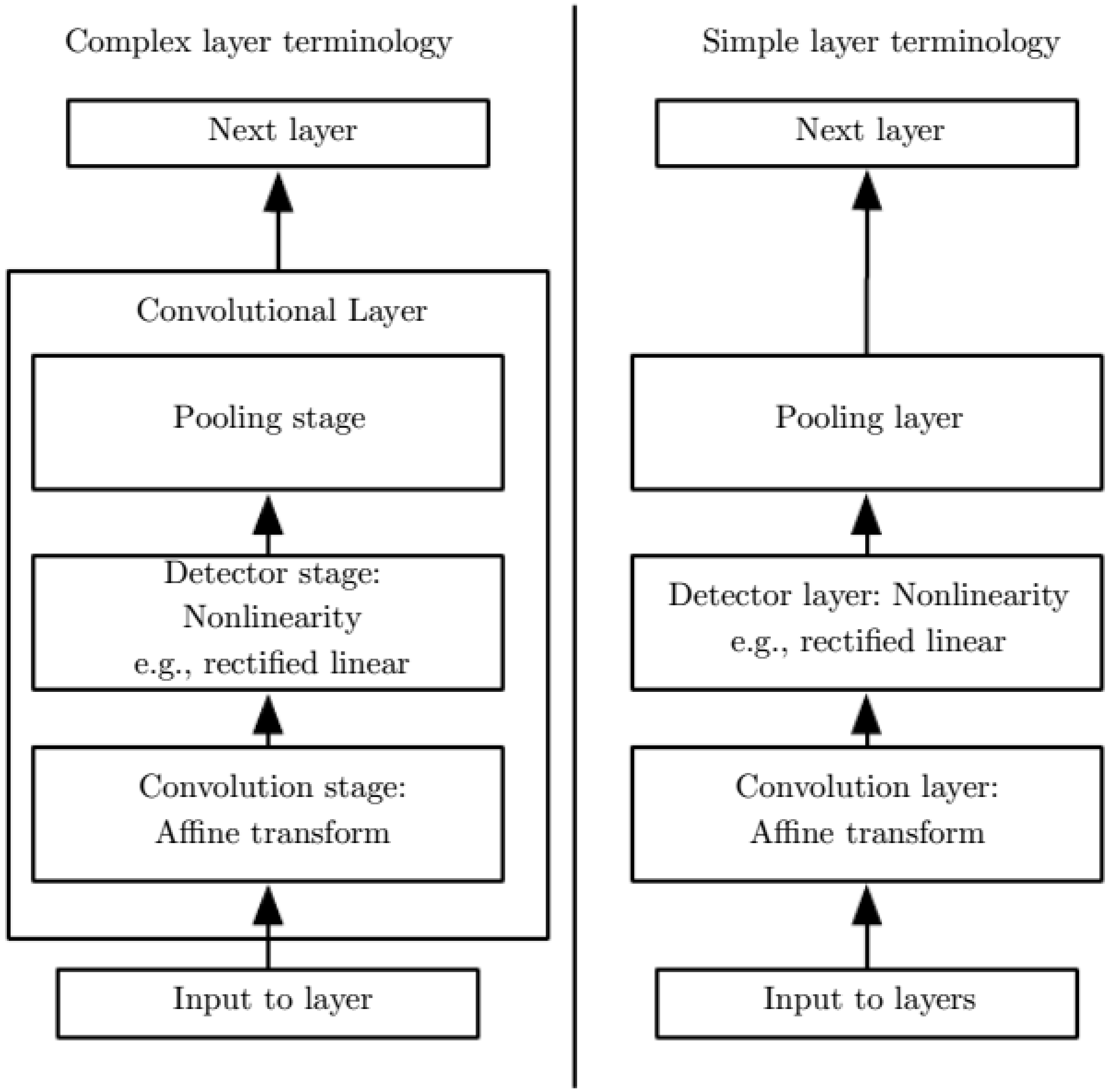
Bias terms

- Locally connected unshared – each unit own bias
- Tiled convolution – share biases in tiling pattern
- Shared convolution
  - share bias
  - separate bias at each location

  compensate differences in the image statistics

Layers in CNNs

- each layer consists of three stages:
    1. convolutions to compute linear combination z
    2. detector stage to compute activation y
    3. pooling function



http://www.deeplearningbook.org/

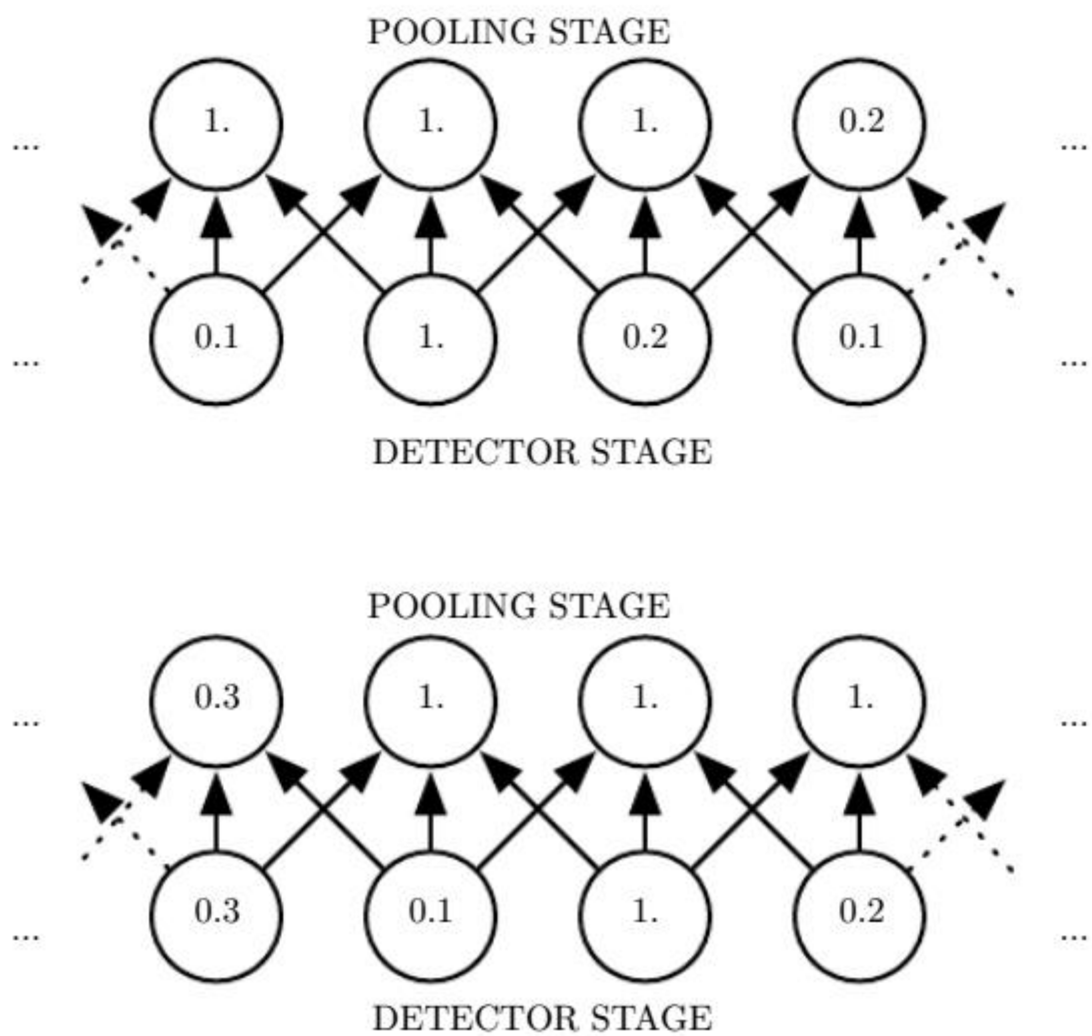
Pooling

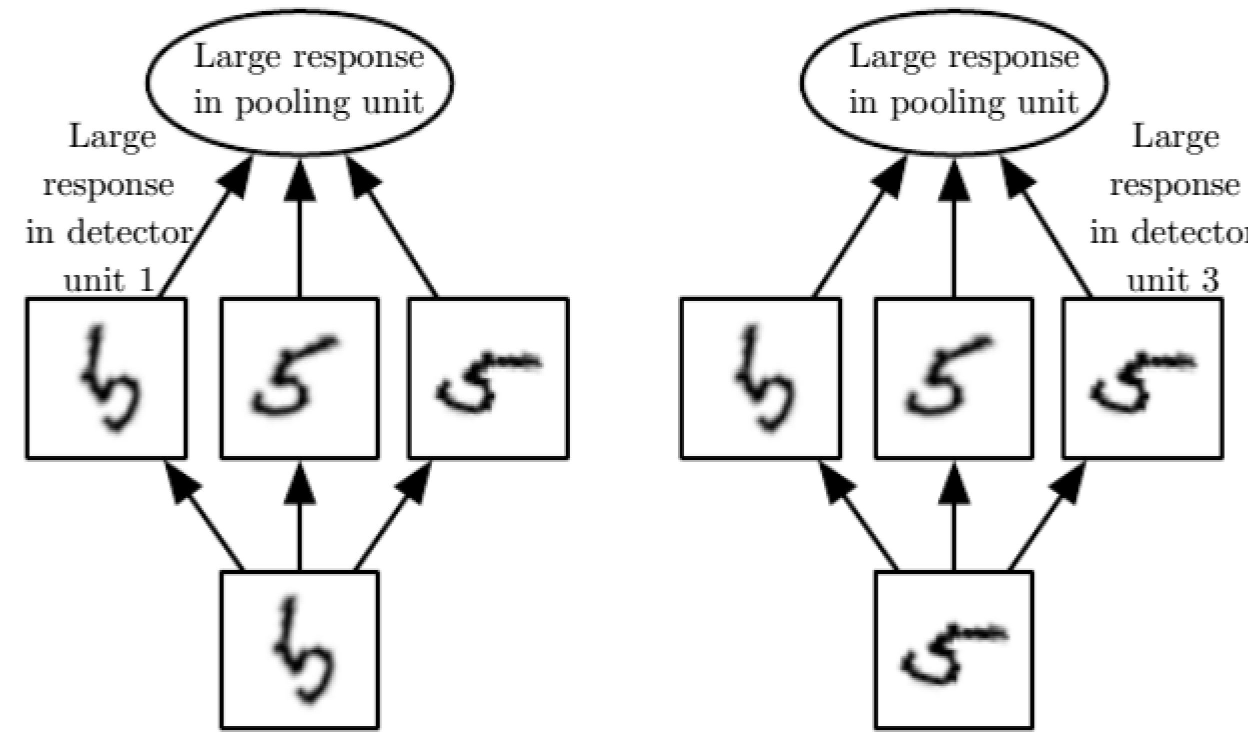
- summary statistics of nearby outputs
  - max pooling [Zhou&Chellappa, 1988]
    maximum output in rectangular region
  - average in rectangular region
  - L2 norm of rectangular region
  - weighted average
    (based on distance from central position)
- approximately invariant to small translations

Invariance and Equivariance

- a function f is invariant (under operation g) if
  - applying g to the input of f does not change its output
  - different inputs (modulo g) have different outputs
- a function f is equivariant (under operation g) if
  - applying g to the input of f changes its output by $\tilde{g}$
  - different inputs have different outputs
- easy for discrete shift operations
- more tricky for rotation and scaling

Pooling and invariance





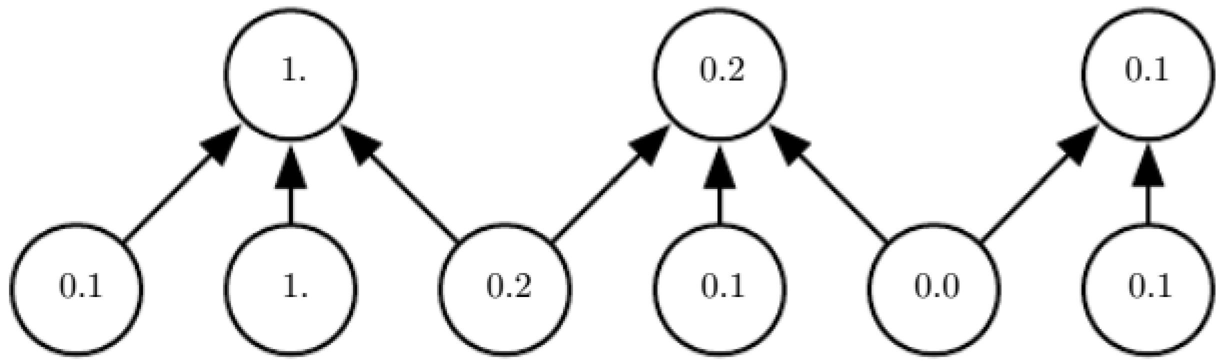
http://www.deeplearningbook.org/

Striding and Activation Functions

LINKÖPINGS UNIVERSITET

Strides

- pooling *s* pixels apart instead of every pixel (stride *s*)

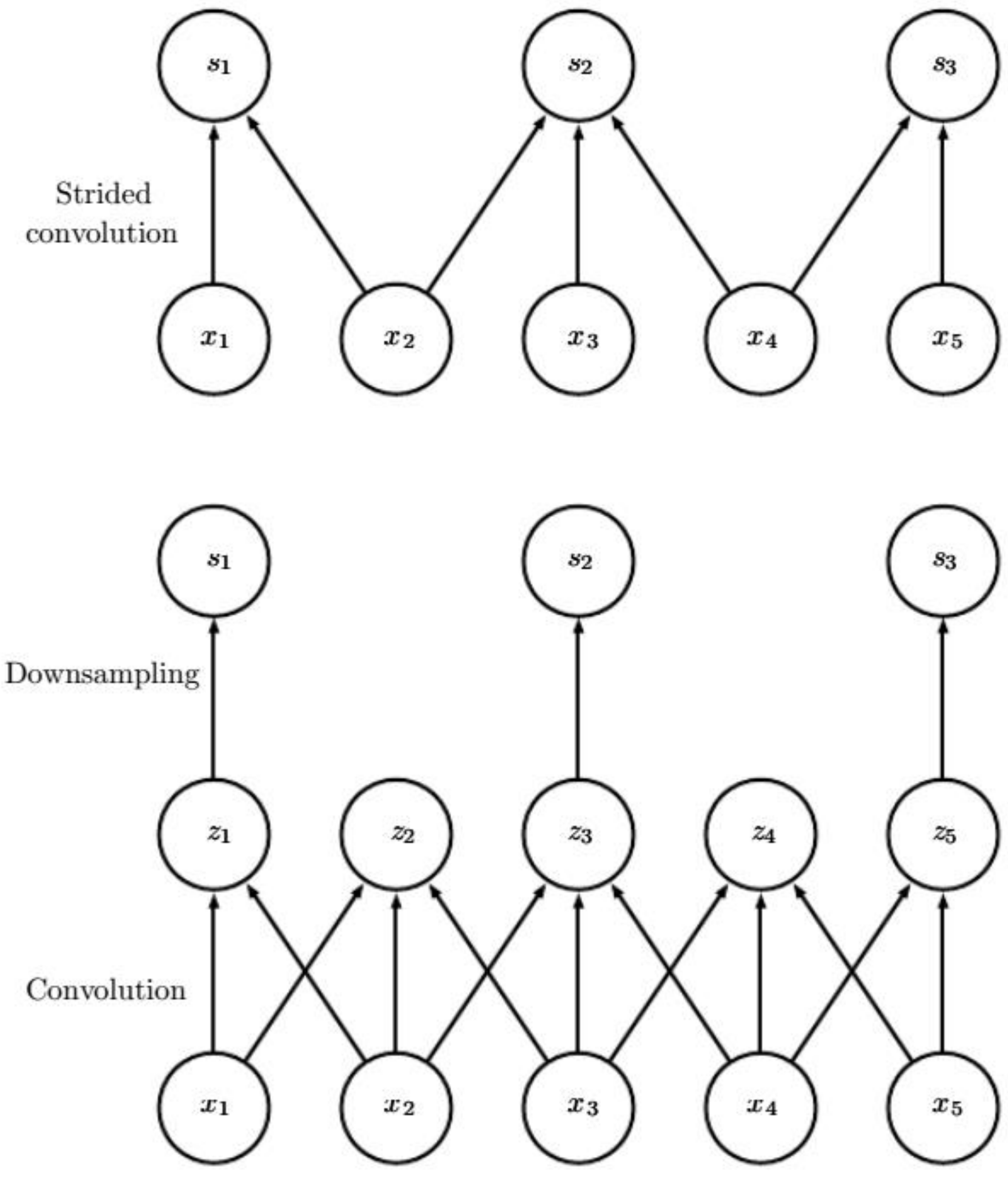


http://www.deeplearningbook.org/

  - improved statistical efficiency
  - reduced memory requirements
  - handling inputs of varying size
  - but: pooling & strides complicate top-down processing (e.g. autoencoders)

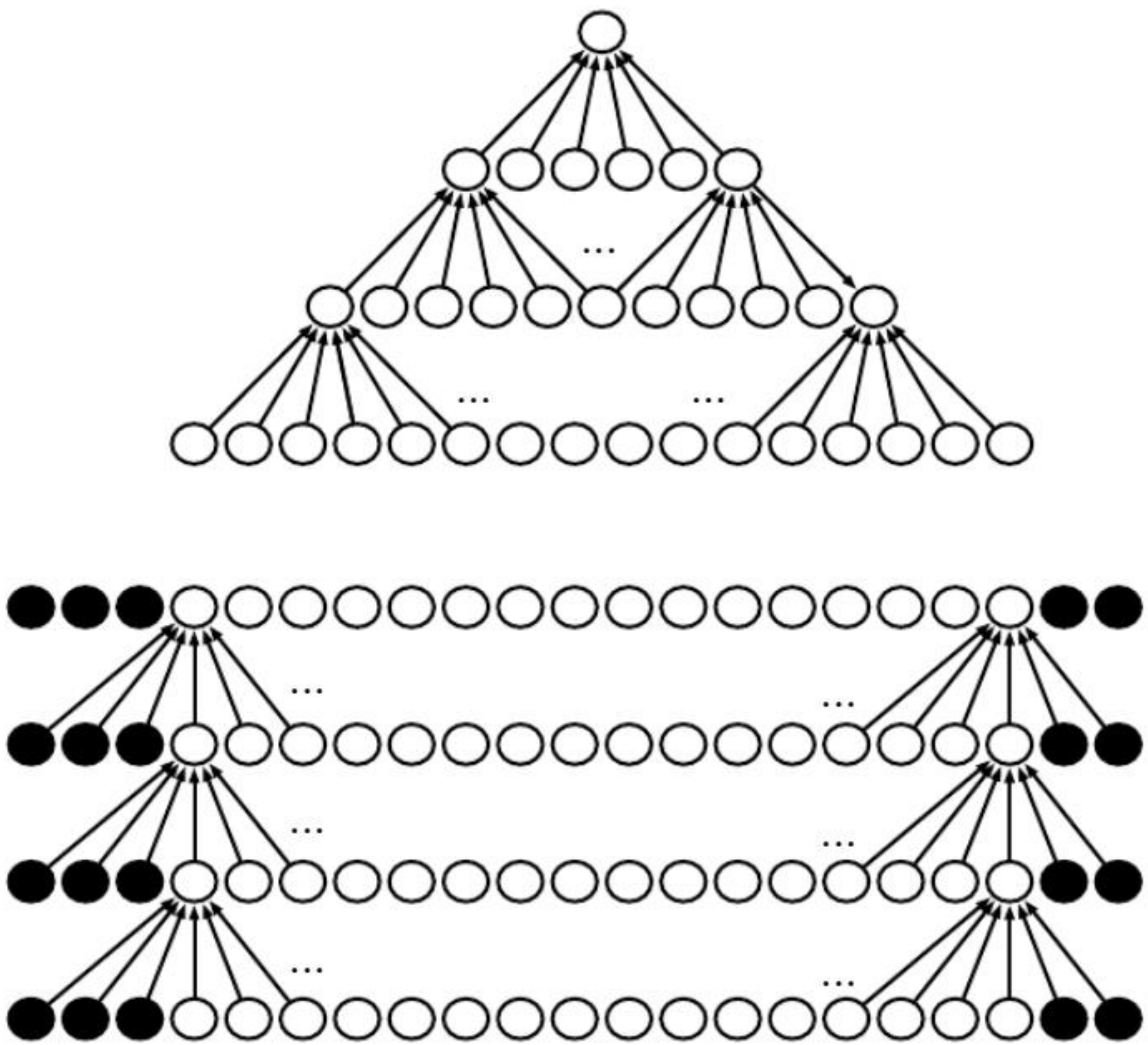
Strided Convolution

Stride vs. sequential convolution and downsampling
(cf. filterbanks/ wavelets; topology, s=z/y)



http://www.deeplearningbook.org/

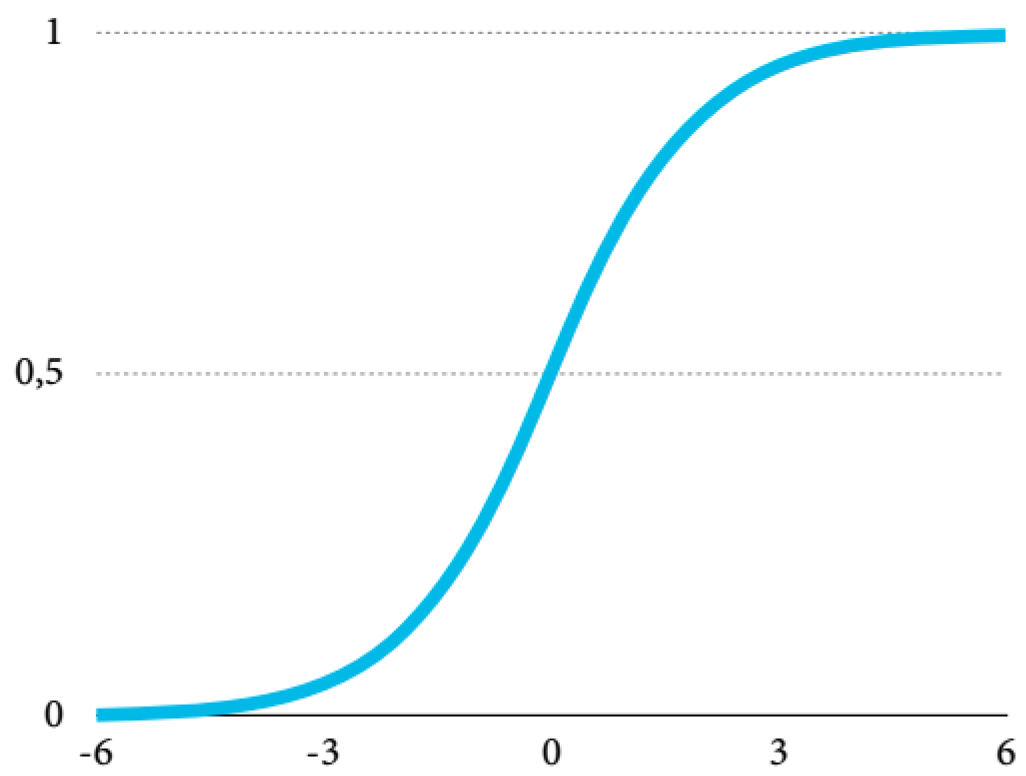
Zero-padding

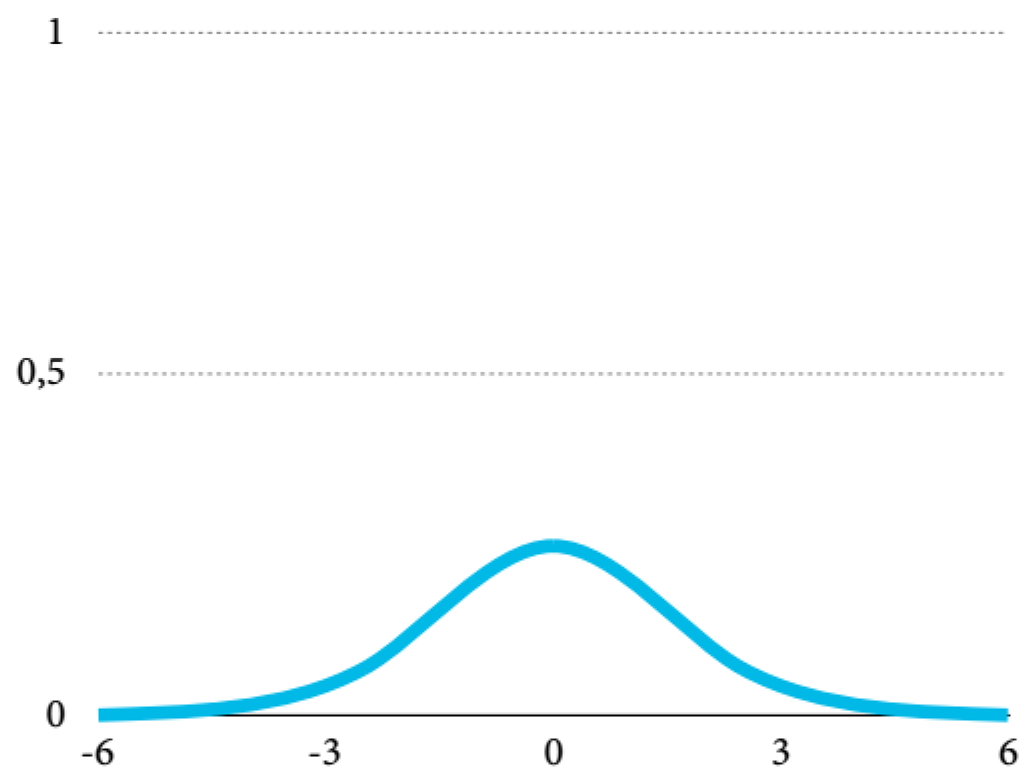


http://www.deeplearningbook.org/

Logistic function

$$z = \boldsymbol{x}^\top \boldsymbol{\theta}$$



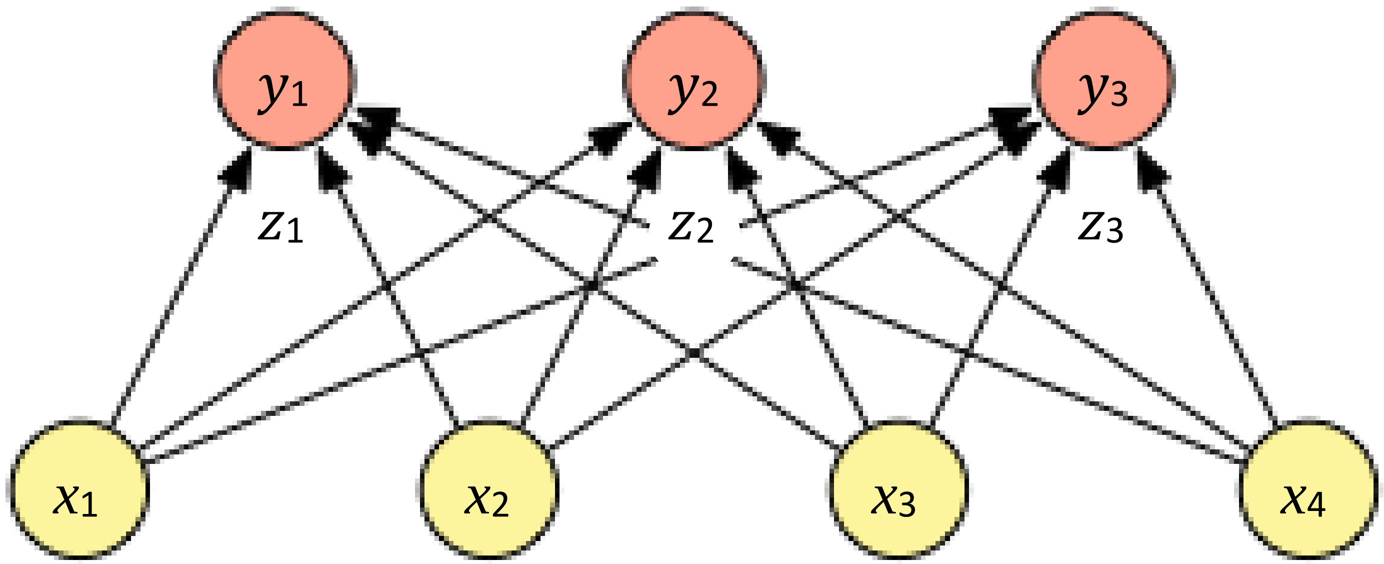


$$f(z) = \frac{1}{1 + e^{-z}}$$

$$\frac{\partial f}{\partial z} = f(z)(1 - f(z))$$

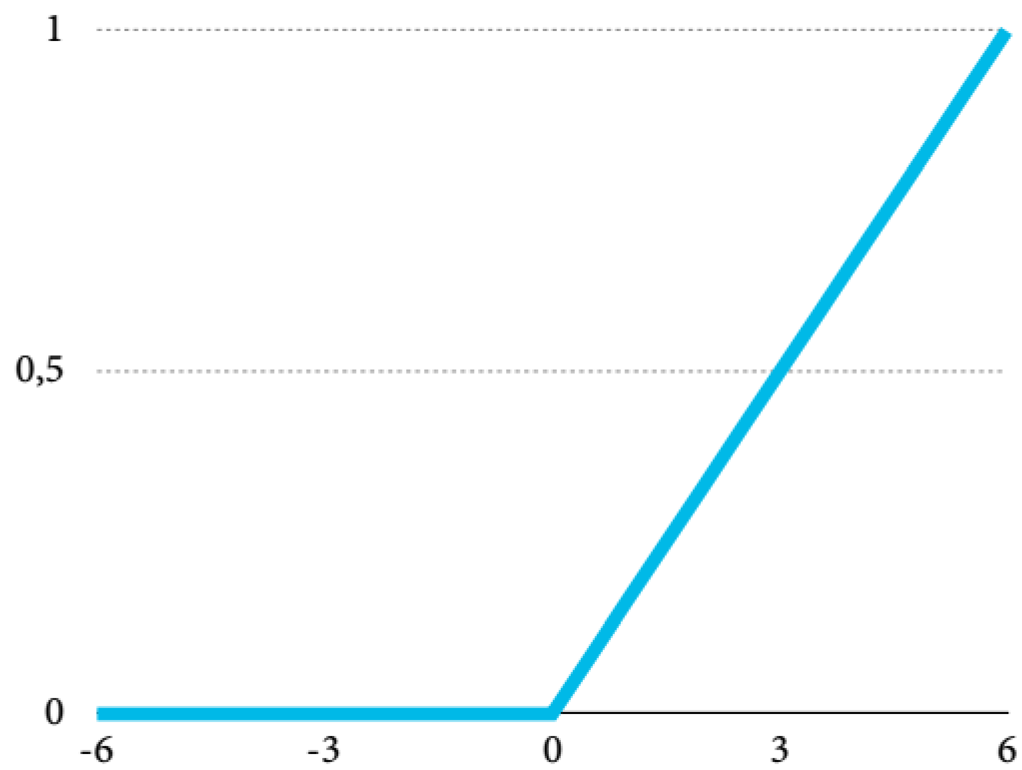
Softmax layer

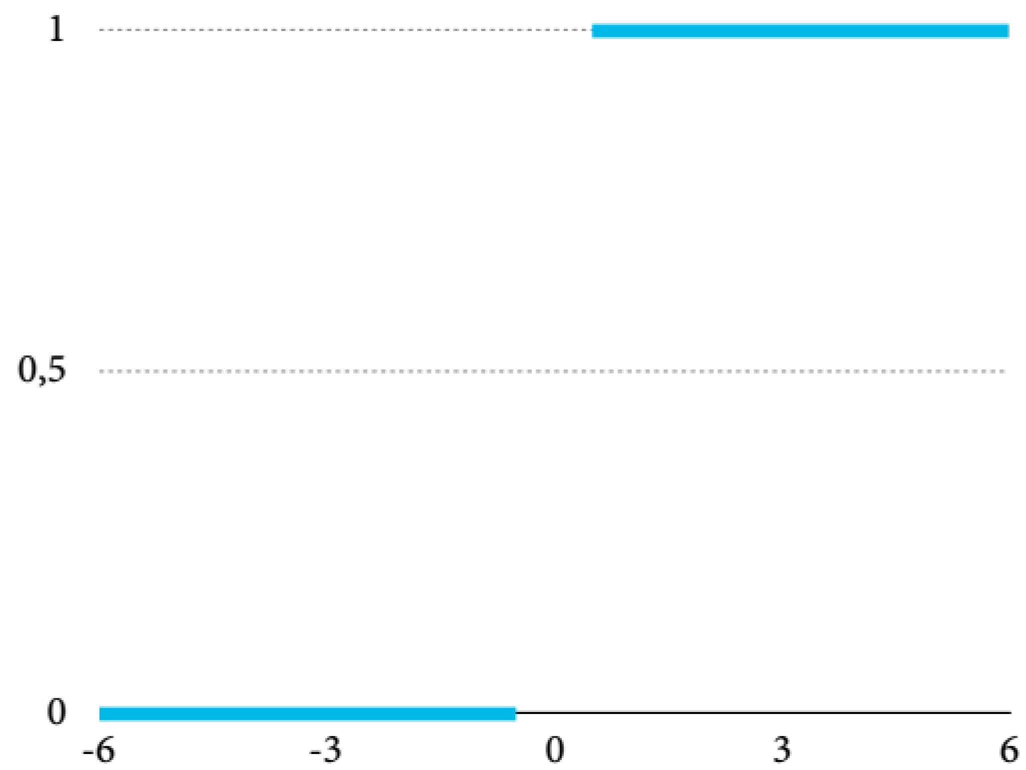
$$y_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$



$$\frac{\partial y_i}{\partial z_j} = \begin{cases} y_i(1 - y_i) & i = j \\ -y_i y_j & i \neq j \end{cases}$$

Rectified linear units





$$f(z) = \begin{cases} 0 & z \leq 0 \\ z & z > 0 \end{cases}$$

$$\frac{\partial f}{\partial z} = \begin{cases} 0 & z \leq 0 \\ 1 & z > 0 \end{cases}$$

Loss functions

Machine learning = optimization?

- Objective $J(\boldsymbol{\theta})$ is minimized
- Expectation over some loss function L
- Ideal: expectation over *data distribution*
- Hope: empirical data (training set) gives the same parameters (empirical risk minimization)
- *Hypothesis*: test set drawn from the same distribution
- Models with high *capacity* memorize training set (overfitting)
- Optimization: direct minimization of objective

Likelihood

- The expected loss over some distribution is given as

$$J(\boldsymbol{\theta}) = \int L(\phi(\mathbf{x}^T\boldsymbol{\theta}), y)p(\mathbf{x})\,d\mathbf{x}$$

- If the training data is drawn from the distribution p

$$J(\boldsymbol{\theta}) \approx \sum L(\phi(\mathbf{x}^T\boldsymbol{\theta}), y)$$

- On the other hand, we want the parameter that maximizes the probability $P(\boldsymbol{\theta}|\mathbf{x}) \propto P(\mathbf{x}|\boldsymbol{\theta})P(\boldsymbol{\theta})$ but the second (prior) term is often unknown and only the first (likelihood) term is maximized

Maximum likelihood estimation

- Family of probability distributions $P(\boldsymbol{X}; \boldsymbol{\theta})$ that assign a probability to any sequence $\boldsymbol{X}$ of N examples.
- The maximum likelihood estimator for $\boldsymbol{\theta}$ is defined as

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{X}; \boldsymbol{\theta})$$

- Assume that the examples are mutually independent and identically distributed, this can be rewritten as

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} P\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log P\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

- If assuming Gaussian noise in $P()$: sum of squares

Michael Felsberg
michael.felsberg@liu.se

www.liu.se

LINKÖPINGS UNIVERSITET