Machine Learning

NN and Deep Learning

Fabio Vandin

December 22nd, 2023

Neural Networks

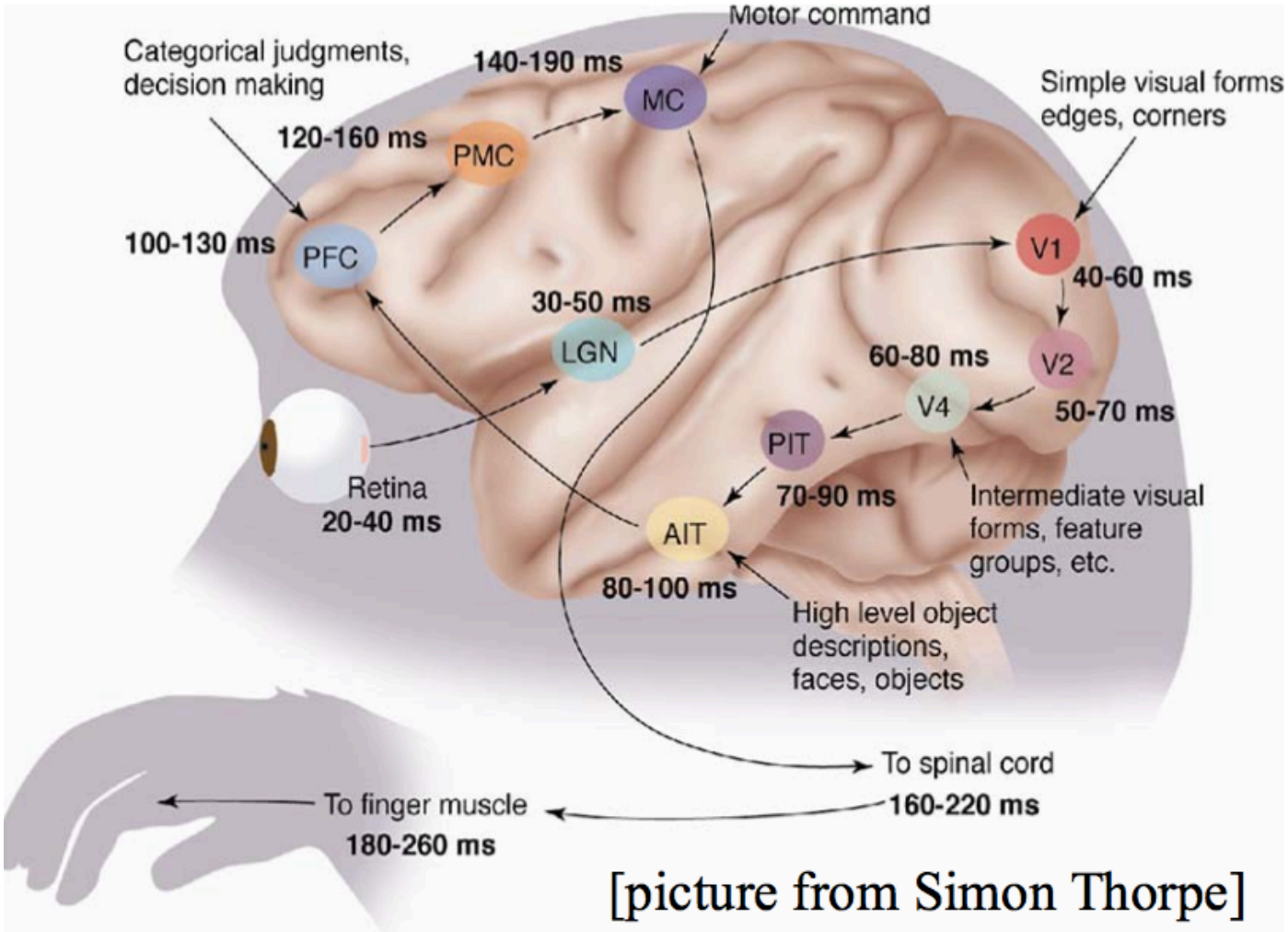
Informal definition: simplified models of the brain

- large number of basic computing units: *neurons*
- connected in a complex network

Deep Learning

Deep Learning = learning hierarchical representations

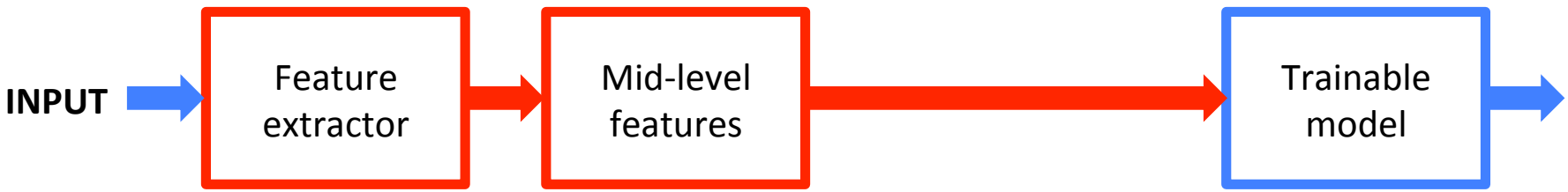
Biological inspiration *e.g. our visual cortex is hierarchical*



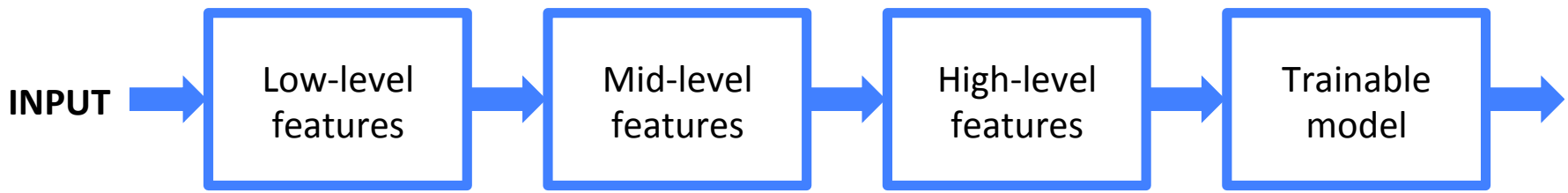
Hierarchical Representations and ML

Traditional Machine Learning: Fixed/Handcrafted Feature Extractor

Advanced Machine Learning: Unsupervised mid-level features



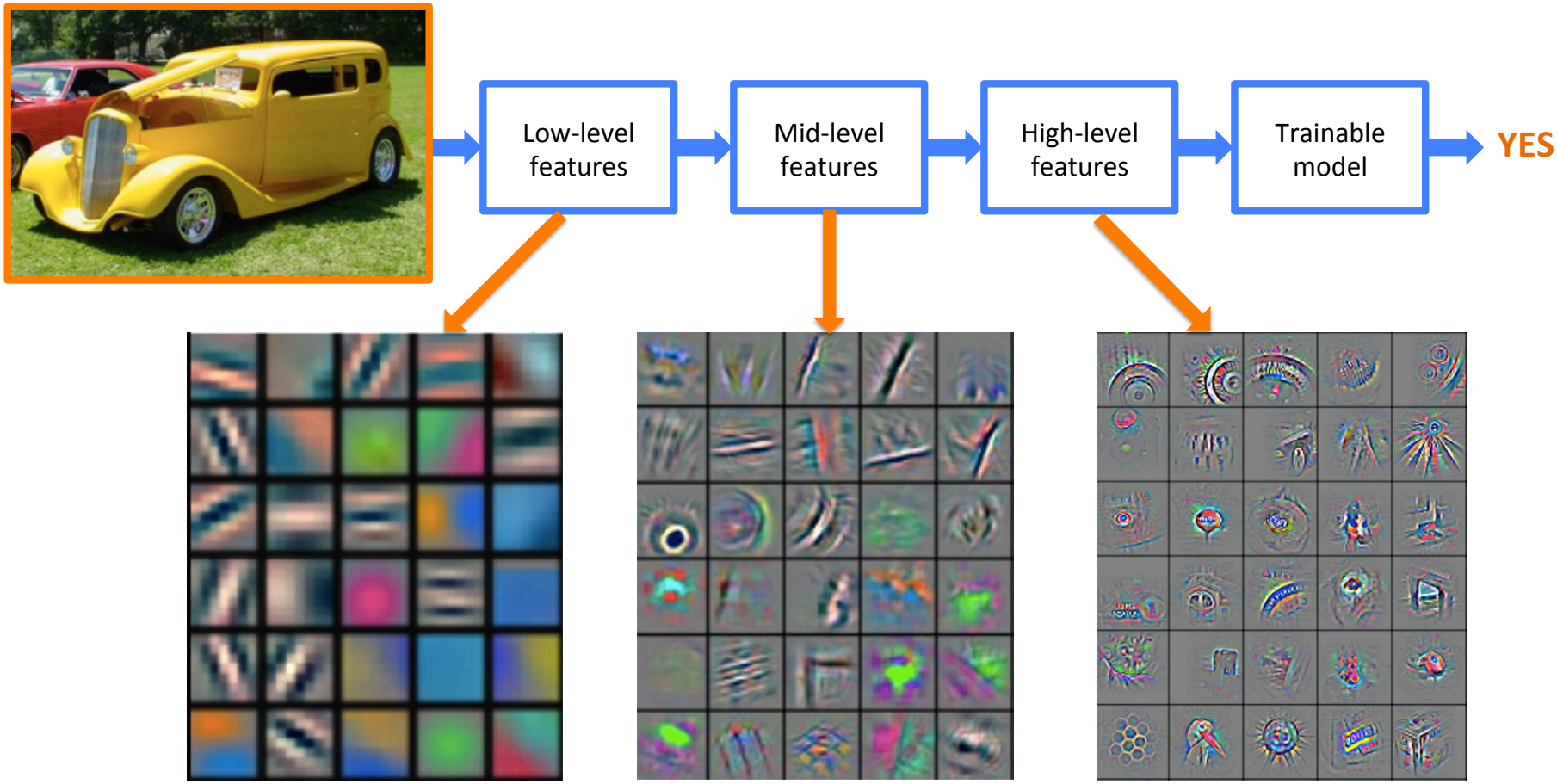
Deep Learning: Representations are hierarchical and trained



Deep Learning $\Rightarrow$ **End-to-end Learning**

Hierarchical Representations and ML (continue)

Example: object recognition (object = car)



Neural Networks: Historical Remarks

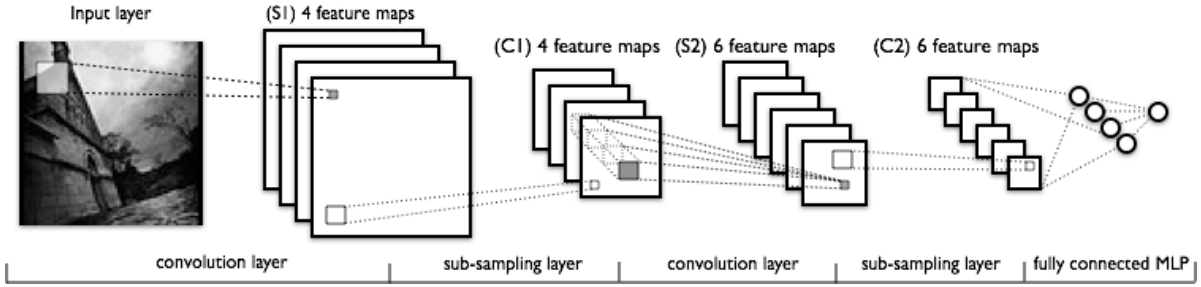
- 1940s-70s:
  - Inspired by learning/modeling the brain (Pitts, Hebb, and others)
  - Perceptron (Rosenblatt), Multilayer perceptron (Minksy and Papert)
  - Backpropagation (Werbos, 1975)
- 1980s-early 1990s:
  - Practical Backpropagation (Rumelhart, Hinton et al., 1986) and SGD (Bottou)
  - Initial empirical success
- 1990s-2000s:
  - Lost favor to implicit linear methods: SVM, Boosting
- 2006-:
  - Regain popularity because of unsupervised pre-training (Hinton, Bengio, LeCun, Ng, and others)
  - Computational advances and several new tricks allow training *huge* networks. Empirical success leads to renewed interest
  - 2012: Krizhevsky, Sustkever, Hinton: significant improvement of state-of-the-art on imagenet dataset (object recognition of 1000 classes), without unsupervised pre-training

Issues with "Traditional NNs"

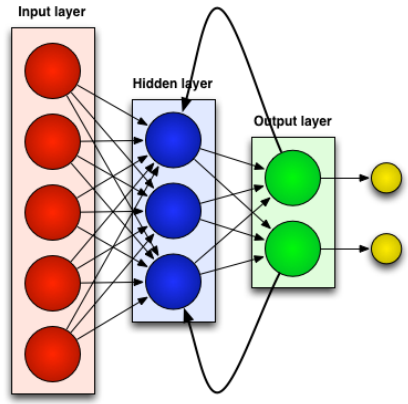
- Huge number of edges $\Rightarrow$ huge number of weights (due to *full connectivity*)
- Domain structure is not taken into account
  - in some domains (e.g., images, text) there is a lot of *structure* $\Rightarrow$ parts of the input have specific relations!
  - what about having *similar/related* neurons?

Modern Neural Networks

- Convolutional Neural Networks (CNN): used mostly for computer vision, imaging, etc.



- Recurrent Neural Networks (RNN): used mostly for time series analysis, speech recognition, machine translation, etc.



- ...

We will see an overview of CNNs

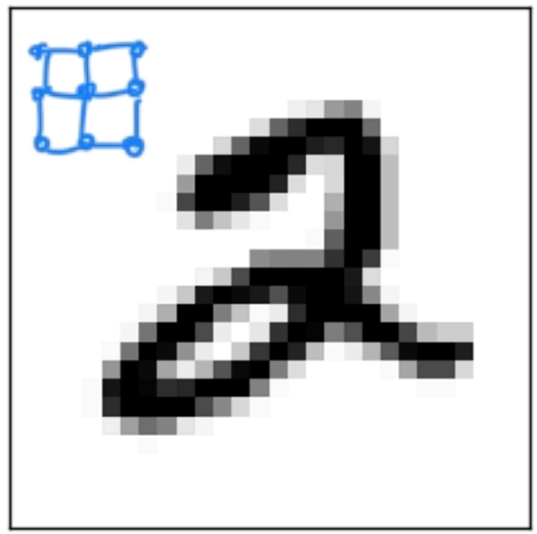
Convolutional Neural Networks (CNNs)

Convolutional NNs: employ the operation of *convolution*

Definition. Convolutional networks are neural networks that use convolution in place of general matrix multiplication in at least one of their layers

CNNs are a specialized kind of neural network for processing data that has a known grid-like topology

Example



Convolution in CNNs

Assume we have a two-dimensional input I and a two-dimensional function K, the (discrete) convolution S of I and K is:

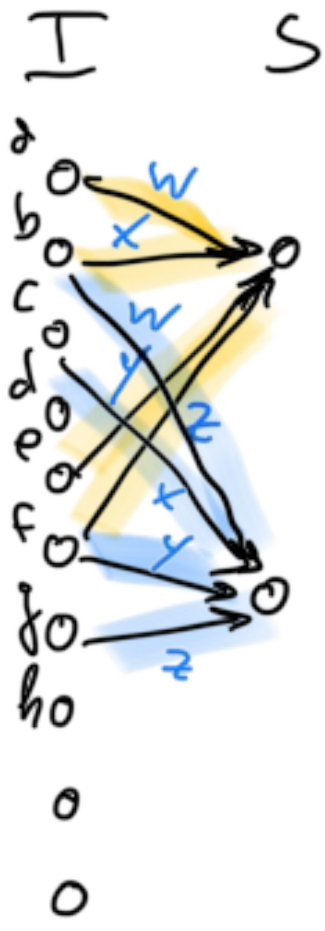
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

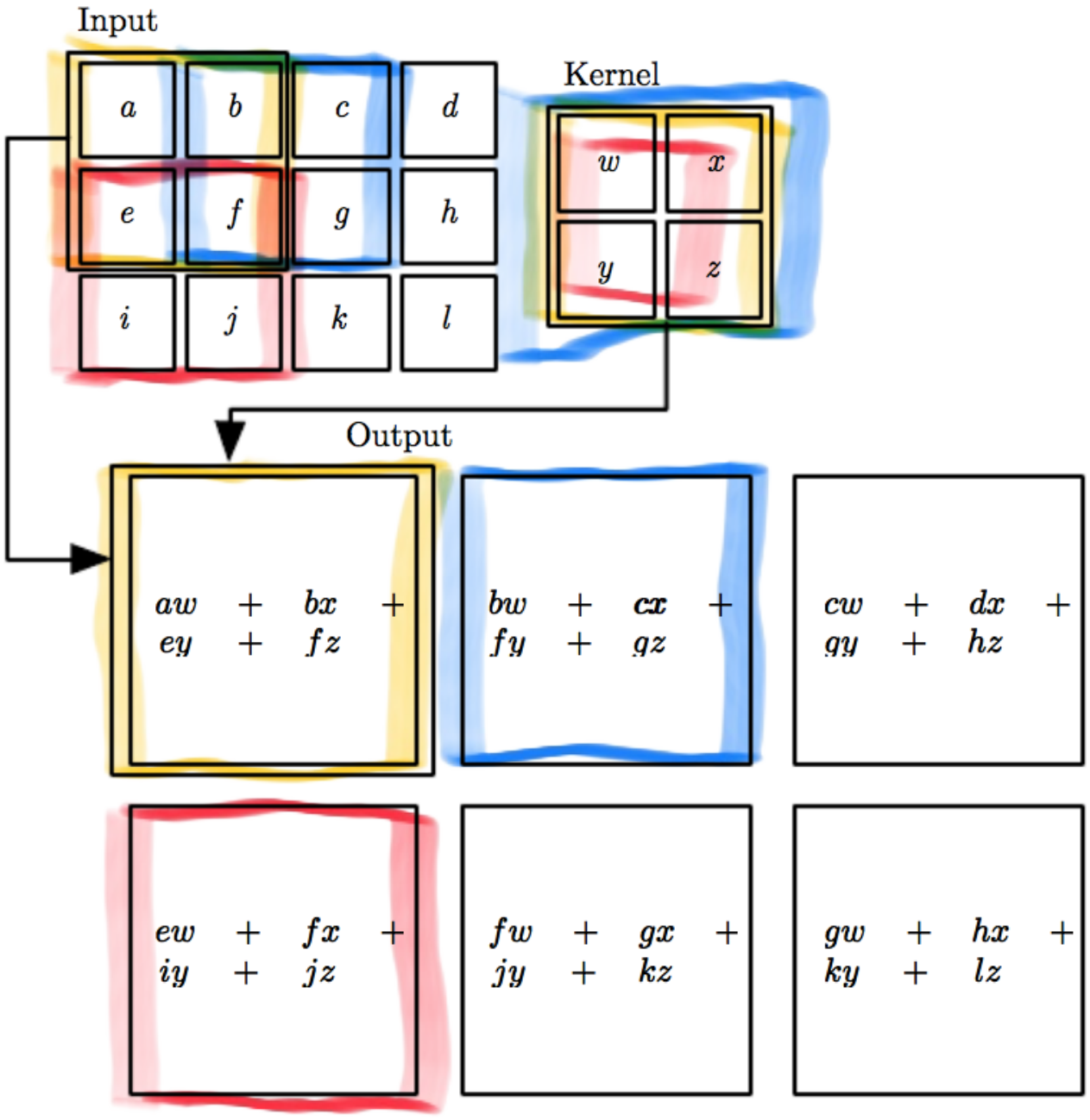
Discrete convolution can be viewed as multiplication by a matrix, but the matrix has several entries constrained to be equal to other entries

Note: K is usually called a *kernel*

In CNNs $K(m,n)$ is zero for all but small values of m, n

Convolution in CNNs (continue)





Convolution: Example

INPUT

OUTPUT

KERNEL

	0	0	0	
	-1	1	0	
	0	0	0	

Filter: *edge enhance*

Note: the kernel is learned from training data

Convolution: Properties

The use of convolution leads to useful properties:

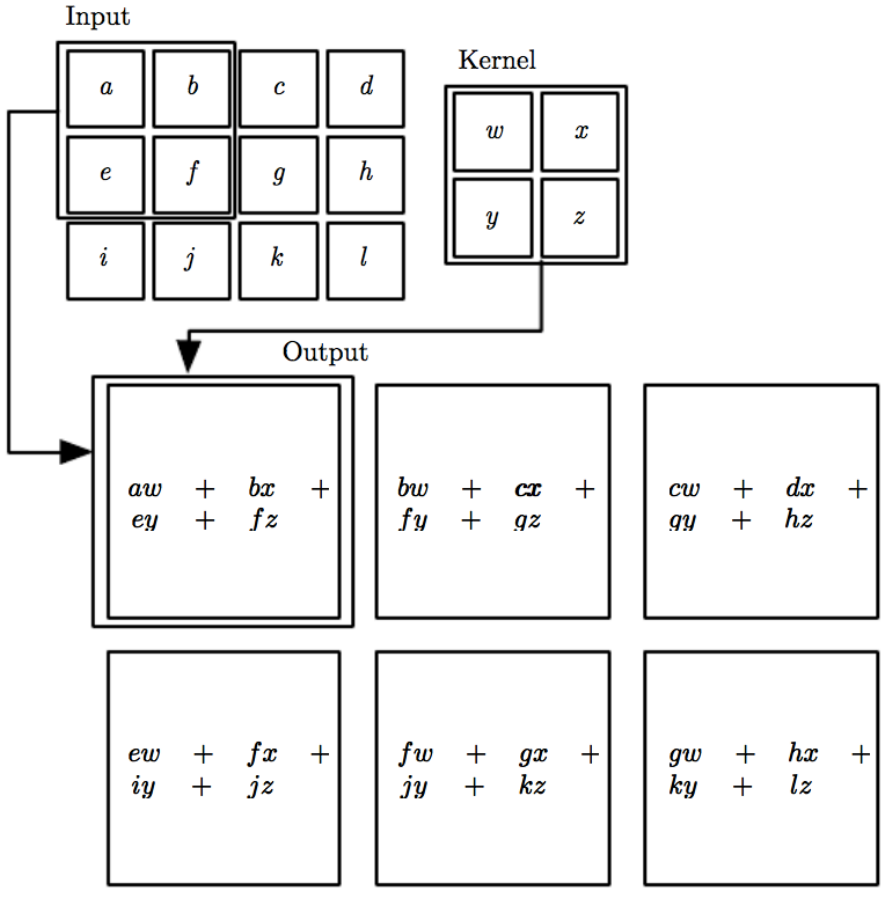
- sparse interactions
- parameter sharing
- equivariant representations

Sparse interactions and parameter sharing: reduce the number of parameters $\Rightarrow$ can be thought as a form of regularization!

Equivariant representations: useful for images (same image/object moved in space)

Convolution: Sparse Interactions

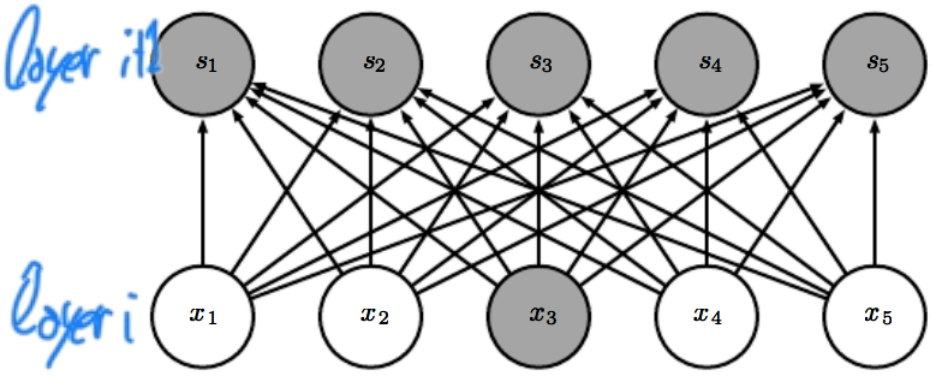
Sparse interactions: many edges do not exist in the network = have 0 weight



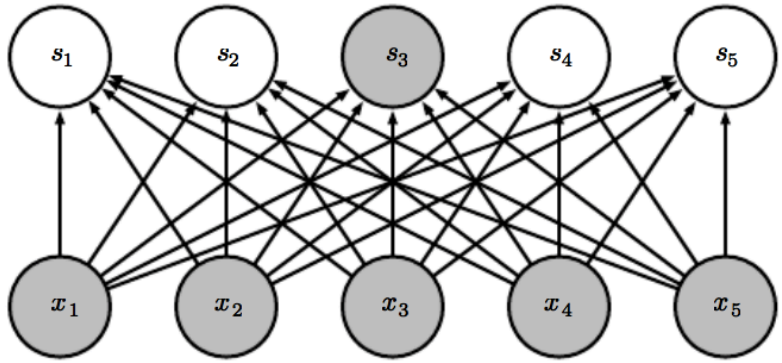
Convolution: Sparse Interactions (continue)

Standard NN

Outgoing edges

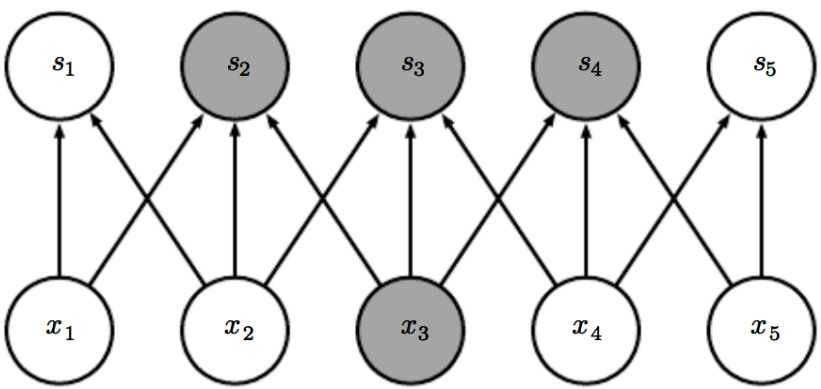


Incoming edges

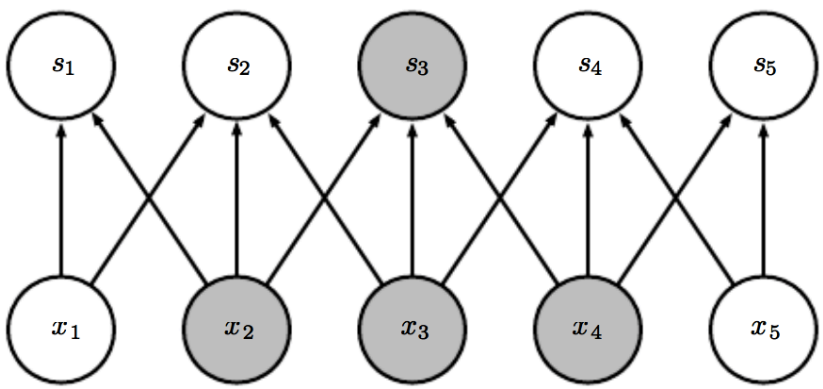


Convolutional NN

Outgoing edges



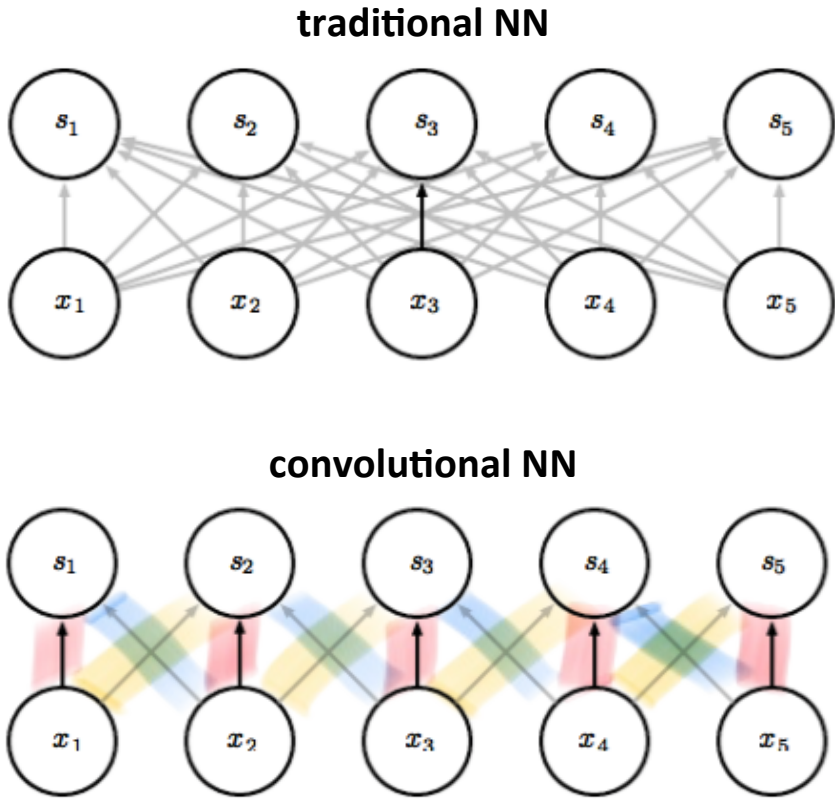
Incoming edges

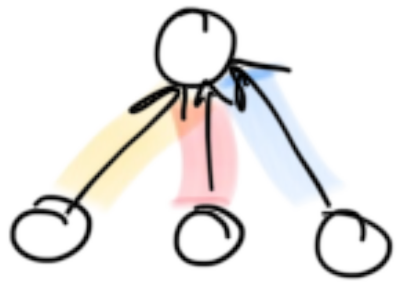


Parameters Sharing

Parameters Sharing = using the same parameter for more than one function in a model

Therefore: rather than learning a separate set of parameters for every input location, we learn only one set

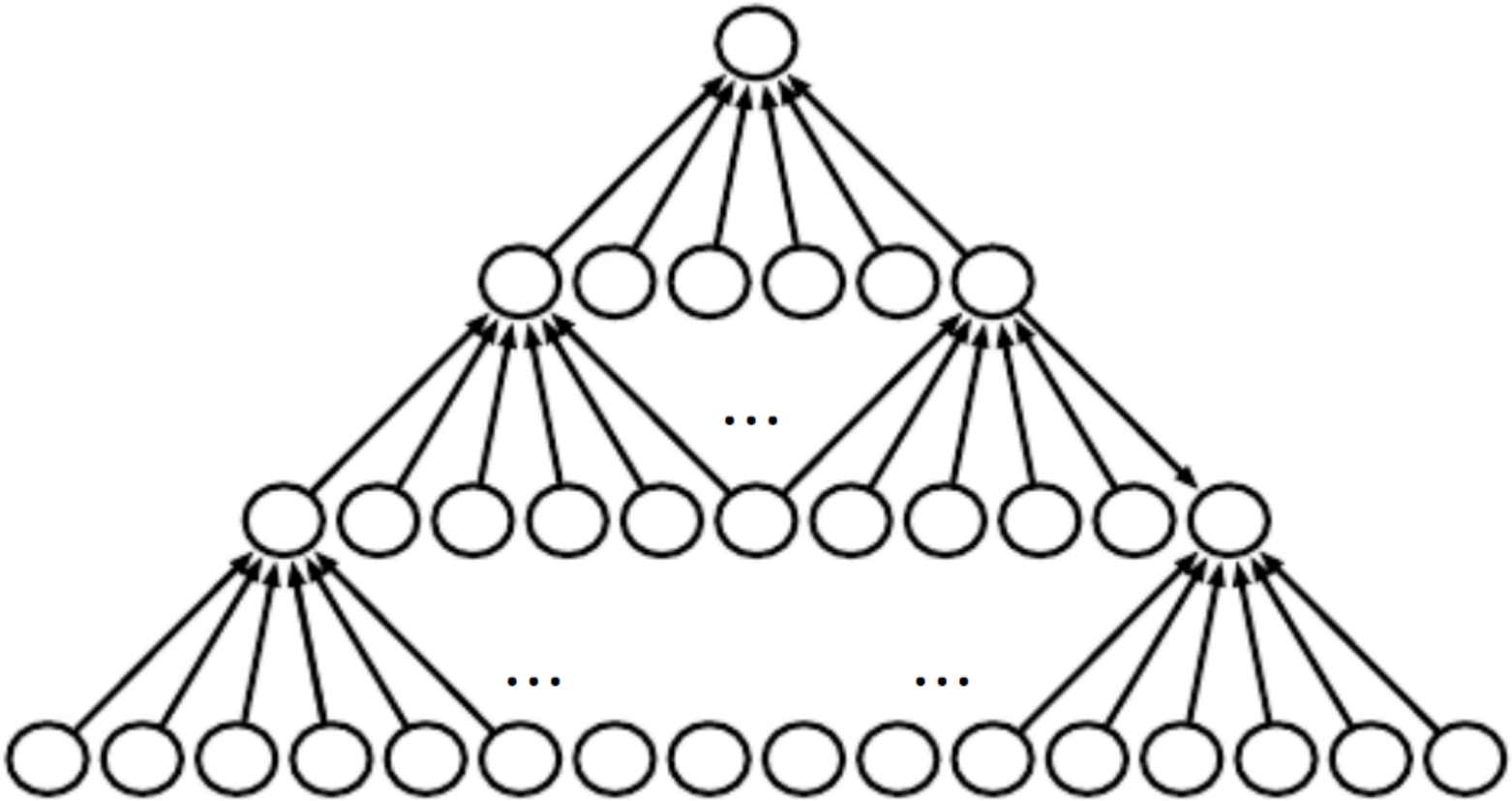




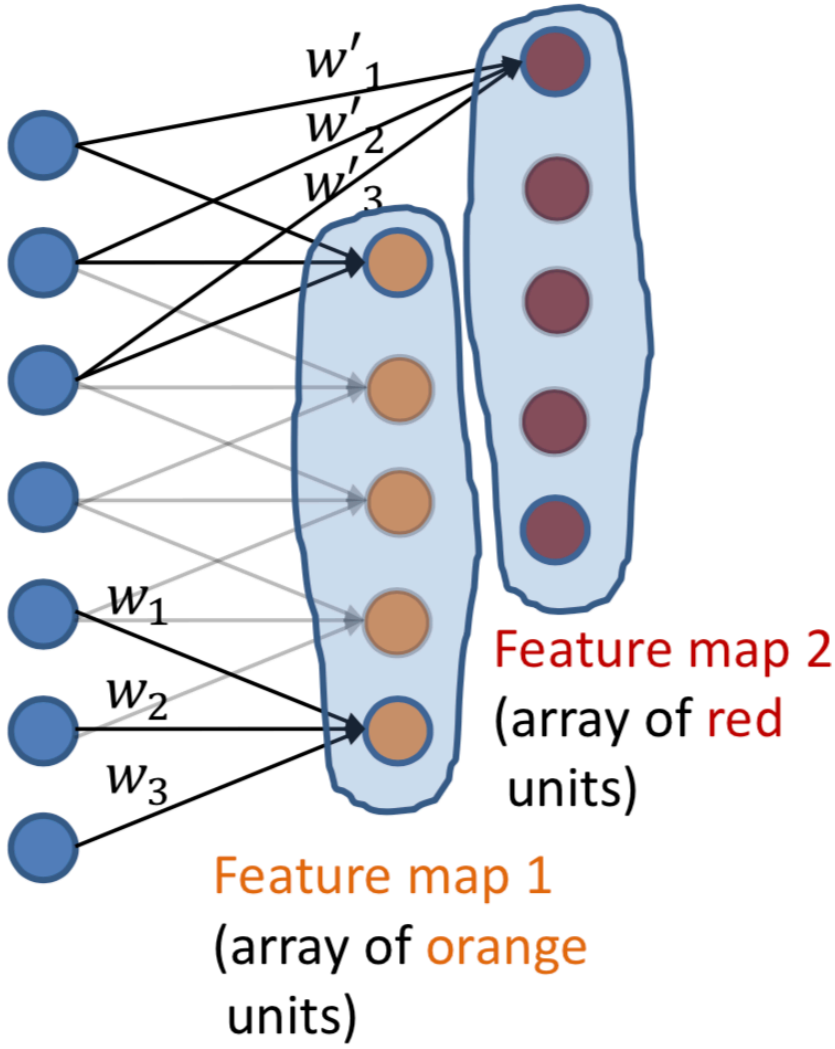
CNNs: Zero Padding

Sometimes want to apply the kernel the same number of times to each input or not to reduce dimension of next layer $\Rightarrow$ need to add extra (virtual) inputs $\Rightarrow$ **zero-padding**

No padding:



Multiple Feature Maps



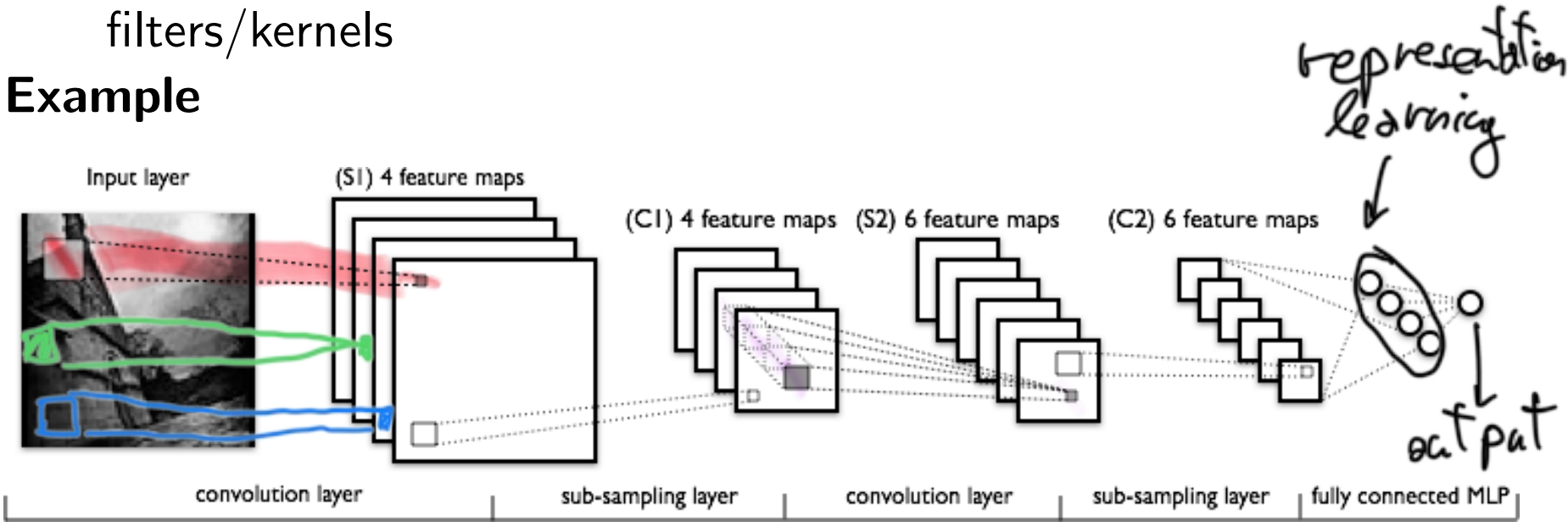
- All **orange units** compute the same function but with a different input windows
- **Orange** and **red** units compute different functions

CNNs: Convolutional Layers

In a CNN:

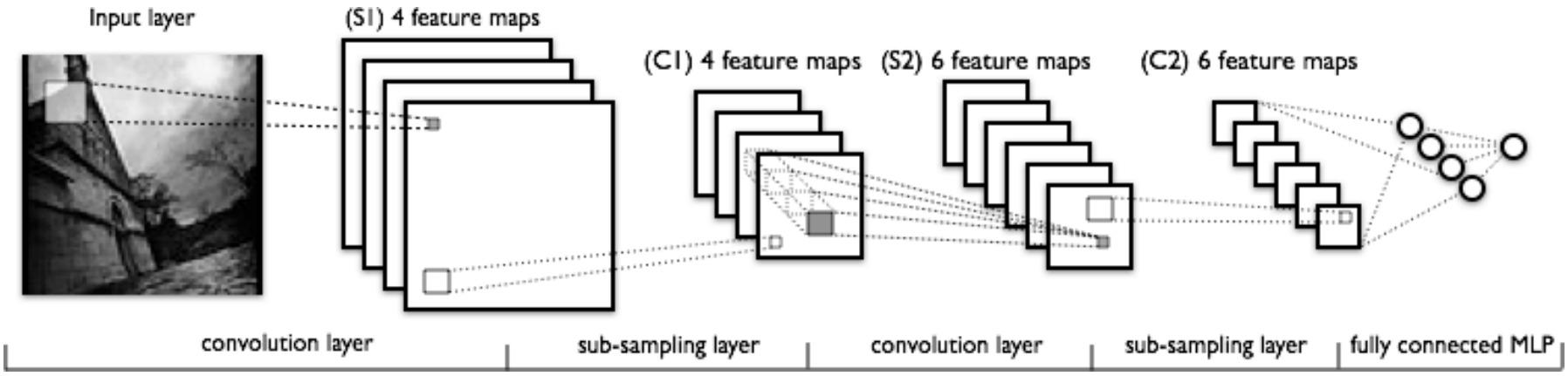
- there are **convolutional layers** (implement convolution) and other types of layers
- the convolutional layer models consists of several filters/kernels

Example



- 4 kernels are learned by the CNN in the first convolutional layer
- 6 kernels are learned by the CNN in the second convolutional layer

CNNs: After Convolution

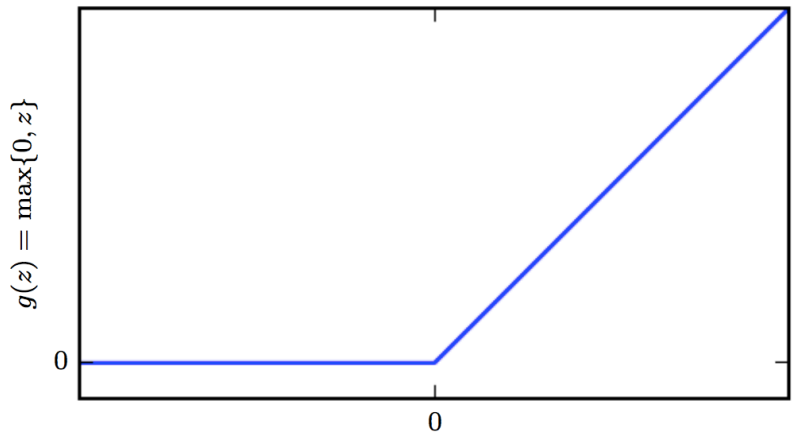


After a convolutional layer:

- nonlinear function: **ReLU (Rectified Linear Unit)**
- **pooling** layer, often combined with **subsampling**

ReLU (Rectified Linear Unit)

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



- helps convergence (almost linear)
- does not hurt expressiveness
- prevents the *vanishing gradient function* (gradient for sigmoid o threshold functions is ≈ 0 when weights have large absolute value $\Rightarrow$ weights do not change)
- most common activation function for deep NNs
- extremely common for CNNs
- leads to sparse activation
- more efficiently computable than other activation functions

The layer with ReLU activation is sometimes called **detector layer**

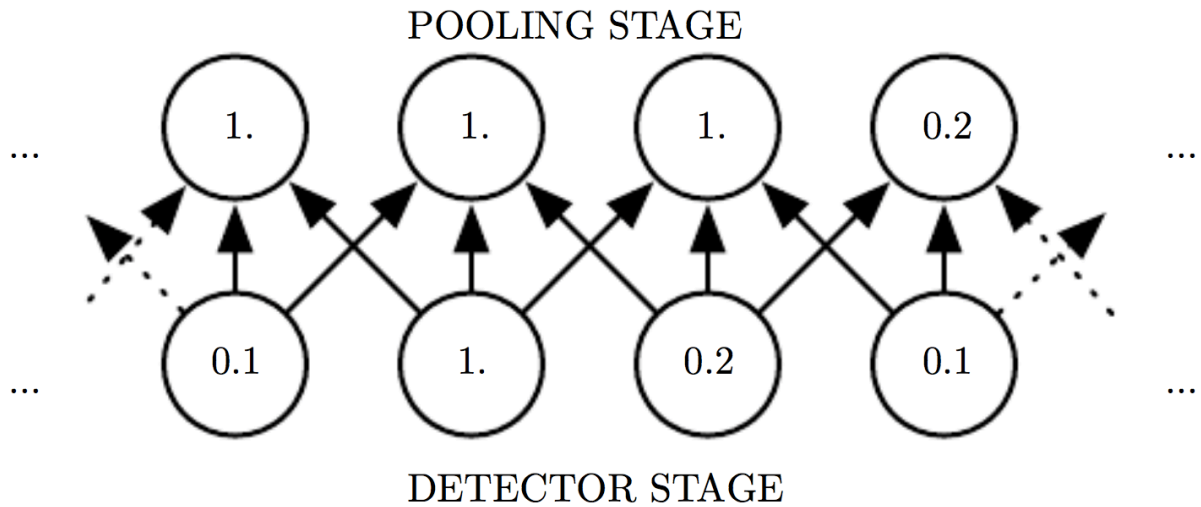
Pooling

Pooling function: replaces the output of the network at a certain location with a summary statistic of the nearby outputs.

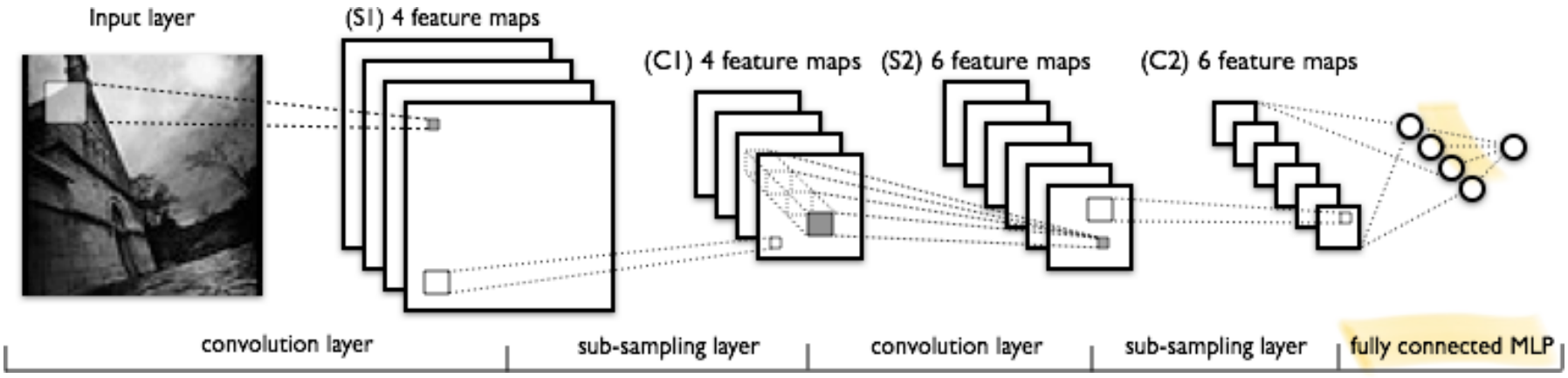
(Popular) Examples:

- *maximum* of a rectangular neighborhood (**max pooling**)
- *average* of a rectangular neighborhood (**average pooling**)
- weighted average based on the distance from the network location
- ...

Example: Max Pooling



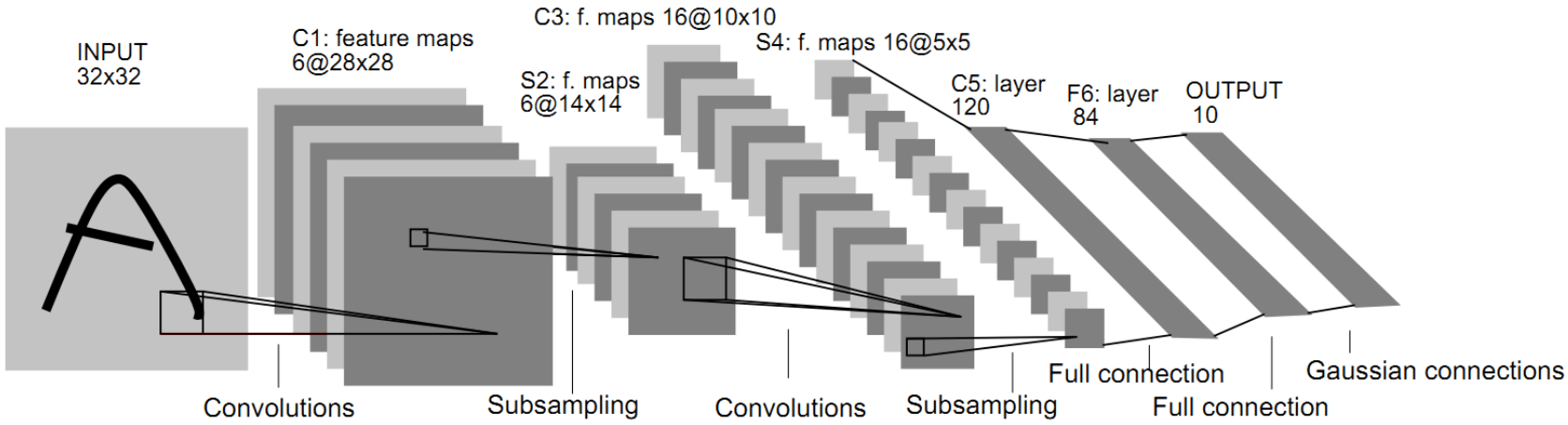
CNN: Last Layer(s)



At the end: fully connected NN (MLP - Multi Layer Perceptron), which learns from the features extracted at the previous hidden layers

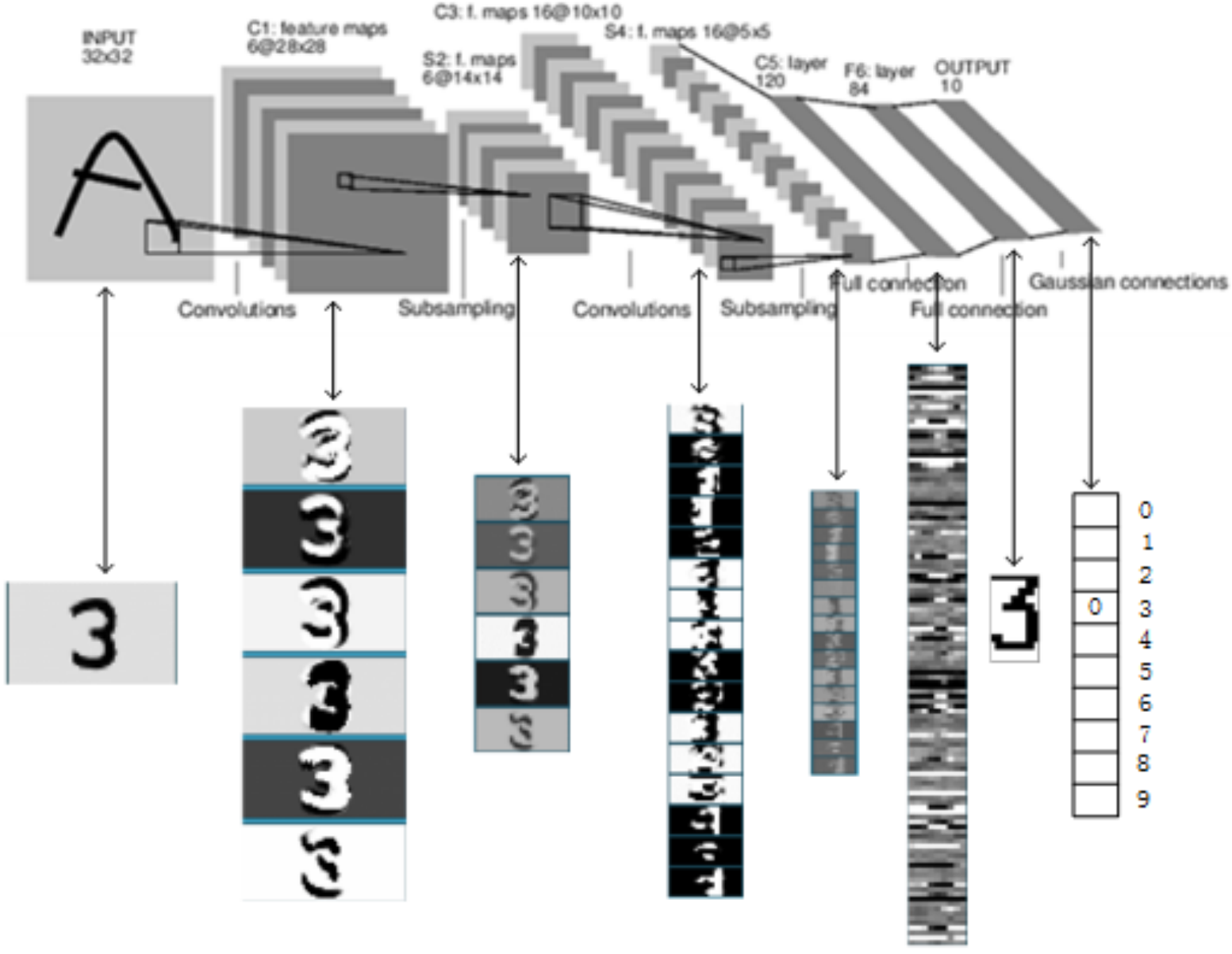
CNN: LeNet-5

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proc. IEEE 86(11): 2278-2324, 1998.

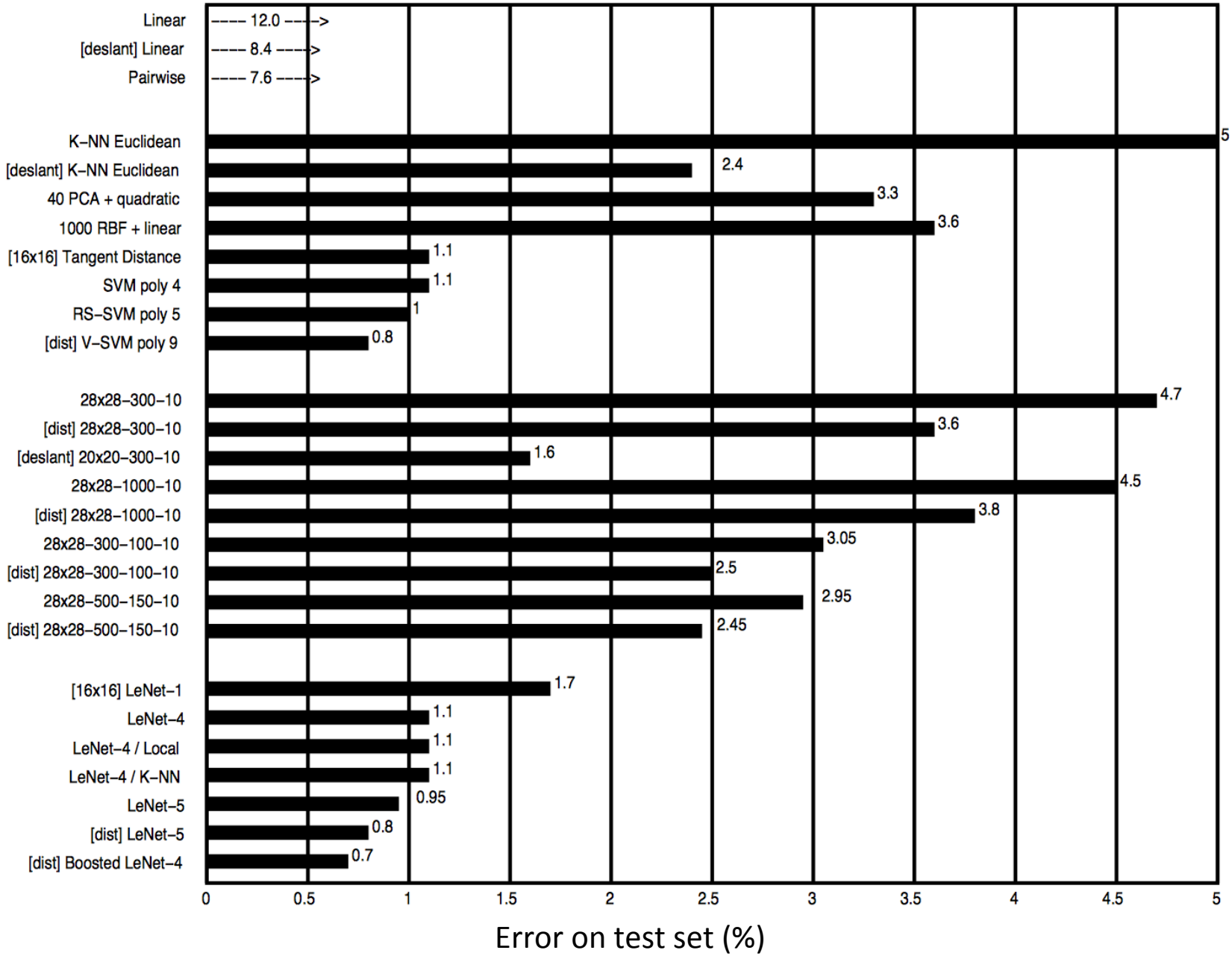


- task: digit recognition (output: $\Pr[\text{INPUT} = i]$ for $i = 0, 1, \ldots, 9$)
- convolutional filters: 5×5
- subsampling and pooling filters: 2×2, stride $= 2$
- used (some form of) average pooling
- trained on MNIST digit dataset with 60K training examples

LeNet-5 Features



LeNet-5 Results



How well do CNN perform?

ImageNet competition: "The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) evaluates algorithms for object detection and image classification at large scale" (http://www.image-net.org/challenges/LSVRC/)

Uses CNN; does not uses CNN

2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

2017: WMW wins with error $< 2.3\%$ - CNN with > 150 layers!

Many more CNNs have been proposed...