

# Machine Learning

## NN and Deep Learning

Fabio Vandin

January 10<sup>th</sup>, 2023

# Neural Networks

**Informal definition:** simplified models of the brain

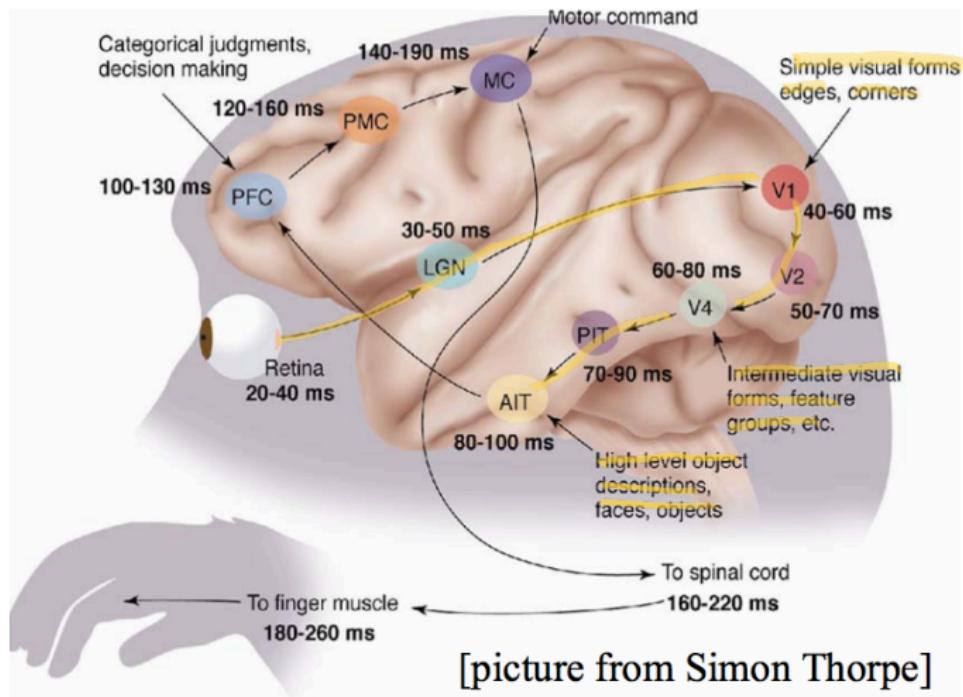
- large number of basic computing units: *neurons*
- connected in a complex network



# 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



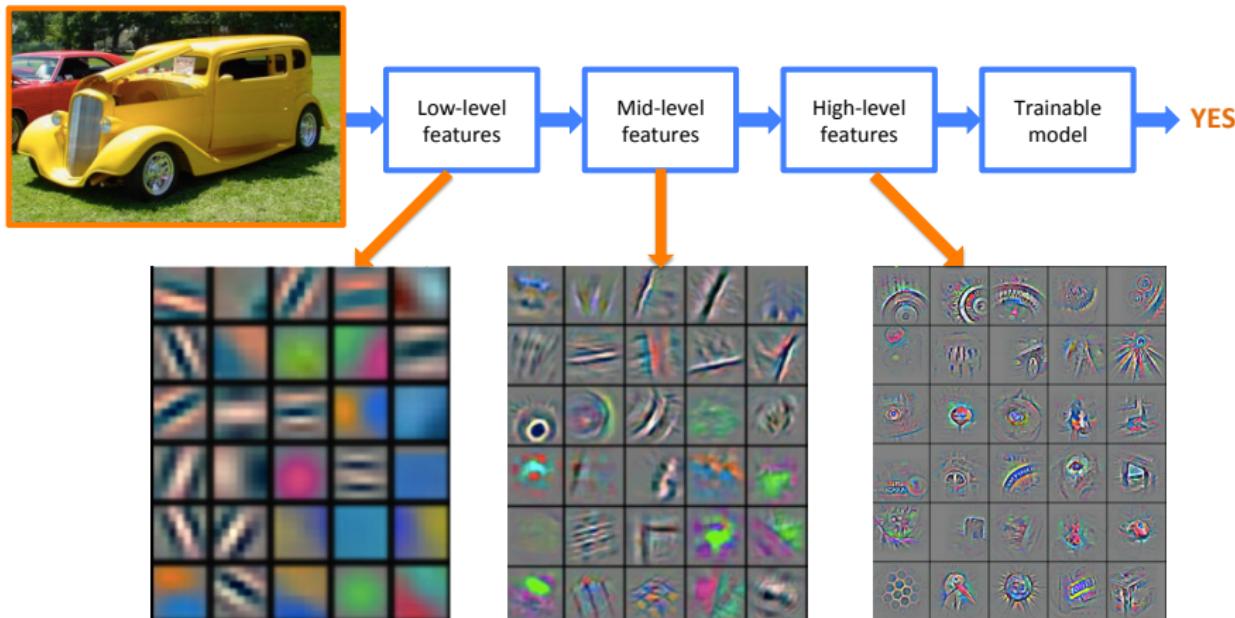
**Deep Learning:** Representations are hierarchical and trained



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

# Hierarchical Representations and ML (continue)

Example: object recognition (object = car)



# Neural Networks: Historical Remarks

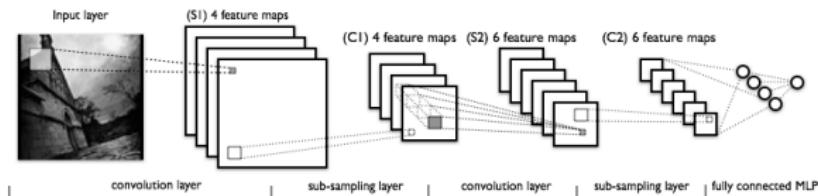
- 1940s-70s:
  - Inspired by learning/modeling the brain (Pitts, Hebb, and others)
  - Perceptron (Rosenblatt), Multilayer perceptron (Minsky and Papert)
  - Backpropagation (Werbos, 1975)
- 1980s-early 1990s:
  - Practical Backpropagation (Rumelhart, Hinton et al., 1986) and SGD (Bottou)
  - Initial 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 imangenet dataset (object recognition of 1000 classes), without unsupervised pre-training

## Issues with “Traditional NNs”

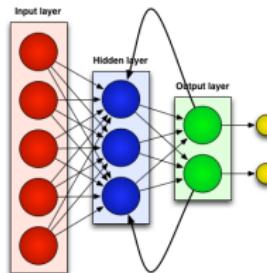
- Huge number of edges  $\Rightarrow$  huge number of weights (due to *full connectivity*)
- Domain structure is not taken into account
  - in some domains (e.g., images, text) there is a lot of *structure*  $\Rightarrow$  parts of the input have specific relations!
  - what about having *similar/related*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



• ...

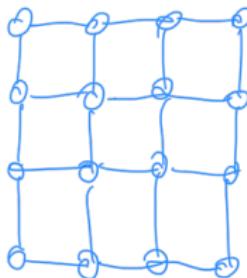
We will see an overview of CNNs

# 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



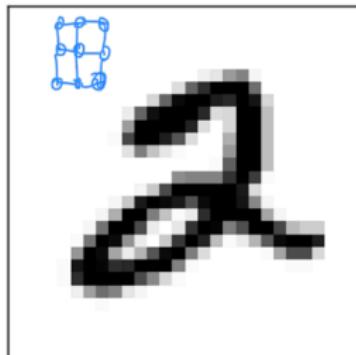
# Convolutional Neural Networks (CNNs)

*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:

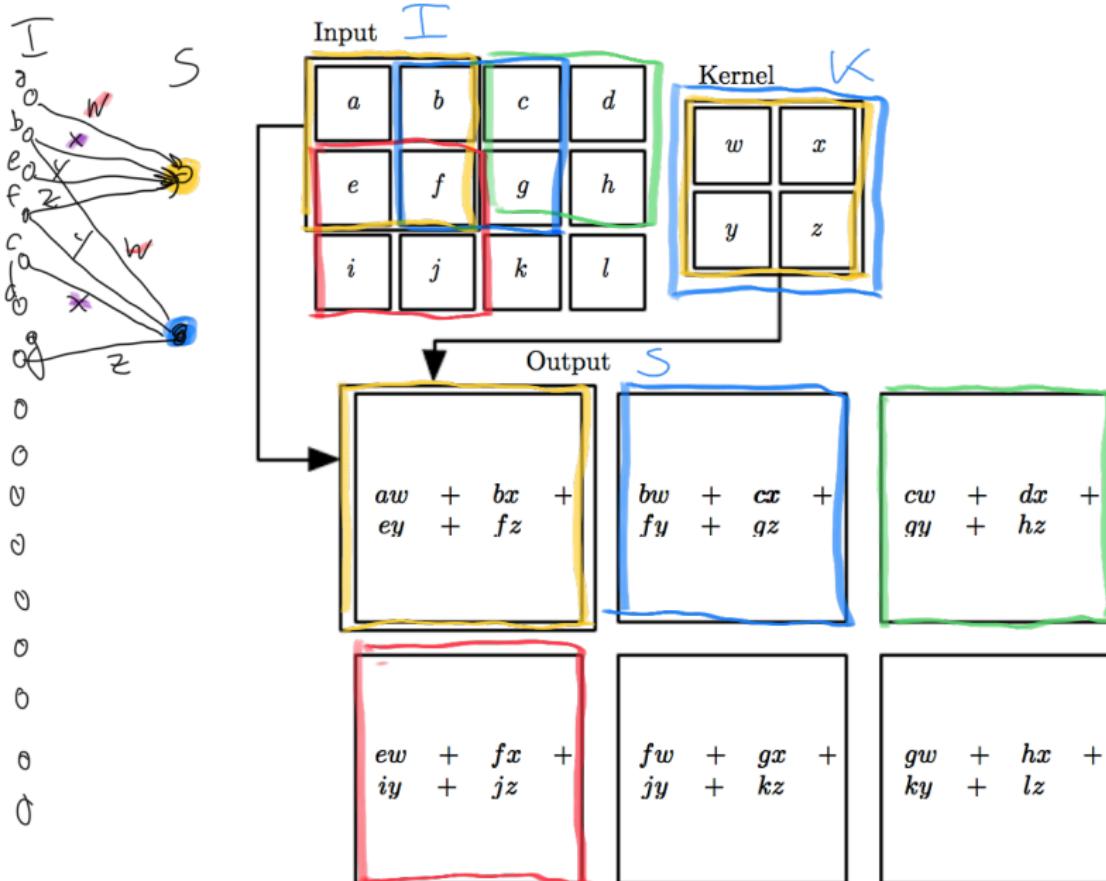
$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n)$$

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

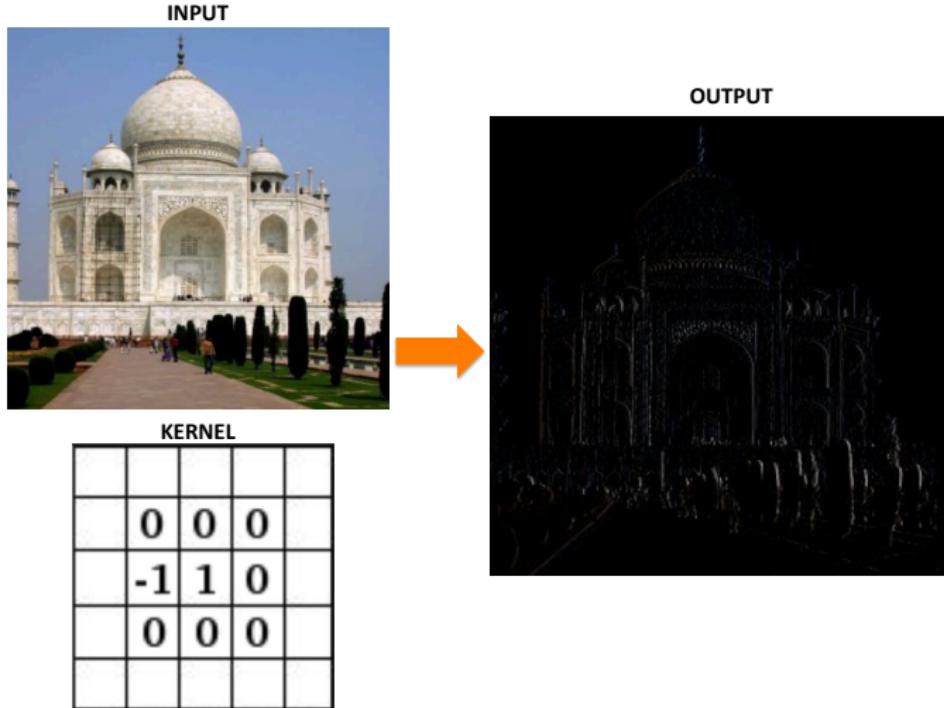
**Note:**  $K$  is usually called a *kernel*

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

# Convolution in CNNs (continue)



# Convolution: Example



**Filter:** *edge enhance*

**Note:** the kernel is learned from training data

## Convolution: Properties

The use of convolution leads to useful properties:

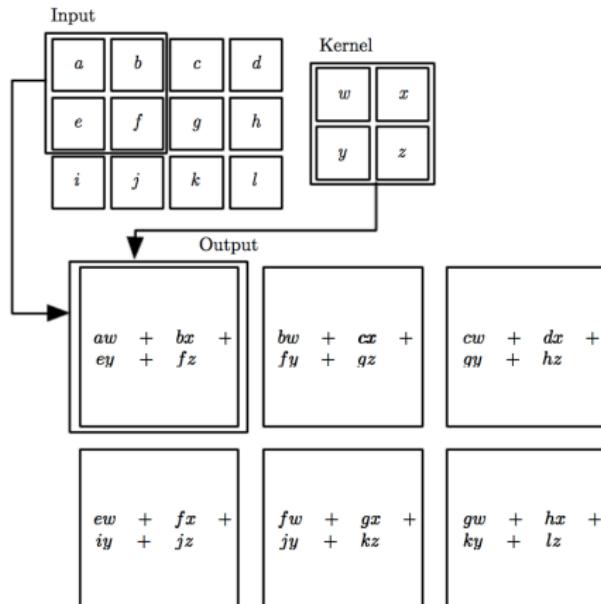
- sparse interactions
- parameter sharing
- equivariant representations

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)

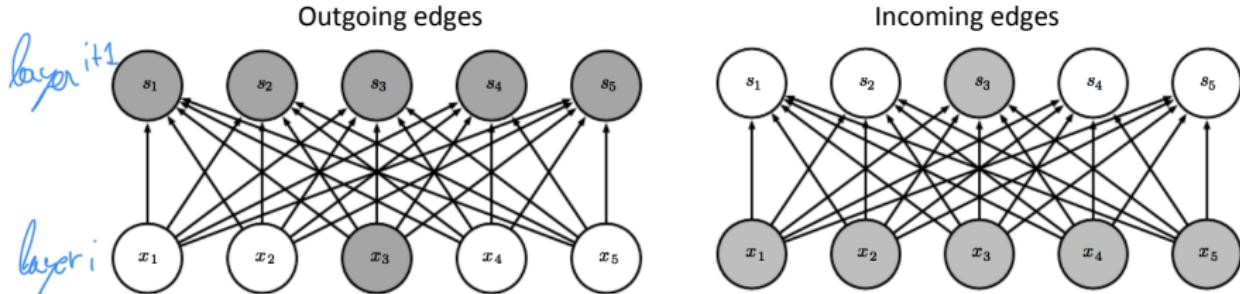
# Convolution: Sparse Interactions

**Sparse interactions:** many edges do not exist in the network  $\Rightarrow$  have 0 weight



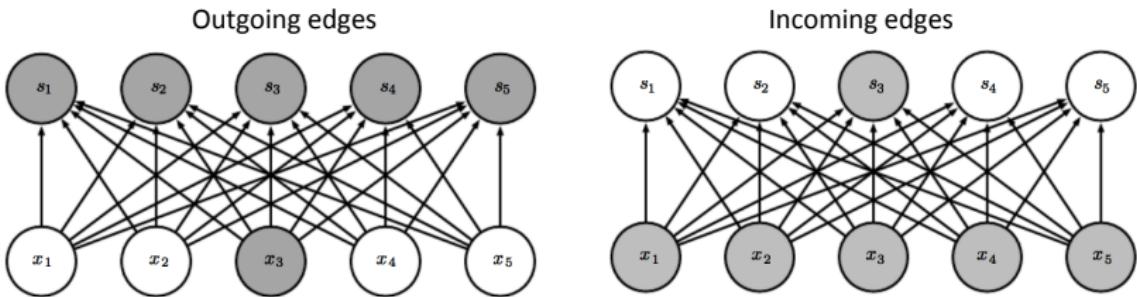
# Convolution: Sparse Interactions (continue)

Standard NN

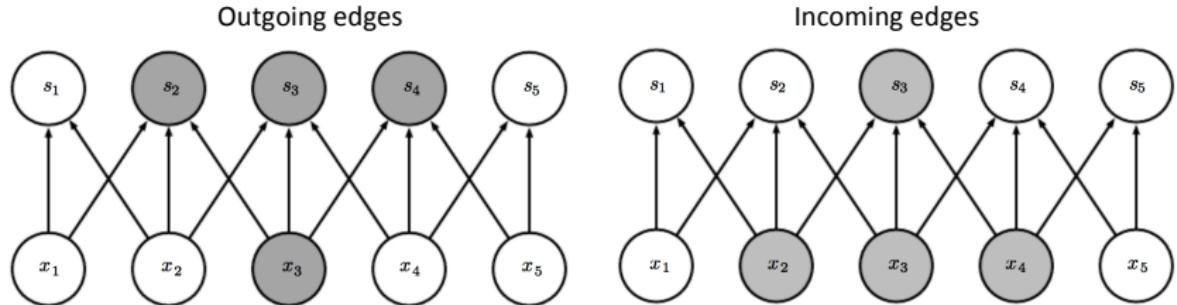


# Convolution: Sparse Interactions (continue)

Standard NN



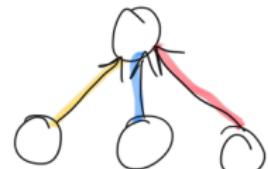
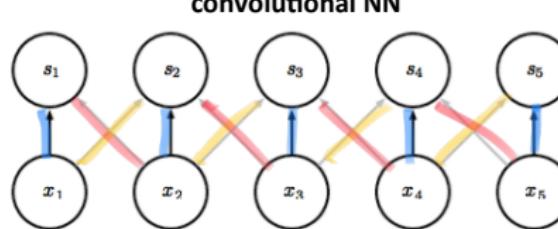
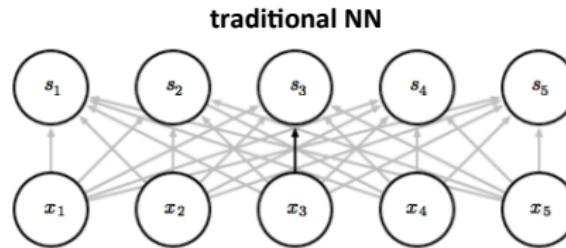
Convolutional NN



# Parameters Sharing

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

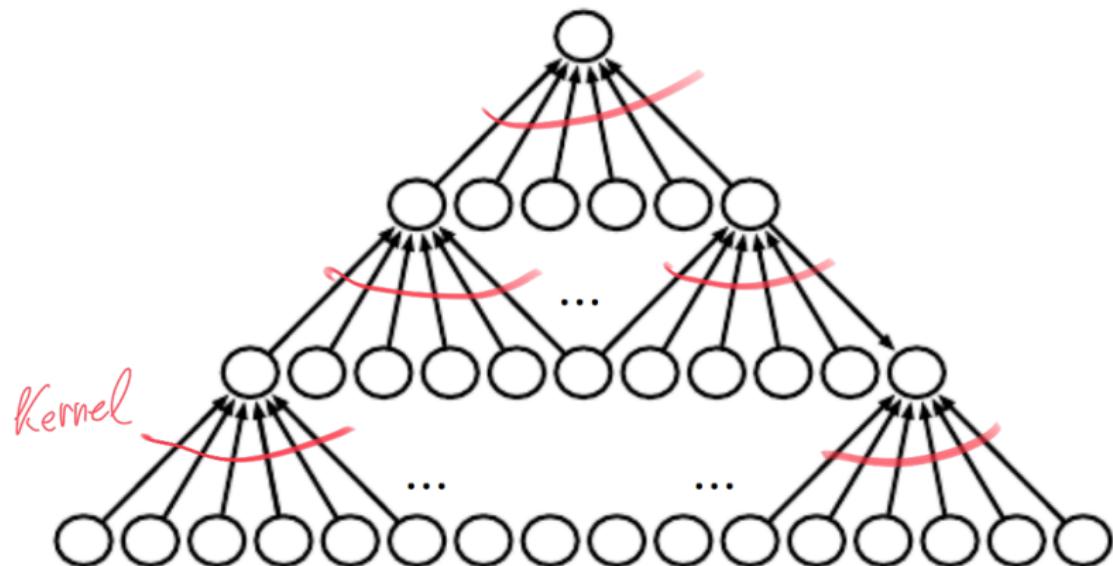
Therefore: rather than learning a separate set of parameters for every input location, we learn only one set



## 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:



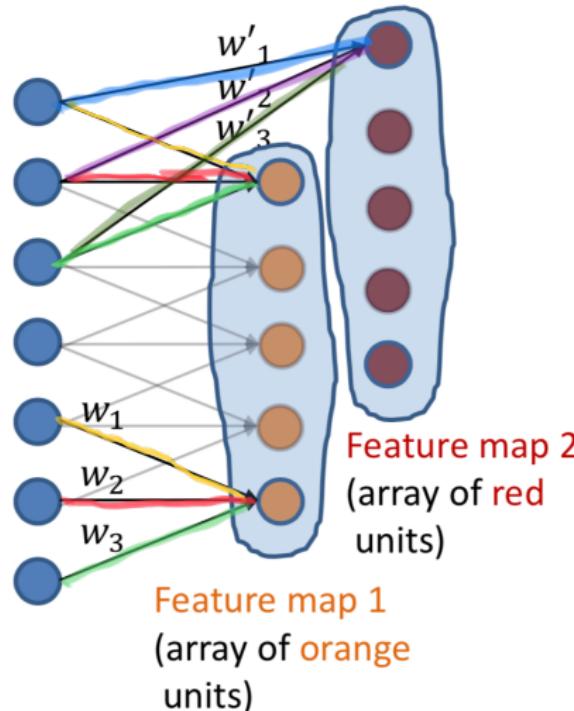
## CNNs: Zero Padding

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**

Zero padding:



# Multiple Feature Maps



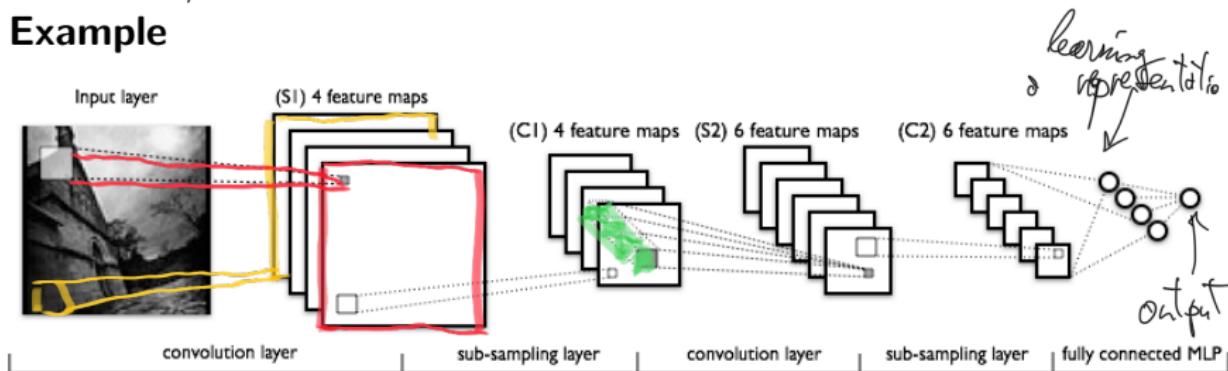
- All **orange units** compute the same function but with a different input windows
- **Orange** and **red** units compute different functions

# 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

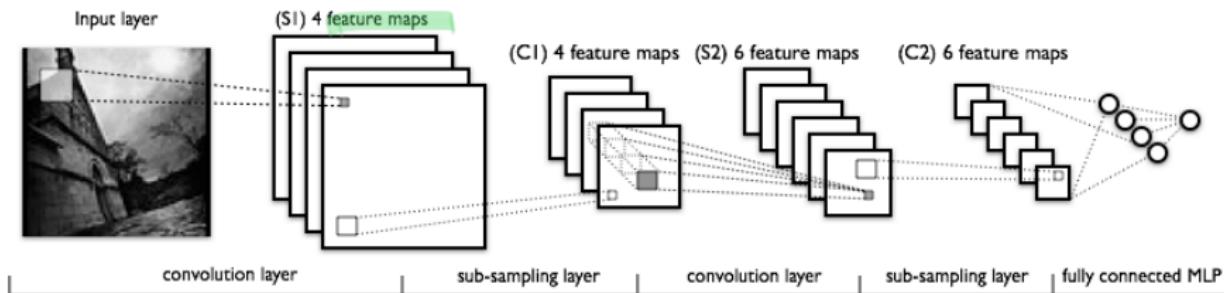


# CNNs: Convolutional Layers

In a CNN:

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

## Example



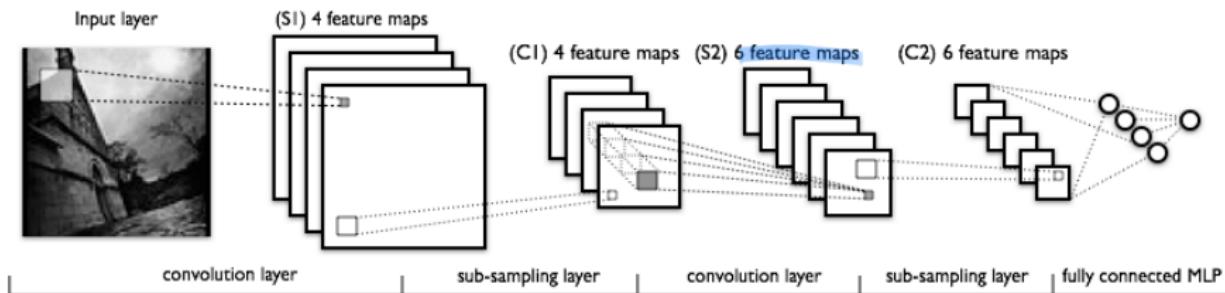
- **4 kernels** are learned by the CNN in the first convolutional layer

# CNNs: Convolutional Layers

In a CNN:

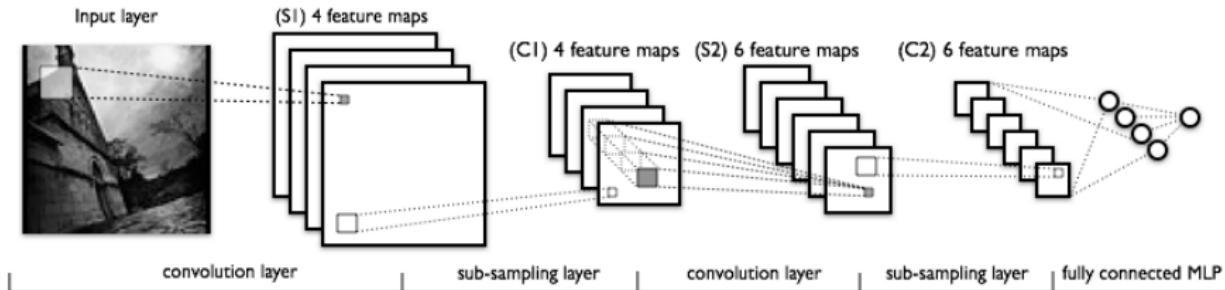
- there are **convolutional layers** (implement convolution) and other types of layers
- the convolutional layer models consists of several filters/kernels

## 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

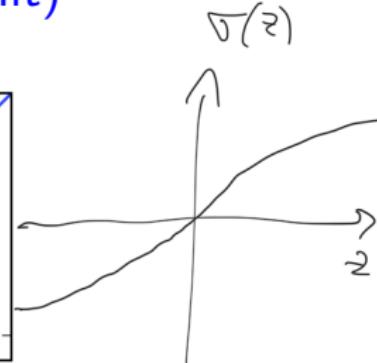
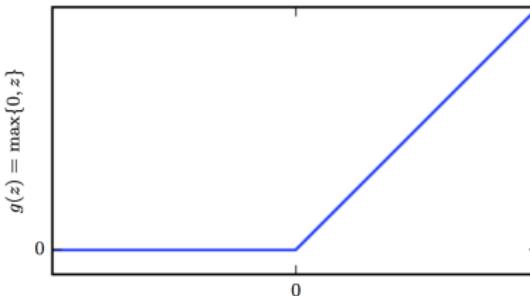


After a convolutional layer:

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

# ReLU (Rectified Linear Unit)

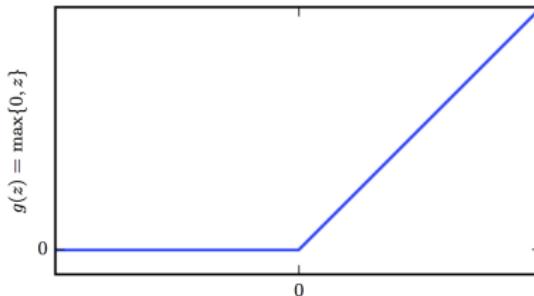
$$\sigma(z) = \max\{0, z\}$$



- helps convergence (almost linear)
- does not hurt expressiveness
- prevents the *vanishing gradient function* (gradient for sigmoid o threshold functions is  $\approx 0$  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

# 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  $\approx 0$  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**

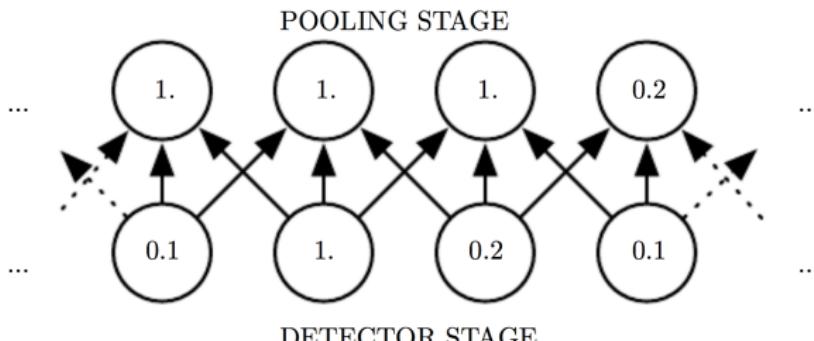
# Pooling

**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
- ...

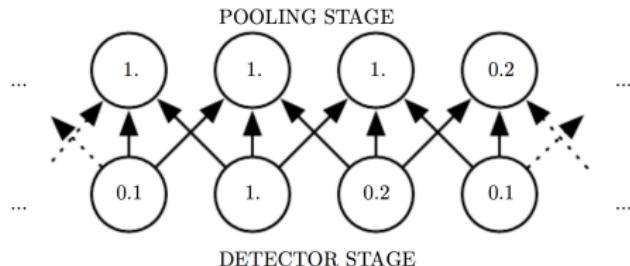
## Example: Max Pooling



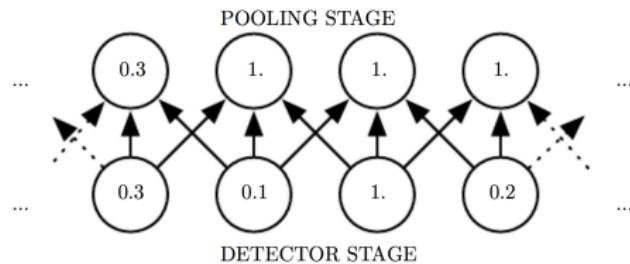
# Max Pooling and Invariance

Max pooling introduces **invariance** to local translation: many neurons have the same output value even if input values are “shifted” a bit

## Before shift



## After shift (to the right by 1)



## Max Pooling and Invariance (continue)

Invariance to local translation: useful if we care more about whether some feature *is present* than exactly *where* it is or *how it appears*

