Introduction to CNNs

Digital Systems M, Module 1

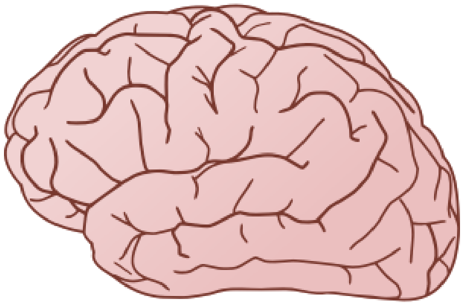
Stefano Mattoccia, Università di Bologna

Neural Networks

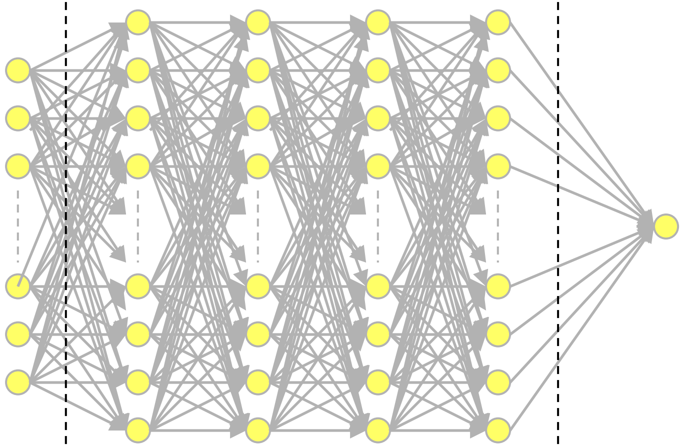
Among Machine Learning (ML) metodologies, neural networks (NN) are bio-inspired frameworks.

According to neuroscience studies, the human brain is made of *billions* of single elements (**neurons**) connected by *billions* of *synapsis*

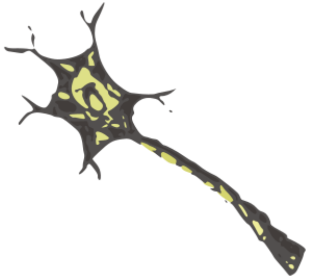
NNs implement a similar mechanism



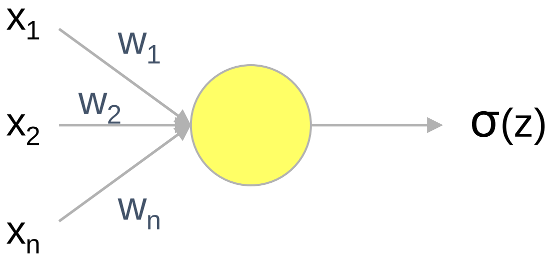
≈

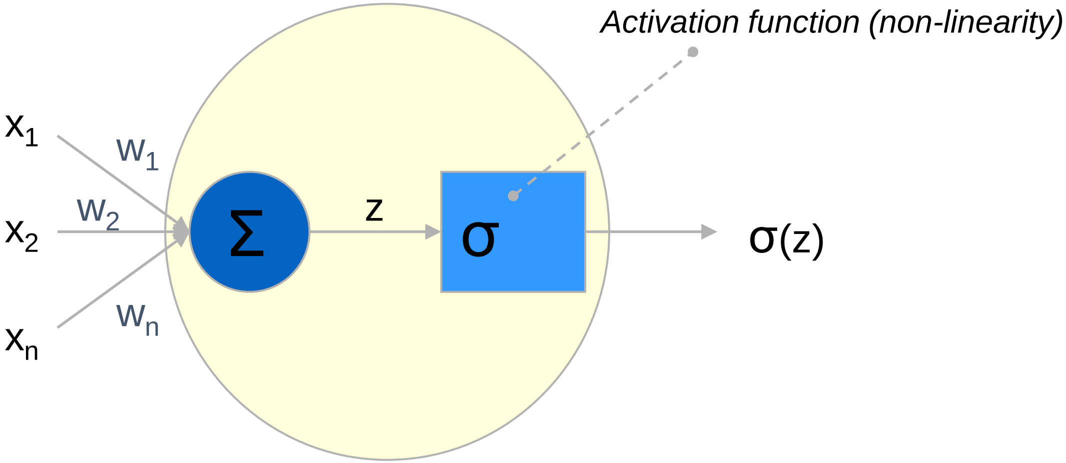


http://neuralnetworksanddeeplearning.com/chap1.html



$\approx$





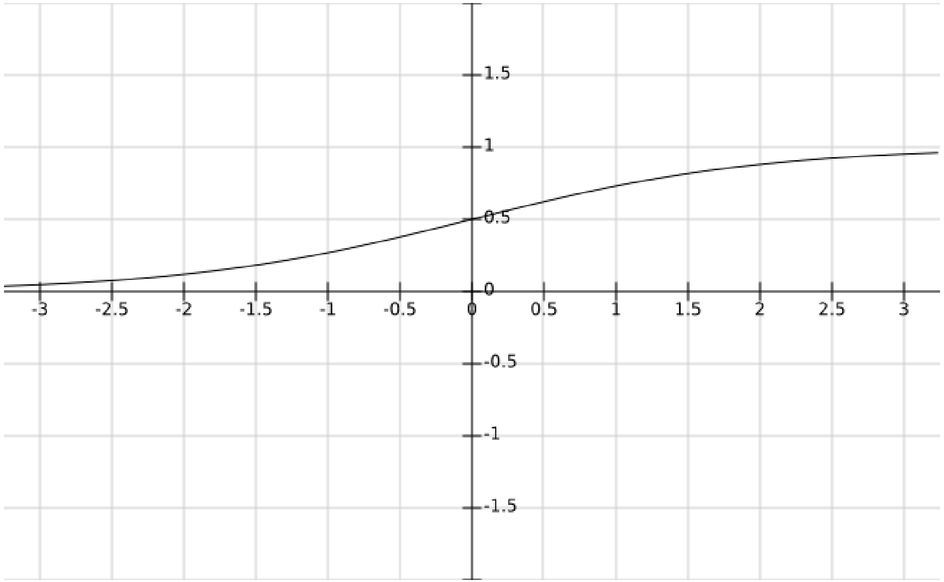
$$z = \sum_i x_i \cdot w_i + b$$

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

σ: sigmoid or *logistic* function

σ: funzione *sigmoid (o logistic)*

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



Introduction to CNNs

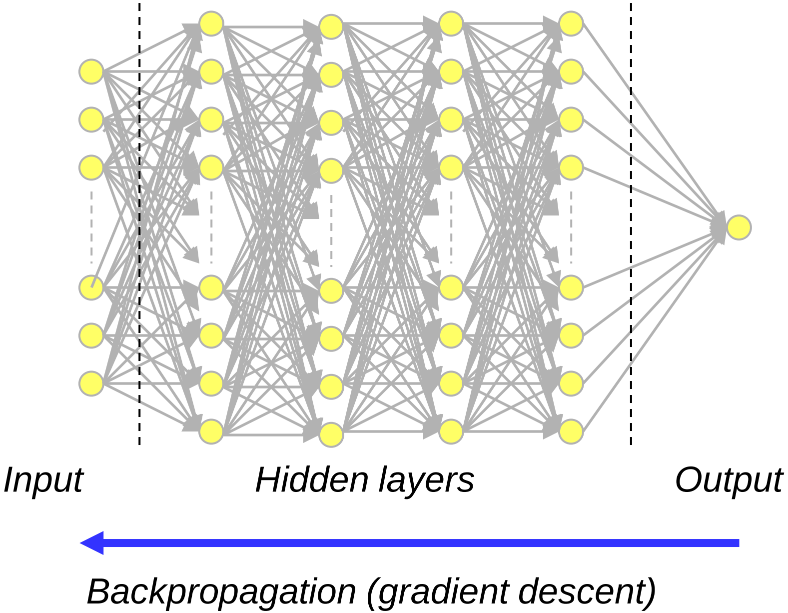
NNs are *trained* in order to find the best values w_i and b for any neuron, by means of the **backpropagation** process

The goal is to find optimal w_i and b that would **minimize** the distance between the network's prediction and the expected value, measured by means of a **loss function** L

The training procedure can be:

- ✓ *Supervised* (We explicitly provide the network with expected values, also known as **labels**)
- ✓ *Unsupervised* (We do not provide the network with labels)
- ✓ . . .

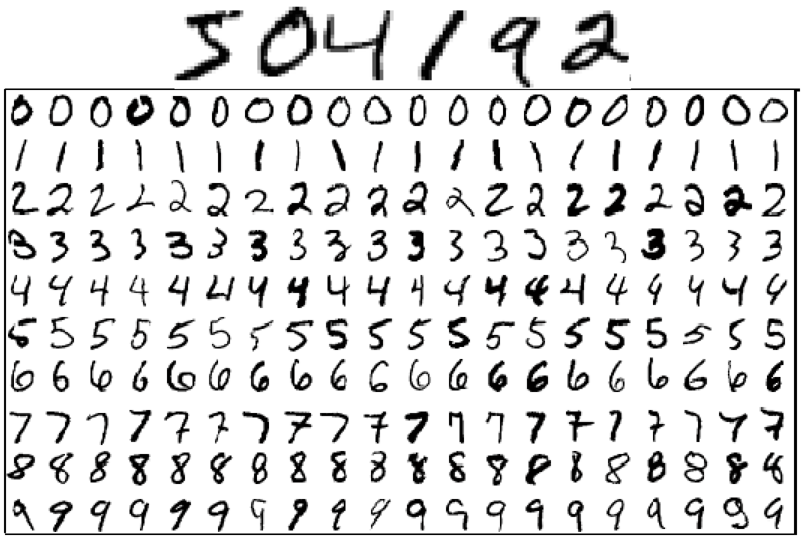
Introduction to CNNs



$$\frac{dL}{dw}$$

An example of loss function: L = Σ(label - output)^2
L needs to be **differentiable**

Example: Digit recognition

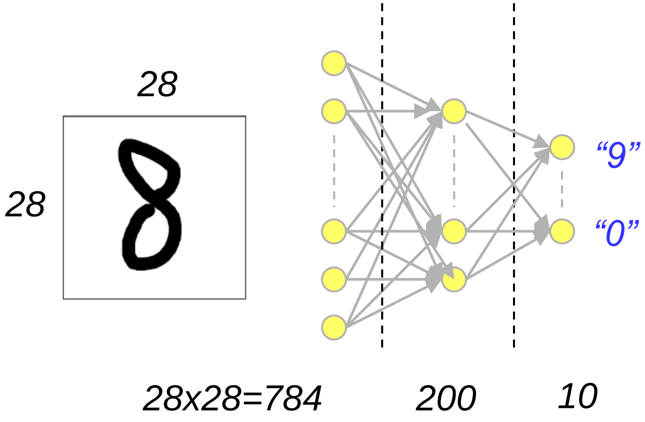


MNIST dataset:
http://yann.lecun.com/exdb/mnist/

http://neuralnetworksanddeeplearning.com/index.html

Introduction to CNNs

Fully connected NN: 1 layer (200 neurons)



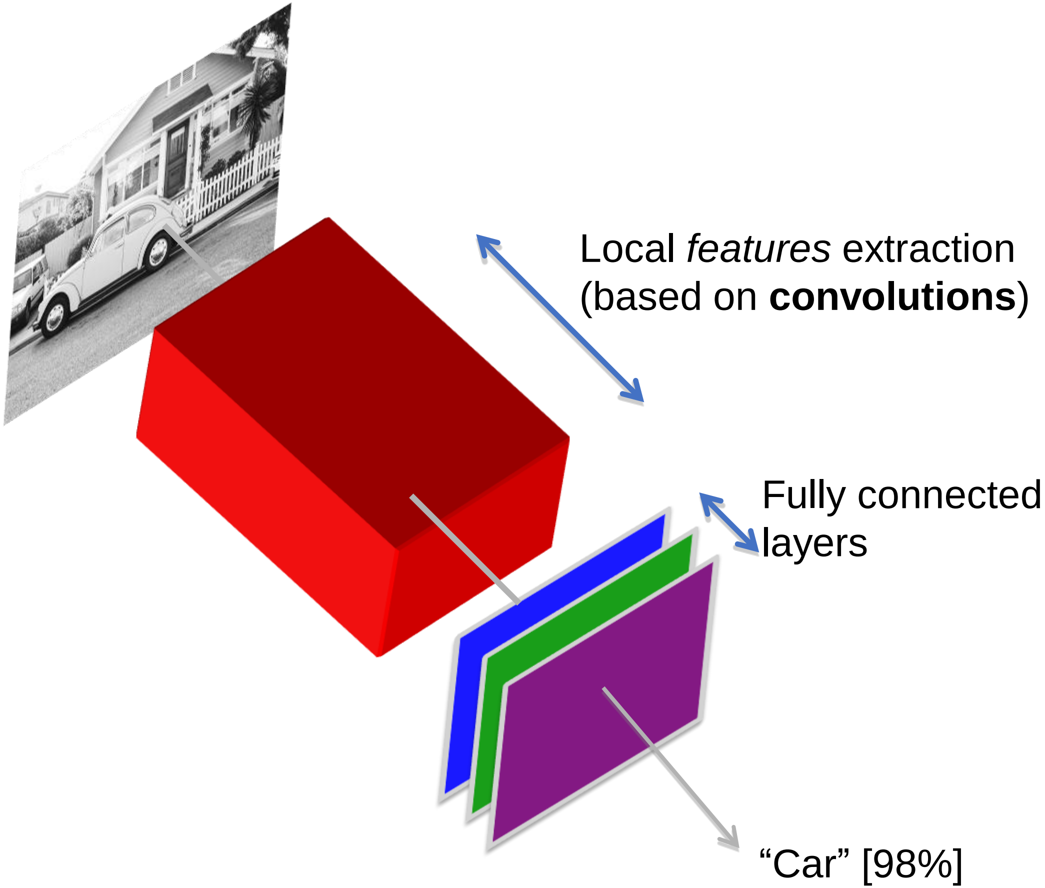
Online demo: http://myselph.de/neuralNet.html

Error: 1.92% (testing dataset)

What happens if the input is not 28x28?
(very likely to happen…)

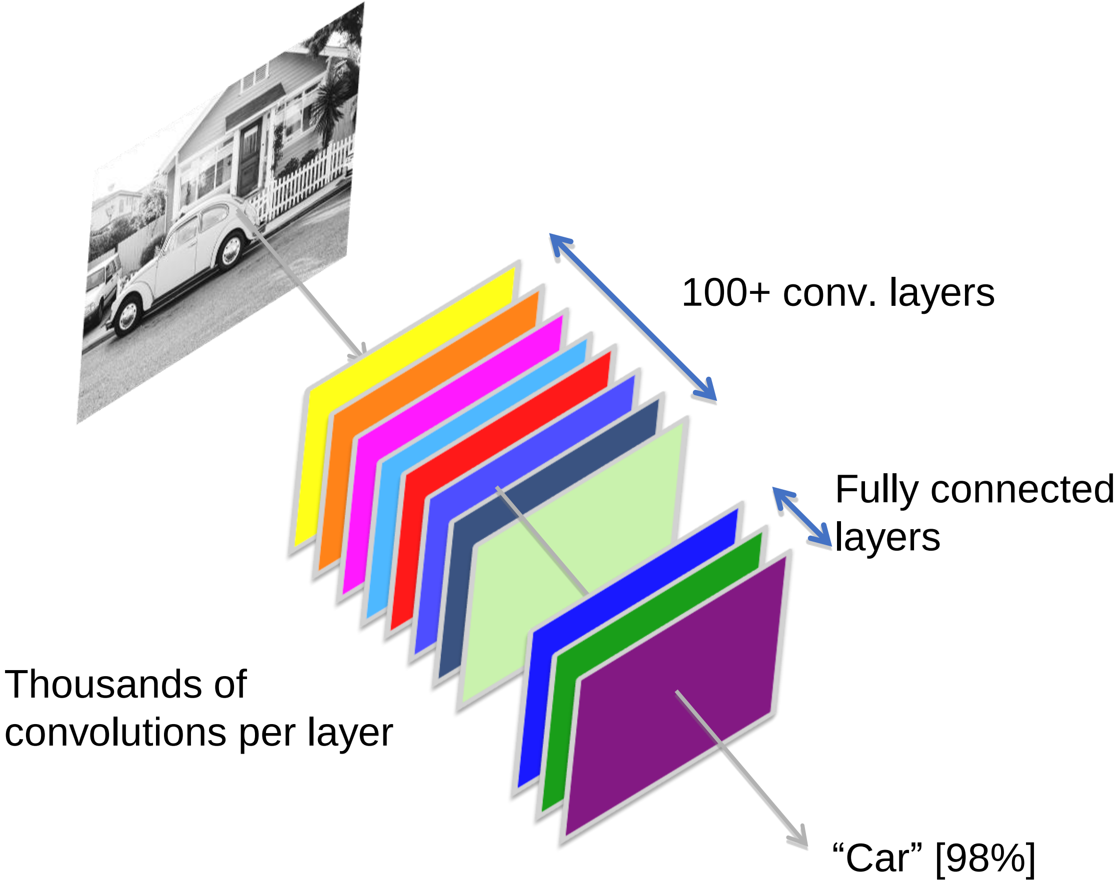
Convolutional Neural Networks

Convolutional NN (CNN)



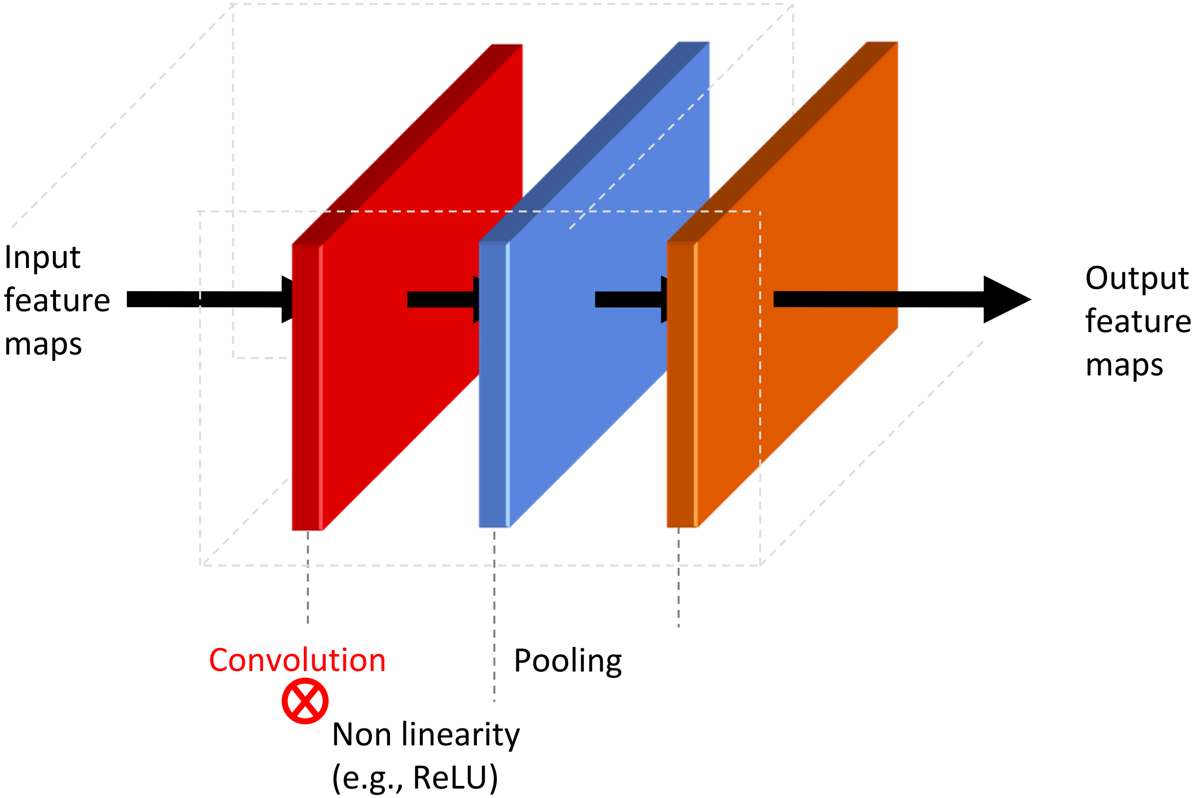
Introduction to CNNs

Convolutional NN (CNN)



Convolutional NN (CNN)

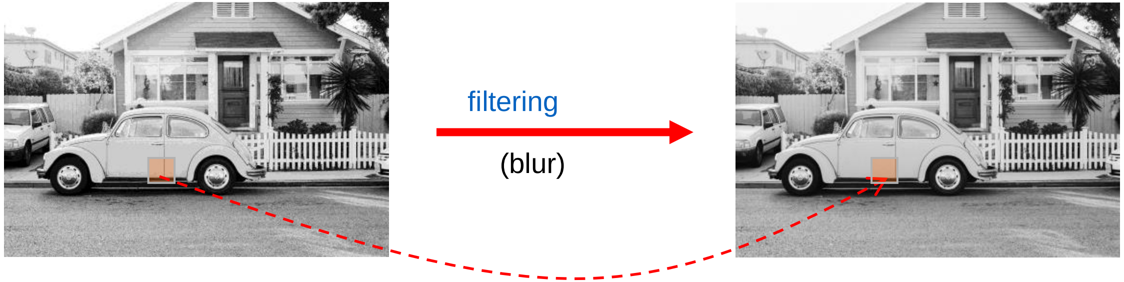
- ✓ A typical CNN is made of multiple layers
- ✓ Early ones are most demanding (convolution filters)



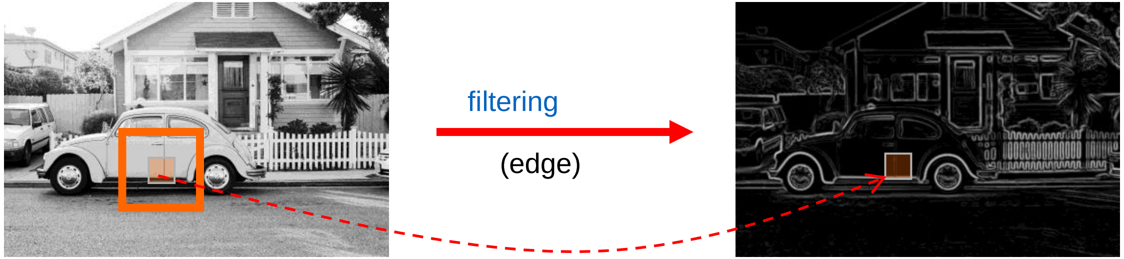
Introduction to CNNs

Image filtering and convolutions

- ✓ Given an input image I, filtering aims at replacing it with a more meaningful representation I’



- ✓ Often (e.g., CNN), I’[x,y] is obtained by processing a patch (<<I) centered in I[x,y]



- ✓ Often, I'[x,y] is a linear combination, according to *kernel coefficients/weights*, of pixels within a patch
- ✓ This operation is known as *convolution* (operator *)

Kernel K (N × M)

1	2	1
2	4	2
1	2	1

K * I

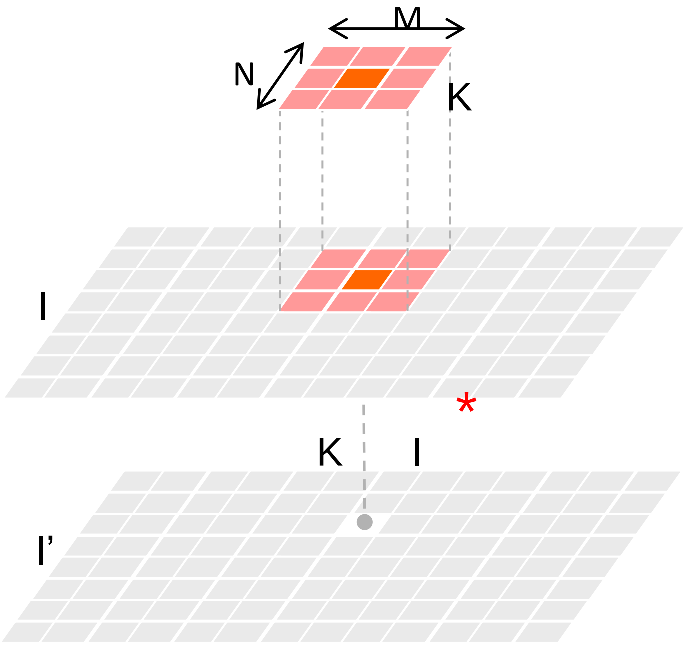
I (H × W)

3	1	5	7	2	2	2	8	7	5	6	7
5	2	7	7	8	8	9	6	6	5	9	8
3	4	6	8	0	1	2	5	6	6	7	7
2	3	0	8	0	2	2	3	5	7	8	9
1	9	4	7	1	1	4	3	4	5	7	2
2	4	5	6	7	8	7	9	9	0	2	3

$$I'[x,y] = K[x,y] * I[x,y] = \sum_{|i|<M/2,\,|j|<N/2} I[x-i, y-i]\cdot K[i,j]$$

$$I'[x,y] = K[x,y] * I[x,y] = \sum_{i,j} I[x-i, y-i]\cdot K[i,j]$$

- ✓ The output image I’, convolution between K and I, is obtained by *sliding* the kernel window K over all the input image I



K (N × M):

1	2	1
2	4	2
1	2	1

K * I

I (H × W):

	0											
0	3	1	5	7	2	2	2	8	7	5	6	7
	5	2	7	7	8	8	9	6	6	5	9	8
	3	4	6	8	0	1	2	5	6	6	7	7
	2	3	0	8	0	2	2	3	5	7	8	9
	1	9	4	7	1	1	4	3	4	5	7	2
	2	4	5	6	7	8	7	9	9	0	2	3

$$I'[1,1] = 1\cdot 3 + 2\cdot 1 + 1\cdot 5 + 2\cdot 5 + 4\cdot 2 + 2\cdot 7 + 1\cdot 3 + 2\cdot 4 + 1\cdot 6 = 59$$

I' (H × W):

	59										

Dot product between K and the image patch

Introduction to CNNs

K (N × M):

1	2	1
2	4	2
1	2	1

K * I

I (H × W):

	0											
0	3	1	5	7	2	2	2	8	7	5	6	7
	5	2	7	7	8	8	9	6	6	5	9	8
	3	4	6	8	0	1	2	5	6	6	7	7
	2	3	0	8	0	2	2	3	5	7	8	9
	1	9	4	7	1	1	4	3	4	5	7	2
	2	4	5	6	7	8	7	9	9	0	2	3

$$I'[1,2] = 1\cdot 1 + 2\cdot 5 + 1\cdot 7 + 2\cdot 2 + 4\cdot 7 + 2\cdot 7 + 1\cdot 4 + 2\cdot 6 + 1\cdot 8 = 88$$

I' (H × W):

	59	88									

N × M kernel K:

1	2	1
2	4	2
1	2	1

K * I

H × W input I:

	0											
0	3	1	5	7	2	2	2	8	7	5	6	7
	5	2	7	7	8	8	9	6	6	5	9	8
	3	4	6	8	0	1	2	5	6	6	7	7
	2	3	0	8	0	2	2	3	5	7	8	9
	1	9	4	7	1	1	4	3	4	5	7	2
	2	4	5	6	7	8	7	9	9	0	2	3

$$I'[1,3] = 1\cdot 5 + 2\cdot 7 + 1\cdot 2 + 2\cdot 7 + 4\cdot 7 + 2\cdot 8 + 1\cdot 6 + 2\cdot 8 + 1\cdot 0 = 101$$

H × W output I’:

	59	88	101								

Gaussian Filter

1	2	1
2	4	2
1	2	1

K_G

*

$1/16 \cdot K_G$

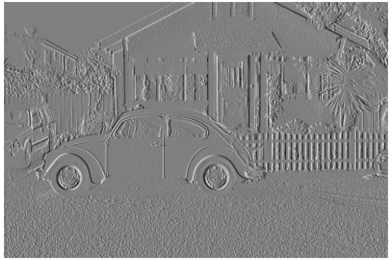
Sobel (Horizontal) filter

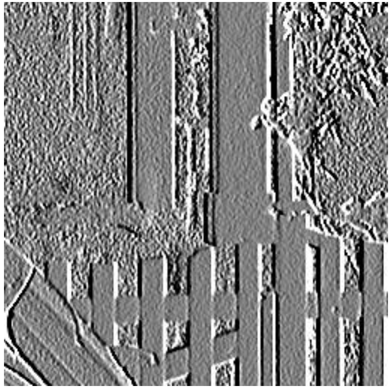
1	0	-1
2	0	-2
1	0	-1

K_{SH}

*

$K_{SH} + 128$





Introduction to CNNs

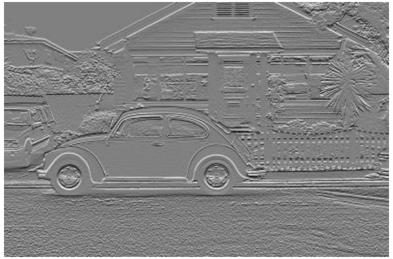
Sobel (vertical) filter

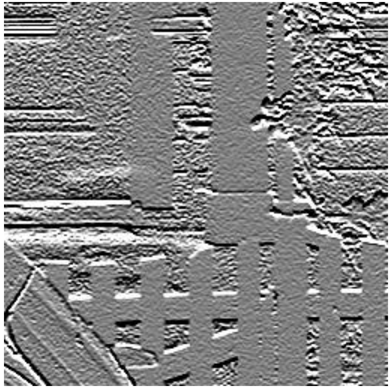
1	2	1
0	0	0
-1	-2	-1

K_{SV}

$*$

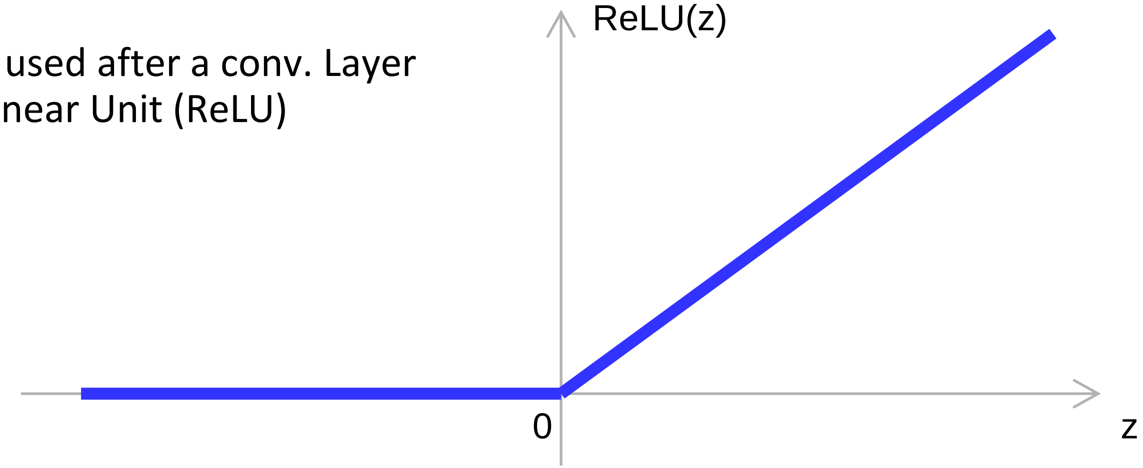
$K_{SV} + 128$





Activation function (non-linearity)

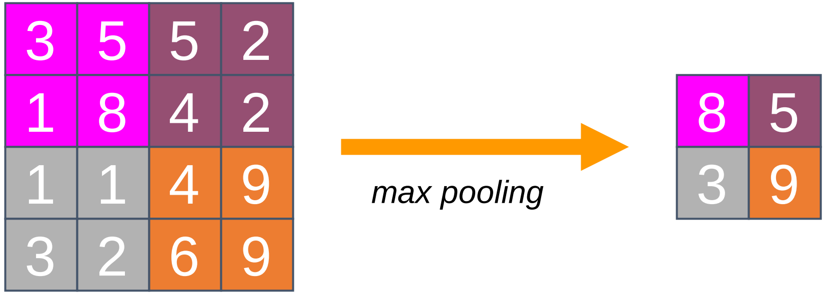
- ✓ Typically, an activation function (remember σ?) that is non-linear
- ✓ In this case, z is the output of a convolution operation
- ✓ A popular activation function used after a conv. Layer inside CNNs is the Rectified Linear Unit (ReLU)



Introduction to CNNs

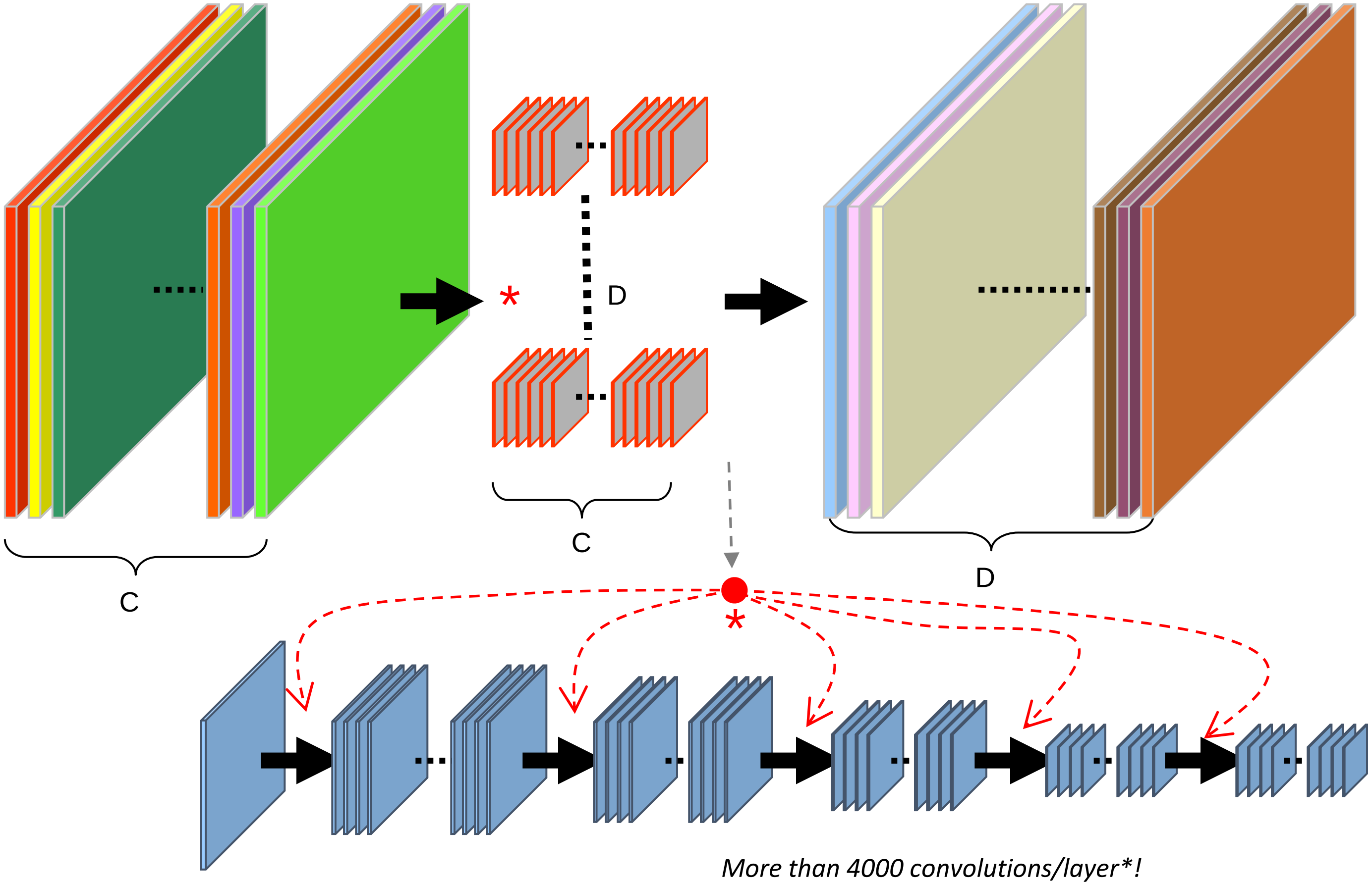
Pooling

- ✓ After a convolution + non-linearity, a **pooling layer** is often deployed
- ✓ The main purpose of this function is to *compact* features, keeping the most important ones. It reduces the image resolution
- ✓ Several strategies exist. For instance, keeping the maximum value in a window (*max pooling*)



- ✓ Activations and pooling are not computationally expensive, compared to convolutions

Introduction to CNNs



More than 4000 convolutions/layer!*