UniGe | MaLGa

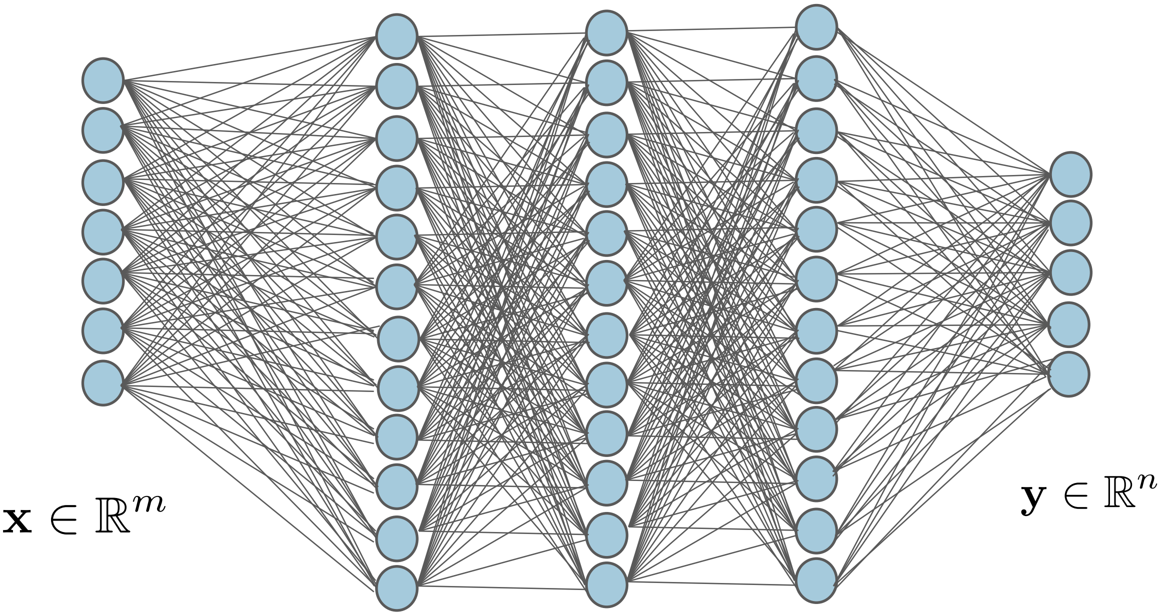
Convolutional Neural Networks

Nicoletta Noceti
Nicoletta.noceti@unige.it

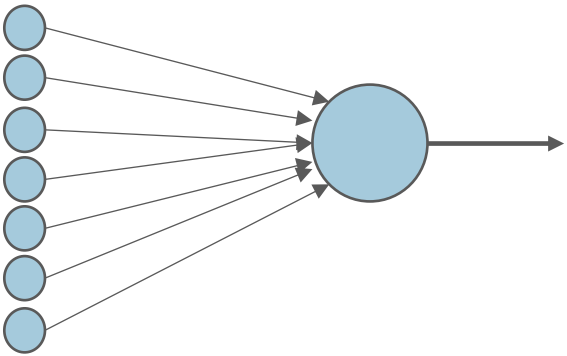
UniGe | MaLGa

A refresh

Deep Neural Network Recap

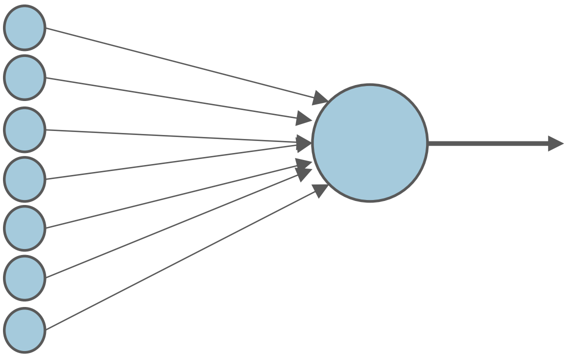


The role of a neuron



$$y = f(\mathbf{x}) = \sigma(\sum_i w_i x_i + b)$$

The role of a neuron



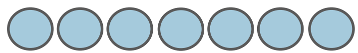
- Each neuron is connected to all the others
- Correlations between input are not taken into account
- As the size of the input and the depth of the architecture increase, the number of parameters increases dramatically

$$y = f(\mathbf{x}) = \sigma(\sum_i w_i x_i + b)$$

Convolutional Neural Networks

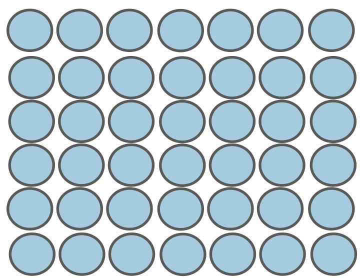
- A specialized kind of neural network for processing data with a known grid-like topology

- Examples:
  - Time-series



1D grid

  - Images



2D grid

Motivations

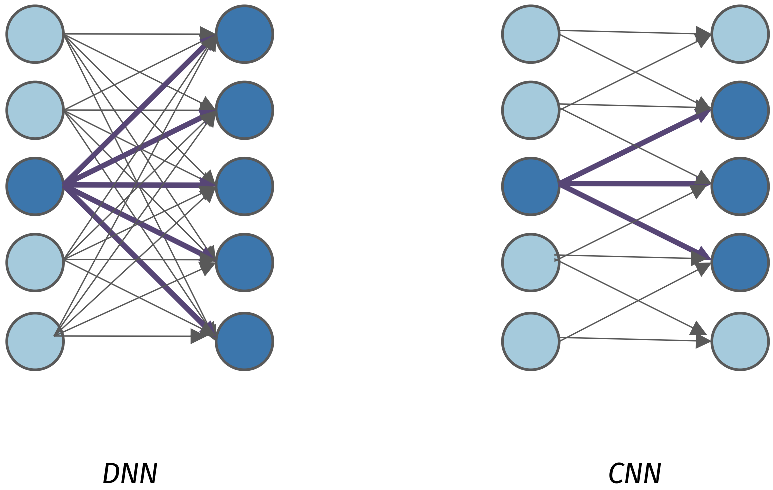
- CNNs leverage two important principles that differ from Dense NNs:
  - **Sparse interaction**
  - **Parameter sharing**

Sparse interactions

- In traditional DNNs every output unit interacts with every input unit
- In CNNs the “field of view” of each neuron is “limited” ma the weights are re-used in different position of the input
- The pros is that we have to learn fewer parameters, with improvements in
  - Memory requirements
  - Statistical efficiency: statistical strength for more samples per weight, reduced variance when estimating the parameters
  - Computations: from O(m x n) to O(k x n) with k << m

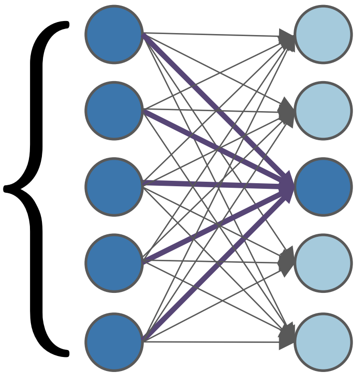
Sparse interaction: intuitions

Focus on the input unit

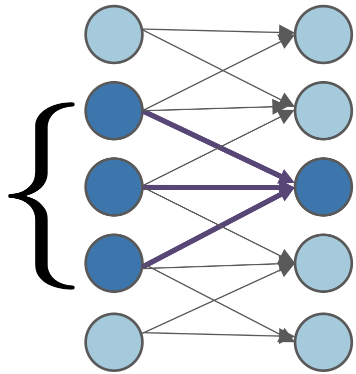


Sparse interaction: intuitions

Focus on the output unit



DNN

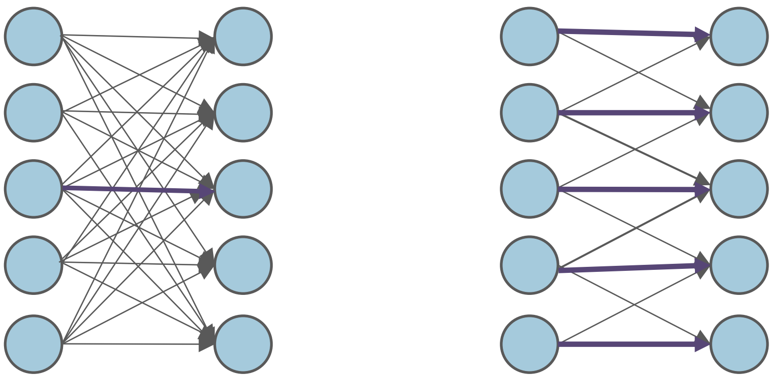


CNN

Receptive fields

Parameter sharing

- The same parameter is used by more than one function in the model
  - In traditional DNNs each weight is used exactly ones when computing the output
  - In CNNs each member of a kernel (a weight) is used at every position of the input



The convolution/cross-correlation operation

For 2D input arrays:

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

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

Cross-correlation (with an example)

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

X_{11}	X_{12}	X_{13}	X_{14}
X_{21}	X_{22}	X_{23}	X_{24}
X_{31}	X_{32}	X_{33}	X_{34}
X_{41}	X_{42}	X_{43}	X_{44}

*

W_{11}	W_{12}
W_{21}	W_{22}

=

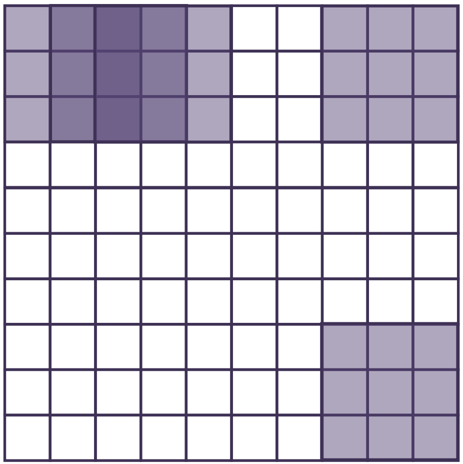
Y_{11}	Y_{12}	Y_{13}
Y_{21}	Y_{22}	Y_{23}
Y_{31}	Y_{32}	Y_{33}

$Y_{11} = X_{11}W_{11} + X_{12} W_{12} + X_{21} W_{21} + X_{22} W_{22}$
$Y_{12} = X_{12}W_{11} + X_{13} W_{12} + X_{22} W_{21} + X_{23} W_{22}$
$Y_{13} = X_{13}W_{11} + X_{14} W_{12} + X_{23} W_{21} + X_{24} W_{22}$
…….

2D convolution

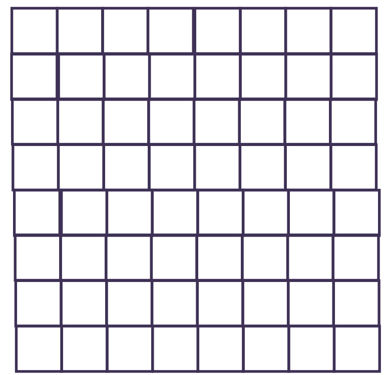
A "feature detector" (kernel) slides over the inputs to generate a feature map

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



Input tensor of size 10x10

Kernel of size 3x3



Output tensor of size 8x8

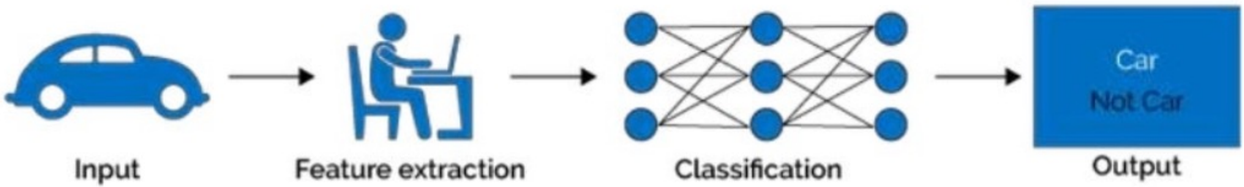
UniGe | MaLGa

NNs don't scale to images!

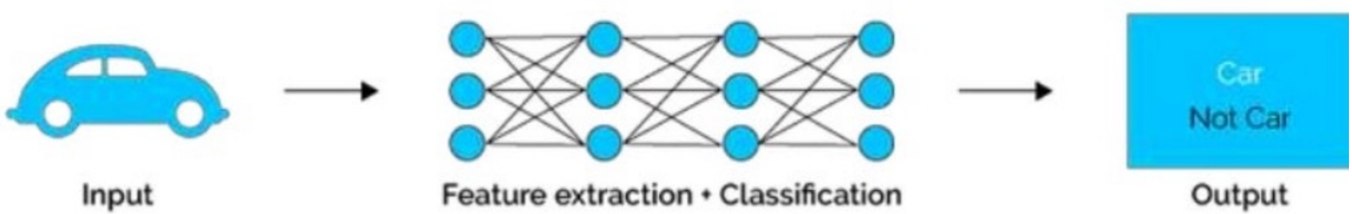
- Let us consider a fully connected network with a single unit
  - Tiny color images of size 32 x 32 x 3
    - Size of the input layer: 32 x 32 x 3 = 3072
    - Size of the weights: 3072
  - Small color images of size 200 x 200 x 3
    - Size of input layer and weights: 200 x 200 x 3 = 120000

From shallow to deep models

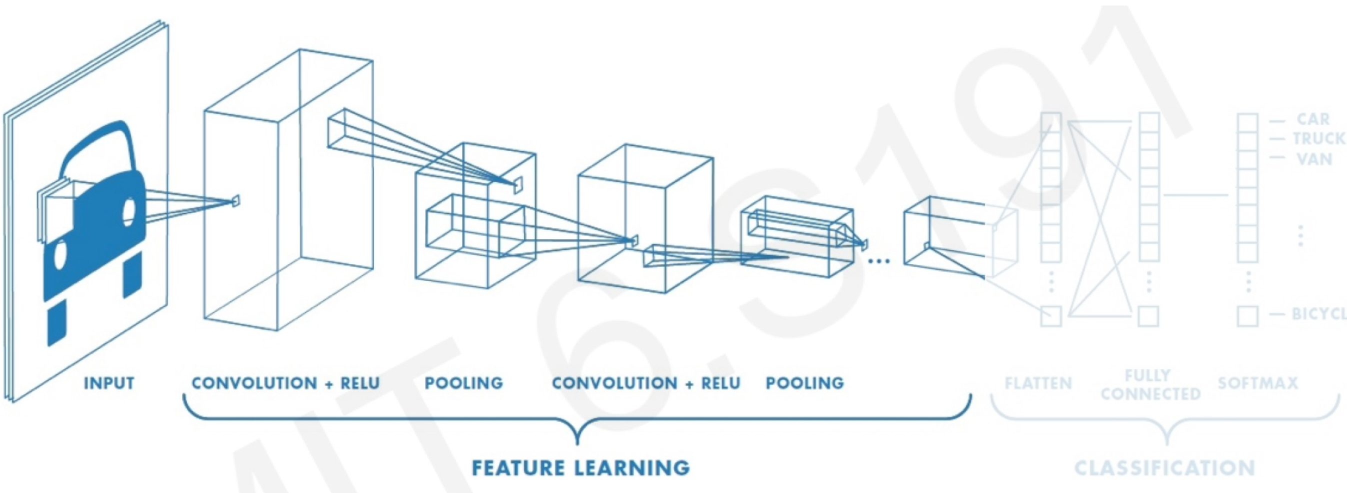
Shallow models



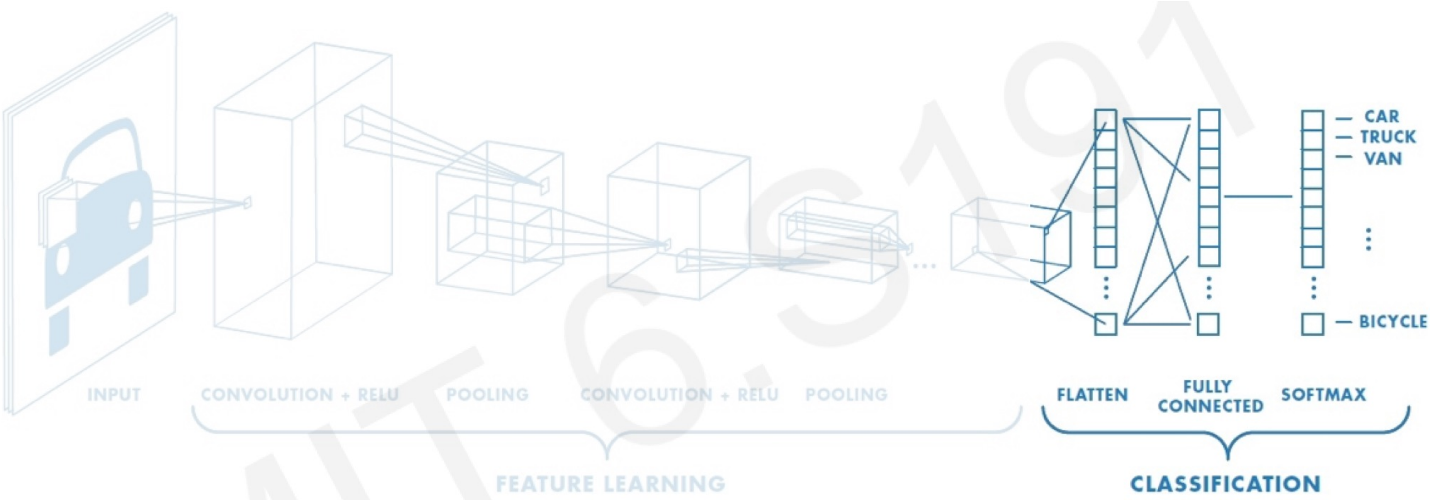
Deep models



A typical CNN

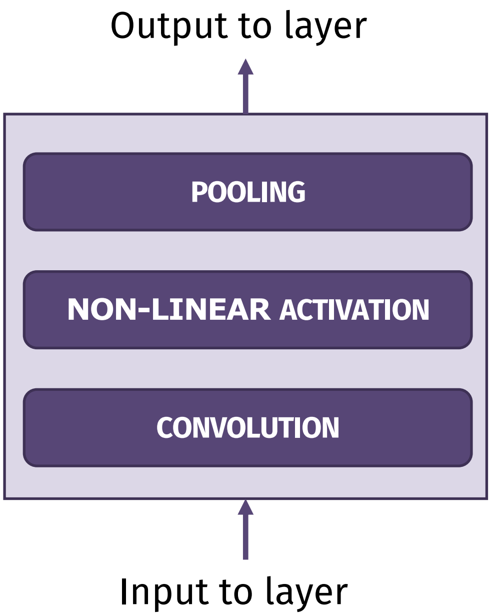


A typical CNN



$$softmax(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}$$

A typical CNN layer



Replaces the output at a certain location with a summary statistic of nearby outputs

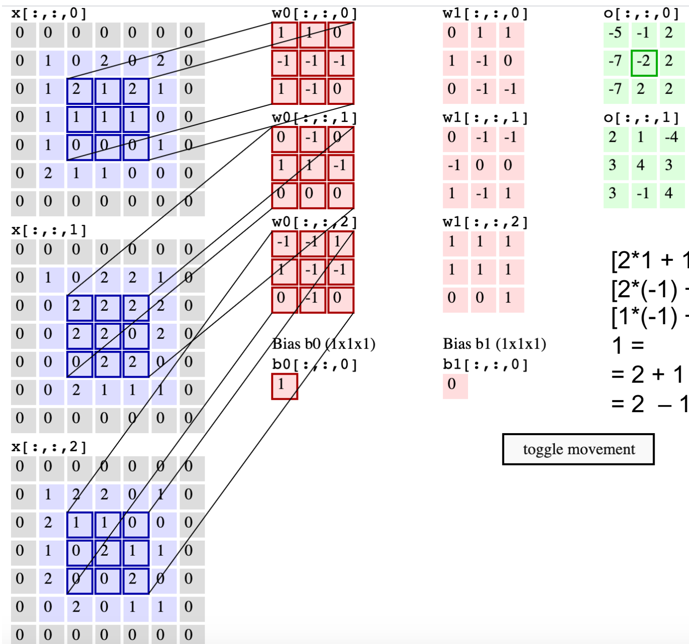
Sometimes called the **detector stage**

Produces a set of linear activations

Parameter sharing

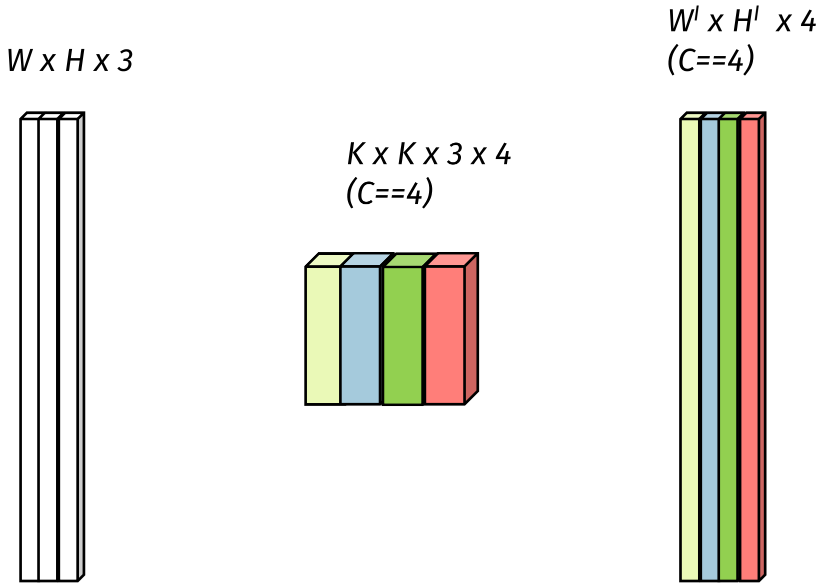
- As the kernel slides on the image**, it is able to capture the same property in different image regions**
- Multiple feature detectors can be used to capture different image properties
- See the demo here: https://cs231n.github.io/convolutional-networks/

An example from the animation



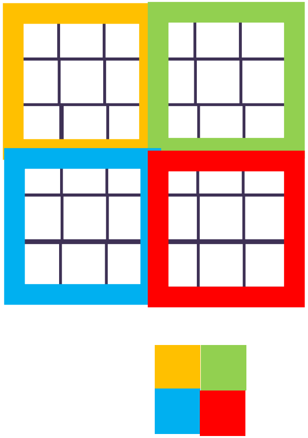
[2*1 + 1*1 + 1*(-1) + 1*(-1) + 1*(-1)]+
[2*(-1) + 2*1 + 2*1]+
[1*(-1) + 1*(-1) + 2*(-1) + 1*(-1)]+
1 =
= 2 + 1 - 1 – 1 - 1 - 2 + 2 + 2 – 1 – 1 – 2 – 1+1
= 2 – 1 - 1 – 1 – 1 = -2

Another animation to clearly understand



Output feature size of conv layers

- Three parameters control the size of the output of a layer
  - **Depth**, the number of filters (kernels) of the layer
  - **Stride,** the step used to slide the filter on the input
    - When stride > 1 we are down-sampling the input data
    - **Tiling** refers to the special case where stride = kernel span
  - **Padding** to enlarge the input and allow for kernels application in each one of the (original) point



Output features size of conv layers

Size BEFORE convolution

Kernel size

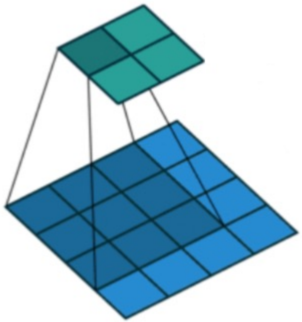
Padding

$$O = \frac{W - K + 2P}{S} + 1$$

Size AFTER convolution

Stride

Output features size of conv layers
Examples



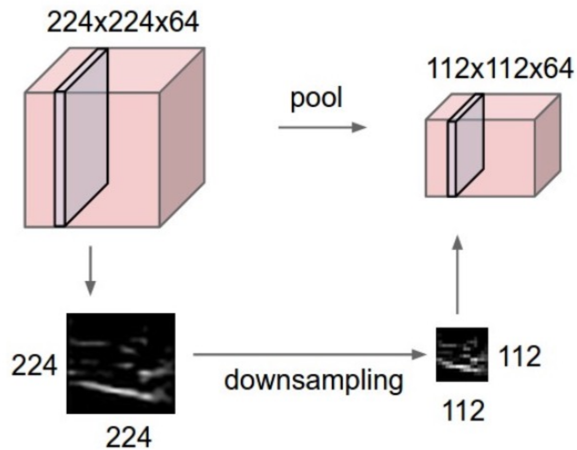
No padding, stride 1

$$O = \frac{\overset{4}{W} - \overset{3}{K} + 2\overset{0}{P}}{\underset{1}{S}} + 1$$

O = 2

Pooling and invariance

- It is a way to further reduce the dimensionality of the description
- It provides invariance to small shifts of the inputs
- Pooling functions:
  - **Average pooling**: average activation of the convolutional layer
  - **Max pooling**: max activation of the convolutional layer



Pooling

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

Max pooling

8	5
6	3

Average pooling

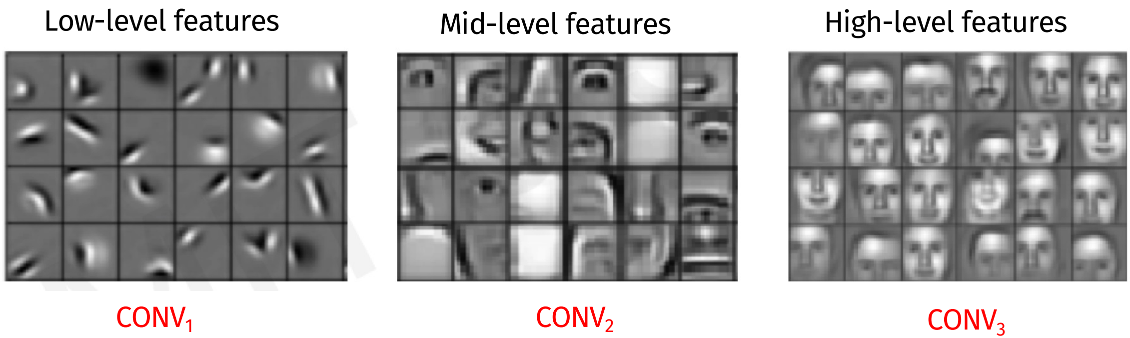
3.8	2.3
3	1.2

Pooling can help with local invariance although some information is lost

No parameter to be estimated here!

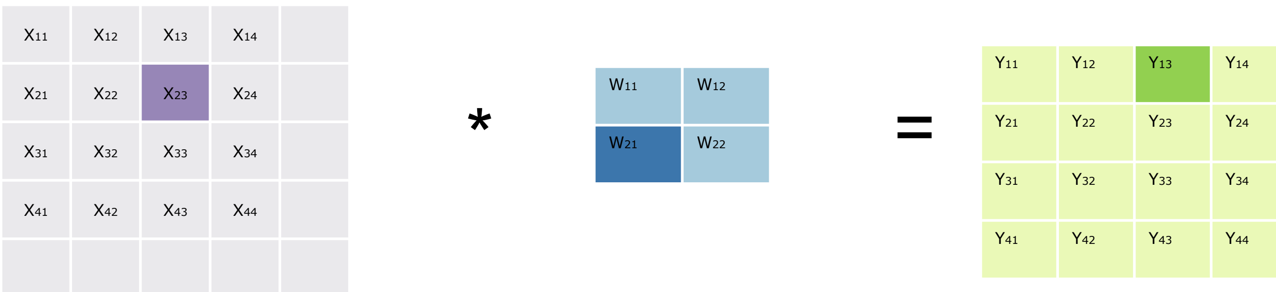
Towards image classification

- Is it possible to learn the most appropriate representation, possibly with a hierarchy, directly from the data?



- From features engineering to features learning

Forward and backpropagation in CNNs



$$Y_{11} = X_{11}W_{11} + X_{12}\,W_{12} + X_{21}\,W_{21} + X_{22}\,W_{22}$$
$$Y_{12} = X_{12}W_{11} + X_{13}\,W_{12} + X_{22}\,W_{21} + X_{23}\,W_{22}$$
$$Y_{13} = X_{13}W_{11} + X_{14}\,W_{12} + X_{23}\,W_{21} + X_{24}\,W_{22}$$
$$Y_{14} = X_{14}W_{11} + X_{24}\,W_{21}$$
$$\ldots\ldots$$

Forward and backpropagation in CNNs

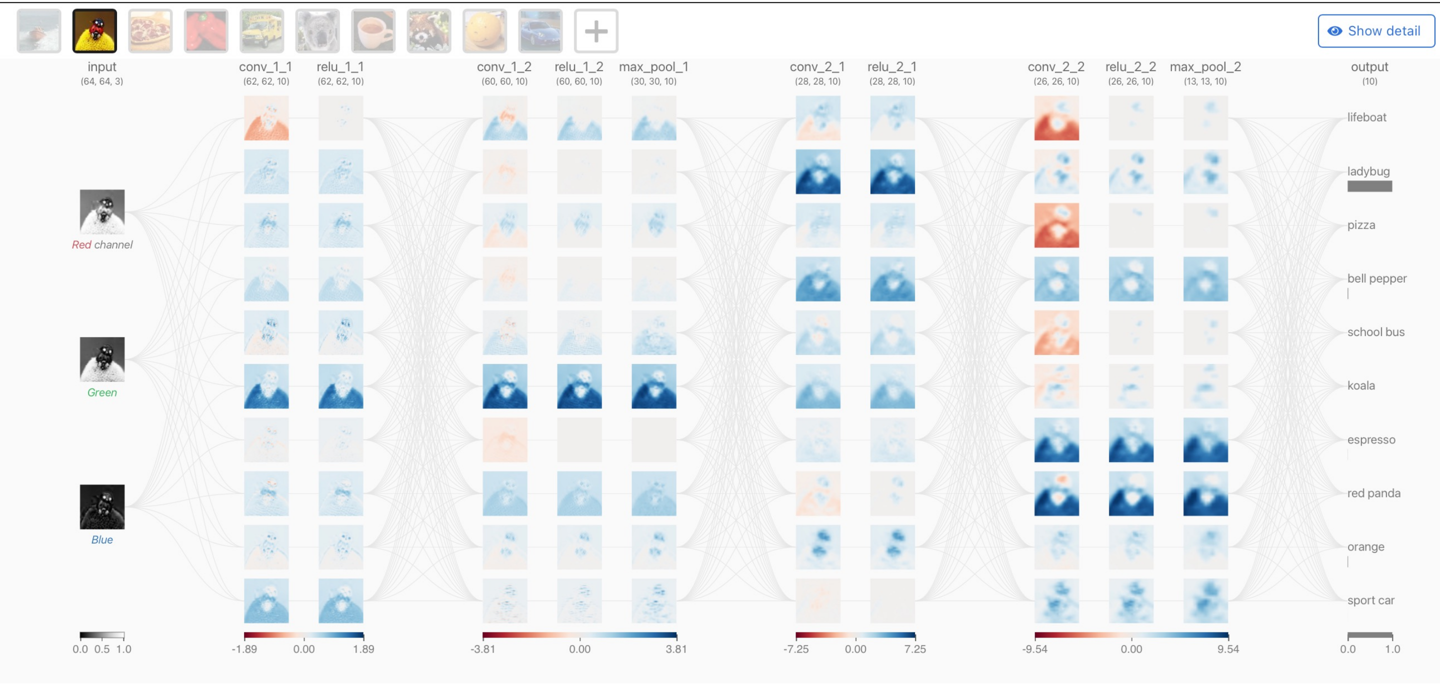
- A specific value in the filter has impact on every output values, thus we need to sum up all the contributions as follow

$$\frac{\partial L}{\partial W(a', b')} = \sum_{r=0}^{N_1-1} \sum_{c=0}^{N_2-1} \frac{\partial L}{\partial Y(r, c)} \frac{\partial Y(r, c)}{\partial W(a', b')}$$

This is a convolution!

A nice visualization

https://poloclub.github.io/cnn-explainer/



UniGe

MaLGa