UniGe | MaLGa

Convolutional Neural Networks

Deep Learning: A hands-on introduction

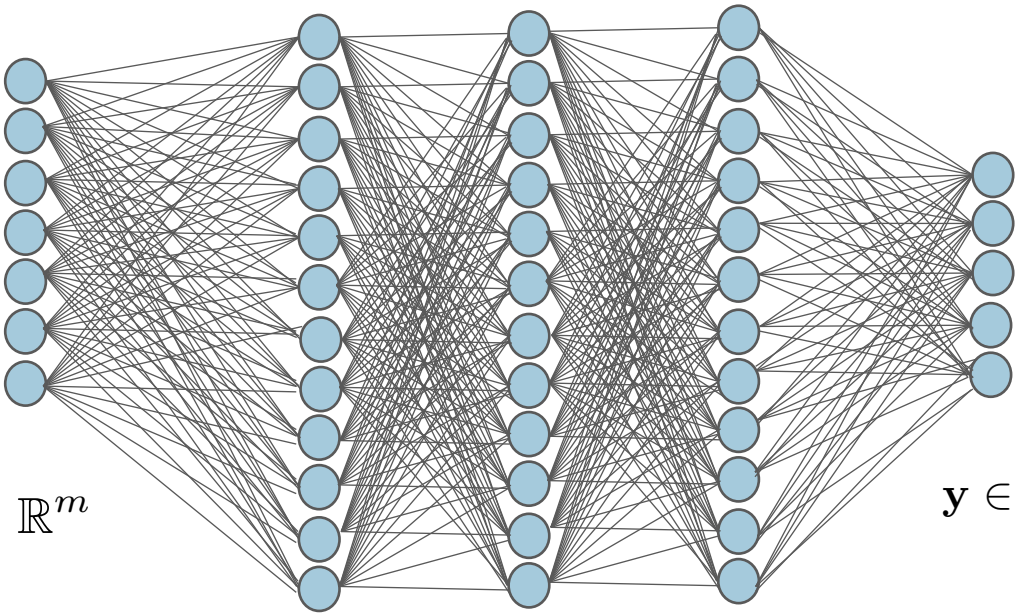
A course offered in the PhD program in Computer Science and Systems Engineering

Nicoletta Noceti
Machine Learning Genoa Center – University of Genoa

UniGe | MaLGa

A refresh

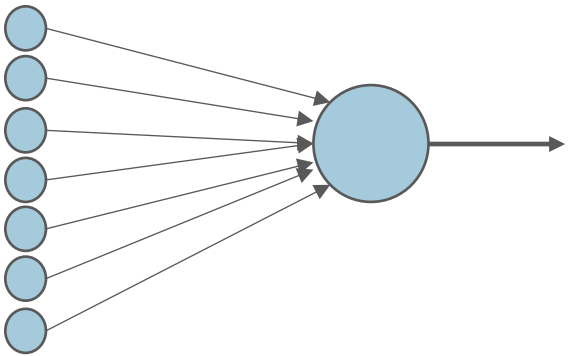
Deep Neural Network Recap



$\mathbf{x} \in \mathbb{R}^m$

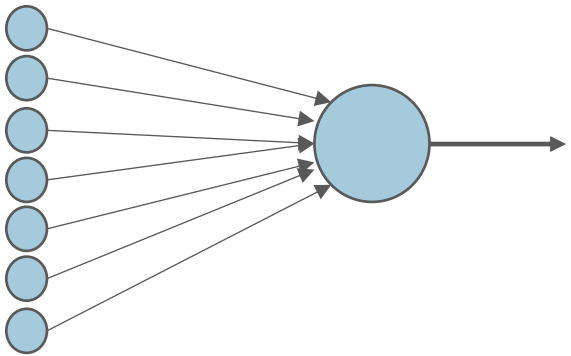
$\mathbf{y} \in \mathbb{R}^n$

The role of a neuron



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

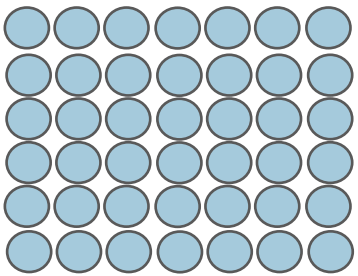
The role of a neuron



- 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

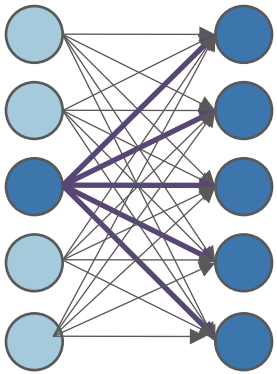
- CNNs leverage three important principles:
  - **Sparse interaction**
  - **Parameter sharing**
  - **Equivariant representations**

Sparse interactions

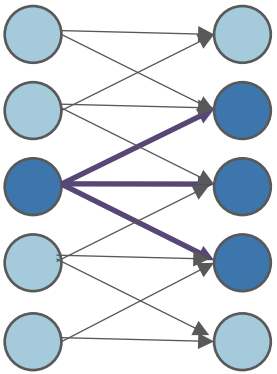
- In traditional DNNs every output unit interacts with every input unit
- In CNNs the kernel is typically smaller than the input, providing sparse interaction (also referred to as sparse connectivity or sparse weights)
- The pros is that we have to learn f ewer parameters, with improvements in
  - Memory 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



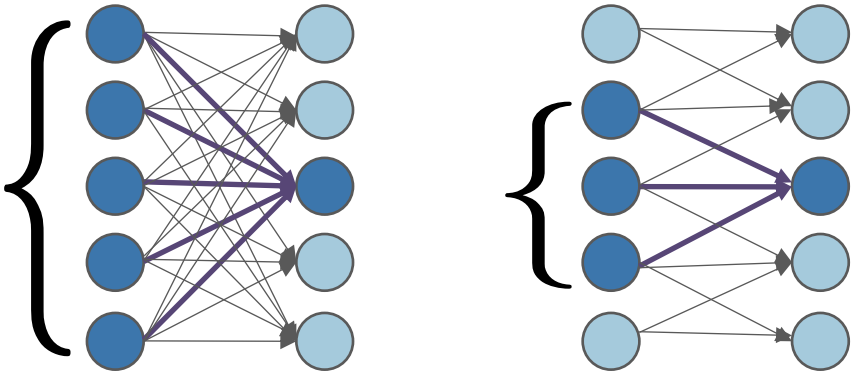
DNN



CNN

Sparse interaction: intuitions

Focus on the output unit

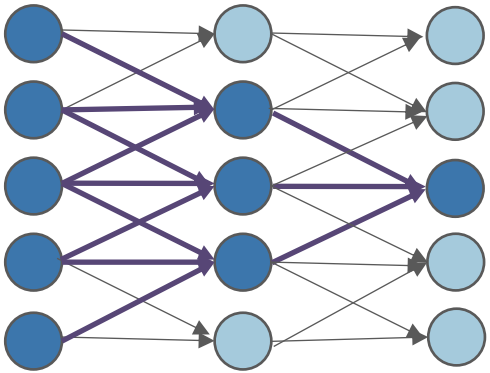


DNN *CNN*

Receptive fields

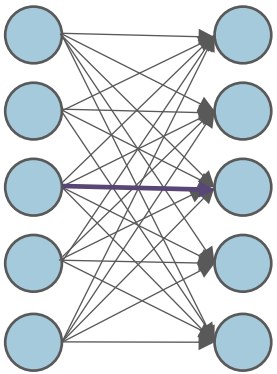
Sparse interaction: intuitions

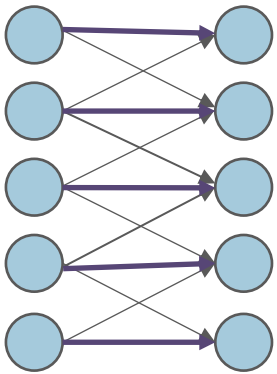
- Units in deeper layers may indirectly interact with a large portion of the input: the network can still describe complex interactions between many variables



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





Equivariant representations

- The specific type of parameter sharing causes the layers to show equivariance to translation
- A function $f(x)$ is equivariant to a function $g(x)$ if

$$f(g(x)) = g(f(x))$$

- In this case g is a function translating (shifting) the input, while f is the convolution operator

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}

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

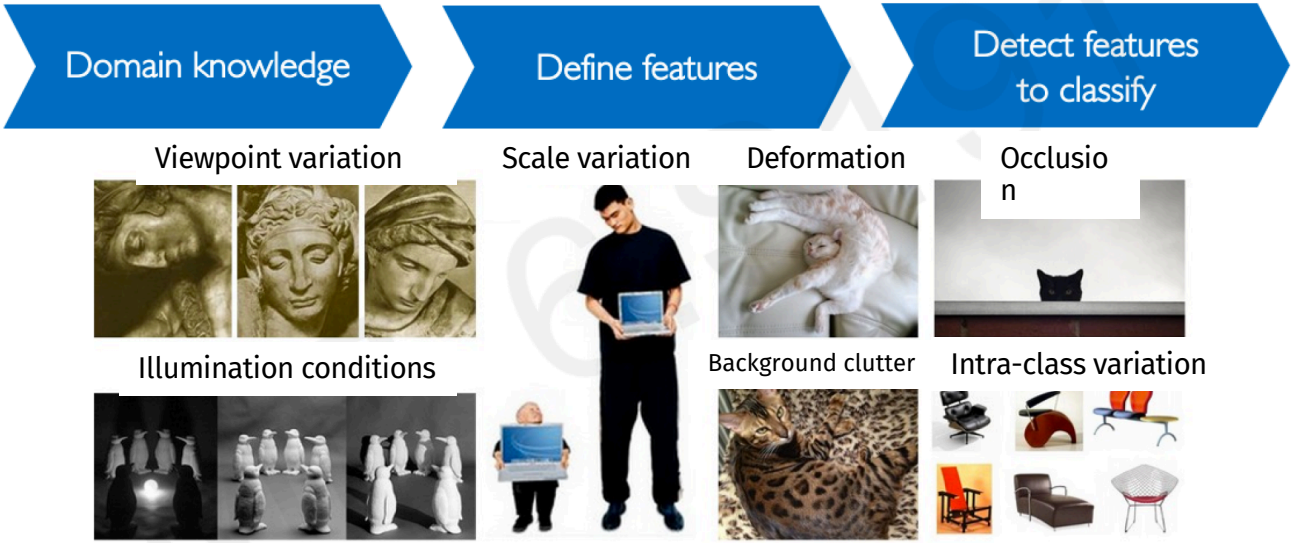
UniGe | MaLGa

Let’s focus on images

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

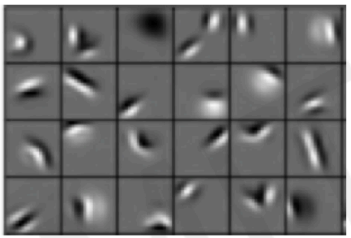
Towards image classification



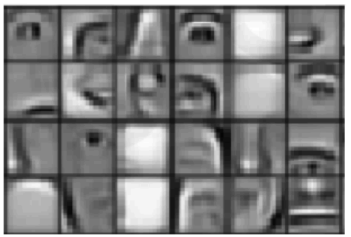
Towards image classification

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

Low-level features



Mid-level features



High-level features

- From features engineering to features learning

Coding-pooling image representations

We compute a set of local descriptors $\mathcal{X}$ associated with an image, so that card($\mathcal{X}$)=n $\quad \mathcal{X} \subset \mathbb{R}^d$

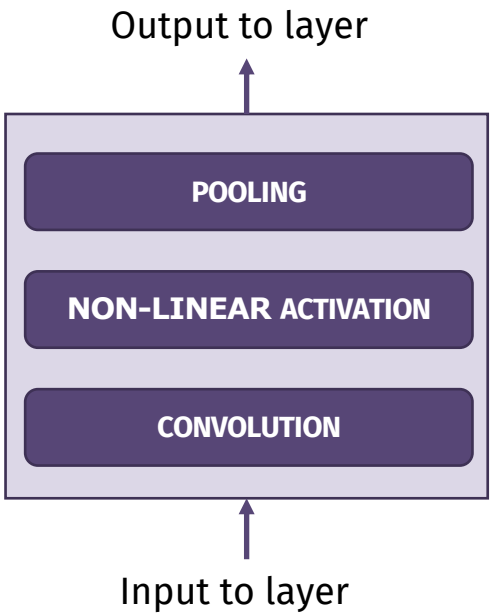
Coding or Embedding step $\quad \phi : \mathbb{R}^d \to \mathbb{R}^D$

encodes each key point $\quad x \in \mathcal{X} \quad x \to \phi(x)$

Pooling or Aggregating step computes a single vector from a set of through an aggregating function $\quad \{\phi(x_1), \ldots, \phi(x_n)\}$

for instance a sum $\quad \psi(\mathcal{X}) = \sum_{x \in \mathcal{X}} \phi(x)$

A typical CNN layer



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

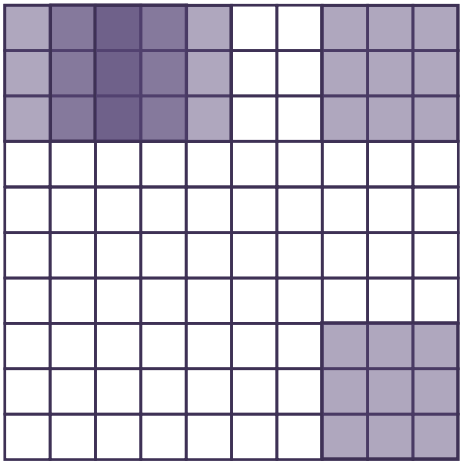
Sometimes called the **detector stage**

Produces a set of linear activations

2D convolution

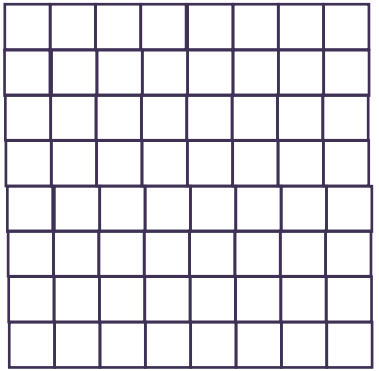
A "feature detector" (kernel) slides over the inputs to generate a feature map

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



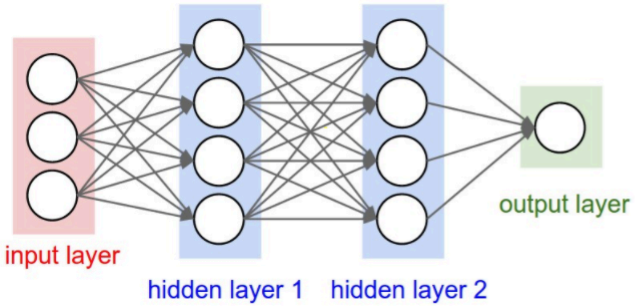
Input tensor of size 10x10

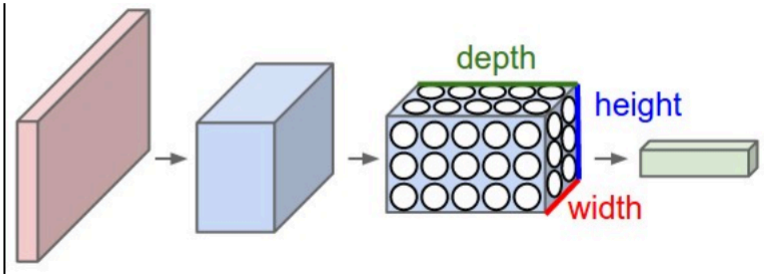
Kernel of size 3x3



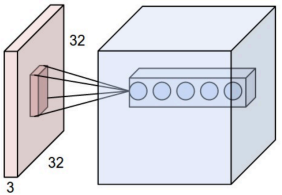
Output tensor of size 8x8

Neurons organization

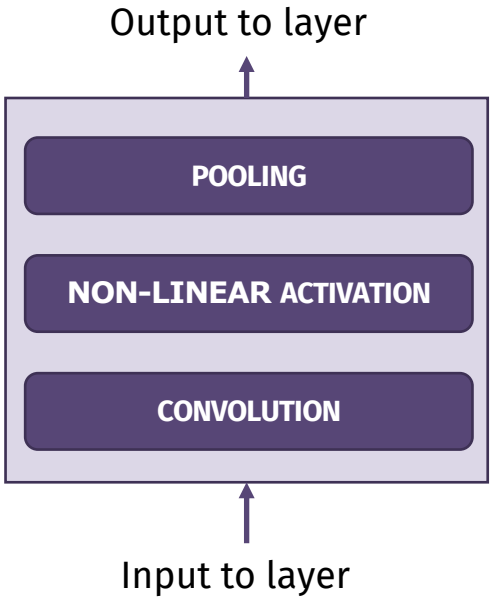




From https://cs231n.github.io/convolutional-networks/



Size: an example



Downsampling, e.g. rpws/2 x cols/2 x depth

rows x cols x depth

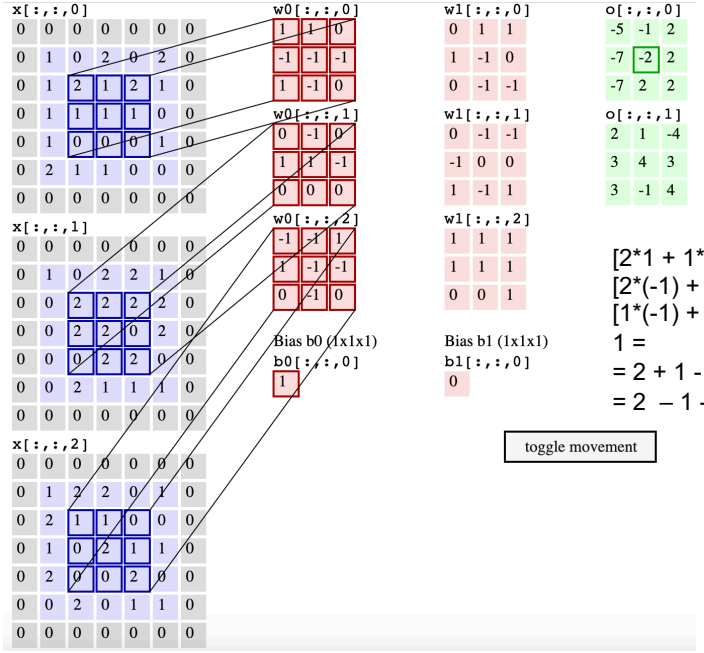
rows x cols x depth, if depth is the number of filters used at this layer

Raw pixels of an image: rows x cols x 3

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

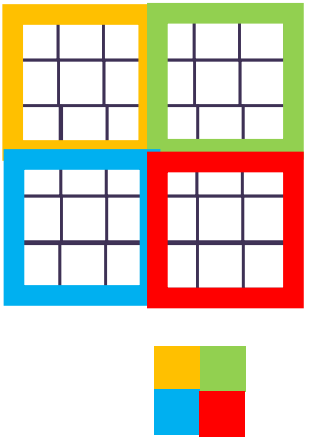


[2*1 + 1*1 + 1*(-1) + 1*(-1) + 1*(-1)]+
[2*(-1) + 2*1 + 2*1]+
[1*(-1) + 1*(-1) + 2*(-1) + 1*(-1)]+
1 =
= 2 + 1 - 1 – 1 - 1 - 2 + 2 + 2 – 1 – 1 – 2 – 1+1
= 2 – 1 - 1 – 1 – 1 = **-2**

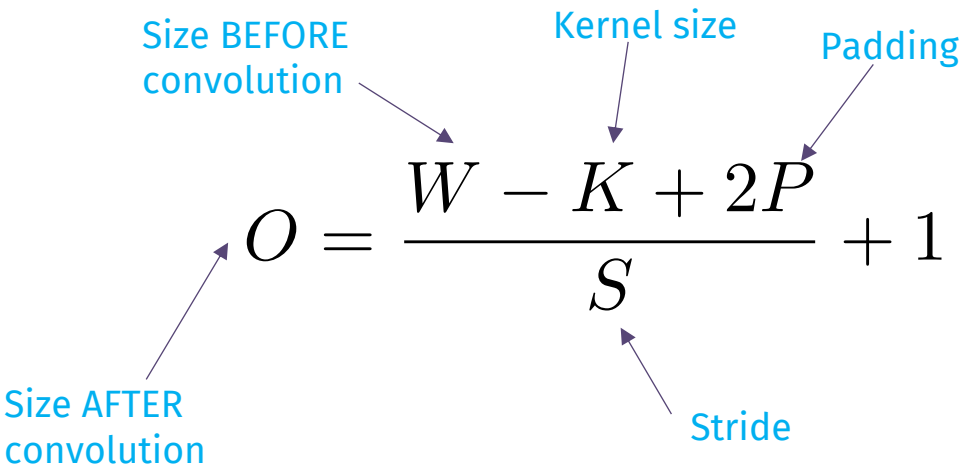
UniGe | MaLGa

Output feature size of conv layers

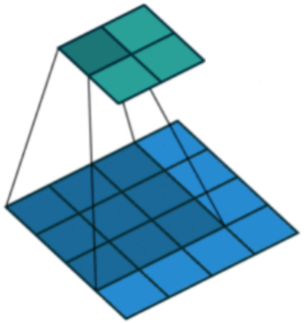
- Three parameters control the size of the output of a layer
  - **Depth**, the number of filters (kernels) of the layer
  - **Stride,** the step used to slide the filter on the input
    - When stride > 1 we are down-sampling the input data
    - **Tiling** refers to the special case where stride = 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



Output features size of conv layers
Examples

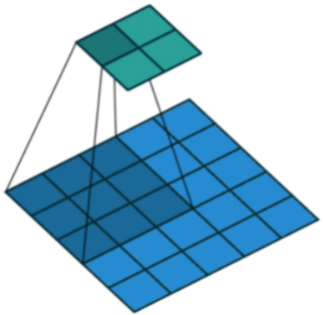


No padding, stride 1

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

4 3 0
2 1

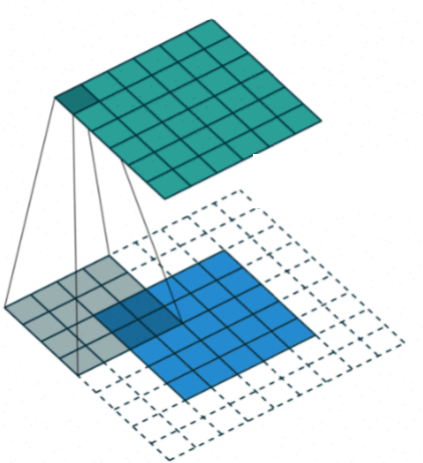
Output features size of conv layers
Examples



No padding, stride 2

$$O_2 = \frac{W^5 - K^3 + 2P^0}{S_2} + 1$$

Output features size of conv layers
Examples



Padding 2, stride 1

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

UniGe | MaLGa

Output features size of conv layers
A summary

- Input size: W_1 x H_1 x D_1
- Parameters:
  - Number of kernels N
  - Kernel size K
  - Stride S
  - Padding P

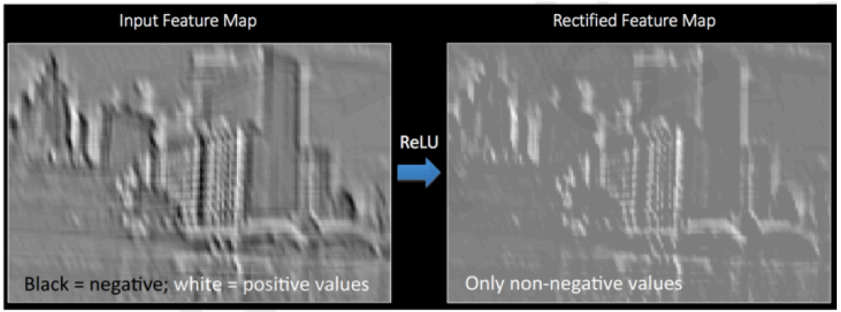
- Output size: W_2 x H_2 x D_2

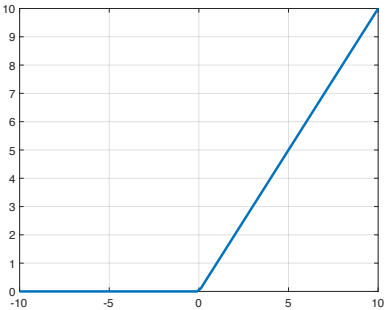
  where
- $W_2 = (W_1 - K + 2P)/S + 1$
- $H_2 = (H_1 - K + 2P)/S + 1$
- $D_2 = N$

- Number of weights per filter: K x K x D_1
- Number of parameters in total:
  - K x K x D_1 x N weights
  - N biases

Introducing non-linearities

- A typical choice is ReLU: it is applied after each convolutional layer and only preserves non-negative values

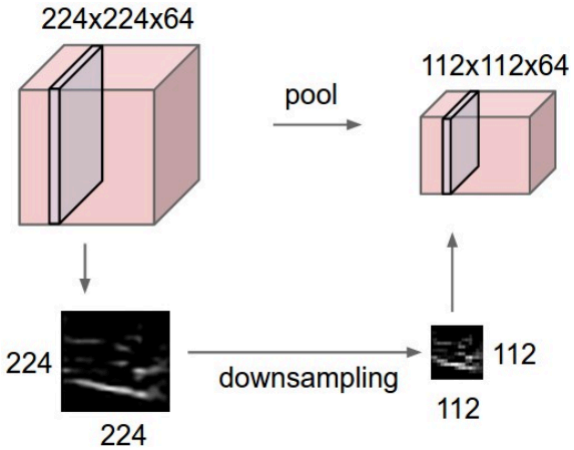




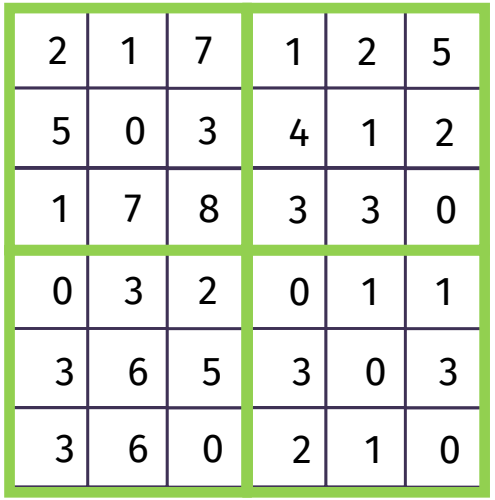
$$f(x) = max(0, x)$$

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



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!

Output features size of pooling layer

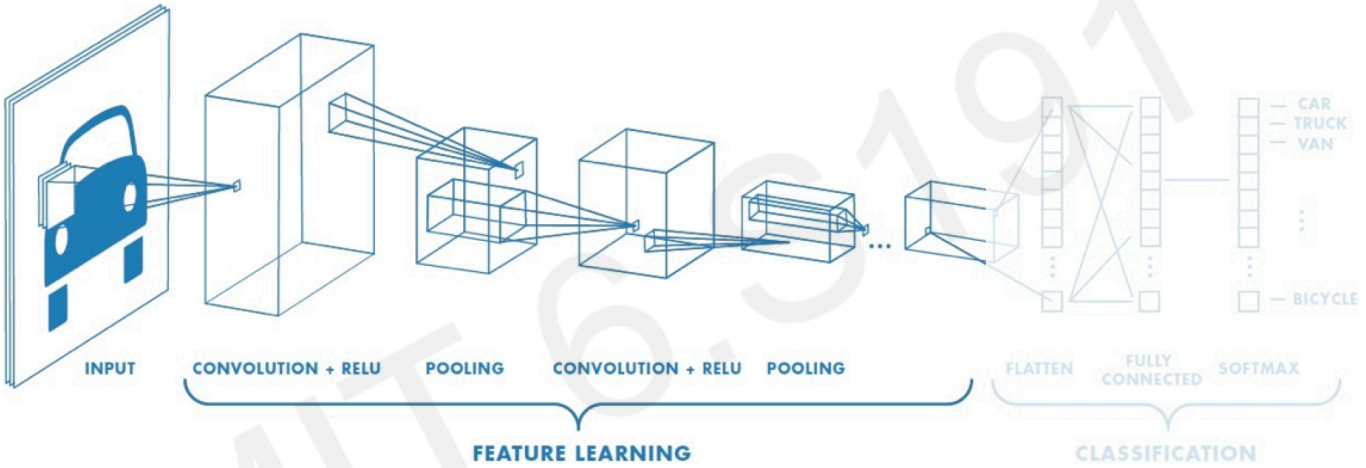
- Input size: W_1 x H_1 x D_1
- Parameters:
  - Window size H
  - Stride S

- Output size: W_2 x H_2 x D_2

  where
- $W_2 = (W_1 - H)/S + 1$
- $H_2 = (H_1 - H)/S + 1$
- $D_2 = D_1$

- Number of weights per filter: K x K x D_1
- Number of parameters in total:
  - K x K x D_1 x N weights
  - N biases

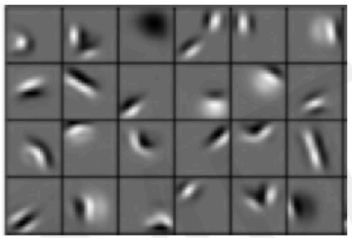
A typical CNN



Towards image classification

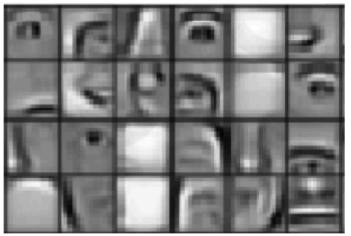
- Is it possible to learn the most appropriate representation, possibly with a hierarchy, directly from the data?

Low-level features



$CONV_1$

Mid-level features



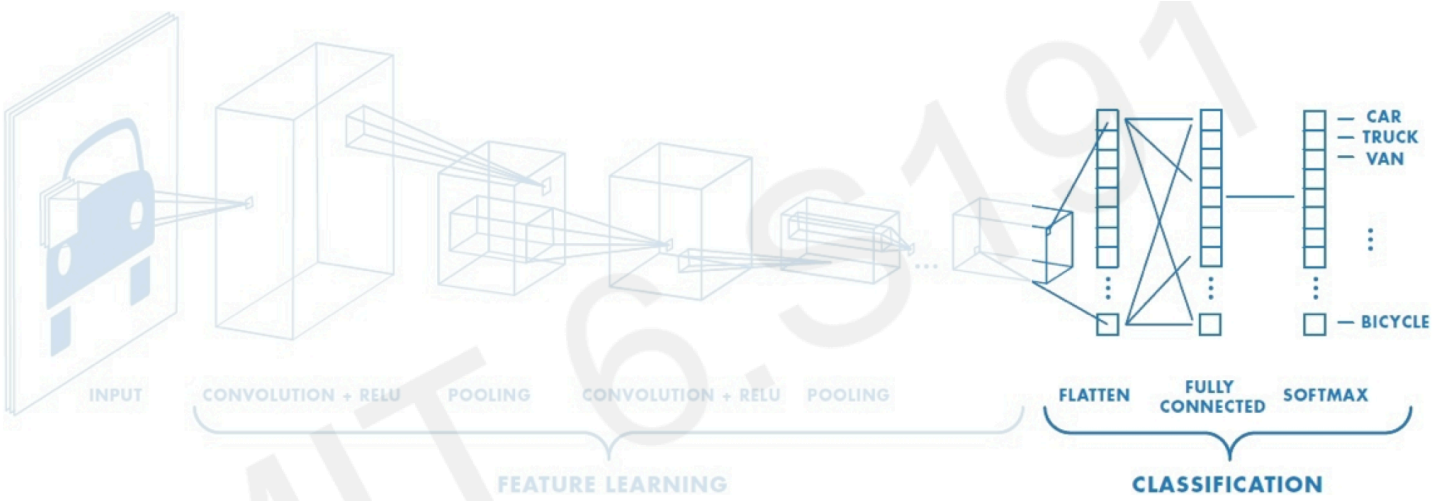
$CONV_2$

High-level features

$CONV_3$

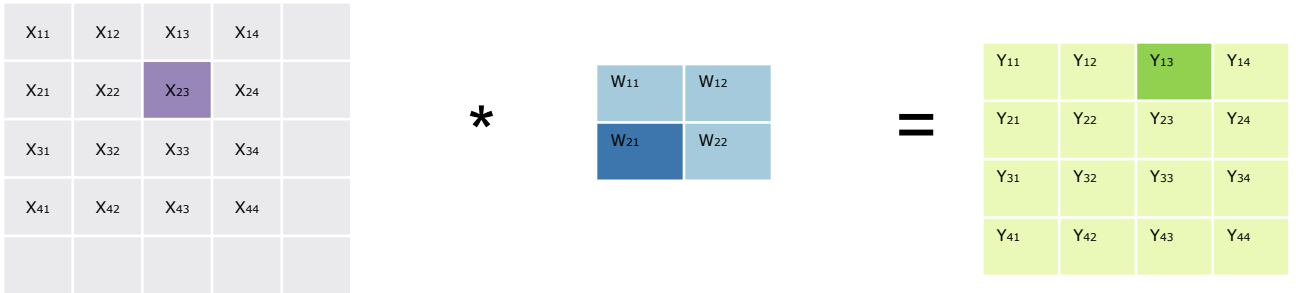
- From features engineering to features learning

A typical CNN



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

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!

UniGe | MaLGa

Relations with neuroscience

Neuroscientific basis for convolutional networks

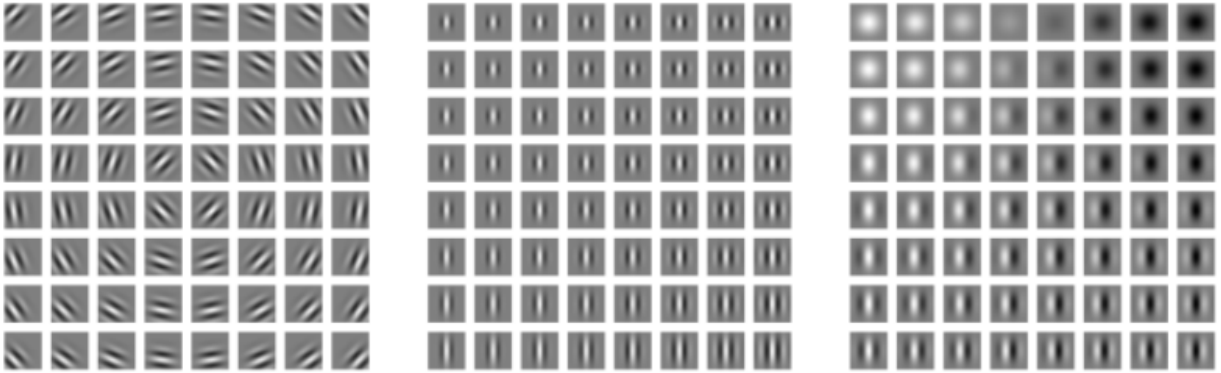
- Some of the design principles of Neural Networks have been drawn from neuroscience
- We now briefly discuss some of the connections between CNNs and a simplified version of the brain functions
- We refer to the primary visual cortex (V1 area), the first one in the brain performing some significantly advanced processing of visual input

V1 area & CNNs

- V1 is arranged in a spatial map
- V1 contains simple cells , that an to some extent be characterized by a linear function (as for the detection step in CNNs)
- V1 also contains complex cells, that show some level of invariance to some changes in the visual input
- It is generally believed that the same basic principles apply to other areas in the visual stream, repeatedly

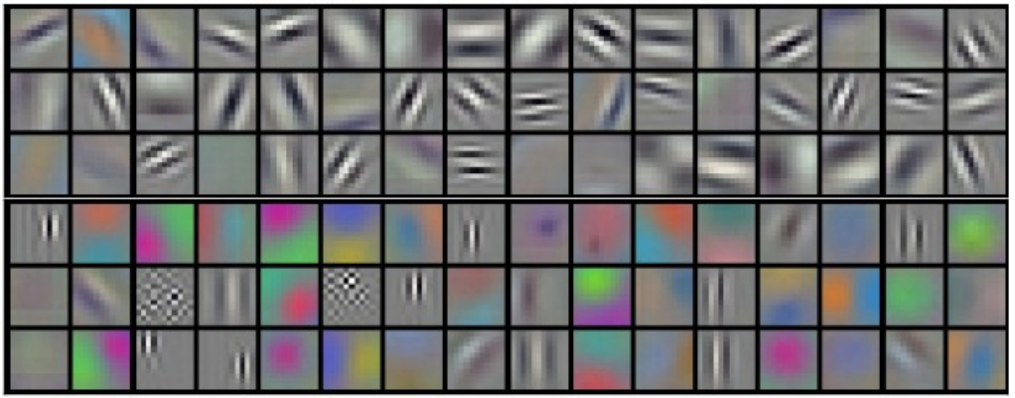
Again on V1 cells

- Experiments showed that most V1 cells have weights that can be described by Gabor functions



What about CNN weights?

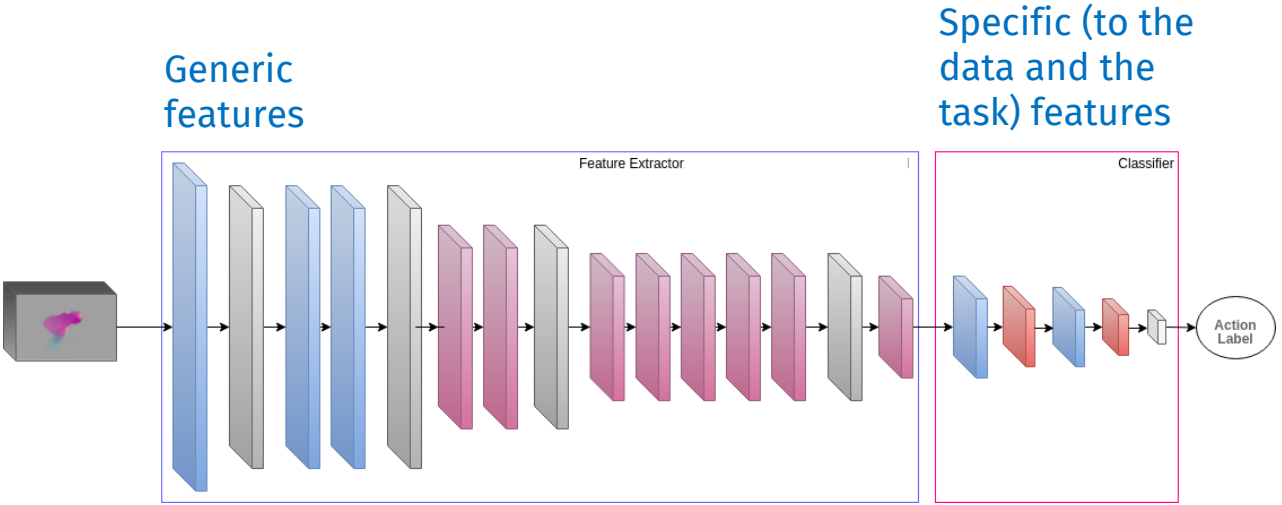
- At the very first layers the weights learnt by a CNN on natural images are very similar to Gabor filters



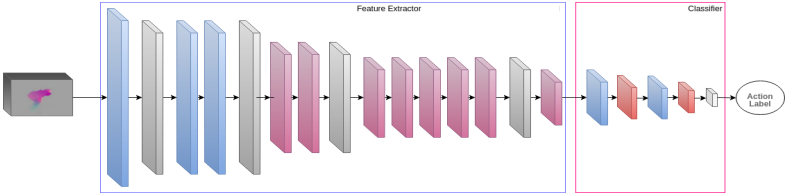
UniGe | MaLGa

Some further discussions

On the properties of weights learnt by convolutional layers



Transfer learning



- It refers to the possibility of exploiting knowledge in terms of pre-trained models that can be used on different data and tasks (with some constraint)

- Fine-tuning is a well-assessed procedure in which the weights are somehow adapted to the new problem/data starting from the pre-trained model

- This may imply a domain shift (also known as covariate shift) problem, due to the fact that the data distribution may change as you change the problem/data

CNN training

- Very data hungry and computationally intensive
- One of the trick for coping with data lack us data augmentation
  - The idea is to generate more data by applying some transformation to the image
  - Examples: rotations, scaling, lighting conditions,...
- An alternative is to use synthetic data, but the model may be affected by domain shift issue (and thus it would need a specific domain adaptation strategy)

UniGe | MaLGa

Popular CNNs

Last years (famous) CNNs

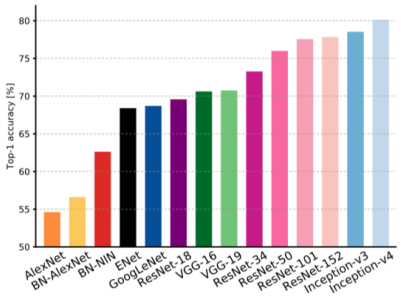


Figure 1: **Top1 *vs.* network.** Single-crop top-1 validation accuracies for top scoring single-model architectures. We introduce with this chart our choice of colour scheme, which will be used throughout this publication to distinguish effectively different architectures and their correspondent authors. Notice that networks of the same group share the same hue, for example ResNet are all variations of pink.

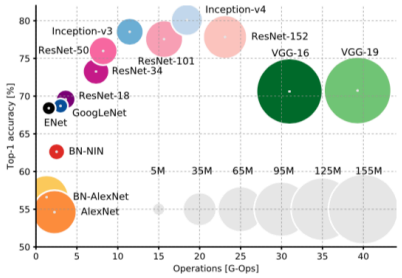
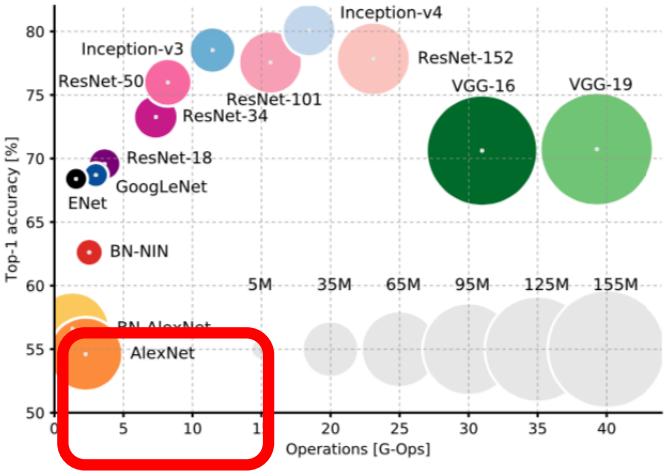


Figure 2: **Top1 *vs.* operations, size $\propto$ parameters.** Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters; a legend is reported in the bottom right corner, spanning from 5×10^6 to 155×10^6 params. Both these figures share the same y-axis, and the grey dots highlight the centre of the blobs.

Canziani, A., Paszke, A., & Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.

Last years (famous) CNNs



AlexNet (2012)

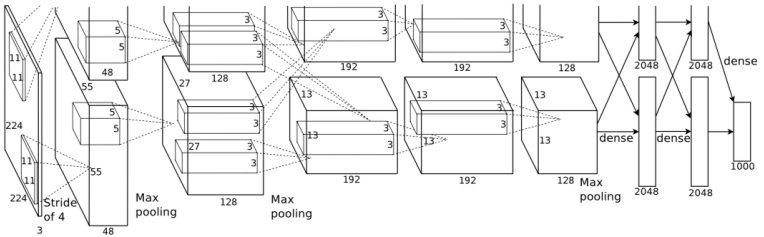
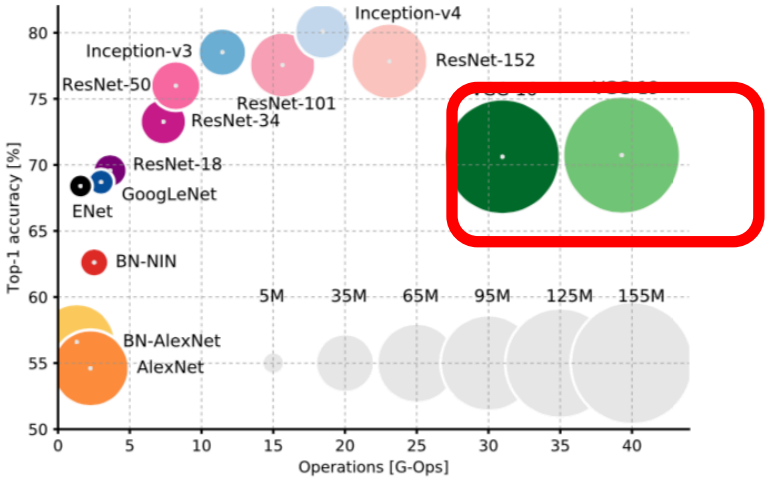


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

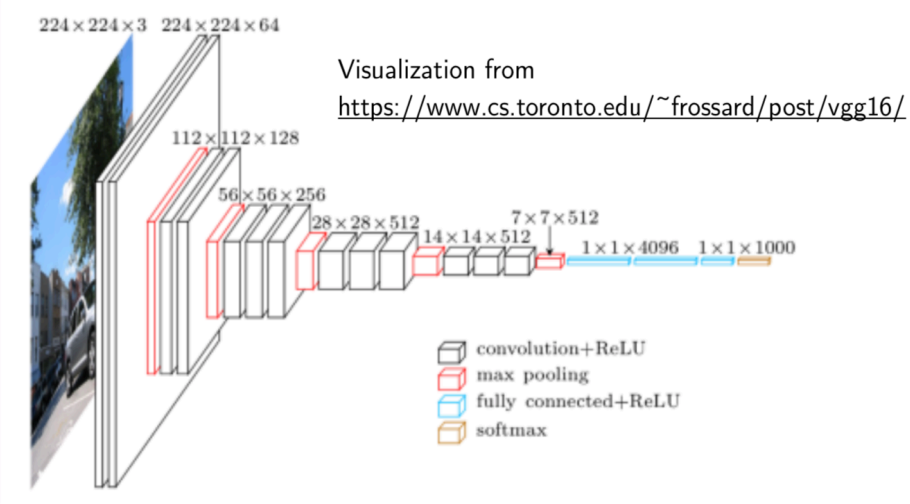
The first work that popularized Convolutional Networks in Computer Vision
Top 5 performance on ImageNet

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Last years (famous) CNNs



VGG (2014)



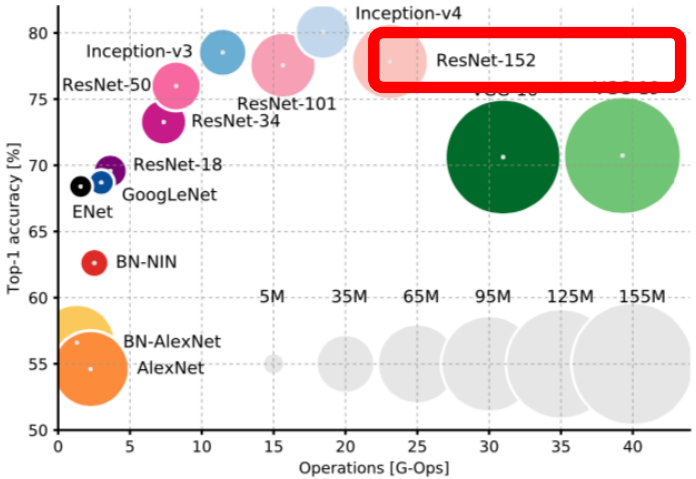
Visualization from
https://www.cs.toronto.edu/~frossard/post/vgg16/

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

It has been shown here that the depth of the network is a critical component for good performance

Cons: lots of parameters, mosyt of them being in the first fully connected layer. It was then found that these FC layers can be removed with no performance downgrade, significantly reducing the number of necessary parameters

Last years (famous) CNNs



ResNet (2015)

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.

With the very simple trick of allowing skip connections, ResNet allows to implement very, very deep architectures

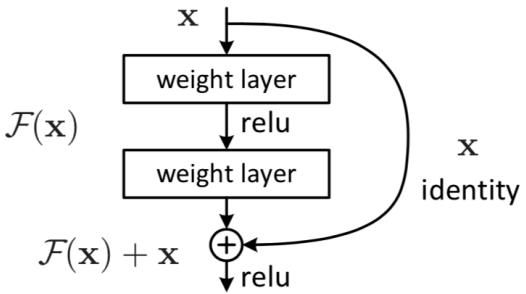
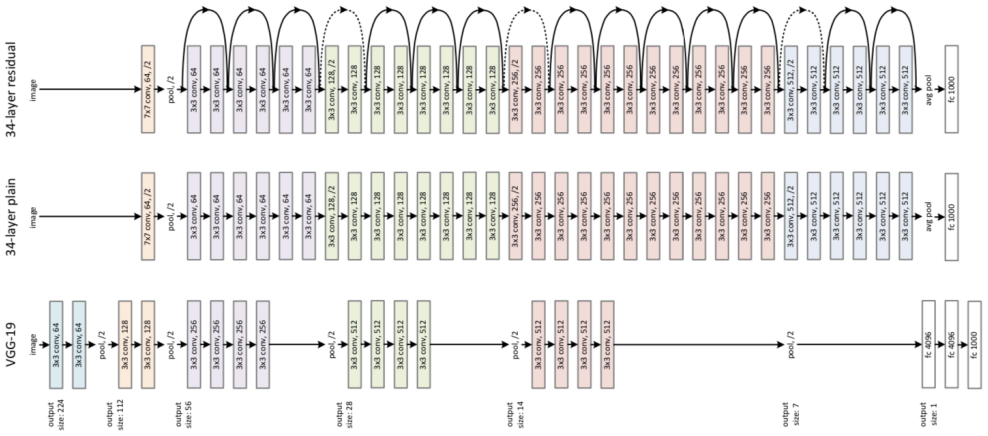


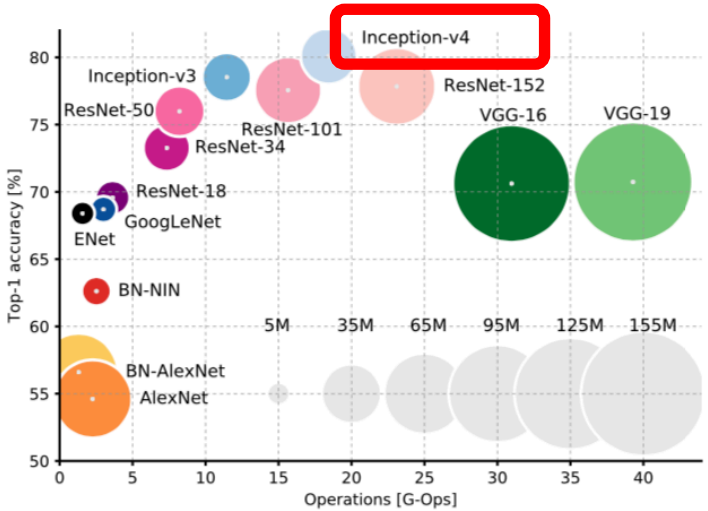
Figure 2. Residual learning: a building block.

ResNet (2015)

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition."
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.



Last years (famous) CNNs



GoogleNet/Inception 92014)

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9. 2015.

F

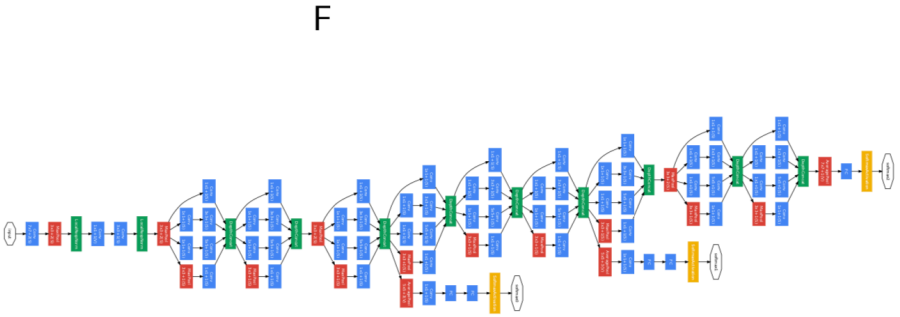
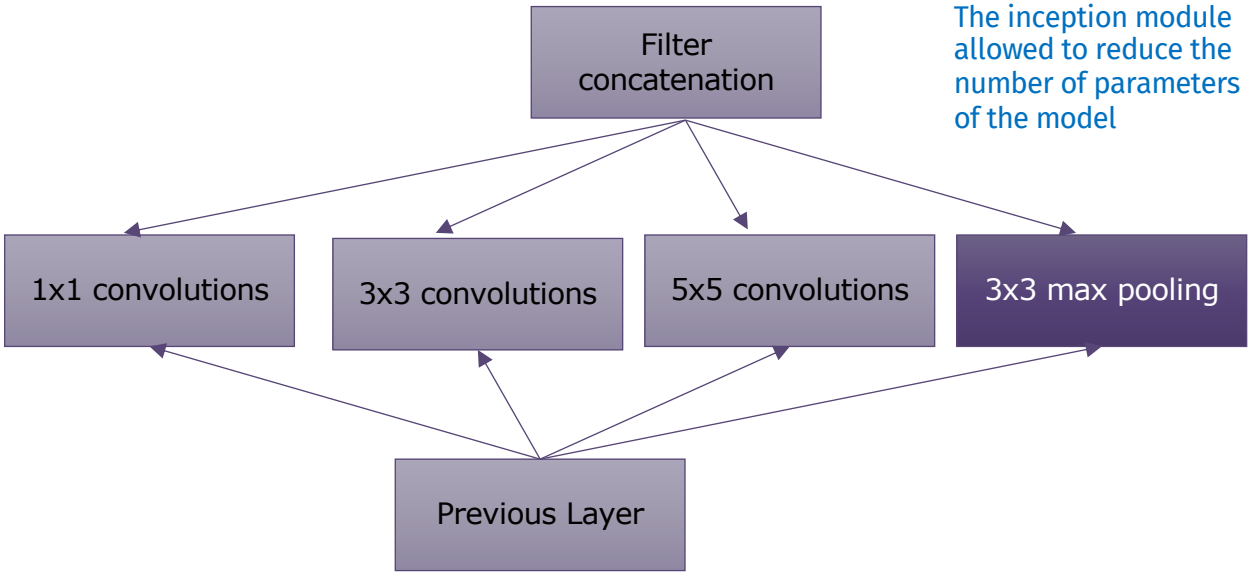


Figure 3: GoogLeNet network with all the bells and whistles.

GoogleNet/Inception (2014)



The inception module allowed to reduce the number of parameters of the model

UniGe