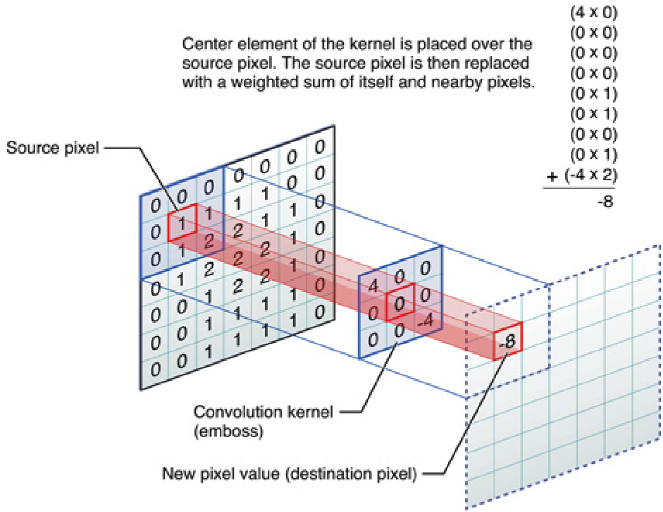
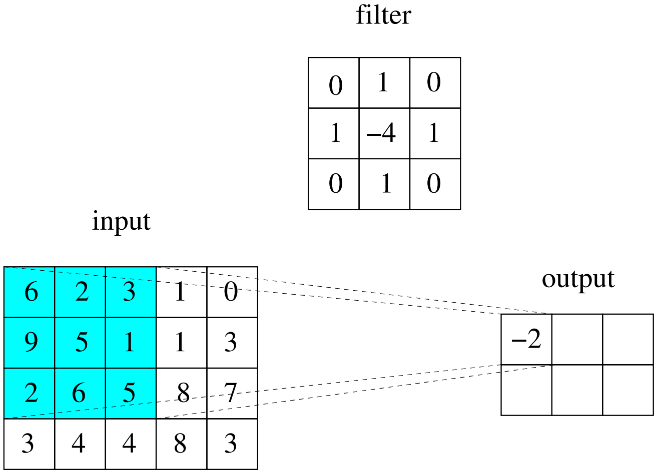
Convolutional Neural Networks

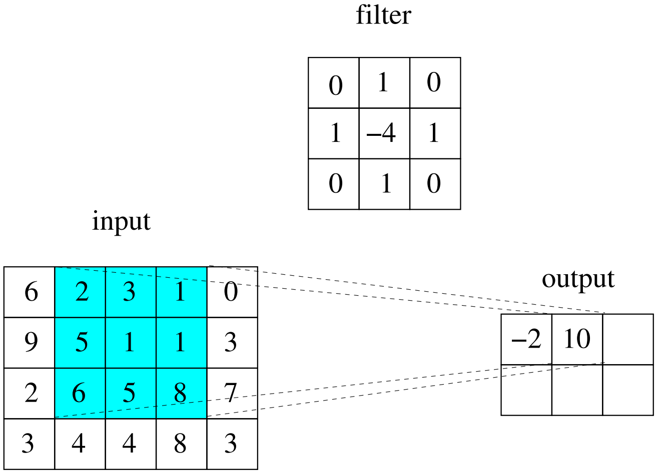
Filters and convolutions



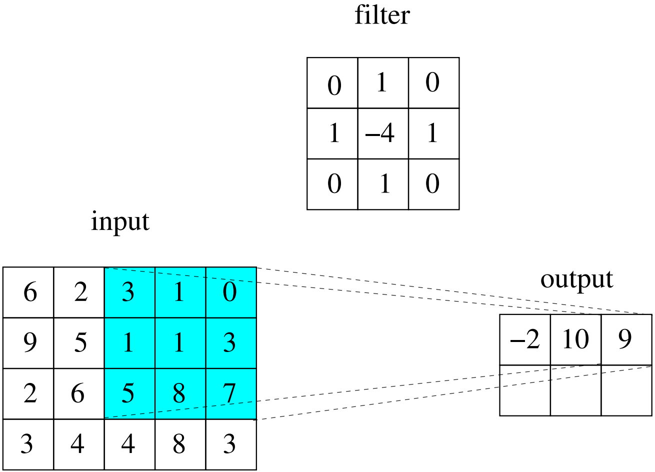
Filters and convolutions



Filters and convolutions



Filters and convolutions



Filters and convolutions

filter

0	1	0
1	–4	1
0	1	0

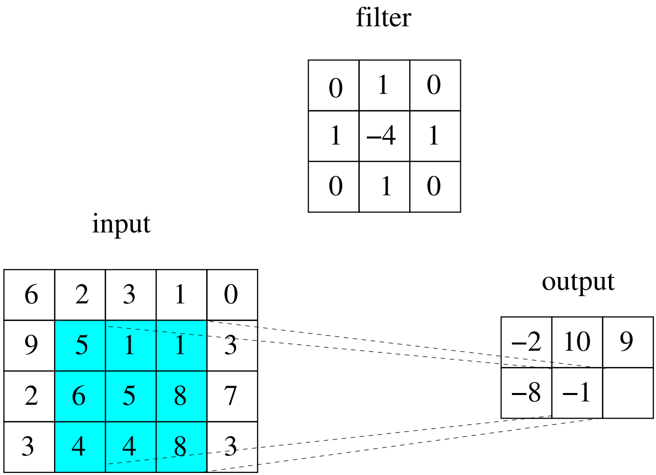
input

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

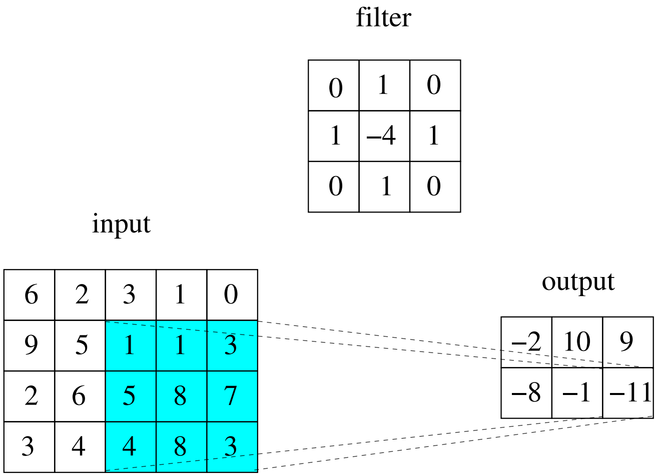
output

–2	10	9
–8		

Filters and convolutions



Filters and convolutions



Loose connectivity and shared weights

- the activation of a neuron is not influenced from all neurons of the previous layer, but only from a small subset of adjacent neurons: his receptive field
- every neuron works as a convolutional filter. Weights are shared: every neuron perform the **same trasformation** on **different areas** of its input
- with a cascade of convolutional filters intermixed with activation functions we get complex non-linear filters assembing local features of the image into a global structure.

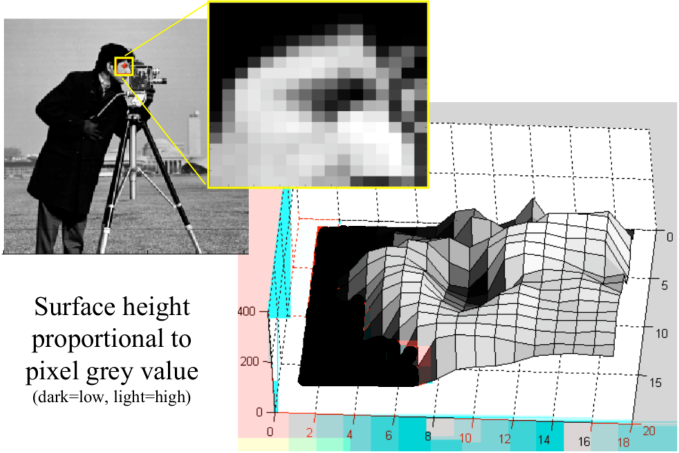
About the relevance of convolutions for image processing

Images are numerical arrays

An image is coded as a numerical matrix (array)
grayscale (0-255) or rgb (triple 0-255)

$$\begin{bmatrix} 207 & 190 & 176 & 204 & 204 & 208 \\ 110 & 108 & 114 & 112 & 123 & 142 \\ 94 & 100 & 96 & 121 & 125 & 108 \\ 95 & 86 & 81 & 84 & 88 & 88 \\ 69 & 51 & 36 & 72 & 78 & 81 \\ 74 & 97 & 107 & 116 & 128 & 133 \end{bmatrix}$$

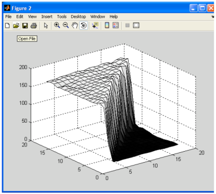
Images as surfaces

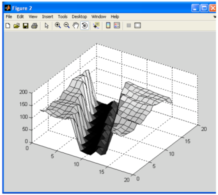


Surface height proportional to pixel grey value
(dark=low, light=high)

Interesting points

Edges, angles, ...: points where there is a discontinuity, i.e. a fast variation of the intensity





More generally, are interested to identify patterns inside the image.

The key idea is that the kernel of the convolution expresses the pattern we are looking for.

Example: finite derivative

Suppose we want to find the positions inside the image where there is a sudden horizontal passage from a dark region to a bright one. The pattern we are looking for is

$$\begin{bmatrix} -1 & 1 \end{bmatrix}$$

or, varying the distance between pixels:

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

The finite derivative at work

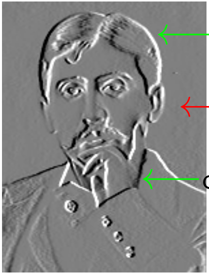
$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \longrightarrow$$

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \longrightarrow$$

Recognizing Patterns

Each neuron in a convolutional layer gets activated by specific patterns in the input image.

$$\text{pattern} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$



pattern found here

no pattern found here

opposite pattern found here

Another example: the finite laplacian

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\longrightarrow$$

Discovering patterns

But how to find good patterns?

Usual idea:

instead of using human designed pre-defined patterns, let the net learn them.

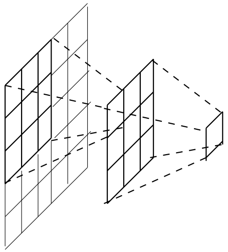
Particularly important in deep architectures, because:

- stacking kernels we **enlarge their receptive fields** (see next slide)
- adding non-linear activations we synthesize complex, **non-linear kernels**

Receptive field

The receptive field of a (deep, hidden) neuron is the dimension of the input region influencing it.

It is equal to the dimension of an input image producing (without padding) an output with dimension 1.



A neuron cannot see anything outside its receptive field!

We may also rapidly enlarge the receptive fields by means of **downsampling** layers, e.g. pooling layers or convolutional layers with non-unitarian stride

Complex (deep) patterns

The intuition is that neurons at higher layers should recognize increasingly complex patterns, obtained as a combination of previous patterns, over a larger receptive field.

In the highest layers, neurons may start recognizing patterns similar to features of objects in the dataset, such as feathers, eyes, etc.

In the final layers, neurons gets activated by "patterns" identifying objects in the category.

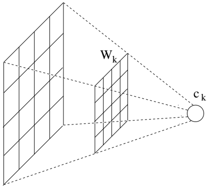
can we confirm such a claim?

Next arguments

How CNNs see the world

Visualization of hidden layers

Goal: find a way to visualize the kind of patterns a specific neuron gets activated by.



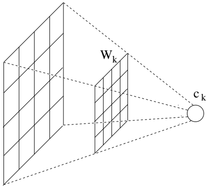
The loss function $\mathcal{L}(\theta, x)$ of a NN depends on the parameters θ and the input x.

During training, we fix x and compute the partial derivative of $\mathcal{L}(\theta, x)$ w.r.t the parameters θ to adjust them in order to decrease the loss.

In the same way, we can fix θ and use partial derivatives w.r.t. input pixels in order to syntehsize images minimizing the loss.

Visualization of hidden layers

Goal: find a way to visualize the kind of patterns a specific neuron gets activated by.



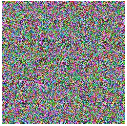
The loss function $\mathcal{L}(\theta, x)$ of a NN depends on the parameters θ and the input x.

During training, we fix x and compute the partial derivative of $\mathcal{L}(\theta, x)$ w.r.t the parameters θ to adjust them in order to decrease the loss.

In the same way, we can fix θ and use partial derivatives w.r.t. input pixels in order to syntehsize images minimizing the loss.

The "gradient ascent" technique

Start with a random image, e.g.



- do a forward pass using this image x as input to the network to compute the activation $a_i(x)$ caused by x at some neuron (or at a whole layer)
- do a backward pass to compute the gradient of $\partial a_i(x)/\partial x$ of $a_i(x)$ with respect to each pixel of the input image
- modify the image adding a small percentage of the gradient $\partial a_i(x)/\partial x$ and repeat the process until we get a sufficiently high activation of the neuron

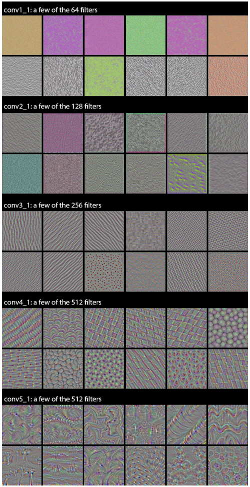
First layers

Some neurons from the first two layers of AlexNet

(Understanding Neural Networks Through Deep Visualization by A.Nguyen et al., 2015)

First features (lower picture) are very simple, and get via via more complex at higher levels, as their receptive field get larger due to nested convolutions.

First layers



For a visualization of the first layers of VGG see:

An exploration of convnet filter with keras

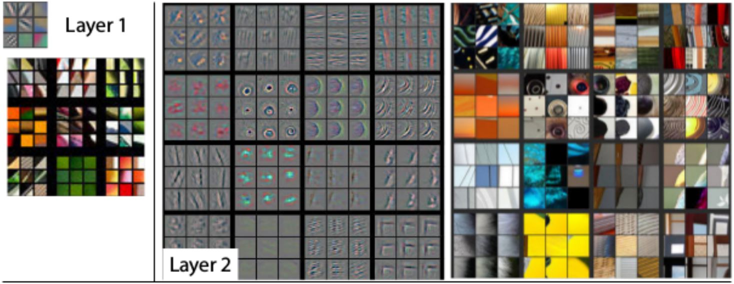
What caused the activation of this neuron **in this image**?

Instead of trying to syntesize the pattern recognized by a given neuron, we can use the gradient ascent technique to emphasize in real images what is causing its activation.

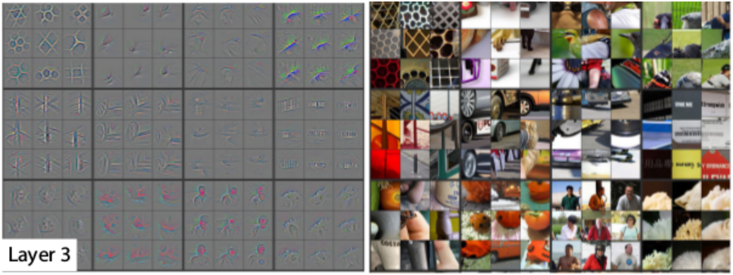
Visualizing and Understanding Convolutional Networks Matthew D Zeiler, Rob Fergus (2013)

results - layers 1 and 2

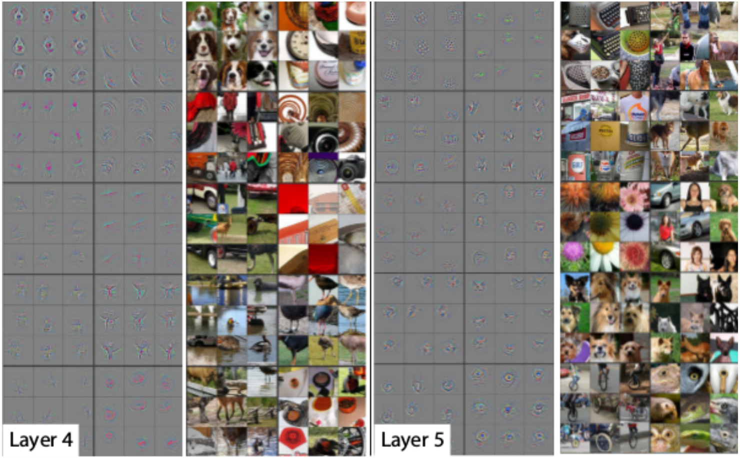
(better viewed in the original article)



results - layer 3



results - layers 4 and 5



Summary

Moving towards higher levels we observe

- growing structural complexity:
  oriented lines, colors $\rightarrow$ angles, arcs $\rightarrow$ textures
- more semantical grouping
- greater invariance to scale and rotation

See also Understanding Deep Image Representations by Inverting Them A. Mahendran, A. Vedaldi (2014)