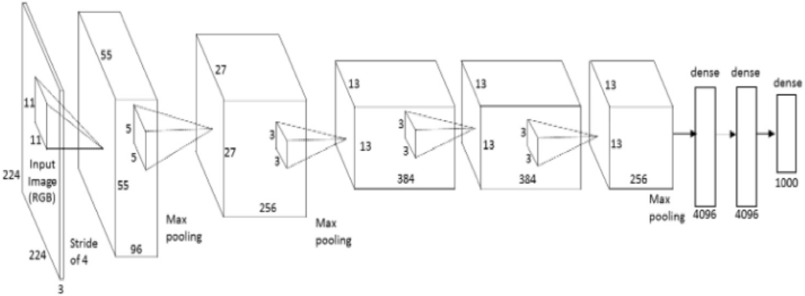
Lesson 5

- Some famous CNNs
- Transfer Learning
- How CNNSs see the world

Famous networks

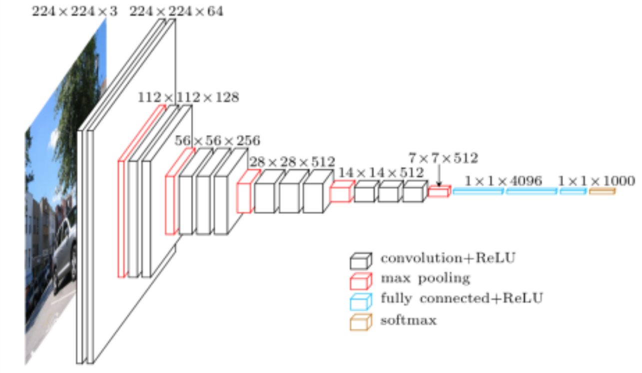
AlexNet

AlexNet Architecture (Krizhevsky, Sutskever e Hinton), winner of the NIPS contest in 2012.



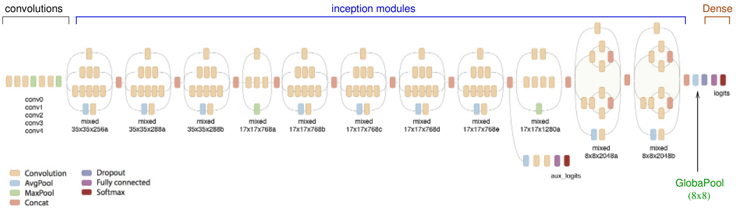
VGG

VGG 16 (Simonyan e Zisserman). 92.7 accuracy (top-5) in ImageNet (14 millions images, 1000 categories).



Picture by Davi Frossard: VGG in TensorFlow

Inception V3

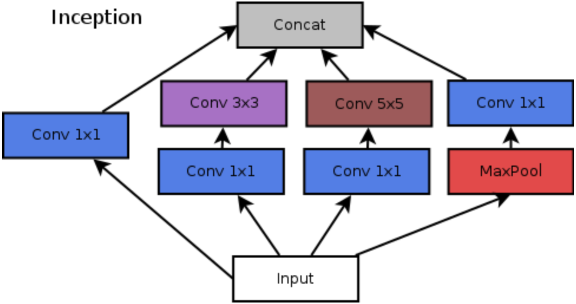


The convolutional part is a long composition of

inception modules

Inception modules

The networks is composed of inception modules (towers of nets):

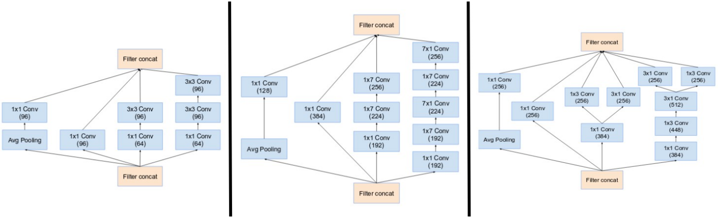


Video from the Udacity course "Deep Learning"

Variants

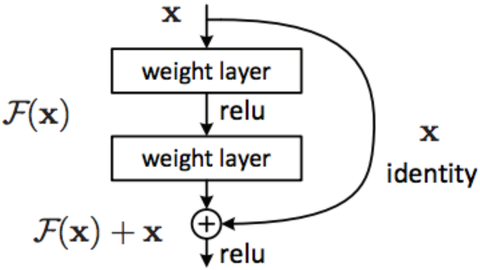
The point is to induce the net to learn different filters.

Many variants proposed and used over years:



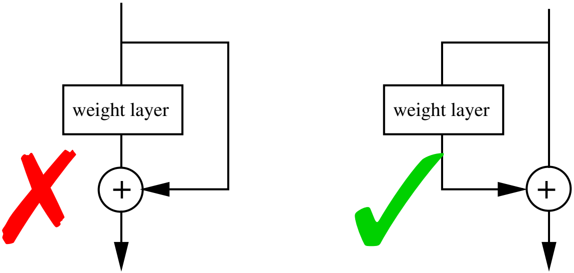
Residual Learning

Another recent topic is residual learning.

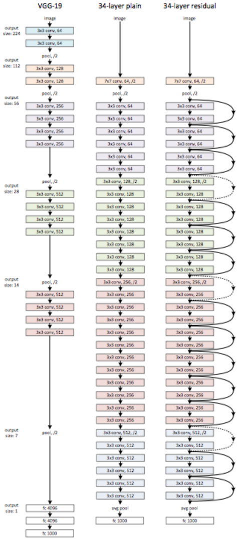


Instead of lerning a function $\mathcal{F}(x)$ you try to learn $\mathcal{F}(x) + x$.

The right intuition



Residual networks



you add a residual shortcut connection every 2-3 layers

Inception Resnet is an example of a such an architecture

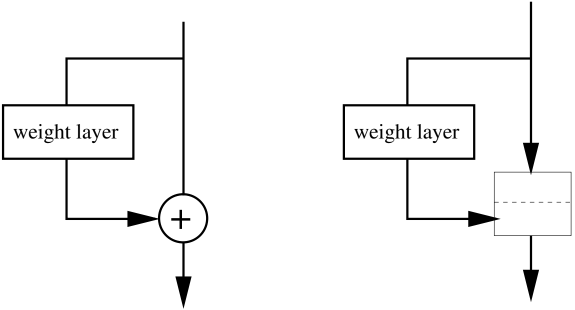
Why Residual Learning works?

Not well understood yet.

The usual explanation is that during back propagation, **the gradient at higher layers can easily pass to lower layers**, withouth being mediated by the weight layers, which may cause vanishing gradient or exploding gradient problem.

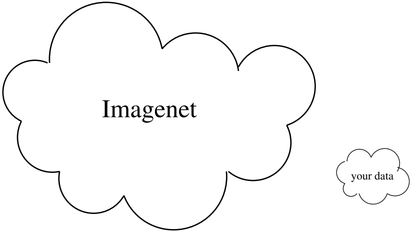
Sum or concatenation?

The "sum" operation can be interpreted in a liberal way.
A common variant consists in concatenating instead of adding (usually along the channel axis):



Next arguments

Transfer Learning



Reusing Knowledge

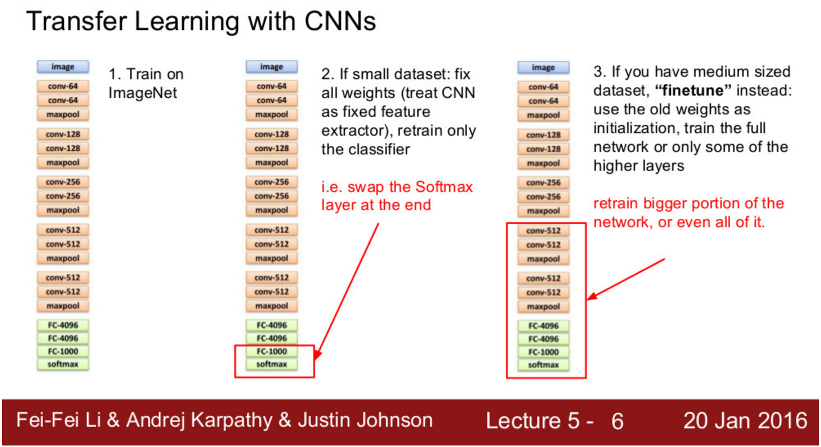
We learned that the first layers of convolutional networks for computer vision compute feature maps of the original image of growing complexity.

The filters that have been learned (in particular, the most primitive ones) are likely to be independent from the particular kind of images they have been trained on.

They have been trained on a huge amount of data and are probably very good.

It is a good idea to try to *reuse them* for other classification tasks.

Transfer Learning with CNNs



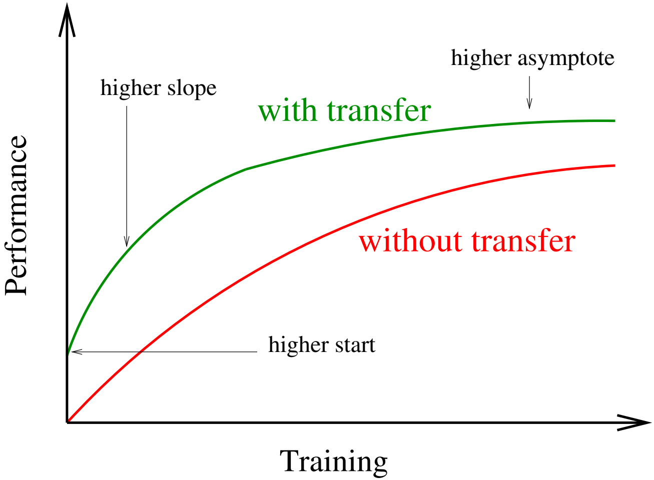
When Transfer Learning makes sense

transferring knowledge from problem A to problem B makes sense if

- the two problems have “similar” inputs
- we have much more training data for A than for B

What we may expect

Faster and more accurate training



Next argument

How CNNs “see” the world

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.

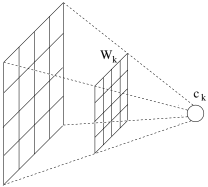
can we confirm such a claim?

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.



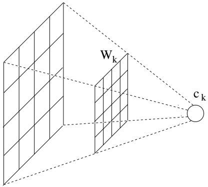
The loss function $\mathcal{L}(\theta, x)$ of a NN depends on the parameters θ and the input x.

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.

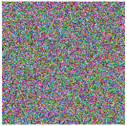


The loss function $\mathcal{L}(\theta, x)$ of a NN depends on the parameters θ and the input x.

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

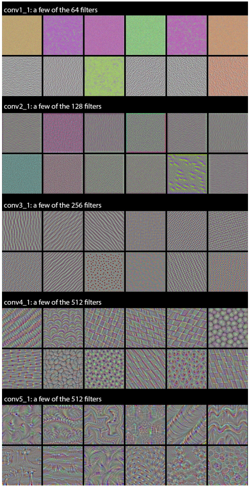
First layers

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

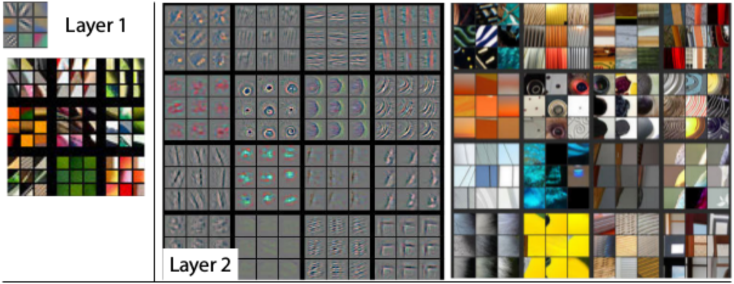
What caused the activation of this neuron **in this image**?

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.

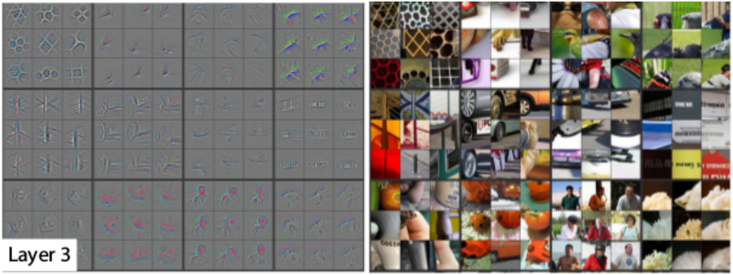
Visualizing and Understanding Convolutional Networks Matthew D Zeiler, Rob Fergus (2013)

results - layers 1 and 2

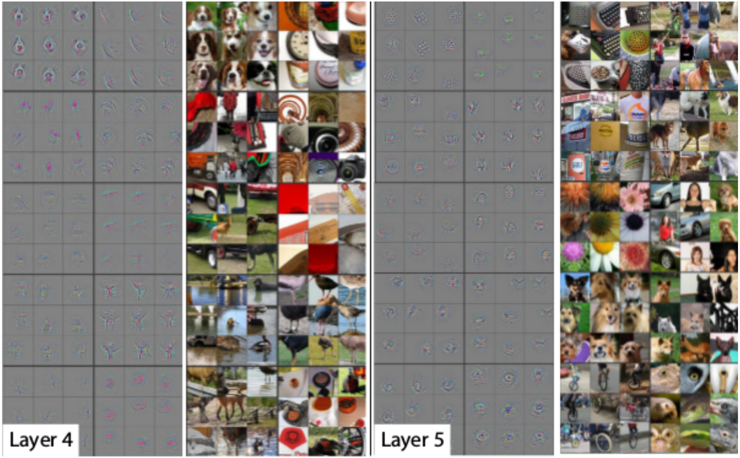
(better viewed in the original article)



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)