Convolutional neural networks

Support: python3 with Tensorflow

July 21st, 2017

Outline:

I. Convolutions.
II. CNNs history
III. CNNs today
IV. Workshop 1: 2D CNN classifier
V. Workshop 2: 3D CNN classifier

I. Convolutions: filters

Basic idea:
slide a weights “small window” across all the image

0.071	0.329	0.259	..	
0.549	0.263	0.612		
0.220	0.706	0.188		
0.271	0.176	0.200		
0.035	0.055	0.192		
..				

Input image

0	1	0
-1	0	1
0	-1	0

0.706 becomes:
1*0.263 + 1*0.188 – 0.176 – 0.22

0.071	0.329	0.259	..	
0.549	0.263	0.612		
0.220	0.055	0.188		
0.271	0.176	0.200		
0.035	0.055	0.192		
..				

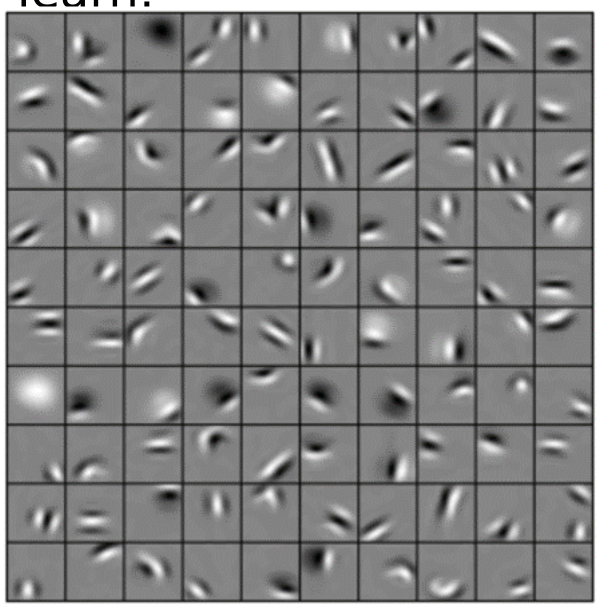
Output image

Typical window sizes: **3*3**, 5*5, 7*7

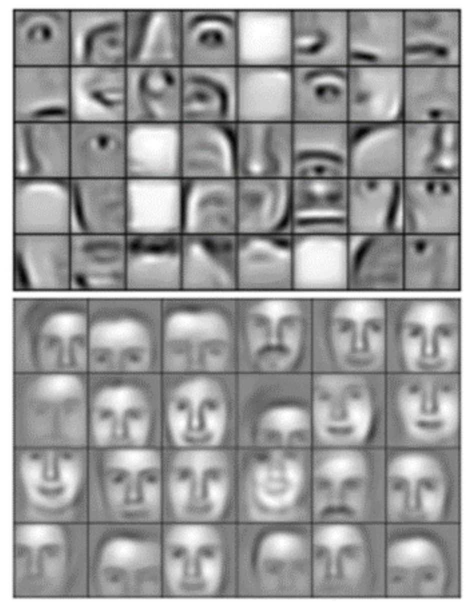
I. Convolutions: intermediate representations

Sliding each filter across the entire image produces a **feature map**.

The **deeper** we go, the more **high-level** concepts these maps learn:



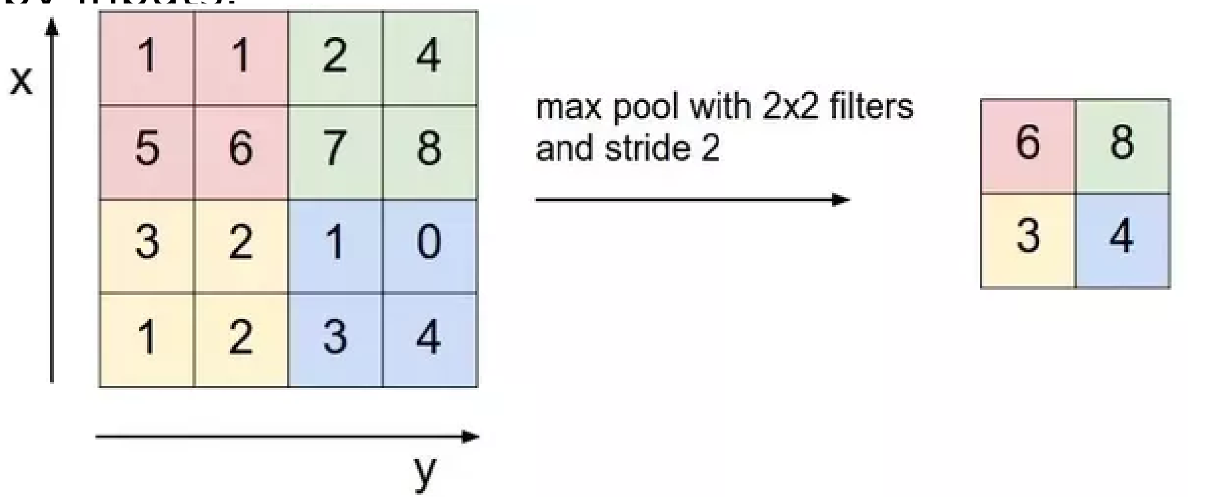
First layers (basic shapes)



Last layers

I. Convolutions: sub-sampling

The idea is to **reduce the dimension** of the input, without loosing “too much” information.
A common way is to take the **local maximum** of group of nearby inputs:



Sub-sampling via max-pooling

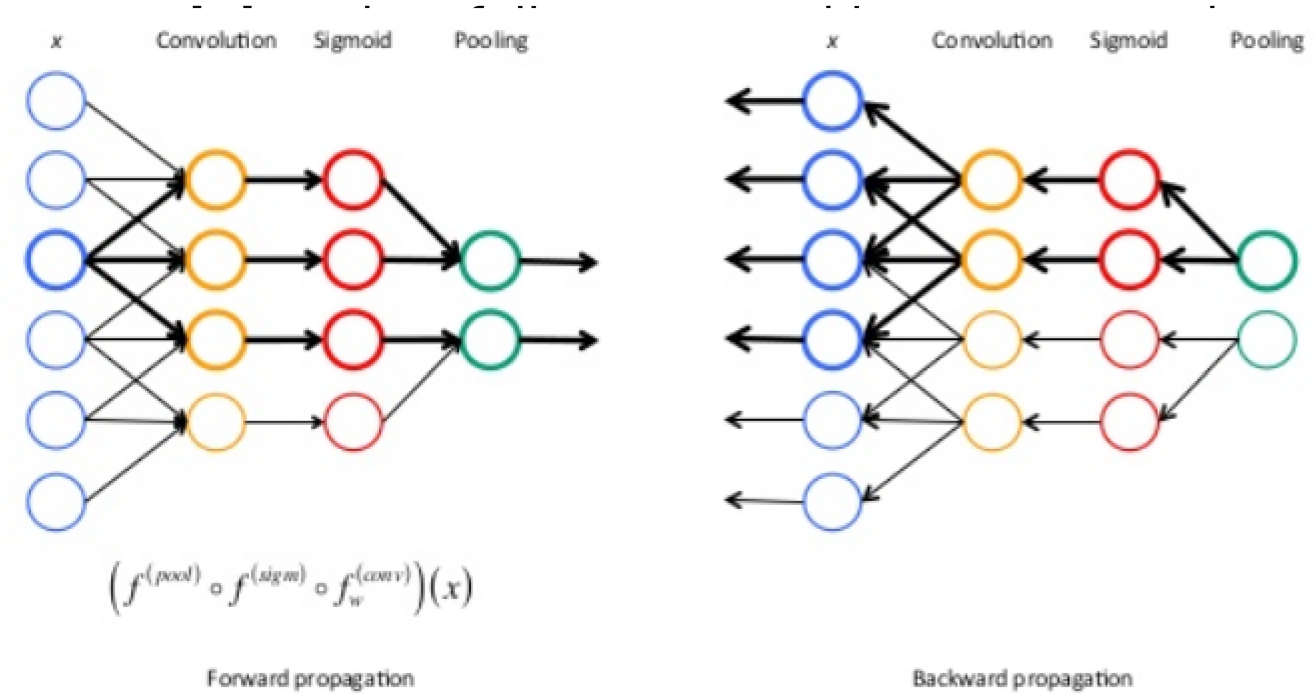
It is also possible to take the **average**, or minimum, etc.
These pooling layers do **not contain any learning**.

CNNs are made of convolutions, sub-sampling layers and dense layers.

I. Convolutions: activations and propagation.

CNNs can use any activation function: sigmoid, tanh, ReLU,
Propagation is done with regards to the filters and pooling zones.

F

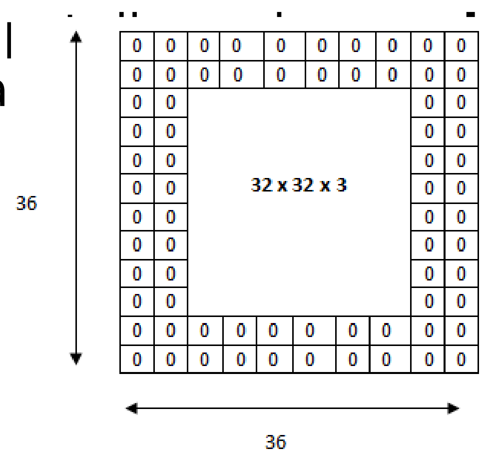


Forward and backward propagations on a Conv-activation-pooling block.

I. Convolutions: strides and padding

Moving filters can be done with **a certain step** in each direction: the stride value. Strides greater than 2 reduce dimension.

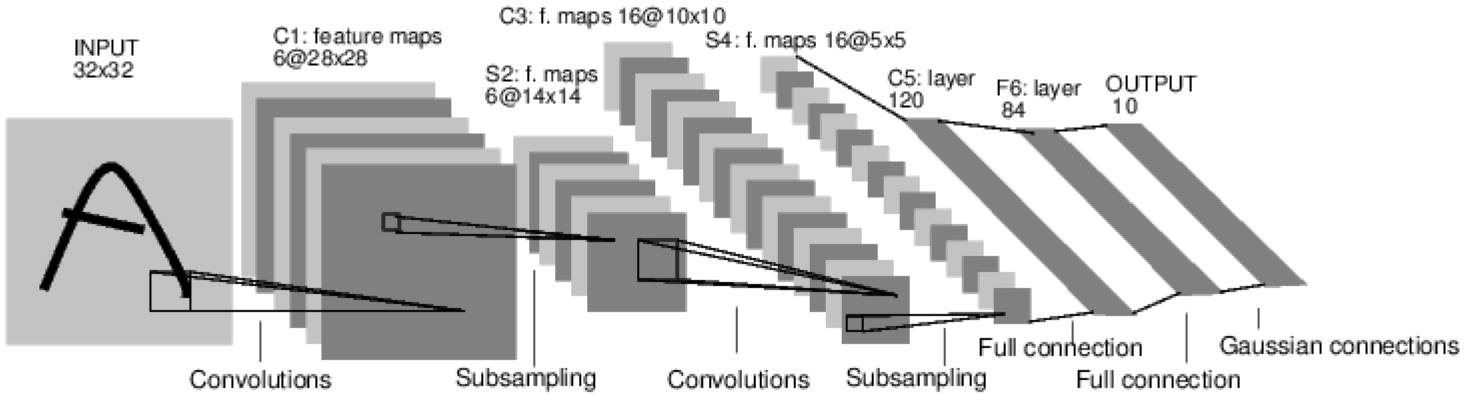
To preserve in| around the image (with zeros for insta



To conclude, setting a convolutional layer requires:
_number of filters
_filter dimension
_stride value
_padding (yes or no)
(+ activation function, initialization, regularization)

II. CNNs history: LeNet (1994)

First successfully implemented CNN, originally for digits recognition (MNIST dataset).



paper: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

2 Convolution + sub-sampling blocks.

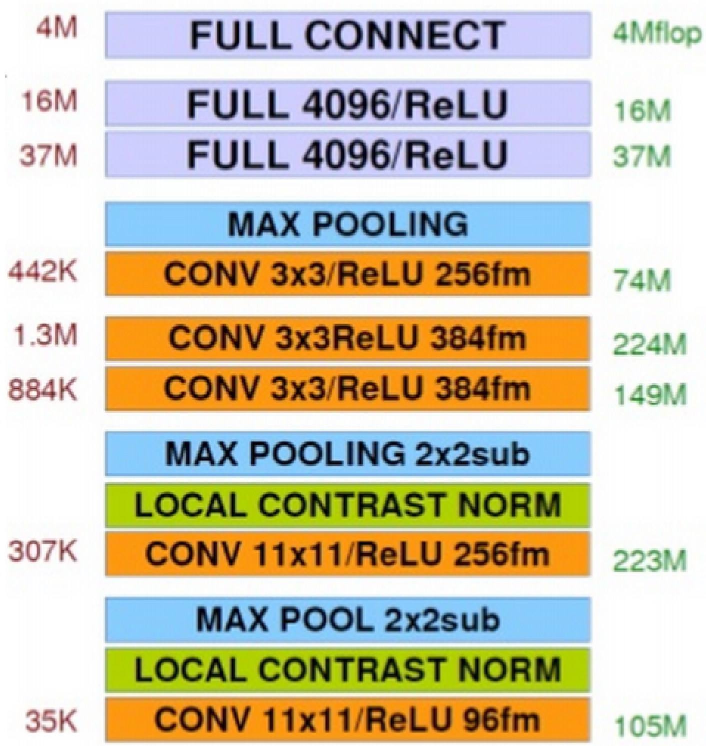
Achieves **96%** classification accuracy on MNIST.

II. CNNs history

AlexNet (2012):

VGG Net

(2014):



paper: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

8 layers (5 convs + 3 FCs)
60 million parameters.
Uses **ReLU** as activations.
Won ImageNet 2012 by 10% margin

19 layers
(16 convs + 3 FCs)
Very heavy: 150 million parameters

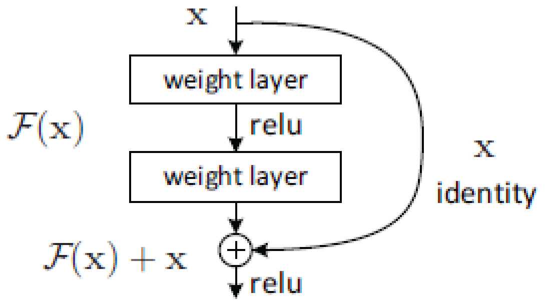
Introduces:
No pooling
Convolution **strides of 1**

Won ImageNet 2014

paper: https://arxiv.org/pdf/1409.1556.pdf

II. CNNs history : ResNet (2015)

Introduces **shortcut** connections:



paper: https://arxiv.org/pdf/1512.03385.pdf

The network keeps in mind residual of the input layer.

As well as:

No more FCs nor pooling layers !
Very deep networks (up to 1000 layers)
Generalized 3*3 filters
Generalized ReLU

ResNet-152 won ImageNet 2015.
It was the deepest network presented to ImageNet at the time, and was still **less complex than VGG**

III. CNNs today: heuristics.

Best working CNNs today typically make use of the following configuration:

- Networks 10 to 1000+ layers deep.

- Strided convolutions with filter size 3*3
  No FCs layers
  No Pooling layers

- ReLU or **Leaky-ReLU** as activation function everywhere

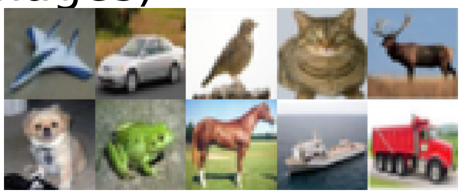
- SGD gradient descent with **Adam** optimizer
  and learning rate 0.0001 to 0.001

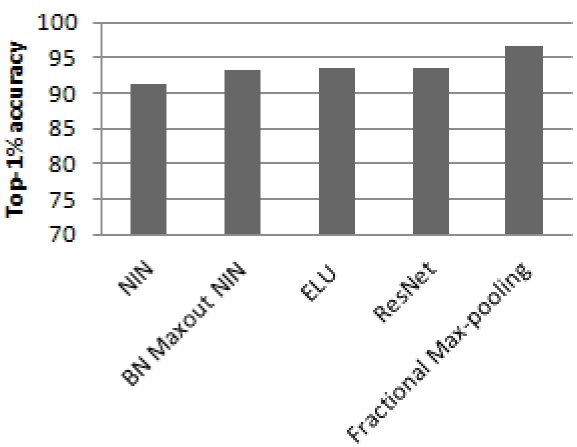
- Regularize with **batch-normalization**.
  It is also possible to use **dropout**.

  papers: https://arxiv.org/pdf/1502.03167.pdf and https://arxiv.org/pdf/1506.02158v6.pdf respectively

III. CNNs today: performance.

On **Cifar-10** (32*32 images)

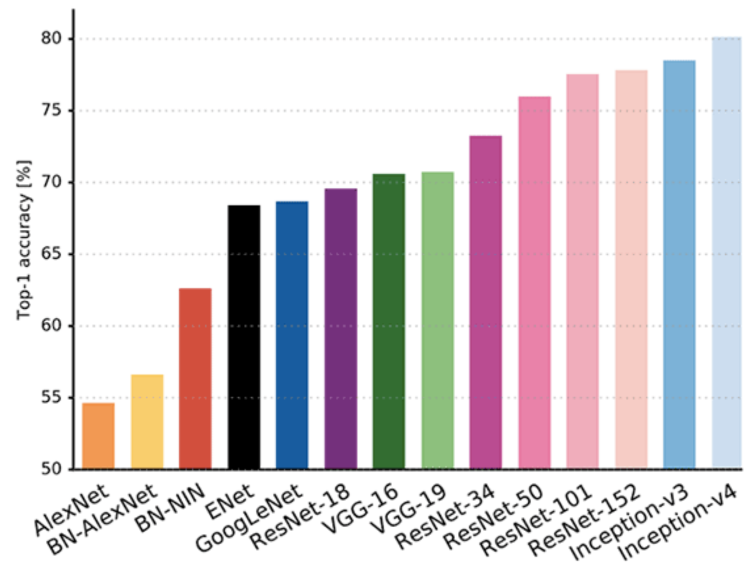




Classification accuracy of some CNNs on Cifar-10, including state-of-the-art, which is **96.5%**

Human performance: 94%

On **ImageNet** (224*224 images):

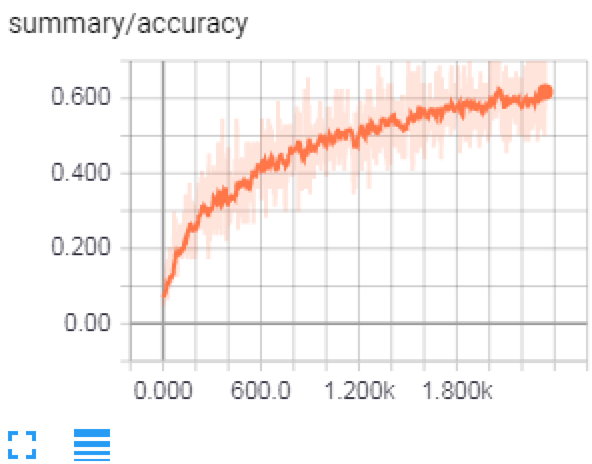


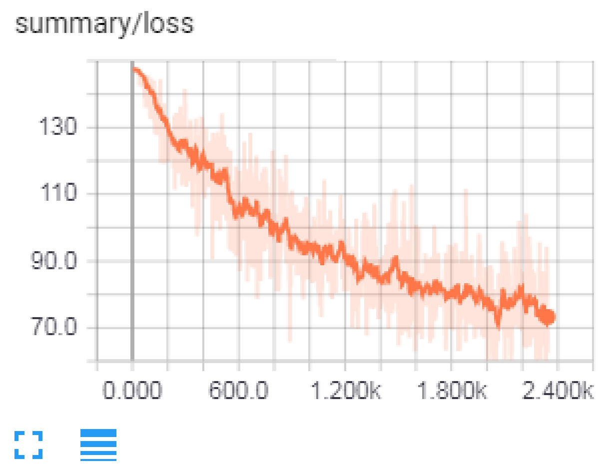
Classification accuracy of some CNNs on ImageNet, including state-of-the-art, which is **80.2%**

IV. Workshop: building a 2D CNN classifier.

<u>Goal</u>: to classify images from the Cifar-10 dataset (32*32 images)

<u>Neural network</u>: adapted version of VGG (6 convolutions +3 dense layers)



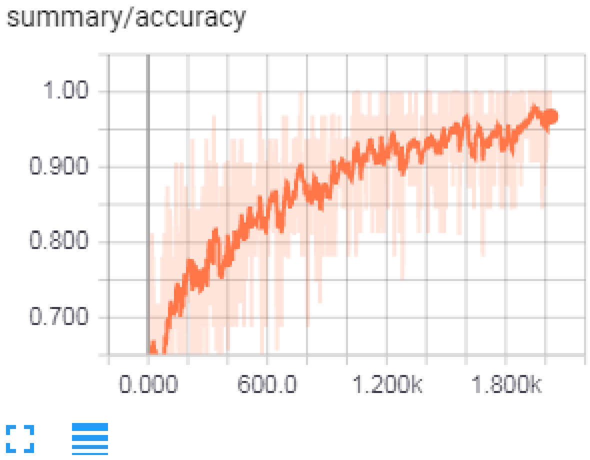


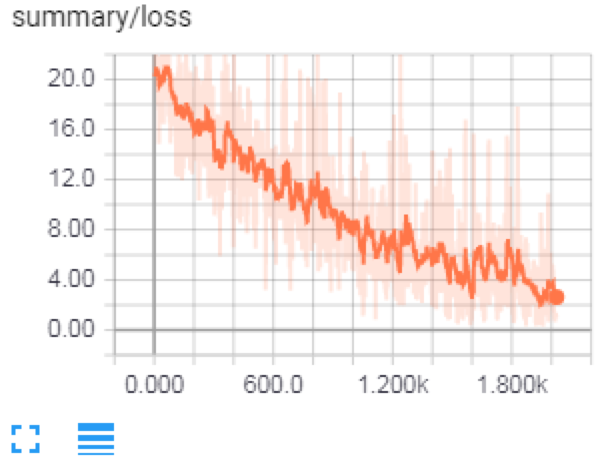
Expected training metrics

V. Workshop: building a 3D CNN classifier

Goal: to classify nodules (32*32*32 cubes) extracted from the LUNA-16 during the Kaggle Data Science Bowl 2017.

Neural network: ResNet-18





Expected training metrics