Convolutional neural networks

Support: python3 with Tensorflow

Mathieu RAVAUT

July 21st, 2017

Outline:

I. Convolutions.
II. CNNs history
III. CNNs today
IV. Workshop 1: 2D CNN classifier on Cifar-10
V. From 2D to 3D
VI. Workshop 2: 3D CNN classifier on nodules

I. Convolutions: filters

Basic idea:
slide a weights “small window” across all the image to capture spatial info

0	1	0
-1	0	1
0	-1	0

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

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

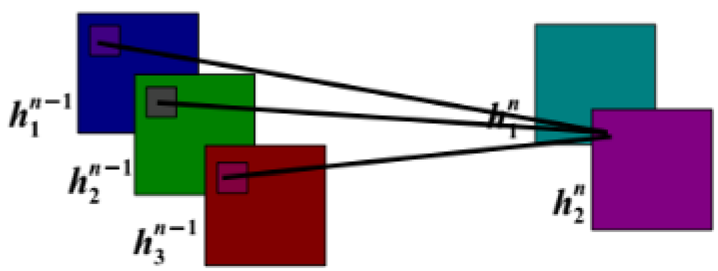
0.071	0.329	0.259	..	
0.549	0.263	0.612		
0.220	0.055	0.188		
0.271	0.176	0.200		
0.035	0.055	0.192		
..				

Output image

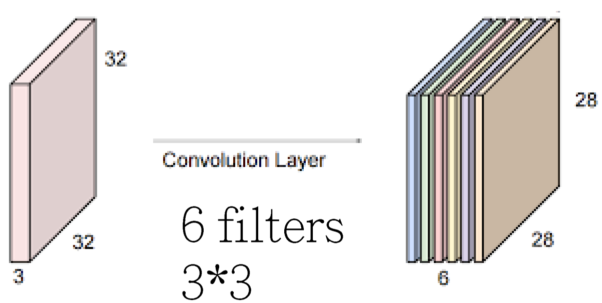
Typical window sizes: 3*3, 5*5, 7*7 (square odd size)

I. Convolutions: filters

Filters on a **multi-channel** input (such as intermediate layers):



Filters get multi-channel, and we sum contributions from each 2D filter

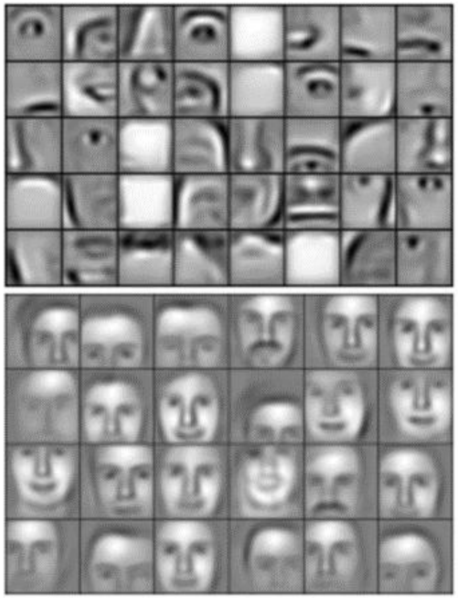


3*6*3*3 = 162 parameters

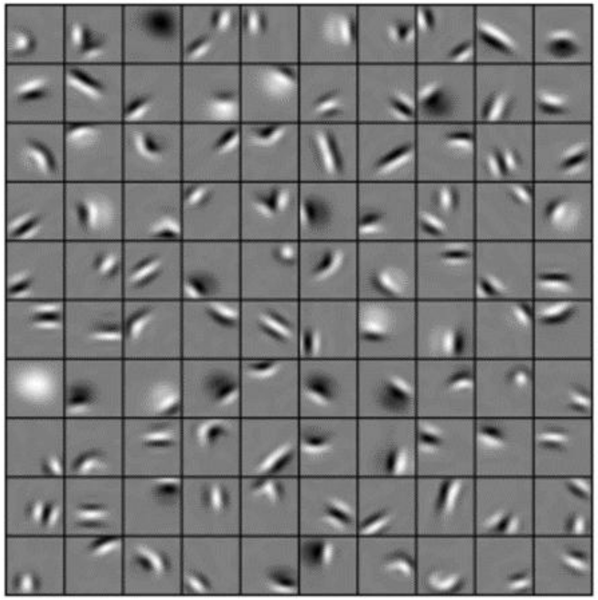
I. Convolutions: intermediate representations

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

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



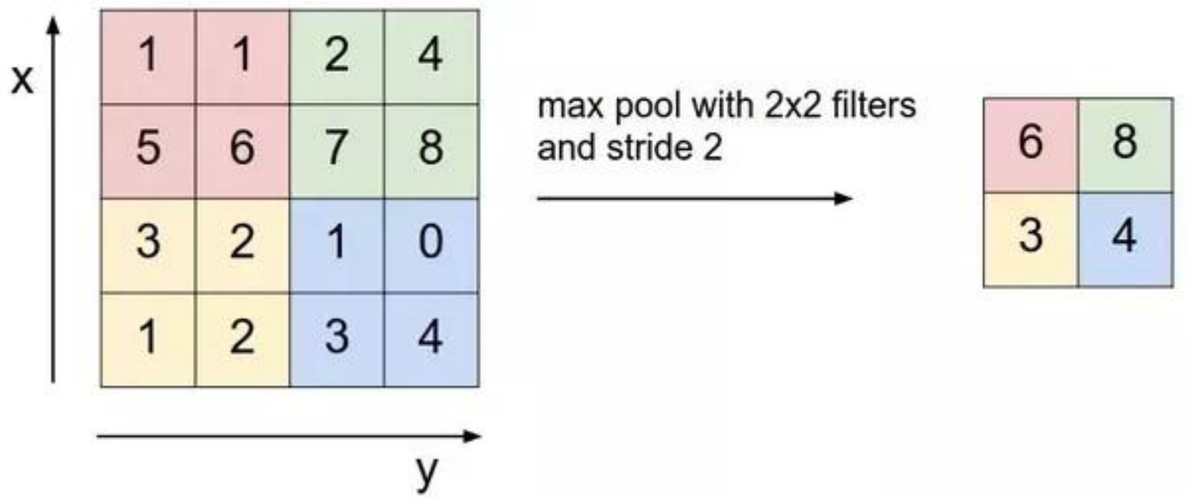
First layers



Last layers (higher-level concepts)

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:



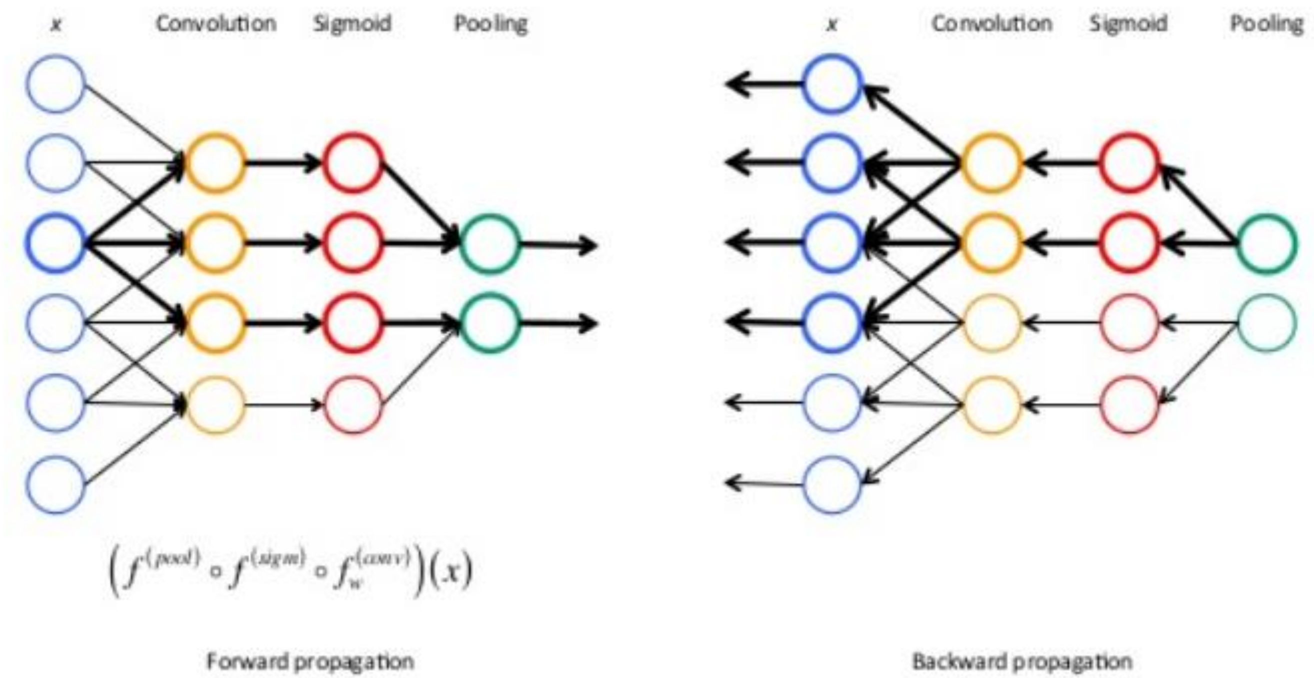
Sub-sampling via max-pooling

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.
Fewer weights than fully connected layers are used.

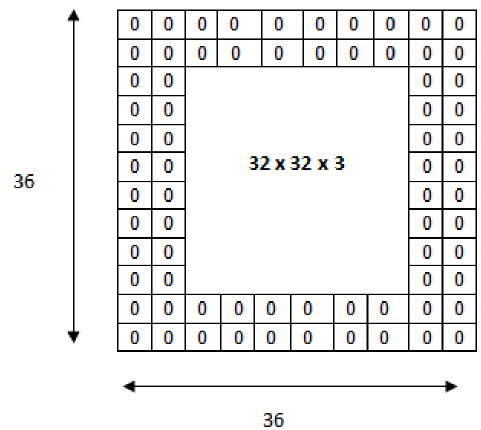


Forward and backward propagations on a Conv-activation-pooling block.

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.

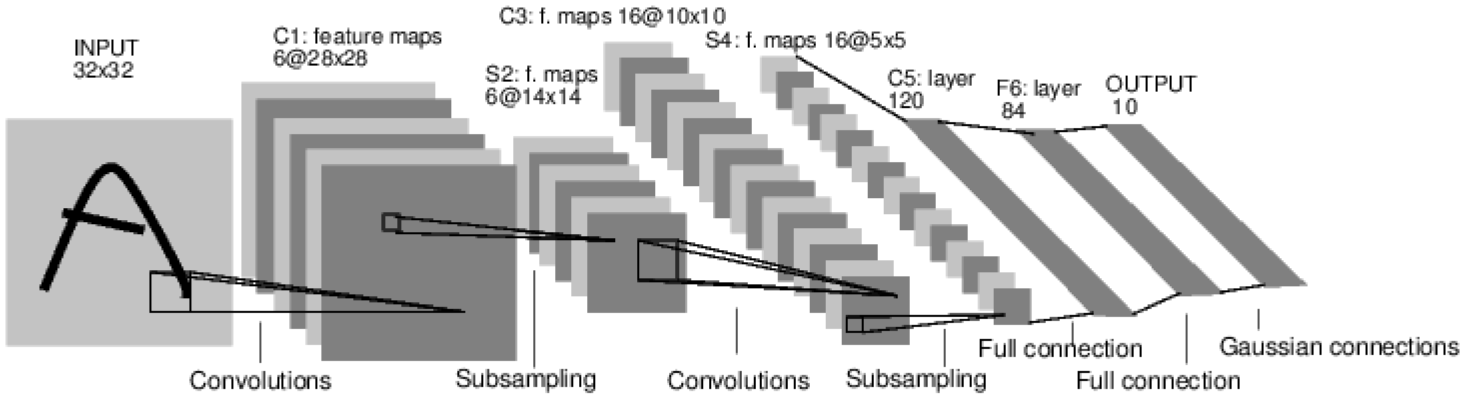
To preserve input dimension: **pad** around the image (with zeros for instance):



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).



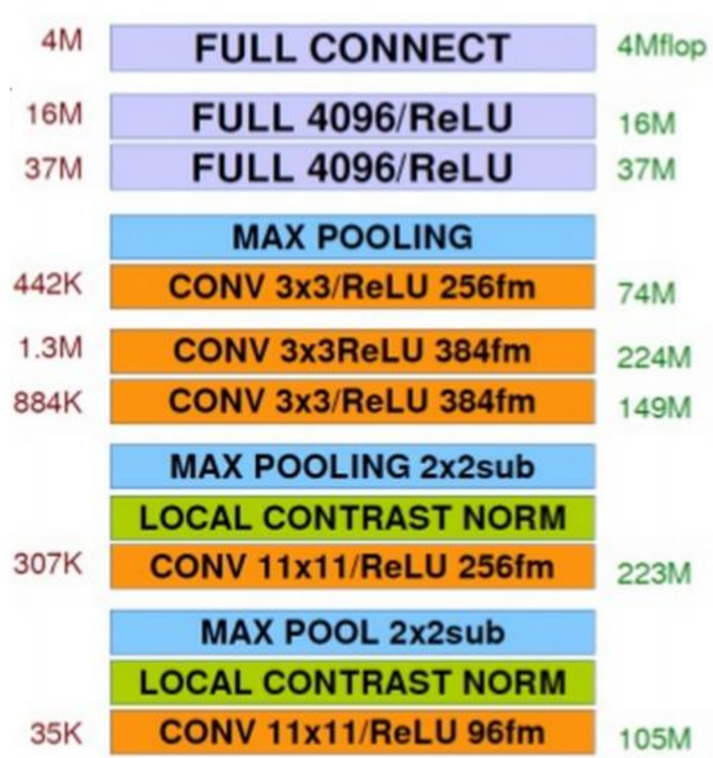
paper: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

2 Convolution + sub-sampling blocks + 3 fully connected layers

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

II. CNNs history

AlexNet (2012):



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

VGG Net (2014):

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

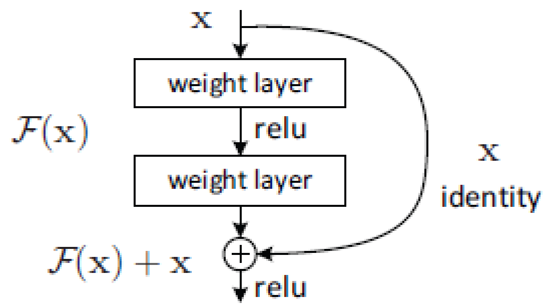
Introduces:
Convolution **strides of 1**

Won ImageNet 2014

paper: [*https://arxiv.org/pdf/1409.1556.pdf*](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 **residuals** of the input layer.

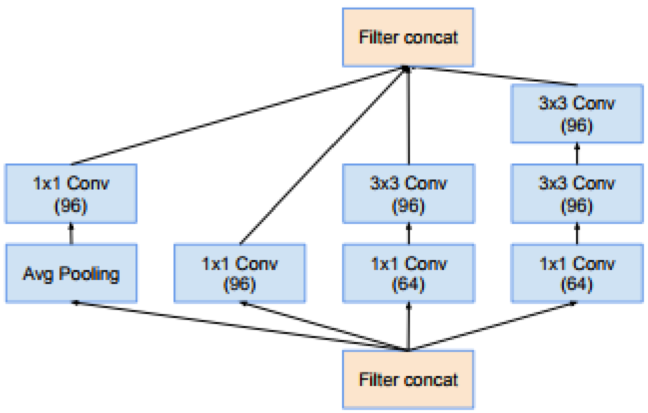
As well as:

Fully convolutional (just 1 global pooling and 1 FC at then end)
Light model (compared to VGG)
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**

II. CNNs history : Inception (v4 in 2016)

Base block:



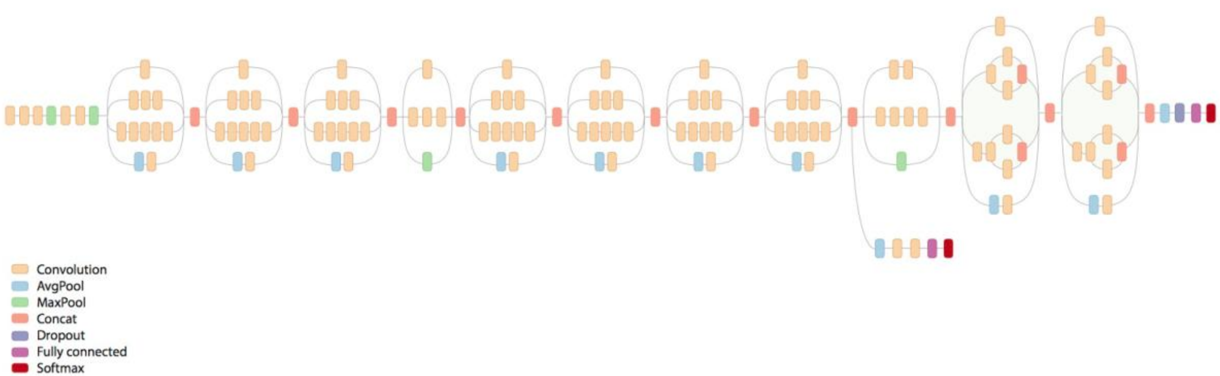
paper: https://arxiv.org/pdf/1602.07261.pdf

Multiple convolution branches with different filter size gathered together.

Produces multi-branches networks.

3.08% top-5 error on ImageNet (state-of-the-art).

Overall architecture:



III. CNNs today: regularization.

Often, the dataset is too small to be used by these huge deep networks.
Thus, **regularization** is crucial.

- Dataset augmentation (crops, flipping, rotations, etc)
- Early stopping
- Dropout
  Convolution layers: 0.7
- Weight decay with L1 and L2 norms

III. CNNs today: guidelines.

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

- Networks 10 to 200+ layers deep.

- Convolutions with filter size 3*3 and **shortcut** connections
  Get rid of FCs layers
  Sub-sample via convolution strides

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

- SGD gradient descent with **momentum** or **Adam** or **RMSProp** optimizer

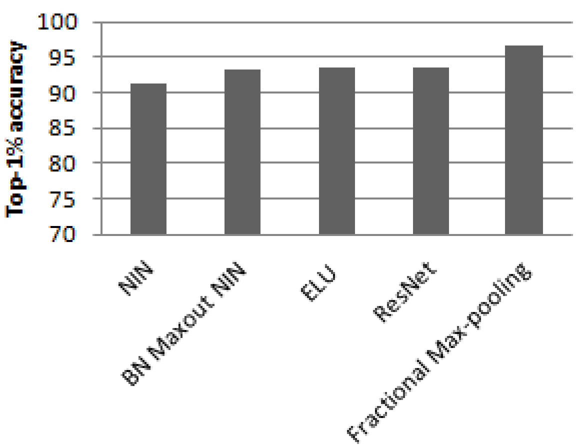
- Regularization with **weight-decay with L2 norm**

- Prevent gradient vanishing with **batch-normalization** -> go very deep

papers: https://arxiv.org/pdf/1502.03167.pdf

III. CNNs today: performance.

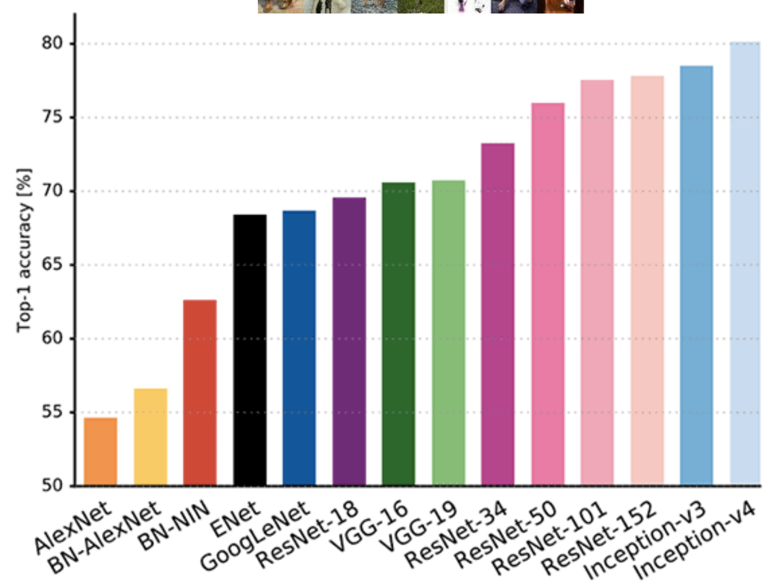
On **Cifar-10** (32*32 images)



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

Human performance: 94%

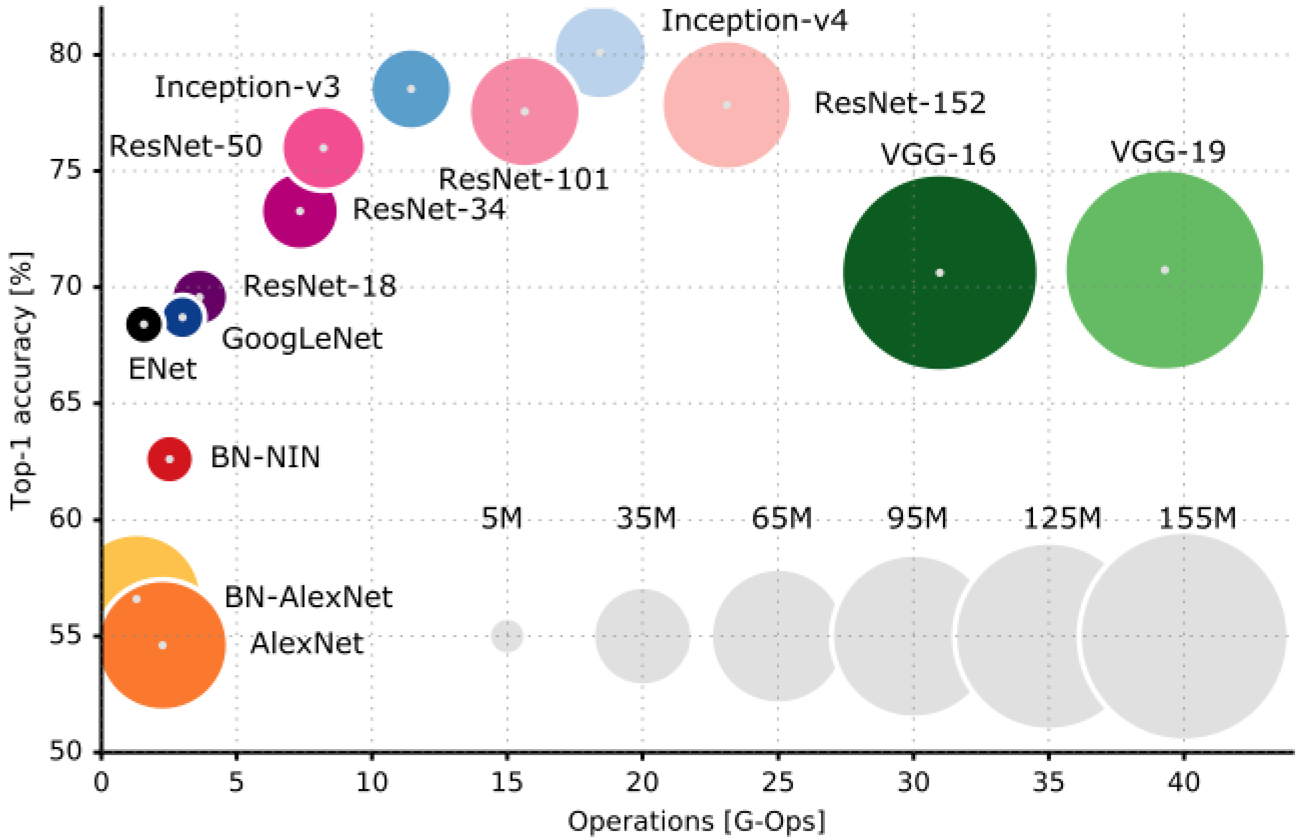
On **ImageNet** (224*224 images):



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

III. CNNs today: trade-off.

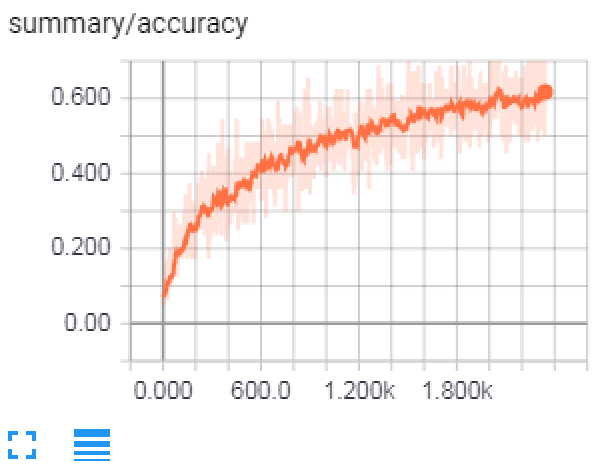
Comparing several famous models on ImageNet:

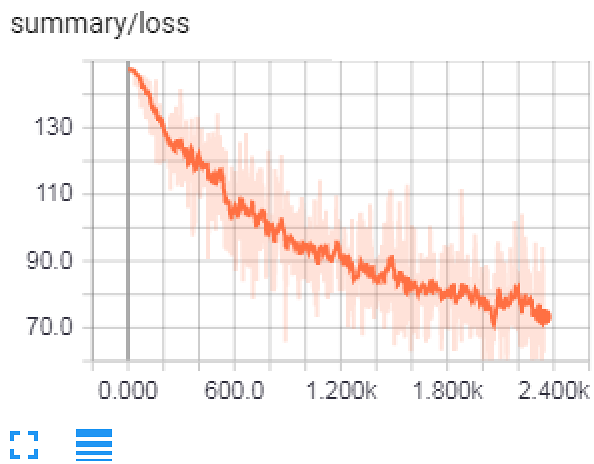


IV. Workshop: building a 2D CNN classifier.

Goal: to classify images from the Cifar-10 dataset (32*32 images)
Multi-class classification problem (10 classes)

Neural network: adapted version of VGG (8 convolutions +3 dense layers)





Expected training metrics

V. From 2D to 3D.

The curse of dimensionality.
3D representations are much, much more complex than 2D ones:

2D representation of ADN

3D representation of ADN

Thus we need **more convolutional filters** to capture spatial information BUT these **filters are bigger**: typically from 3*3 to 3*3*3

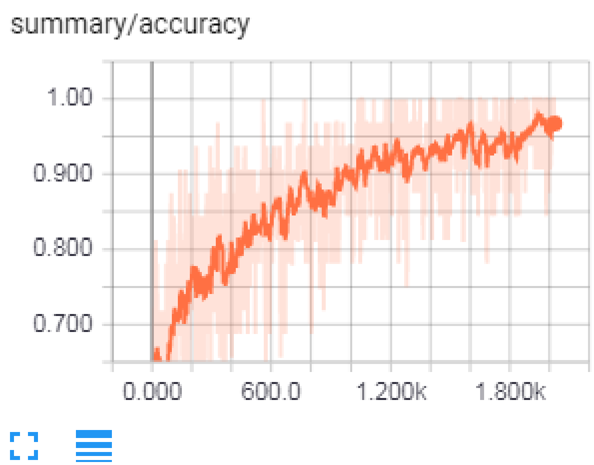
Input tensor is 5D: *(batch size, dim1, dim2, dim3, channels)*
⇨ Intermediate representations make RAM blow up quickly

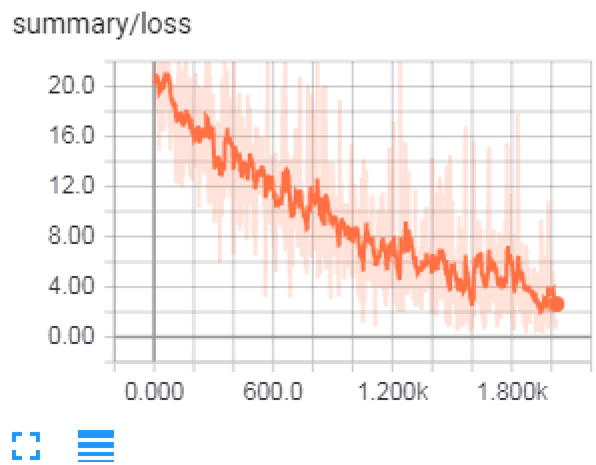
Filters are moved in **3 directions** (x,y,z)

VI. 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.
Binary classification problem.

Neural network: ResNet-18





Expected training metrics