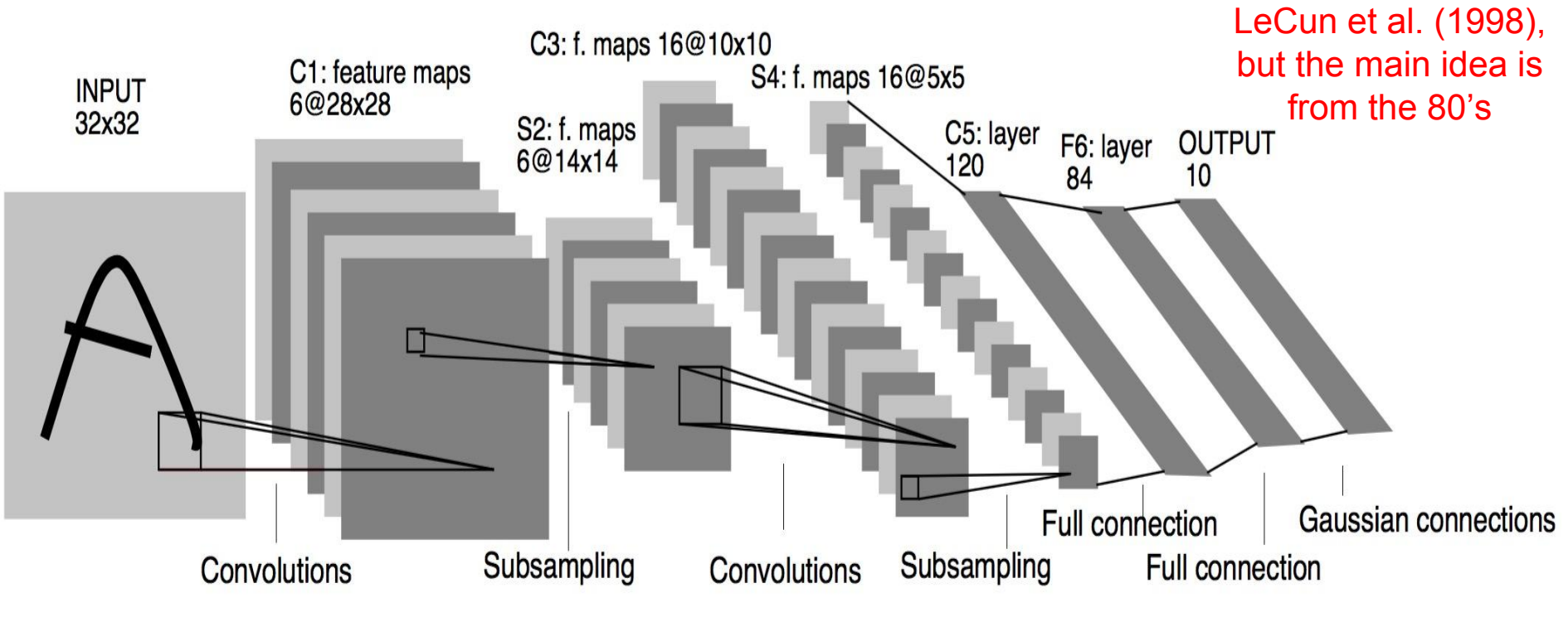
Convolutional neural networks (convnets)

Part I - introduction

Convnets are not new!



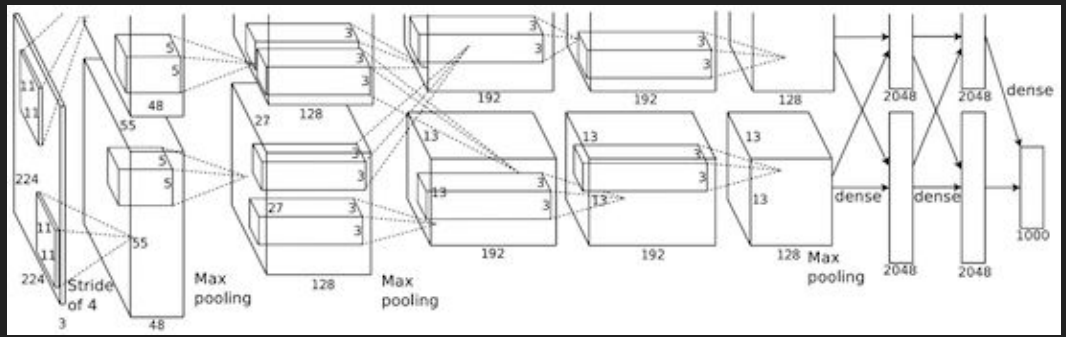
LeCun et al. (1998), but the main idea is from the 80's

Big breakthrough - 2012, renaissance of convnets begins

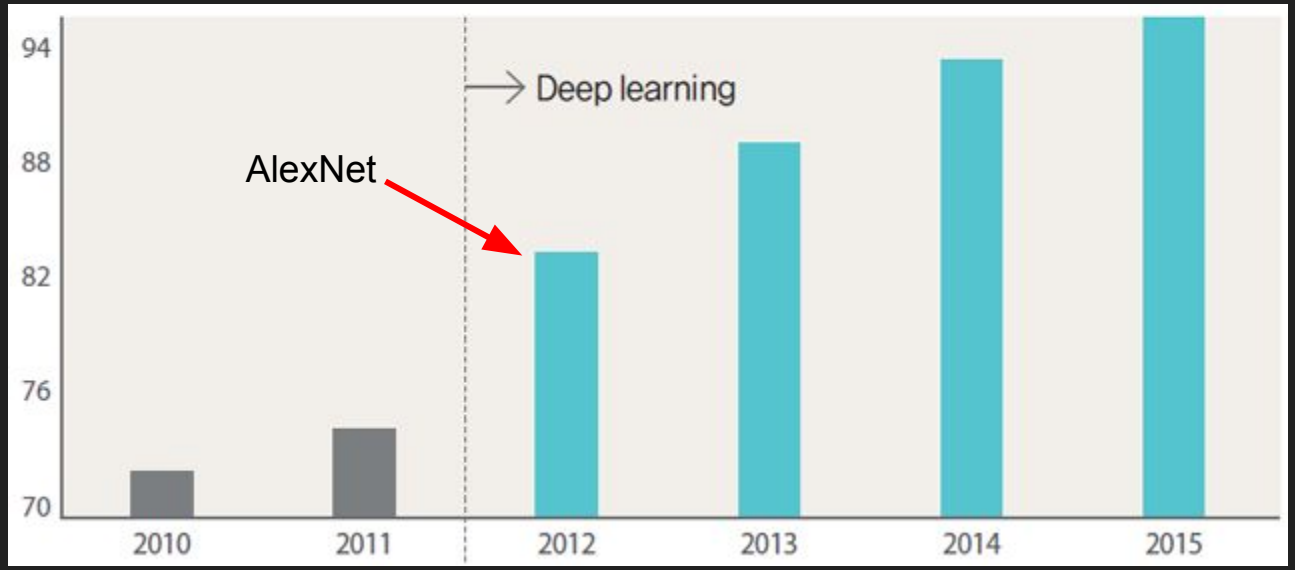
ImageNet Classification with Deep Convolutional Neural Networks
(NIPS 2012, over 10,000 citations**)**

- Alex Krizhevsky
- Ilya Sutskever
- Geoffrey E. Hinton

"AlexNet"

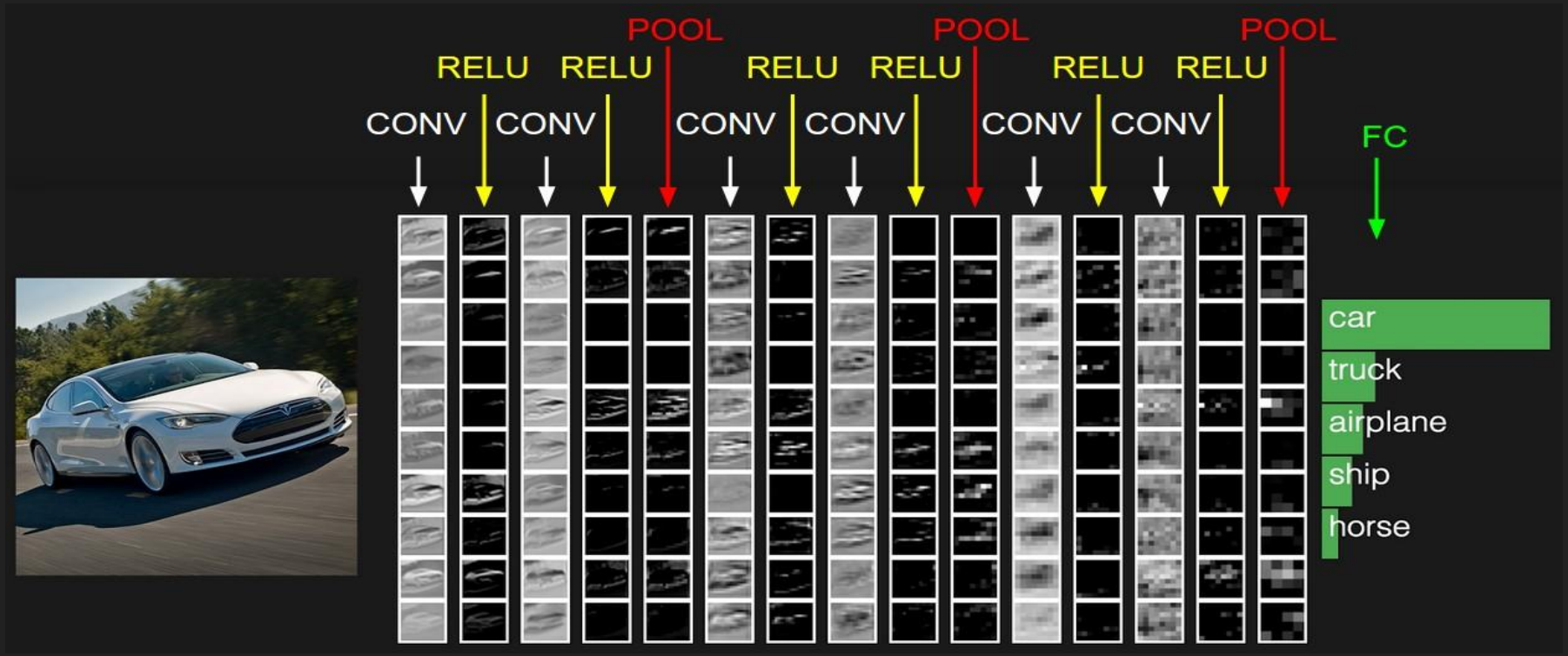


ImageNet top-5 accuracy
(image classification for 1000 classes)



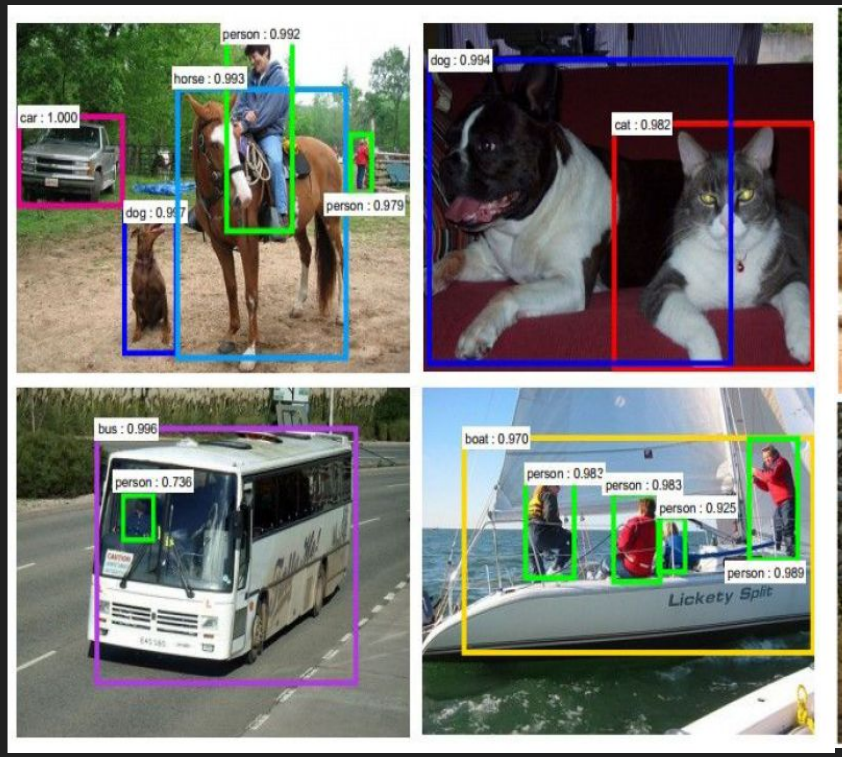
Nowadays convnets are ubiquitous!

Image classification



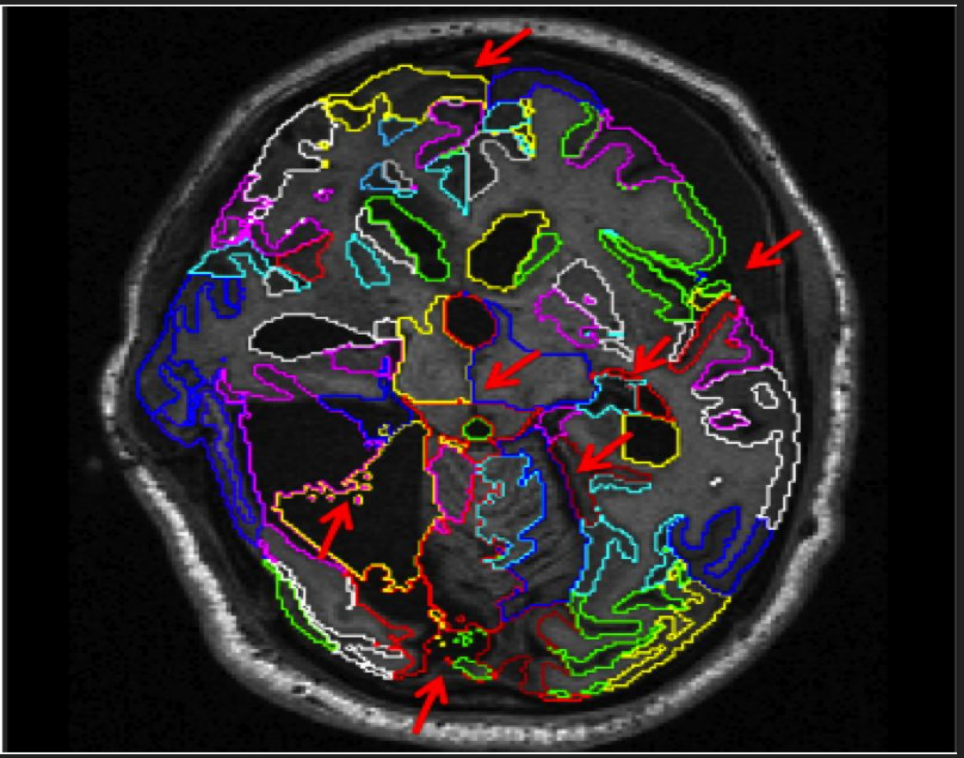
Detection

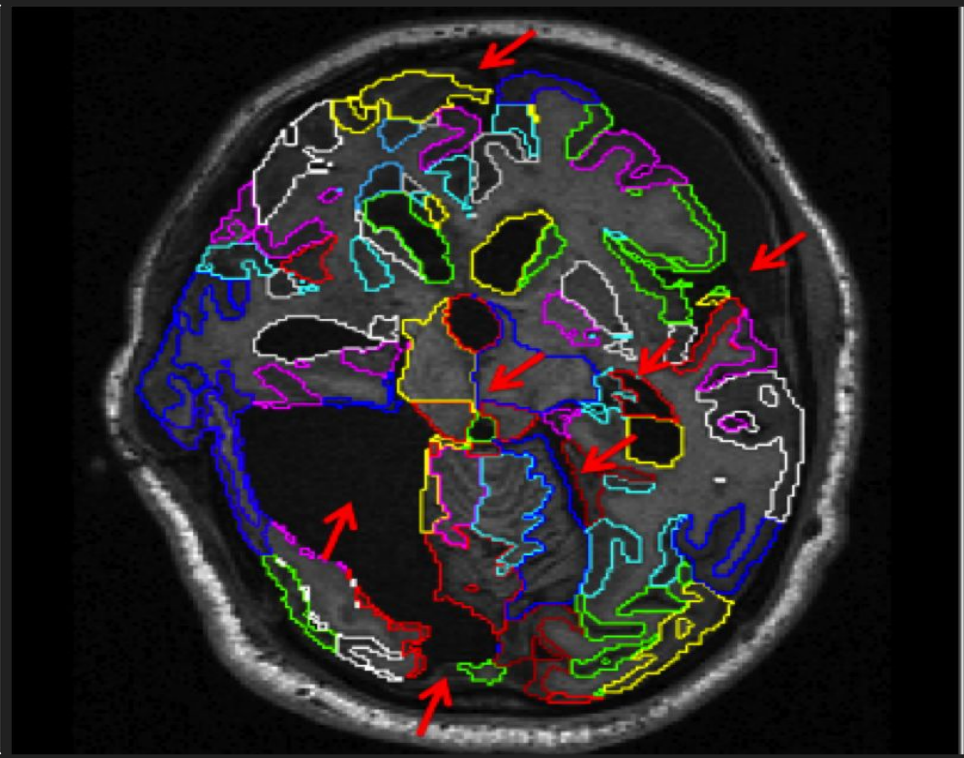
Segmentation




Cruise
13
SPEED
LIMIT
30

Medical image segmentation





Ledig et al. (2015)

nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE

At last — a computer program that can beat a champion Go player PAGE 484

ALL SYSTEMS GO

CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452

RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459

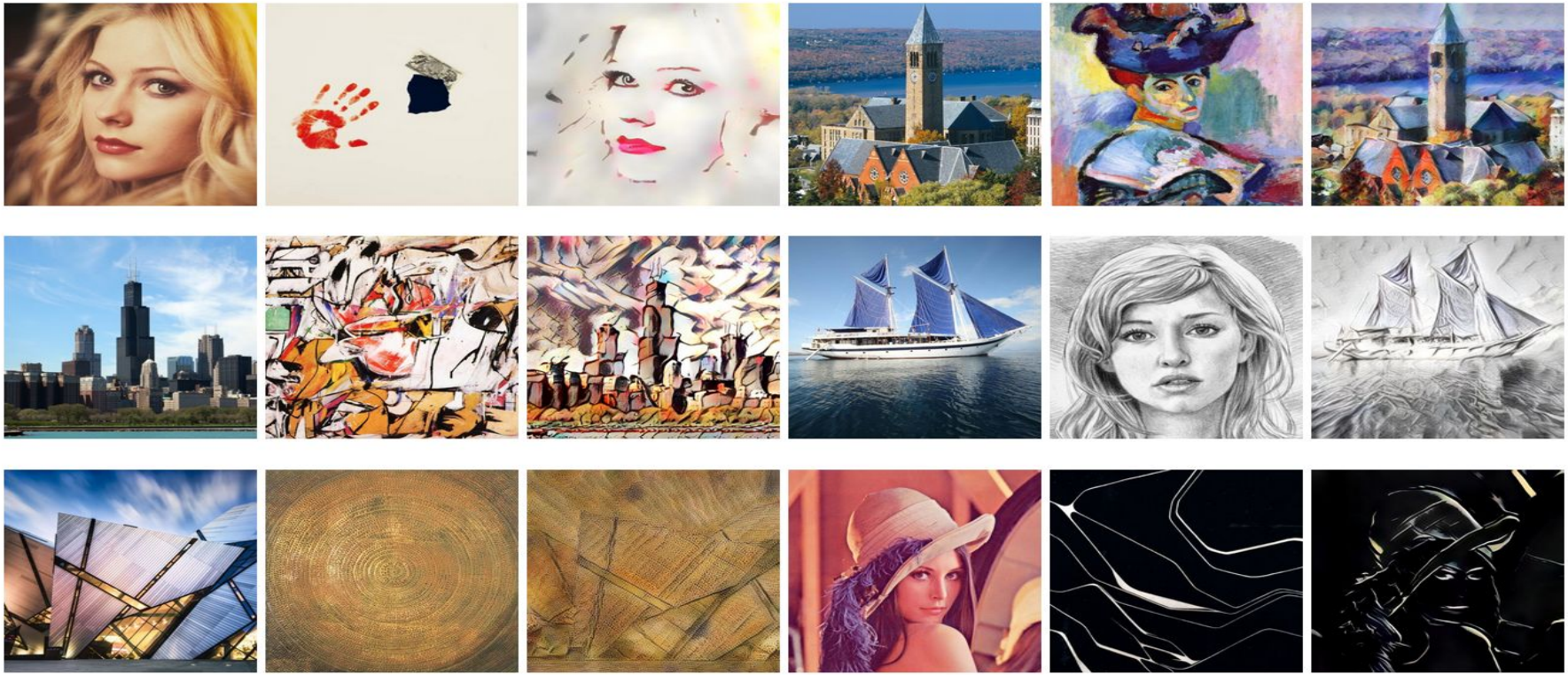
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card forty years on
PAGE 462

NATURE.COM/NATURE
28 January 2016 £10
Vol. 529, No. 7587
04
9 770028 083095

© AP Photo/Lee Jin-man

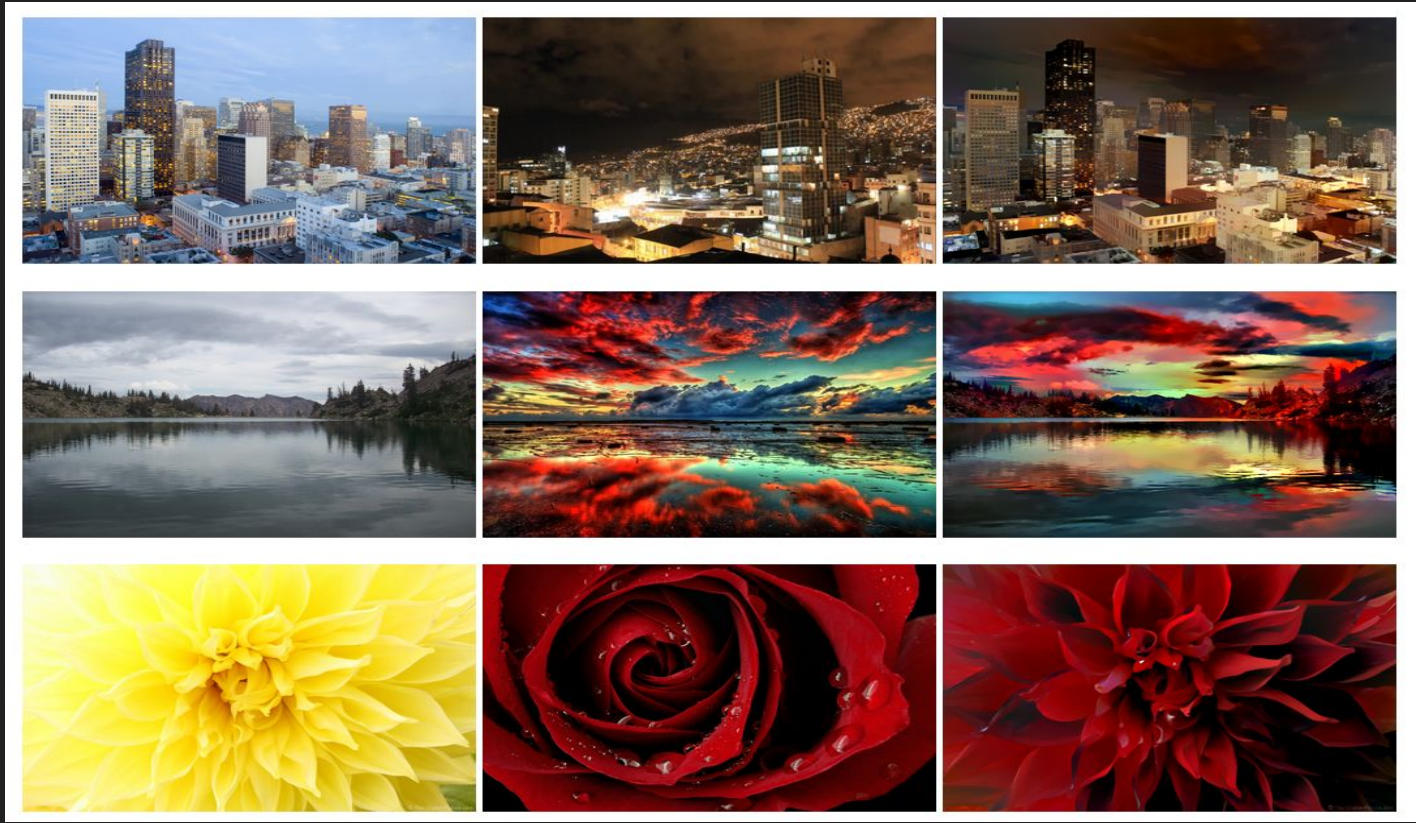
Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization

Xun Huang, Serge Belongie (03.2017) https://github.com/xunhuang1995/AdaIN-style



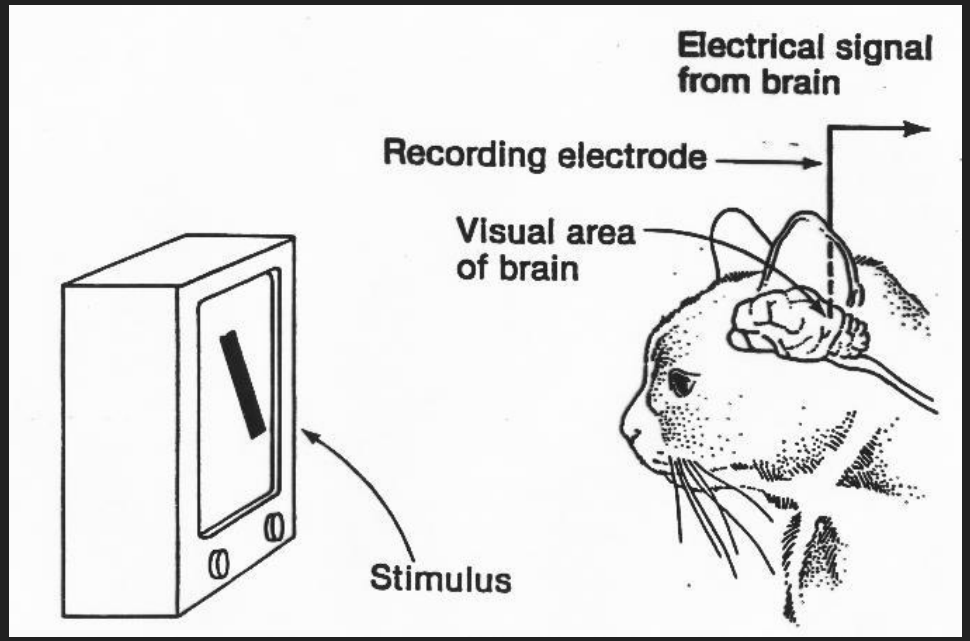
Deep Photo Style Transfer

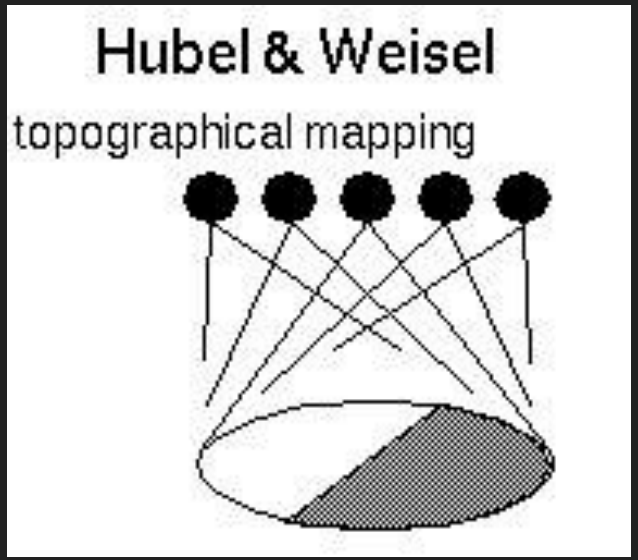
Fujun Luan, Sylvain Paris, Eli Shechtman, Kavita Bala (03.2017) https://github.com/luanfujun/deep-photo-styletransfer



Analogies to human brain

Hubel & Wiesel - nobel prize winning research made in 1959



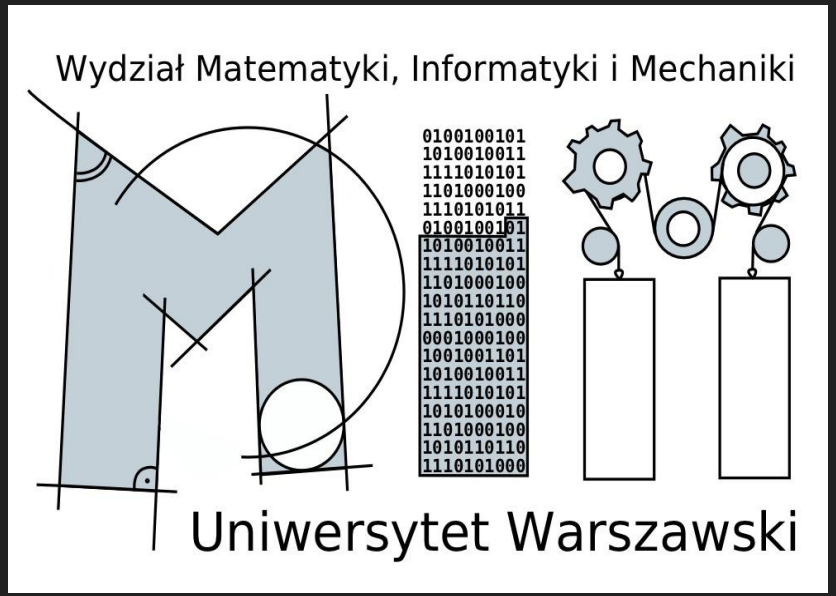


Convnets’ usual components:

- Convolution layer
- Pooling layer
- Batch Normalization
- ReLU (activation)
- Fully connected layers

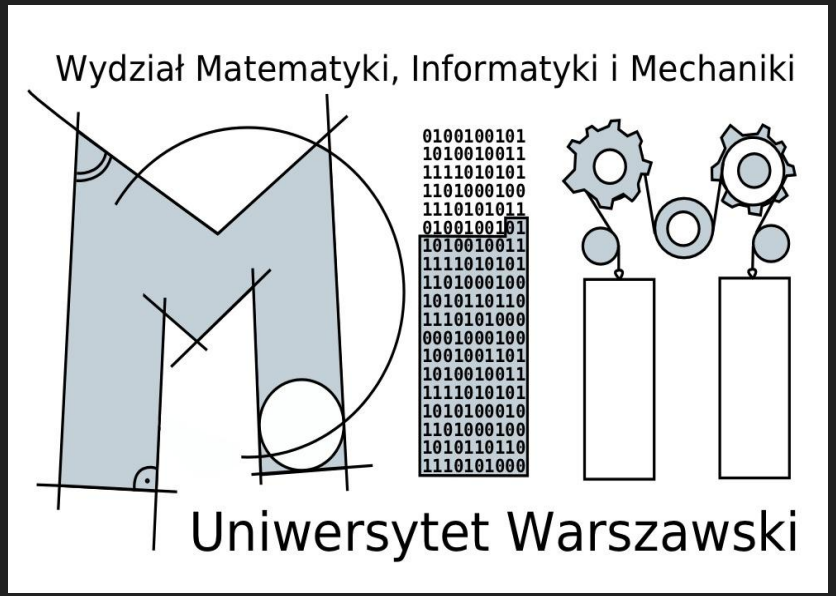
Convolution

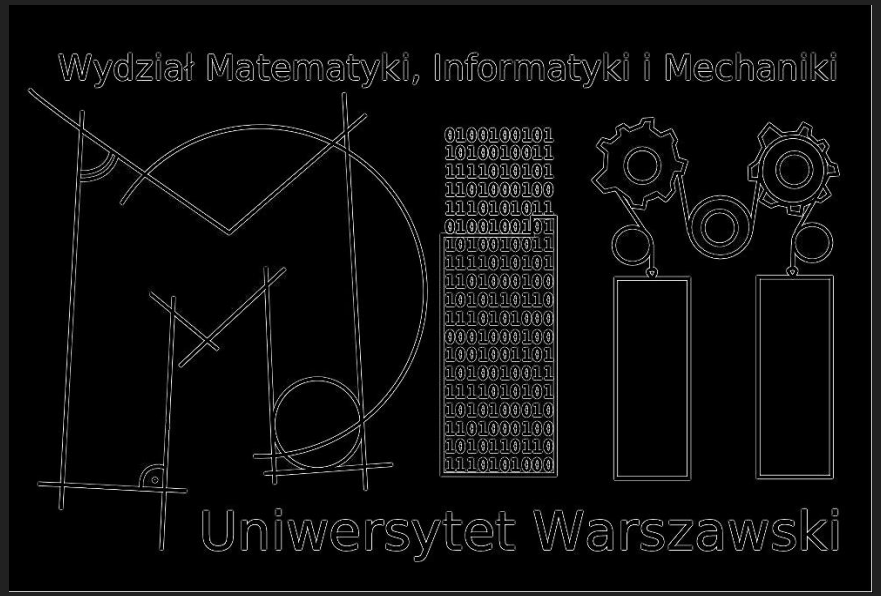
What will happen, if we replace each pixel by an average of all pixels within (maximum) distance 2?



Convolution

What will happen, if we subtract from each pixel
the average of neighbouring pixels?





Convolution

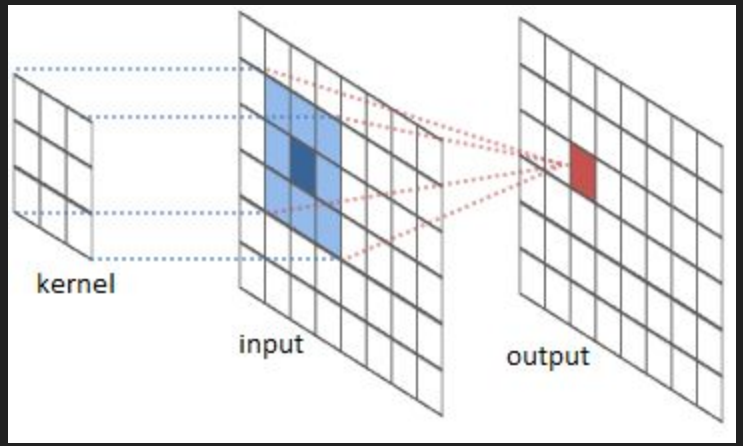


Image taken from http://intellabs.github.io/RiverTrail/tutorial/

What are the kernel matrices for the two examples: blur and edge detection?

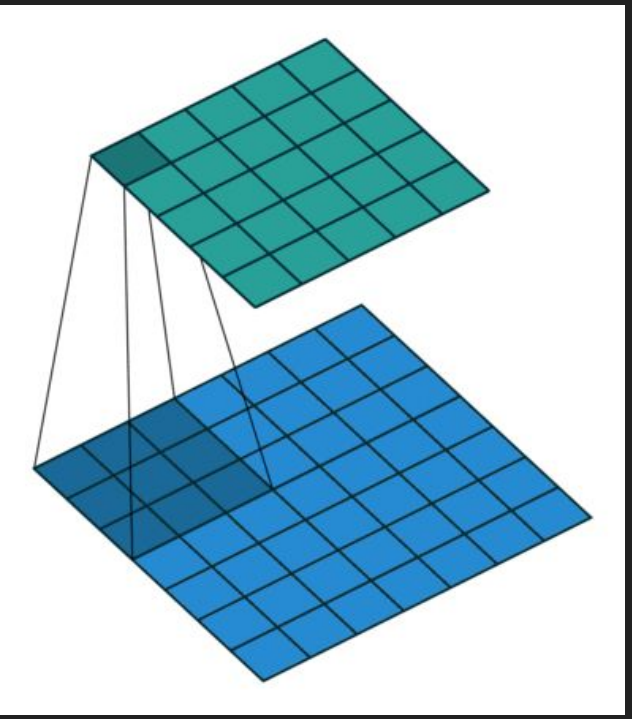
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04

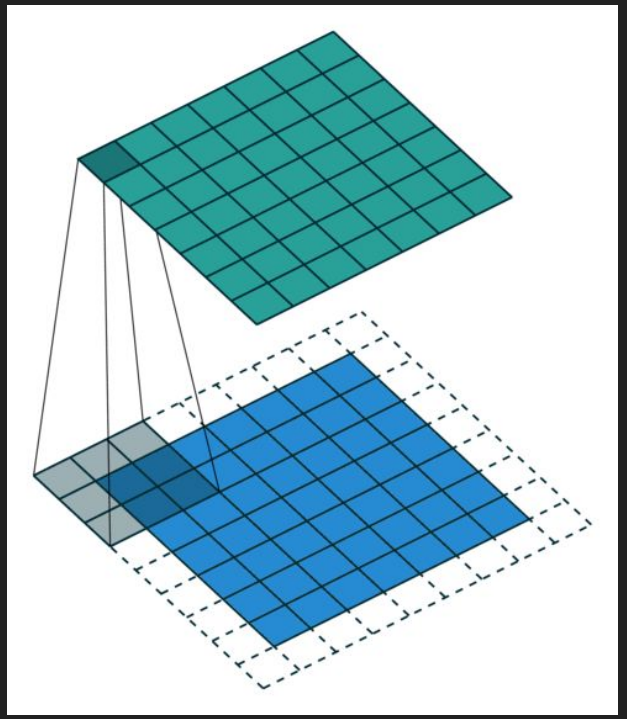
blur

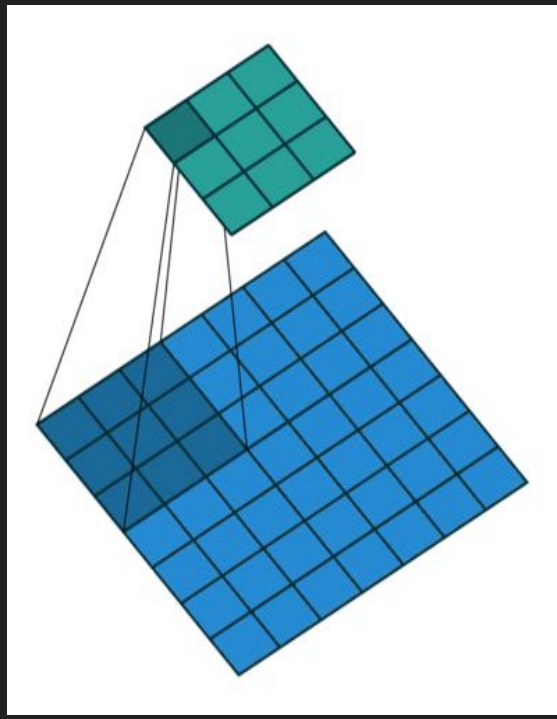
0	-1/4	0
-1/4	1	-1/4
0	-1/4	0

edge detection

Convolution - padding and strides







Remark: when stride=1 it is is common to use padding of type SAME (preserving size)

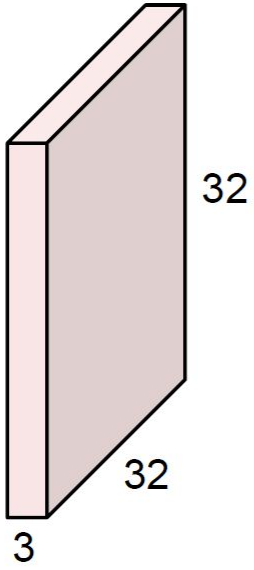
Images generated by using https://github.com/vdumoulin/conv_arithmetic

Convolution

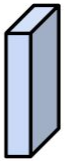
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



5x5x3 filter

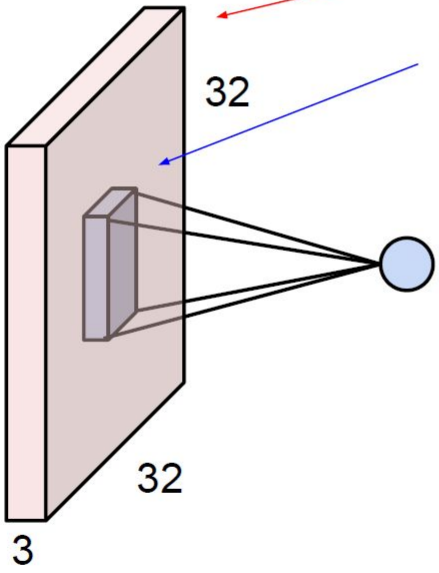


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Slide taken from CS231n@stanford

Convolution

Convolution Layer



32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Slide taken from CS231n@stanford

Convolution

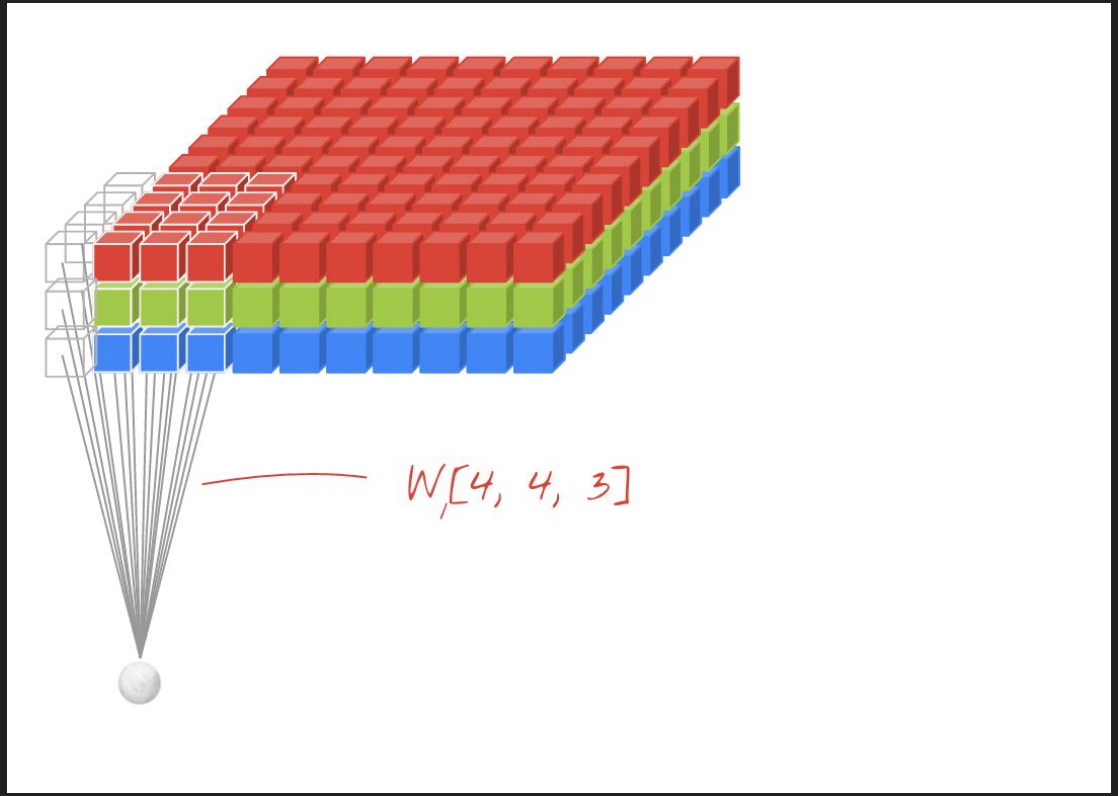
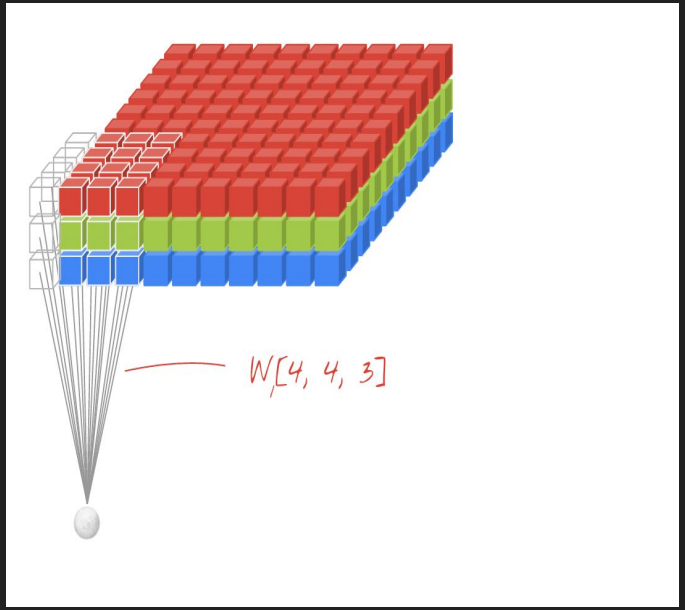
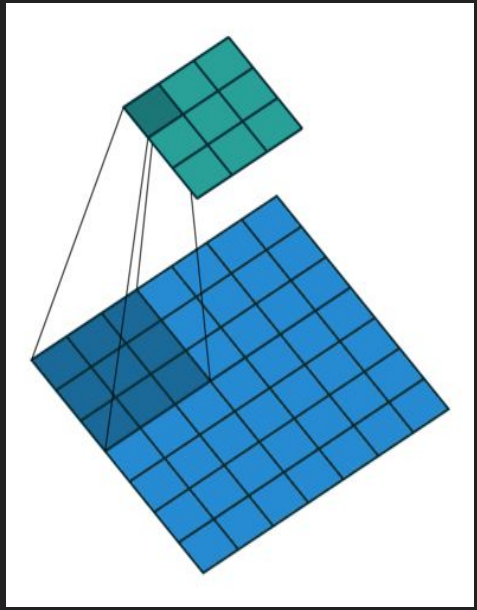


Image from https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#10

Convolution - exercise





Exercise:
Input volume 33x33x5
Apply 10 filters of size 3x3, stride 2, no padding

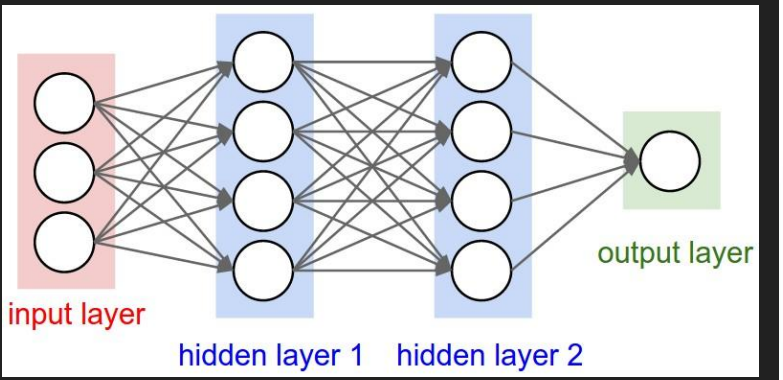
What is the output size:

16x16x10

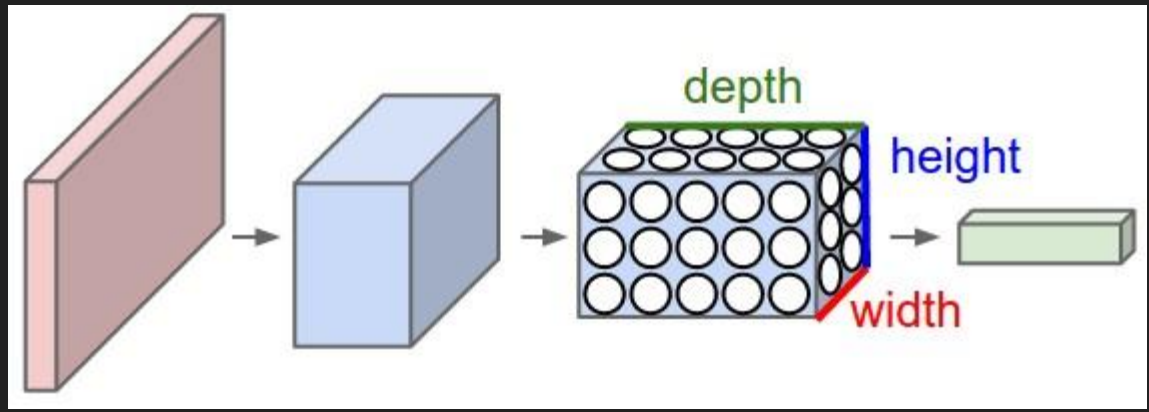
What is the number of parameters:

10 * (3 * 3 * 5 + 1) = 460

MLP vs CNN



Multilayer perceptron (MLP)
Layer transforms sequence to sequence



Convolutional Neural Network (convnet, CNN)
Layer transforms volume to volume

Images taken from cs231n.github.io/convolutional-networks/

Pooling layer

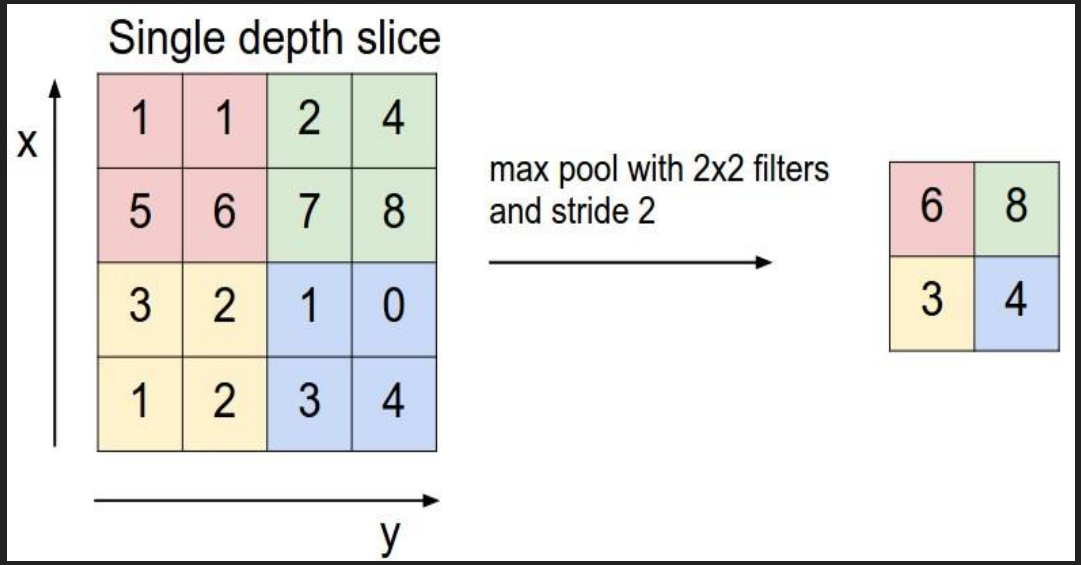
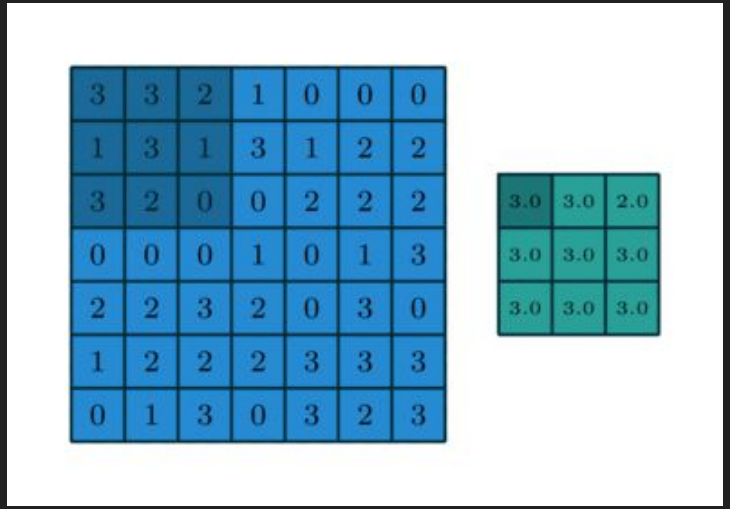


Image taken from cs231n.github.io/convolutional-networks/



Generated by https://github.com/vdumoulin/conv_arithmetic

Pooling layer

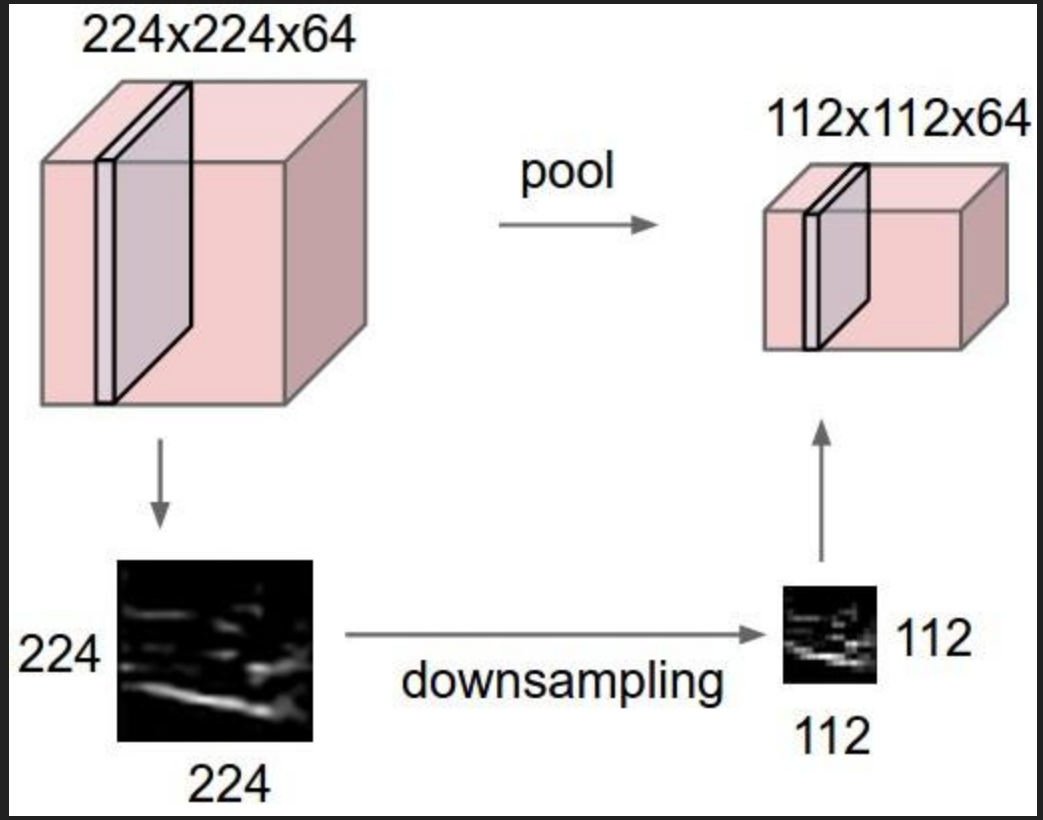


Image taken from cs231n.github.io/convolutional-networks/

- Pooling layers have no parameters!
- Max pooling much more common than average pooling
- Common choices:
  2x2 max pool with stride 2
  3x3 max pool with stride 2

Batch normalization

S. Ioffe, C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* ICML 2015: 448-456

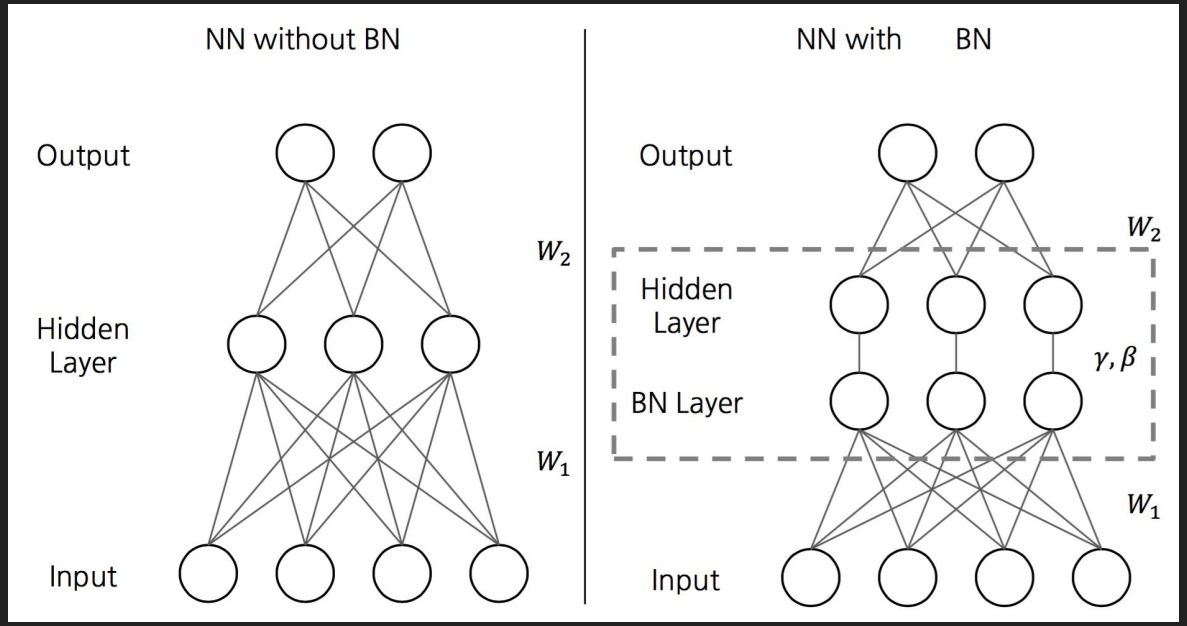


Image taken from https://wiki.tum.de/display/lfdv/Batch+Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

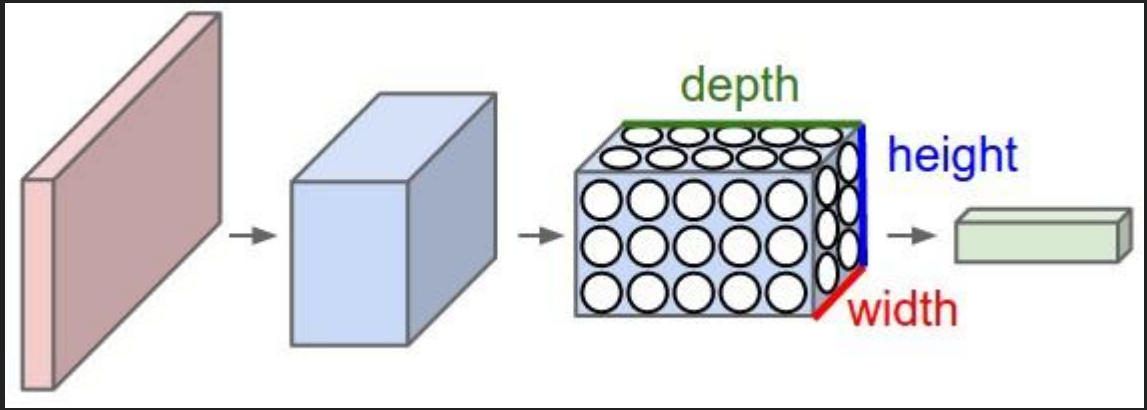
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Batch normalization

S. Ioffe, C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* ICML 2015: 448-456



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

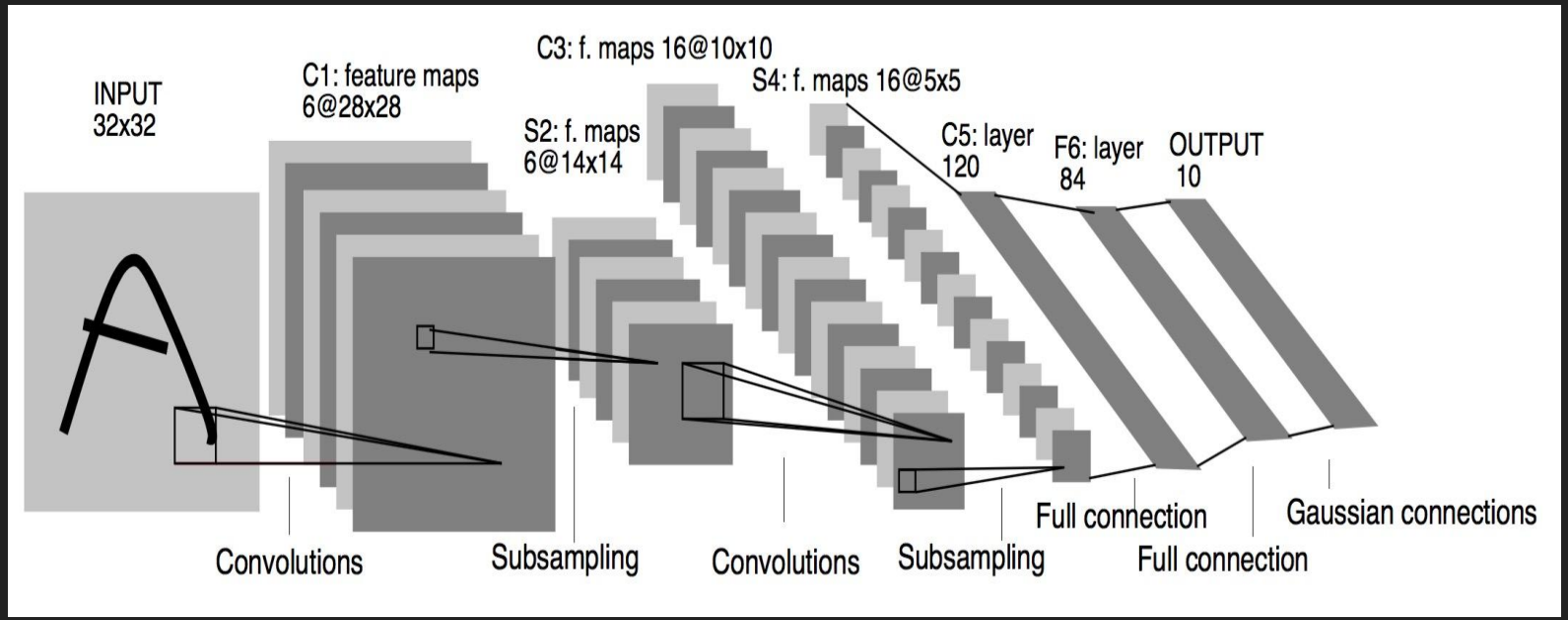
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Important: for convolution batch norm computes mean/variance not only across mini-batch, but also across spatial locations (3 out of 4 dimensions)

Intuitively we are rescaling color values across all images.

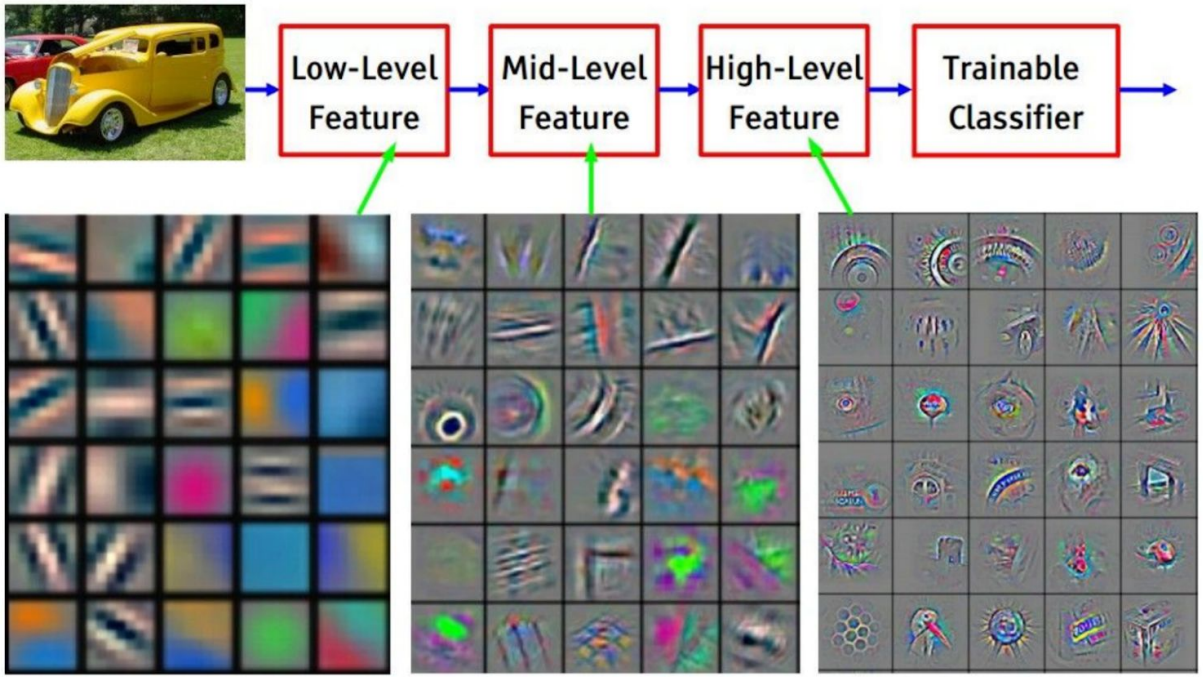
LeNet-5

All conv filters are 5x5, stride 1
Max pooling 2x2, stride 2



Hierarchical features in CNNs

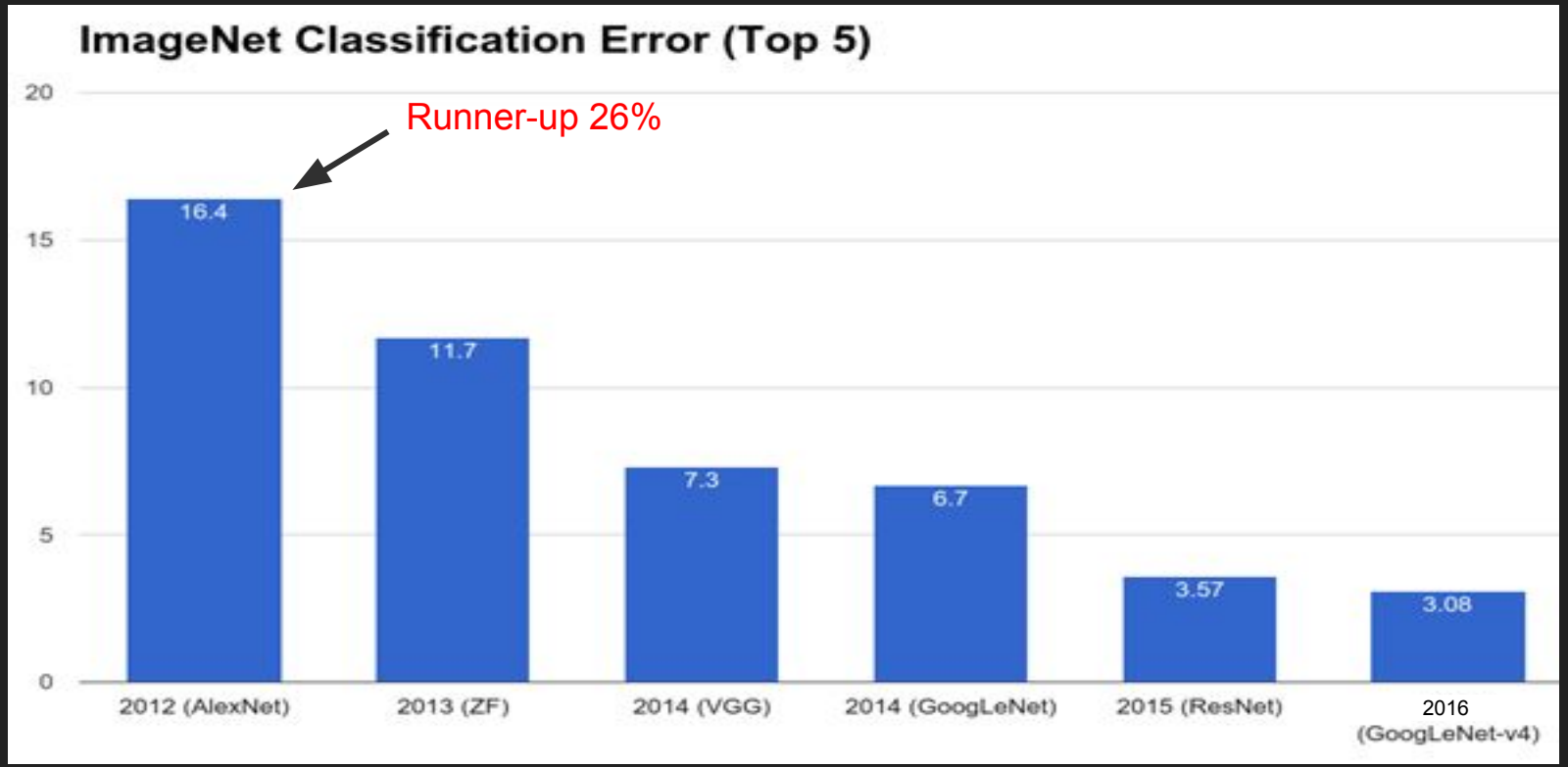
[From recent Yann LeCun slides]



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

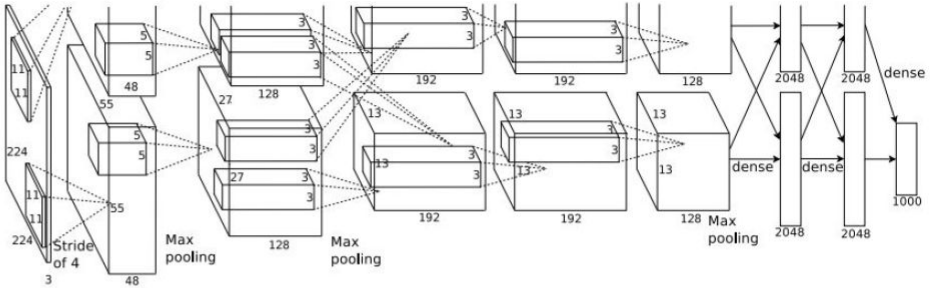
Case study: well known architectures

Image Net winning solutions - case study



Case Study: AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

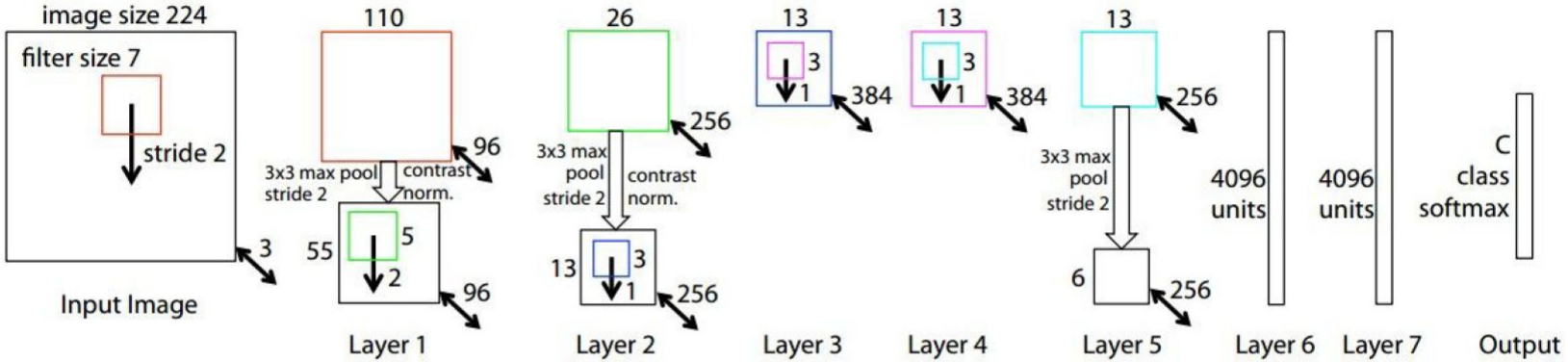
Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Trained on two GTX 580 GPUs for **five to six days**.

Slide taken from CS231n@stanford

Case Study: ZFNet

[Zeiler and Fergus, 2013]



AlexNet but:

CONV1: change from (11x11 stride 4) to (7x7 stride 2)

CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

Trained on a GTX 580 GPU for **twelve days**.

ImageNet top 5 error: 15.4% -> 14.8%

Slide taken from CS231n@stanford

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

best model

11.2% top 5 error in ILSVRC 2013
->
7.3% top 5 error

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

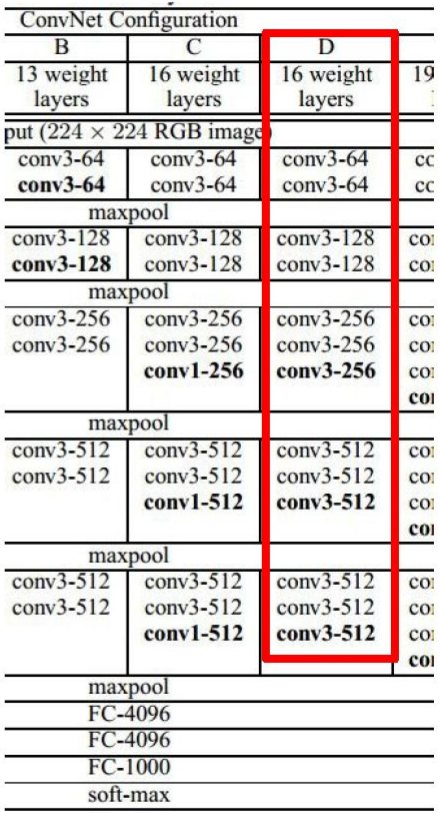
Slide taken from CS231n@stanford

Case study: VGG net

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters



ConvNet Configuration			
B	C	D	
13 weight layers	16 weight layers	16 weight layers	19
put (224 × 224 RGB image)			
conv3-64	conv3-64	conv3-64	co
conv3-64	conv3-64	conv3-64	co
maxpool			
conv3-128	conv3-128	conv3-128	co
conv3-128	conv3-128	conv3-128	co
maxpool			
conv3-256	conv3-256	conv3-256	co
conv3-256	conv3-256	conv3-256	co
	conv1-256	**conv3-256**	co
			co
maxpool			
conv3-512	conv3-512	conv3-512	co
conv3-512	conv3-512	conv3-512	co
	conv1-512	**conv3-512**	co
			co
maxpool			
conv3-512	conv3-512	conv3-512	co
conv3-512	conv3-512	conv3-512	co
	conv1-512	**conv3-512**	co
			co
maxpool			
FC-4096			
FC-4096			
FC-1000			
soft-max			

Slide taken from CS231n@stanford

Case study: VGG net

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

Note:

Most memory is in early CONV

Most params are in late FC

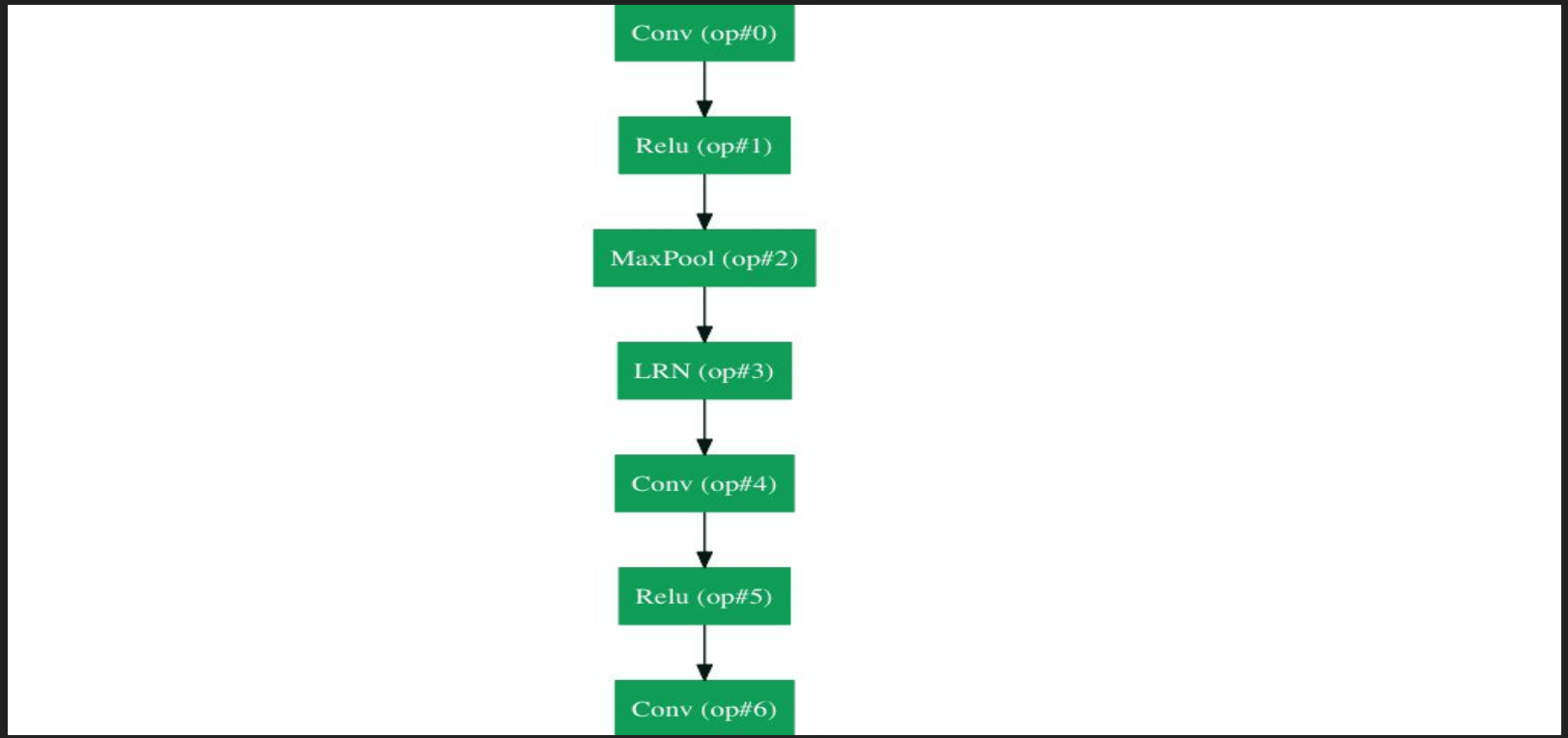
Slide taken from CS231n@stanford

Case study: VGG net

- batch size 256
- momentum 0.9
- weight decay 5×10^{-4}
- dropout regularization for the first two fully-connected layers 0.5
- SGD learning rate was initially set to 10^{-2}, and decreased by a factor of 10 when the validation set accuracy stopped improving. The learning rate decreased 3 times in total.
- Training time 2-3 weeks (per model) on four NVIDIA Titan Black GPUs.

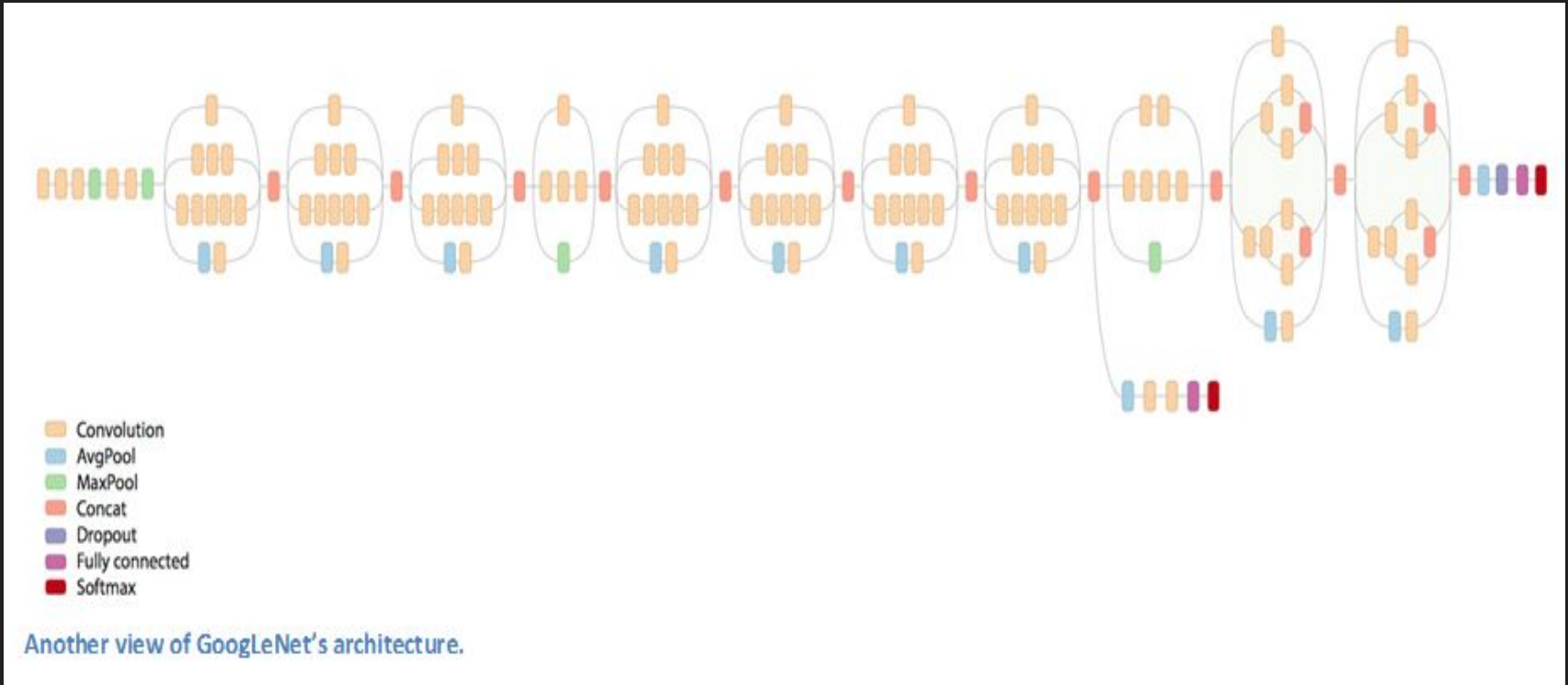
Side note:
Architecture simple to implement and train (especially with batch norm added).

Case study: GoogLeNet (2014 winner, 6.7%)



https://github.com/google/inception/blob/master/inception.ipynb

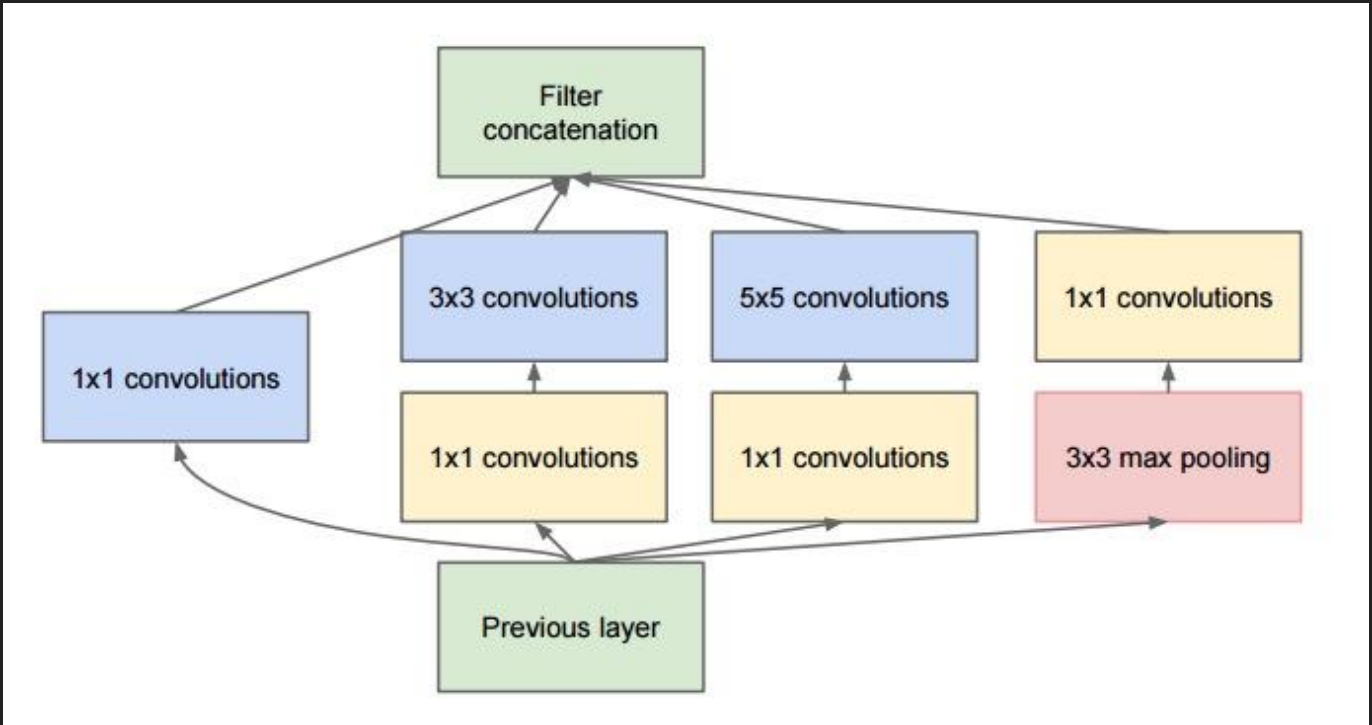
Case study: GoogLeNet



Another view of GoogLeNet's architecture.

https://github.com/google/inception/blob/master/inception.ipynb

Case study: GoogLeNet
inception module



Case study: GoogLeNet

From the paper:

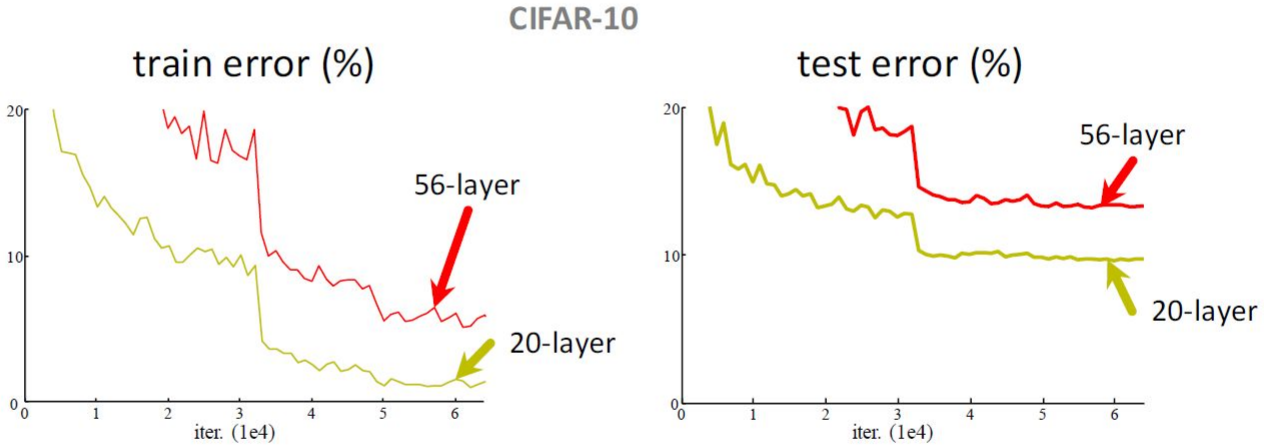
GoogLeNet networks were trained using the DistBelief distributed machine learning system using modest amount of model and data-parallelism. Although we used a CPU based implementation only, a rough estimate suggests that the GoogLeNet network could be trained to convergence using few high-end GPUs within a week, the main limitation being the memory usage. Our training used asynchronous stochastic gradient descent with 0.9 momentum, fixed learning rate schedule (decreasing the learning rate by 4% every 8 epochs).

Interesting facts:

- Used 9 Inception modules in the whole architecture, with over 100 layers in total!
- No use of fully connected layers! Average pool is used instead, to go from a 7x7x1024 volume to a 1x1x1024 volume.
- Uses 12x fewer parameters than AlexNet.

Case study: ResNet

Simply stacking layers?

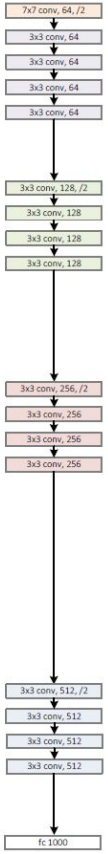


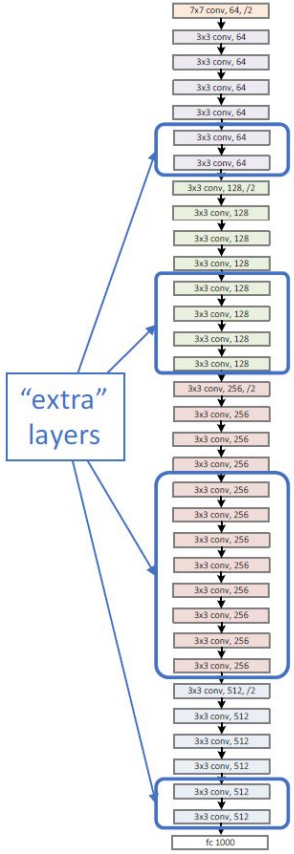
- *Plain* nets: stacking 3x3 conv layers…
- 56-layer net has **higher training error** and test error than 20-layer net

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Case study: ResNet

a shallower model (18 layers)



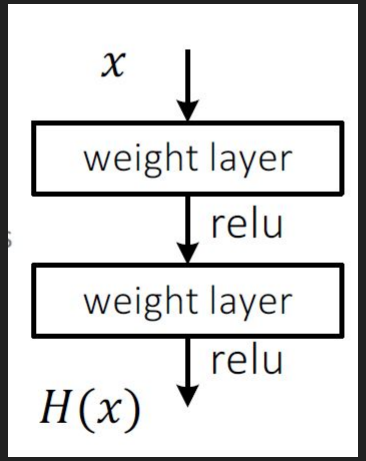


a deeper counterpart (34 layers)

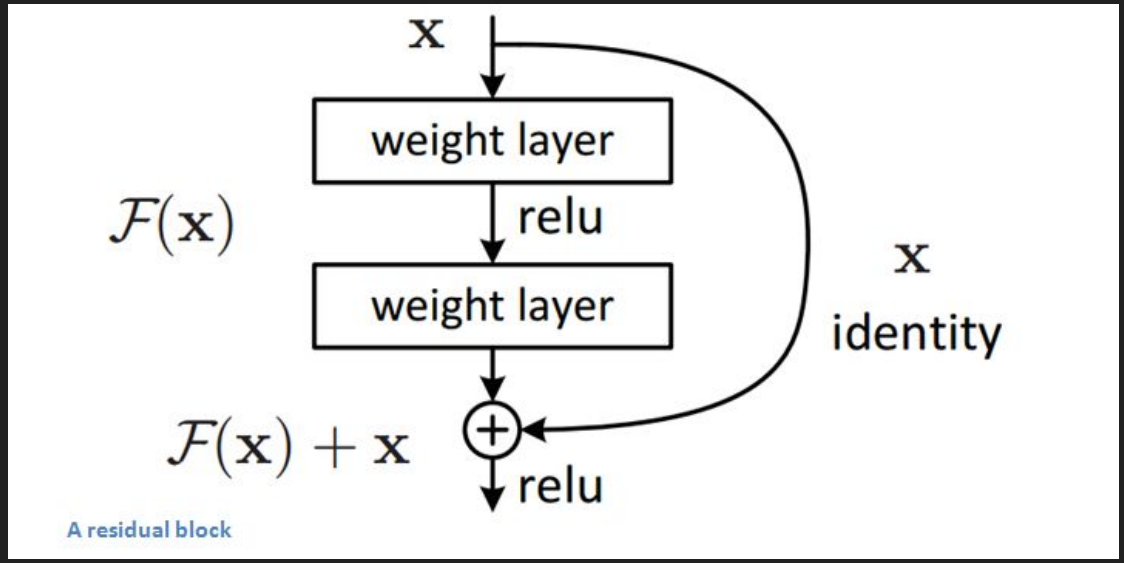
- Richer solution space
- A deeper model should not have **higher training error**
- A solution *by construction*:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
- Optimization difficulties: solvers cannot find the solution when going deeper…

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Case study: ResNet solution - residual blocks

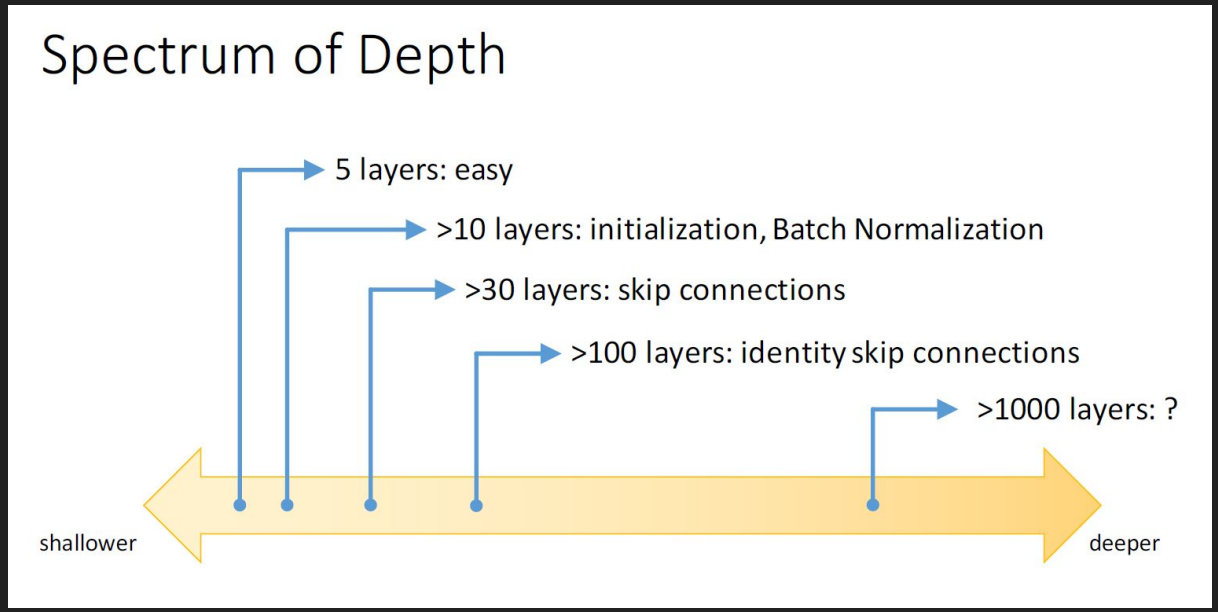


Plain net



K He, X Zhang, S Ren, J Sun, *Deep Residual Learning for Image Recognition* Computer Vision and Pattern Recognition (CVPR), 2016 `

Summary



- Nowadays filters are usually small (3x3, 5x5, 1x1)
- More and more layers
- Towards fully convolutional architectures
- Training large network for large dataset can take a week or two

http://kaiminghe.com/icml16tutorial/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

Lab session - administrative items

- You can sign up for AWS EC2 educate program as a University of Warsaw student to get 70$ (or 100$) credits.
- Everyone will be given access to the sylvester server (4x Pascal Titan X)
- Sign the ICM account form (and give your email) if you want to get access to GPUs at ICM (4xPascal Titan X).
- In the case of high demand we might add one more server with 4 GPUs in May.
- First homework will be released by tomorrow with a deadline in two weeks, it will be related to MNIST and convnets, so it is worth having a good solution for today's lab session.