MADMO
Introduction to …

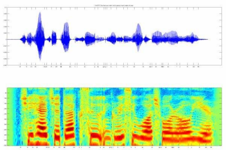
Deep Learning: CNNs

Taras Khakhulin
Deep Learning Engineer Samsung AI Center
Skoltech and MIPT Master Student

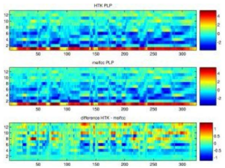
t.khakhulin@gmail.com
https://github.com/khakhulin/
https://twitter.com/t_khakhulin

Real world problems

Audio Features

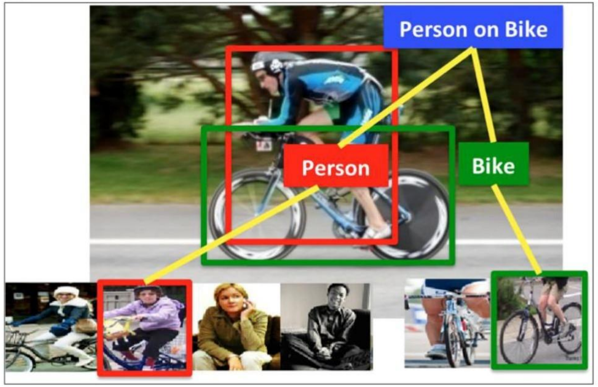


Spectrogram



MFCC

- Object detection
- Action classification
- Image captioning
- …

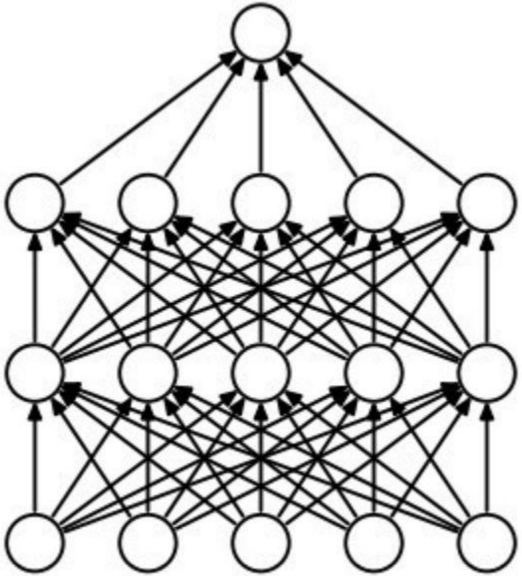


"man in black shirt is playing guitar."

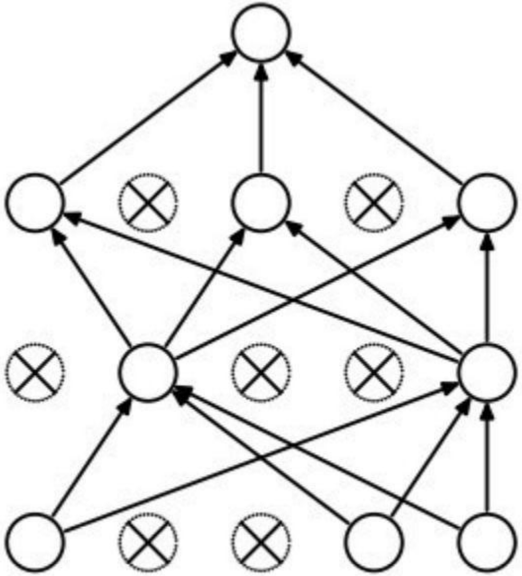
Regularization: Dropout

Some neurons are “dropped” during training.

Prevents overfitting.



(a) Standard Neural Net



(b) After applying dropout.

Regularization

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \boxed{\lambda R(W)}$$

Adding some extra term to the loss function.

Common cases:

- L2 regularization: $R(W) = \|W\|_2^2$
- L1 regularization: $R(W) = \|W\|_1$
- Elastic Net (L1 + L2): $R(W) = \beta \|W\|_2^2 + \|W\|_1$

source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

Batch normalization

- Normalize activation of a hidden layer

  (zero mean unit variance)

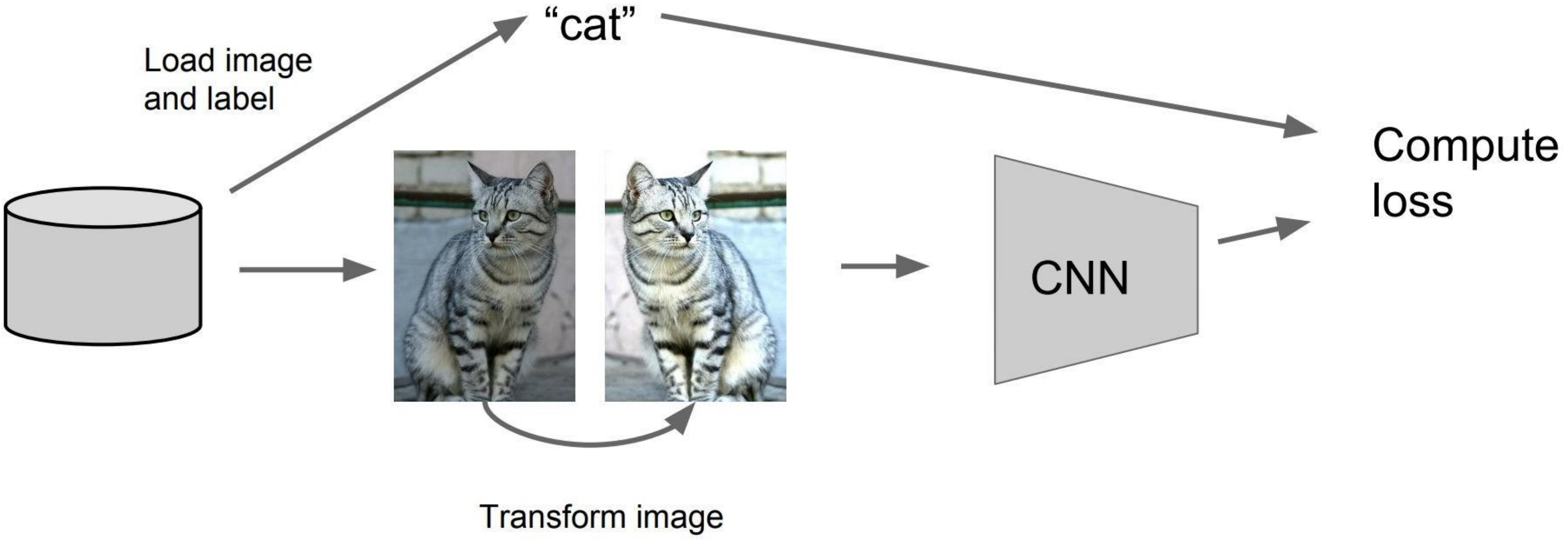
$$h_i = \frac{h_i - \mu_i}{\sqrt{\sigma_i^2}}$$

- Update μ_i, σ_i^2 with moving average while training

$$\mu_i := \alpha \cdot mean_{batch} + (1 - \alpha) \cdot \mu_i$$

$$\sigma_i^2 := \alpha \cdot variance_{batch} + (1 - \alpha) \cdot \sigma_i^2$$

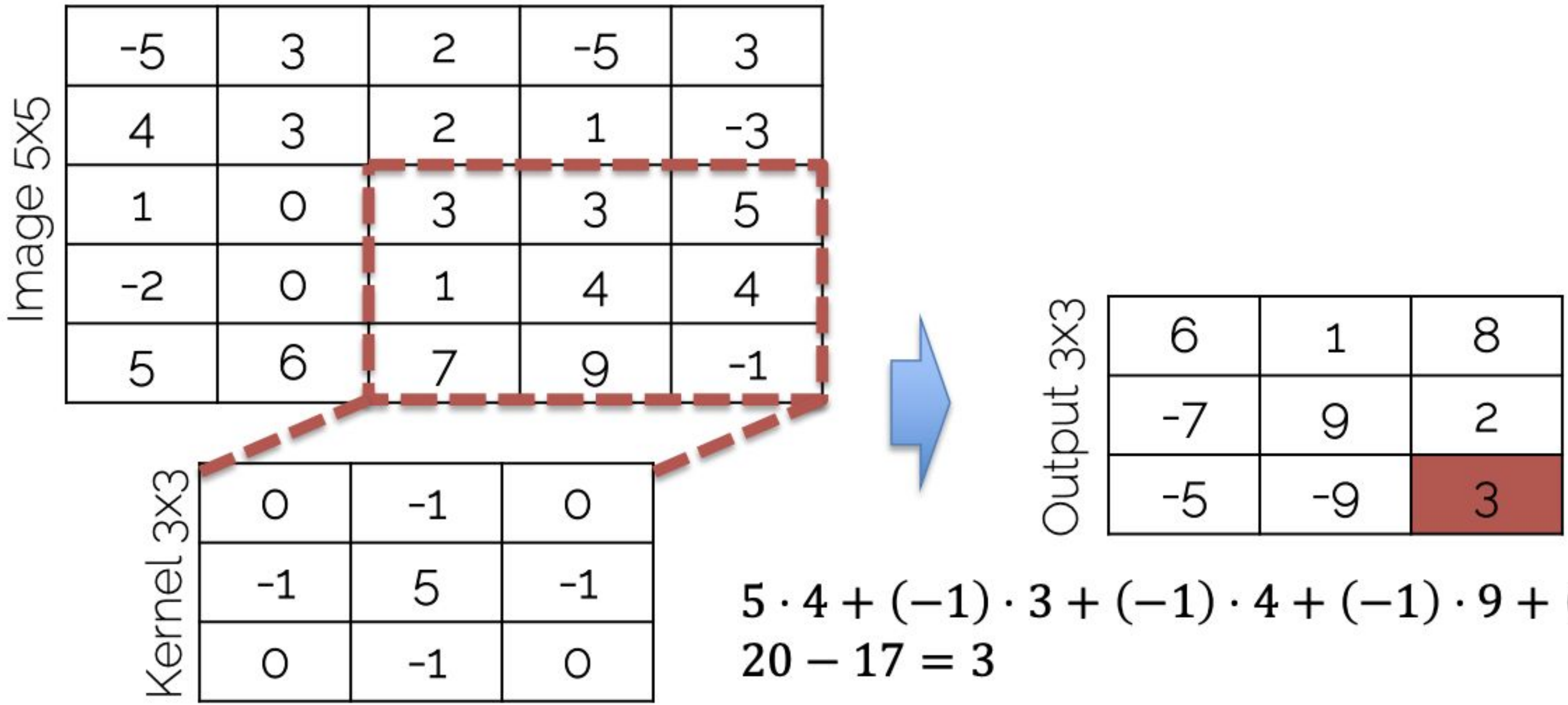
Regularization: data augmentation



source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

CNN: dive into sea

CNN: image recap



$$5 \cdot 4 + (-1) \cdot 3 + (-1) \cdot 4 + (-1) \cdot 9 + (-1) \cdot 1 =$$
$$20 - 17 = 3$$

CNN: image recap

- Each kernel gives us a different image filter

Input

Edge detection

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Box mean

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpen

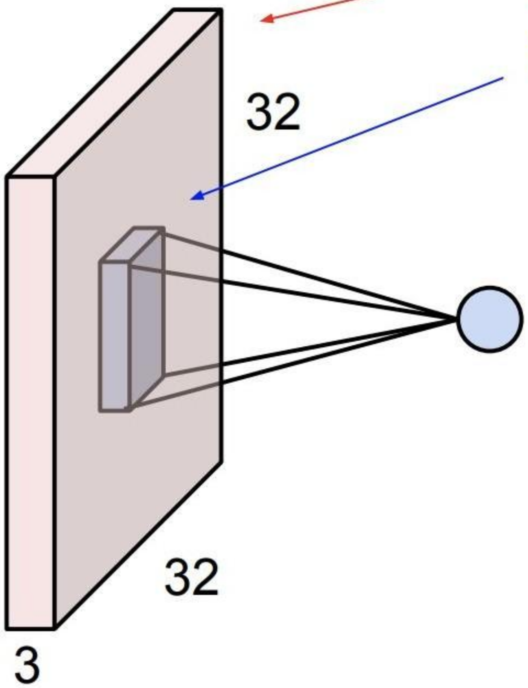
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Gaussian blur

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

LET'S LEARN THESE FILTERS!

Convolutional layer



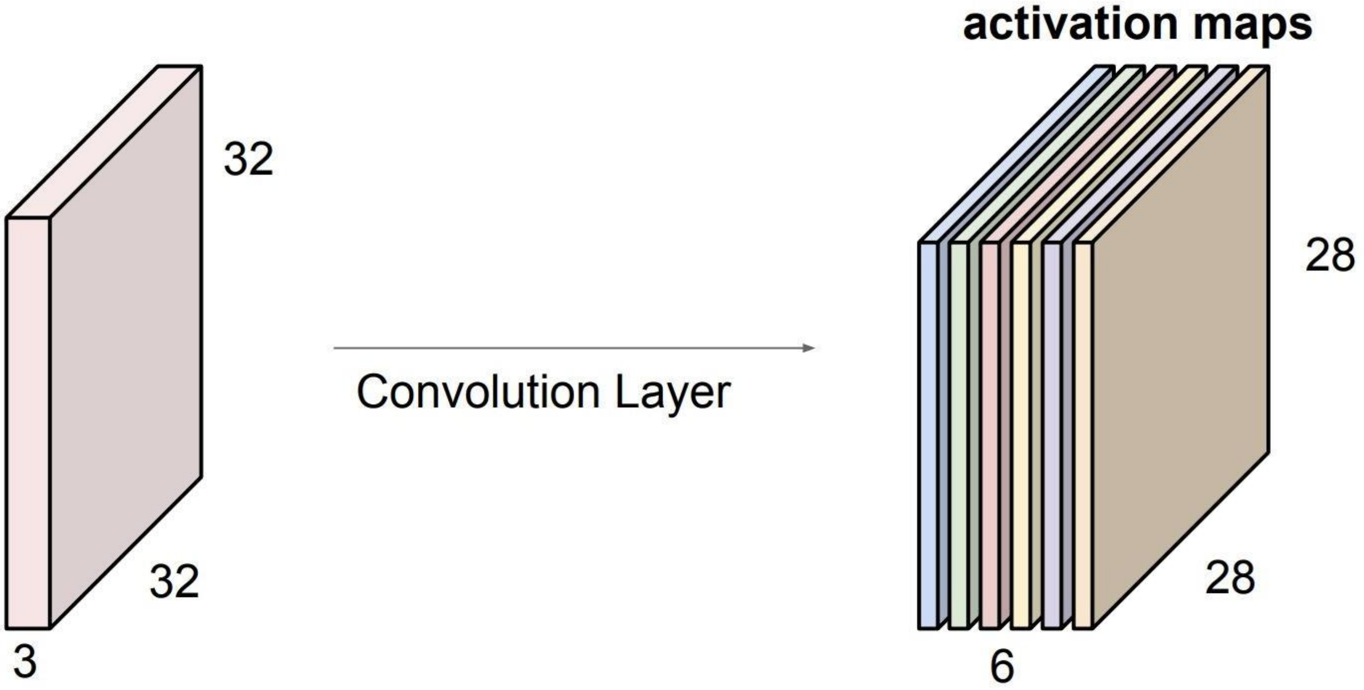
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

source: http://cs231n.stanford.edu/slides/2016/winter1516_lecture7.pdf

Convolutional layer

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

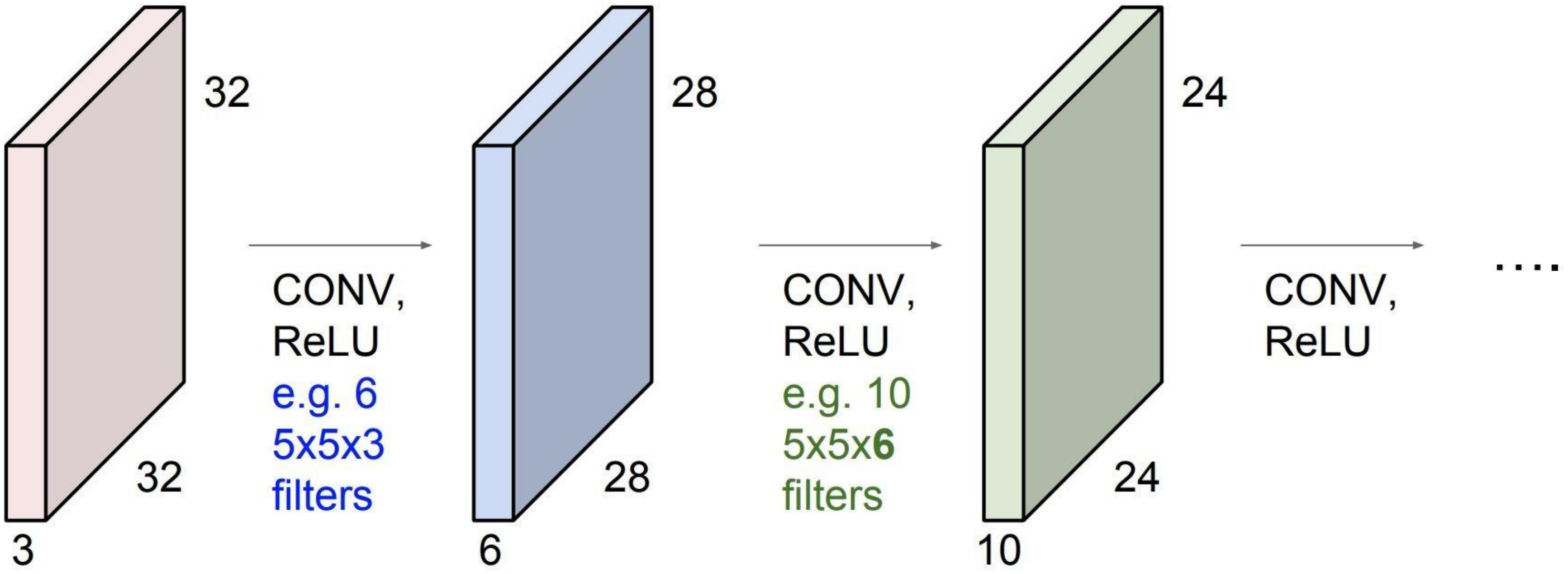


We stack these up to get a "new image" of size 28x28x6!

source: http://cs231n.stanford.edu/slides/2016/winter1516_lecture7.pdf

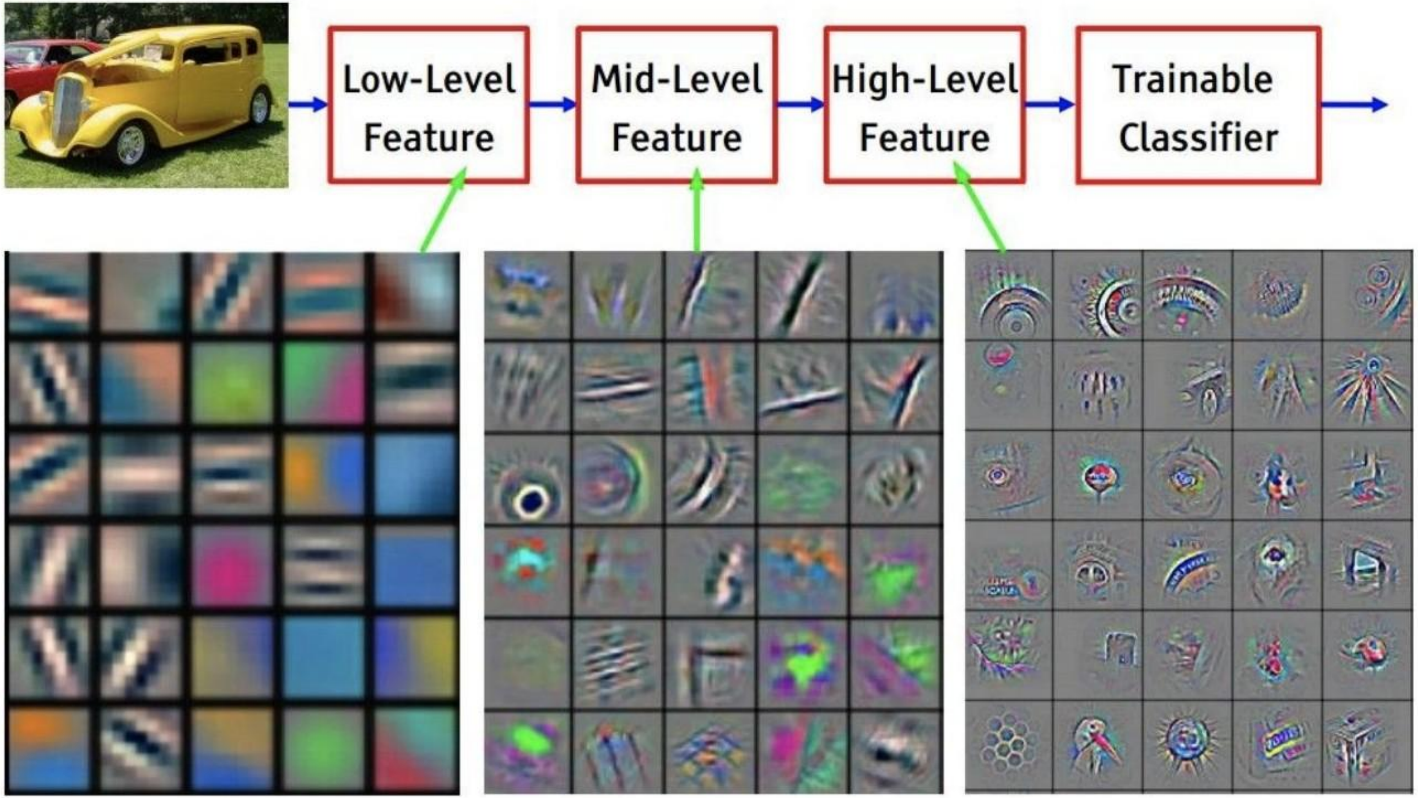
Convolutional layer

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



source: http://cs231n.stanford.edu/slides/2016/winter1516_lecture7.pdf

Convolutional layer

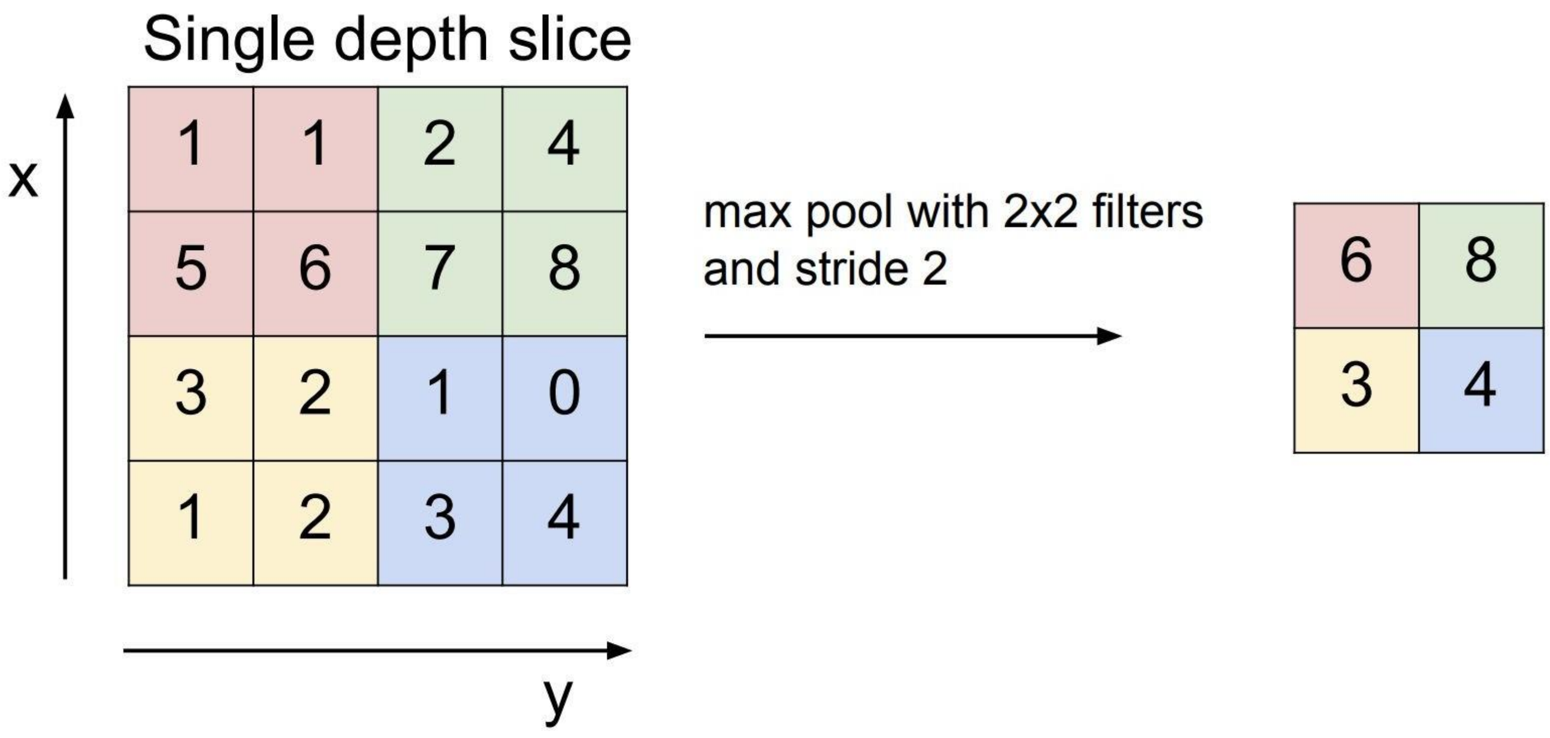


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

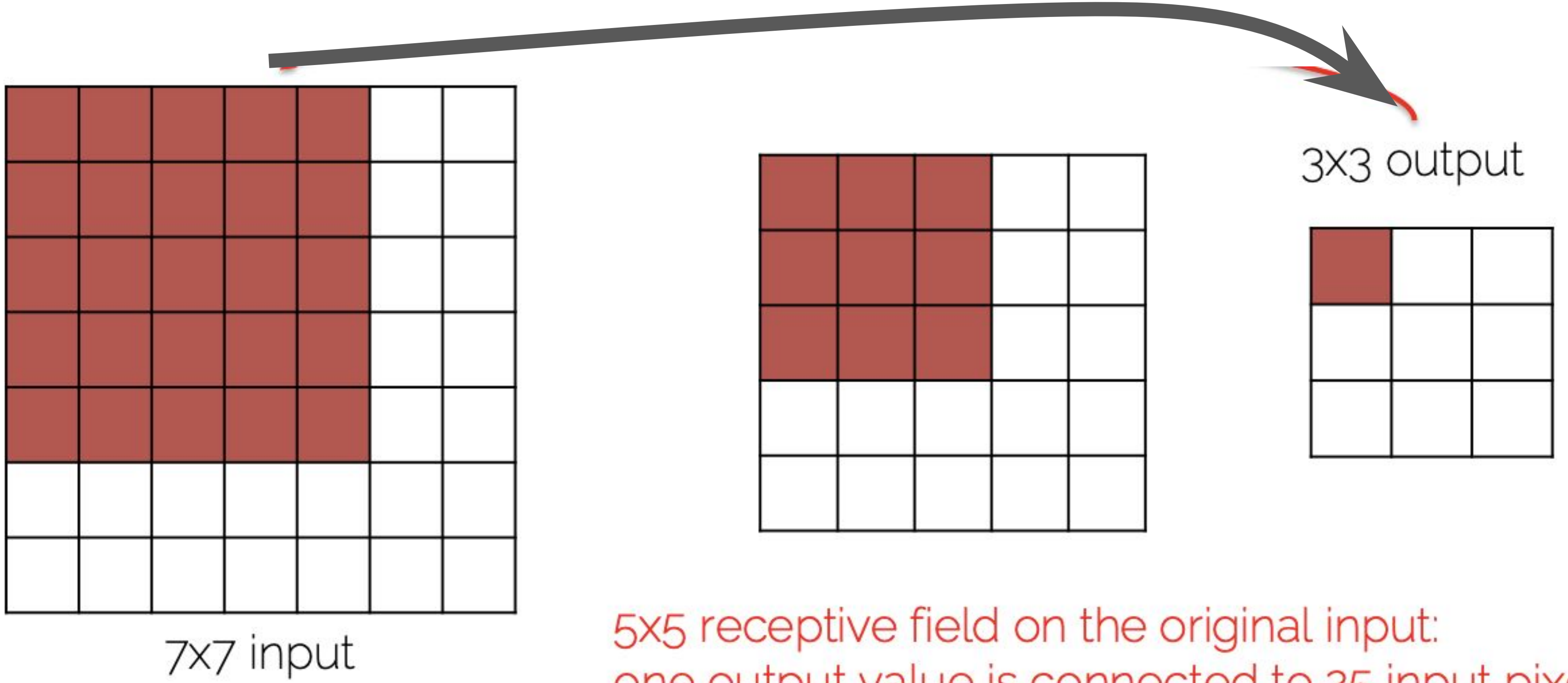
[From Yann LeCun slides]

source: http://cs231n.stanford.edu/slides/2016/winter1516_lecture7.pdf

Max pooling



source: http://cs231n.stanford.edu/slides/2016/winter1516_lecture7.pdf

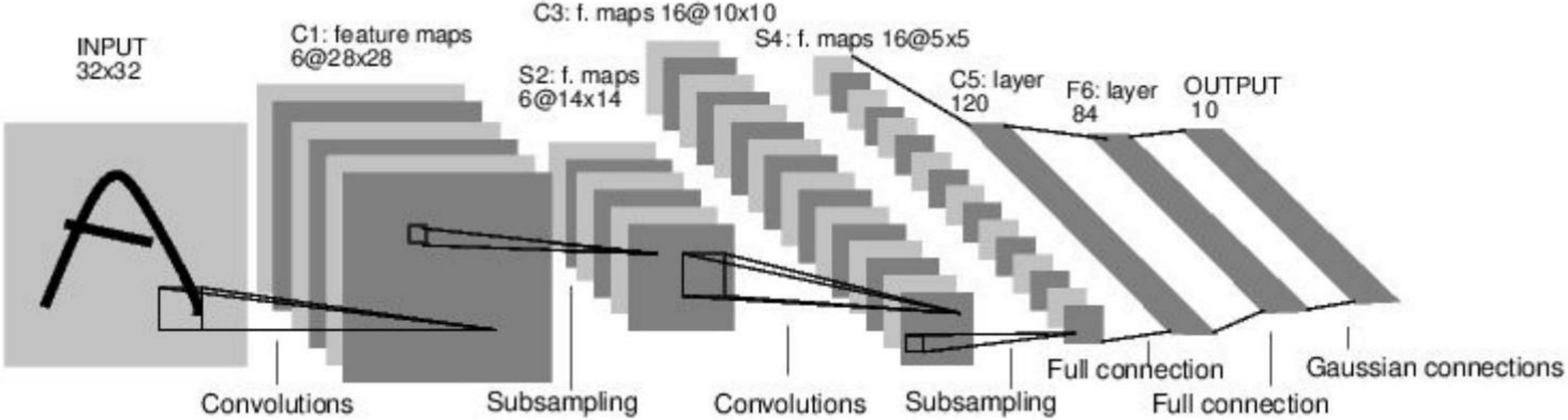


5x5 receptive field on the original input:
one output value is connected to 25 input pixels

y

Architectures overview

CNN



[LeNet-5, LeCun 1998]

LeNET

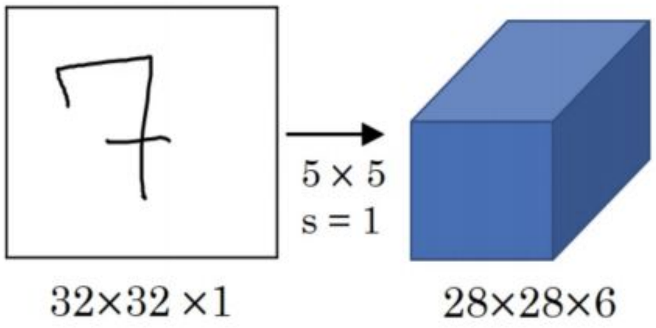
- Digit recognition: 10 classes

$32 \times 32 \times 1$

Input: 32×32 grayscale images
This one: Labeled as class "7"

LeNET

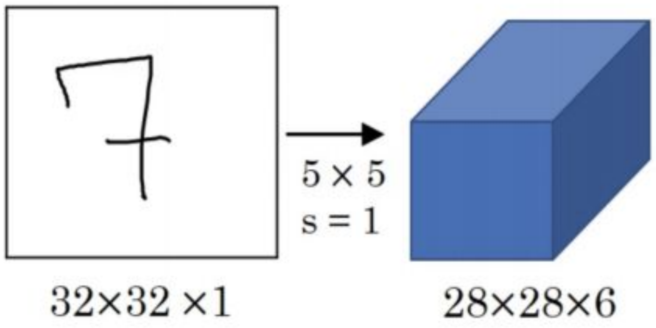
- Digit recognition: 10 classes



- Valid convolution: size shrinks
- How many conv filters are there in the first layer?

LeNET

- Digit recognition: 10 classes

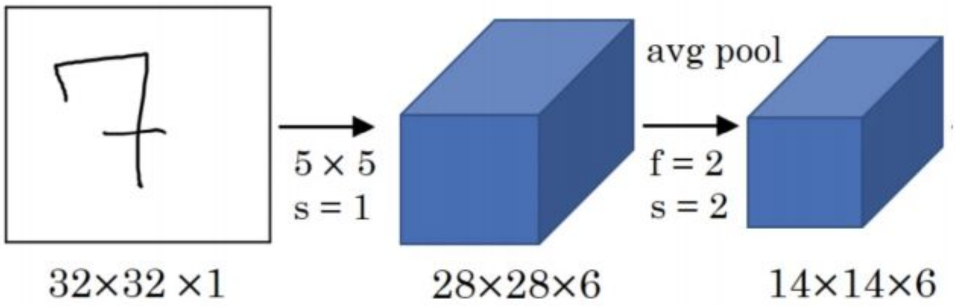


- Valid convolution: size shrinks
- How many conv filters are there in the first layer?

6

LeNET

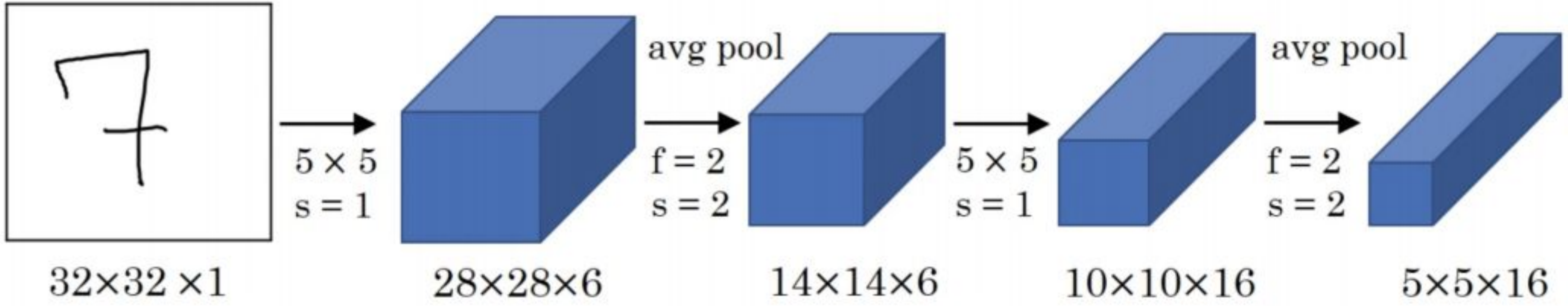
- Digit recognition: 10 classes



- At that time average pooling was used, now max pooling is much more common

LeNET

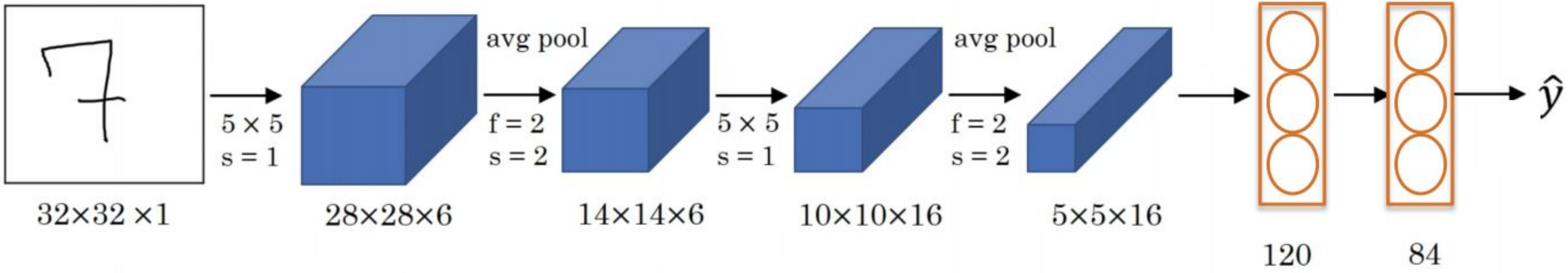
- Digit recognition: 10 classes



- Again valid convolutions, how many filters?

LeNET

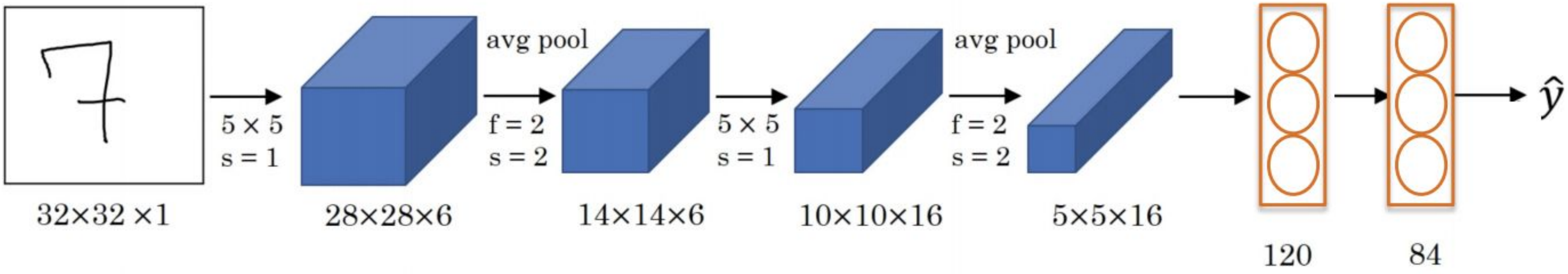
- Digit recognition: 10 classes



- Use of tanh/sigmoid activations → not common now!

LeNET

- Digit recognition: 10 classes

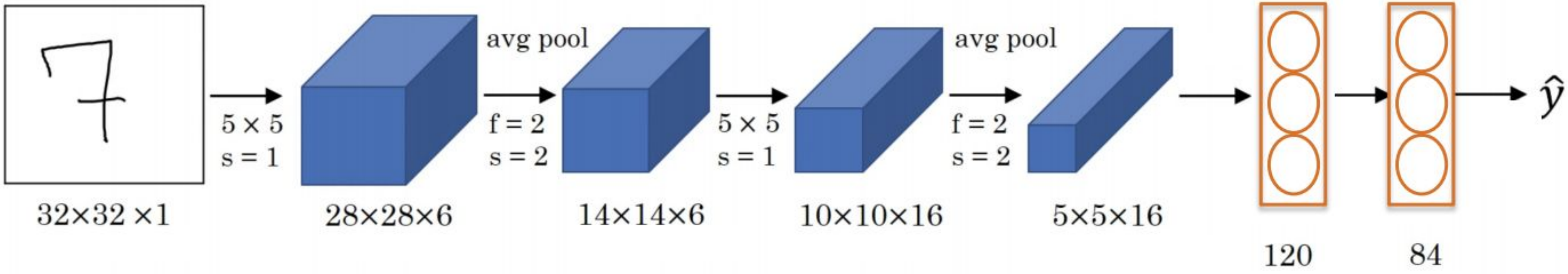


- Conv -> Pool -> Conv -> Pool -> Conv -> FC

LeNET

60k parameters

- Digit recognition: 10 classes

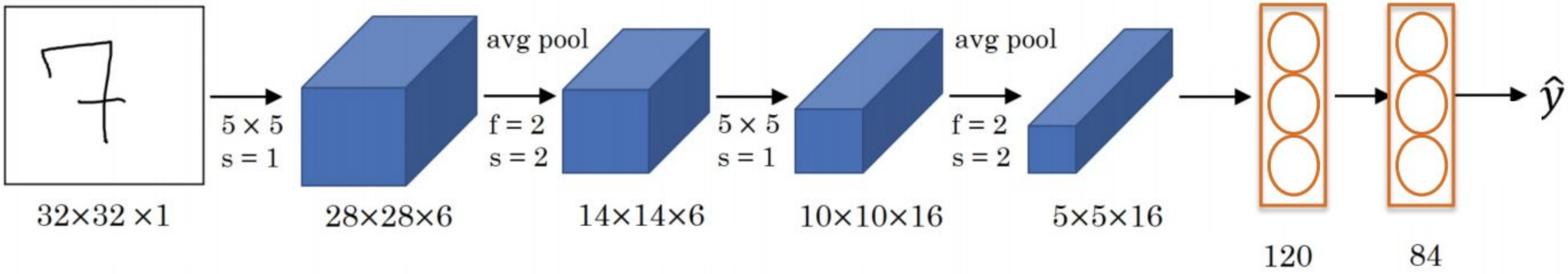


- Conv -> Pool -> Conv -> Pool -> Conv -> FC
- As we go deeper: Width, Height ↓ Number of Filters ↑

LeNET

60k parameters

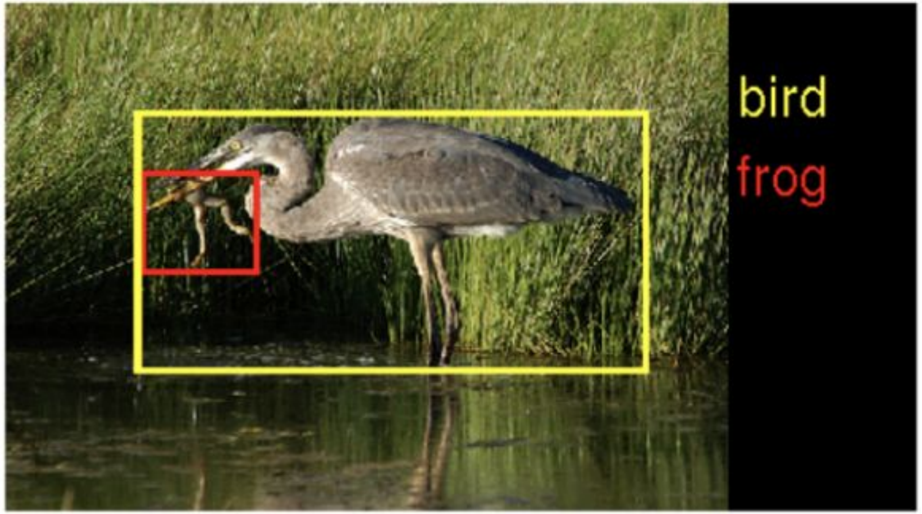
- Digit recognition: 10 classes

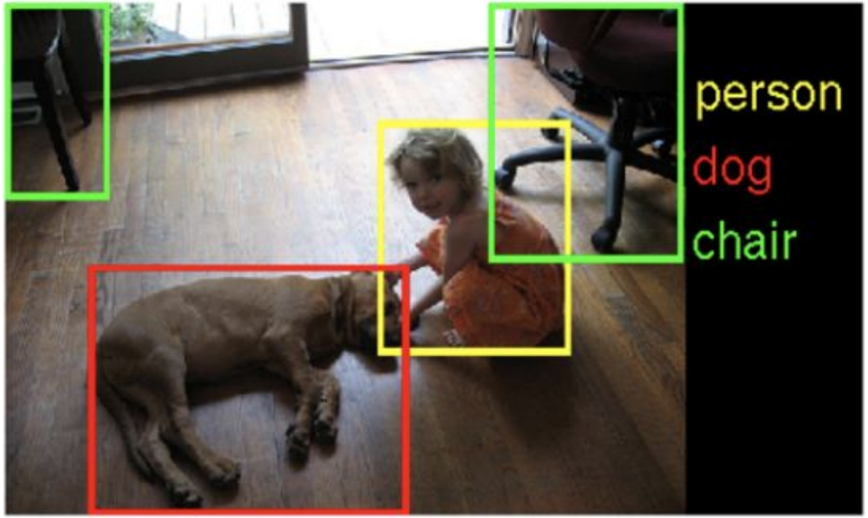


- Conv -> Pool -> Conv -> Pool -> Conv -> FC
- As we go deeper: Width, Height ↓ Number of Filters ↑

ImageNet

- ImageNet Dataset:
  ImageNet Large Scale Visual Recognition Competition (ILSVRC)



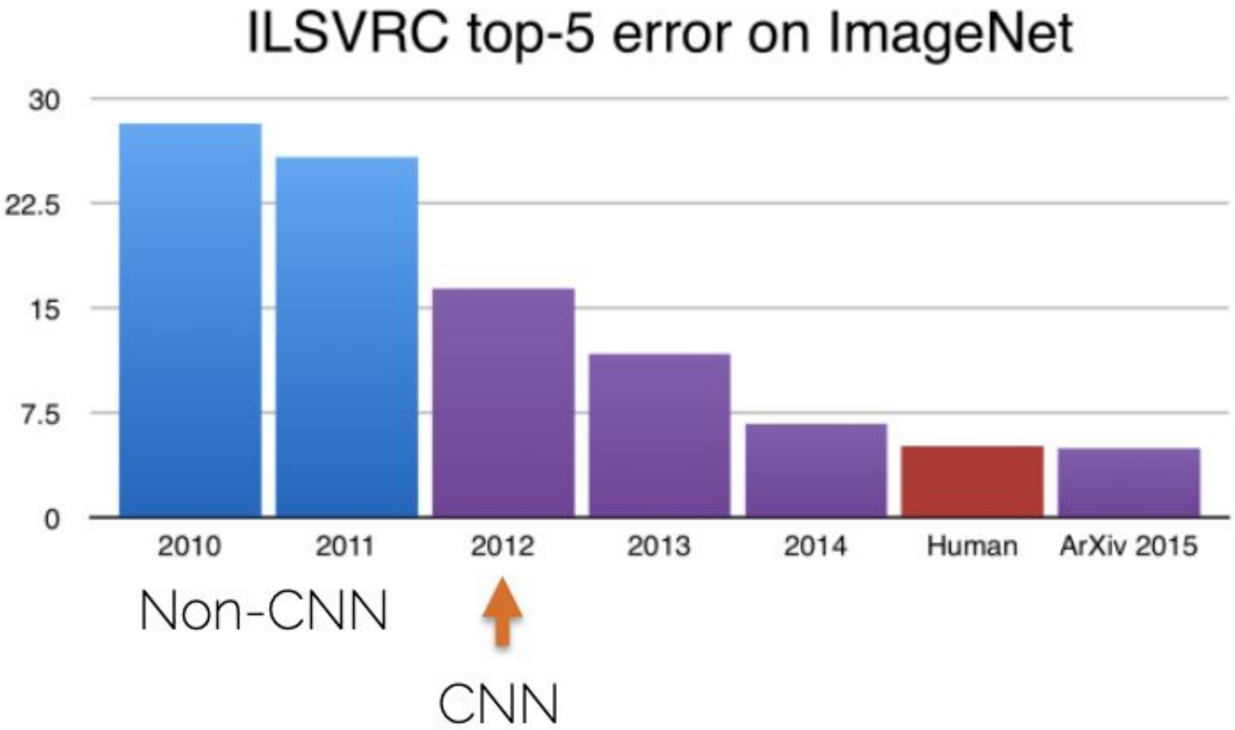


[Russakovsky et al., IJCV'15] "ImageNet Large Scale Visual Recognition Challenge."

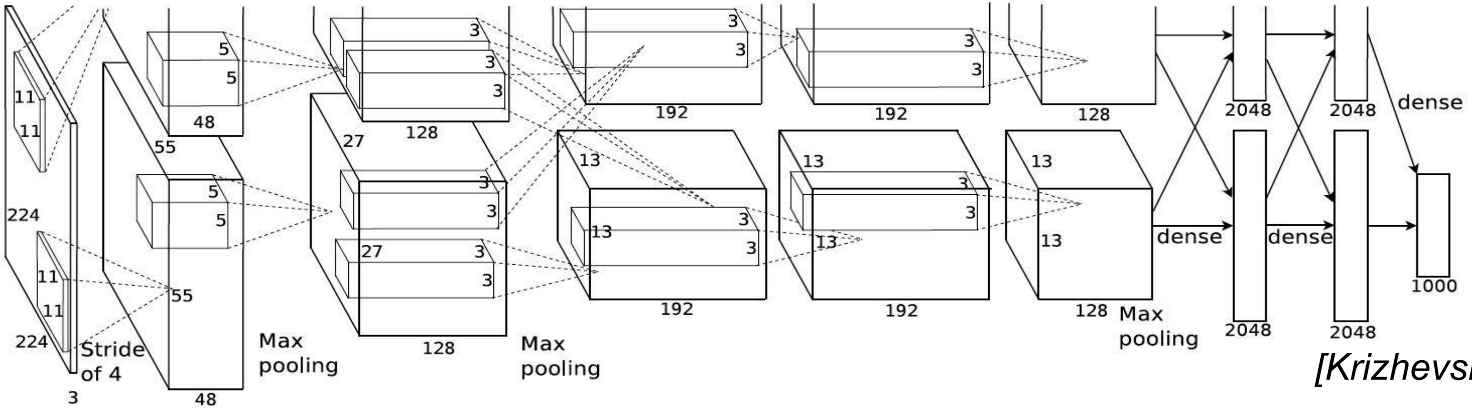
Common Terms

- Top-1 score: check if a sample's top class (i.e. the one with highest probability) is the same as its target label
- Top-5 score: check if your label is in your 5 first predictions (i.e. predictions with 5 highest probabilities)
- Top-5 error: percentage of test samples for which the correct class was not in the top 5 predicted classes

AlexNet


ILSVRC top-5 error on ImageNet
30
22.5
15
7.5
0
2010
2011
2012
2013
2014
Human
ArXiv 2015
Non-CNN
CNN

AlexNet

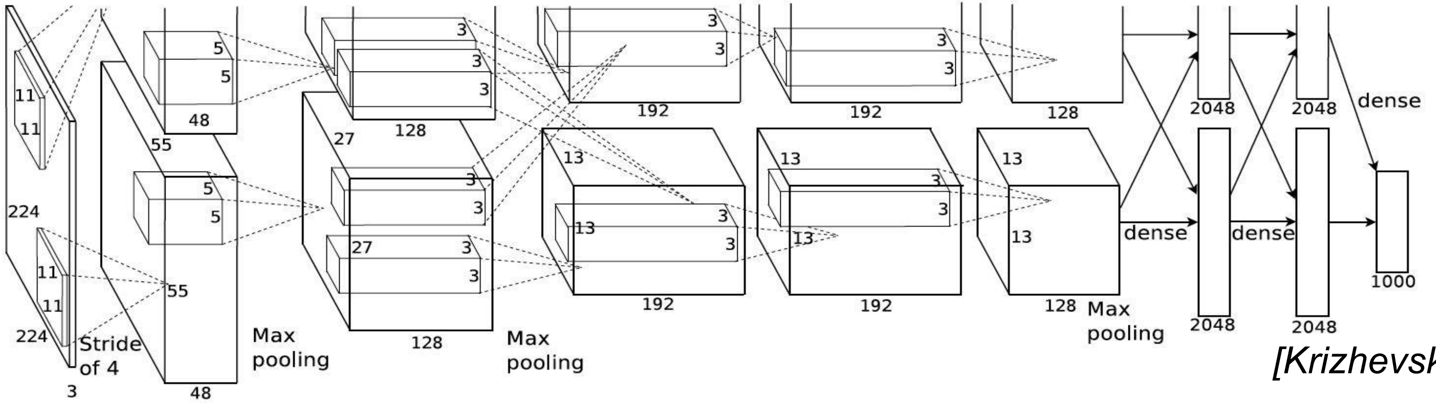


[Krizhevsky et al. 2012]

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)

source: http://cs231n.stanford.edu/slides/2016/winter1516_lecture7.pdf

AlexNet



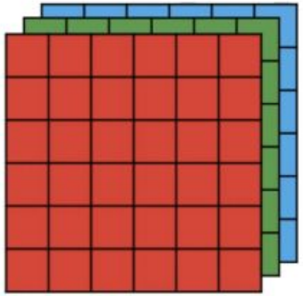
[Krizhevsky et al. 2012]

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%

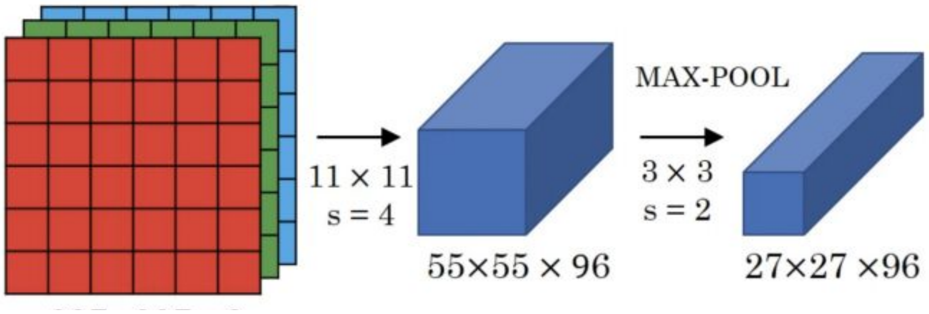
source: http://cs231n.stanford.edu/slides/2016/winter1516_lecture7.pdf

Alex Net how does it work?

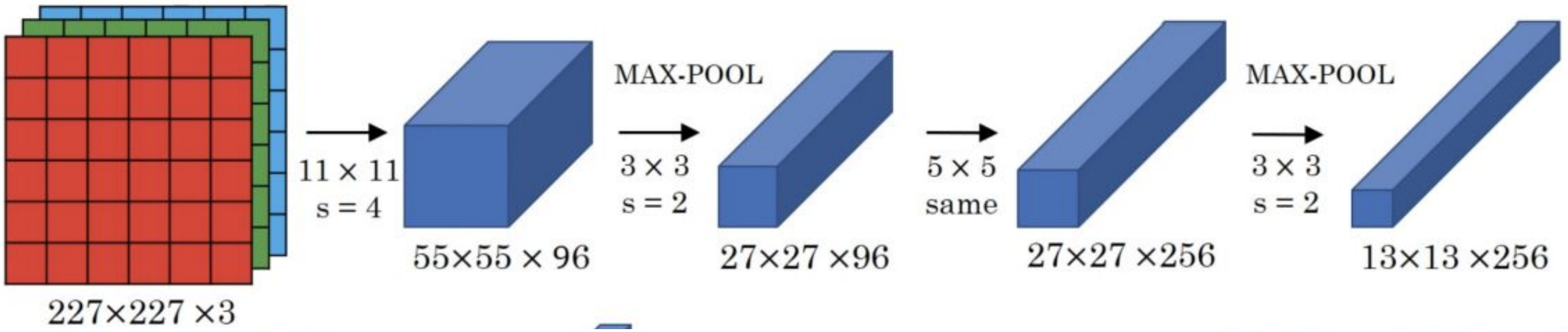


227×227 ×3

Alex Net how does it work?

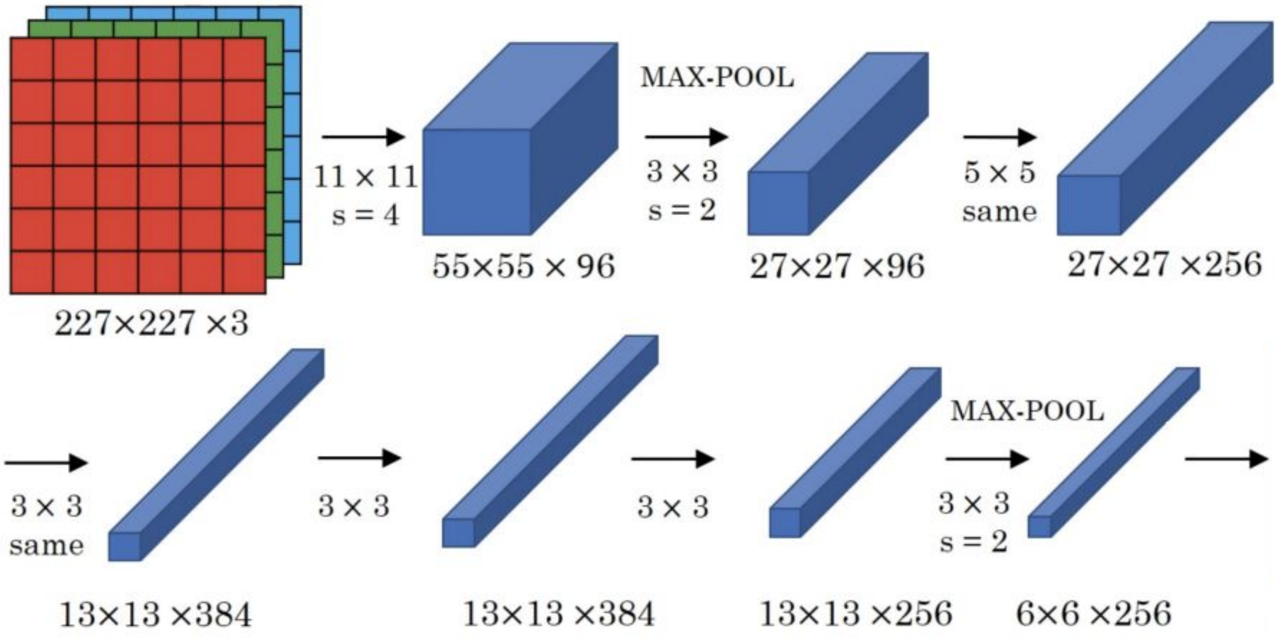


Alex Net how does it work?

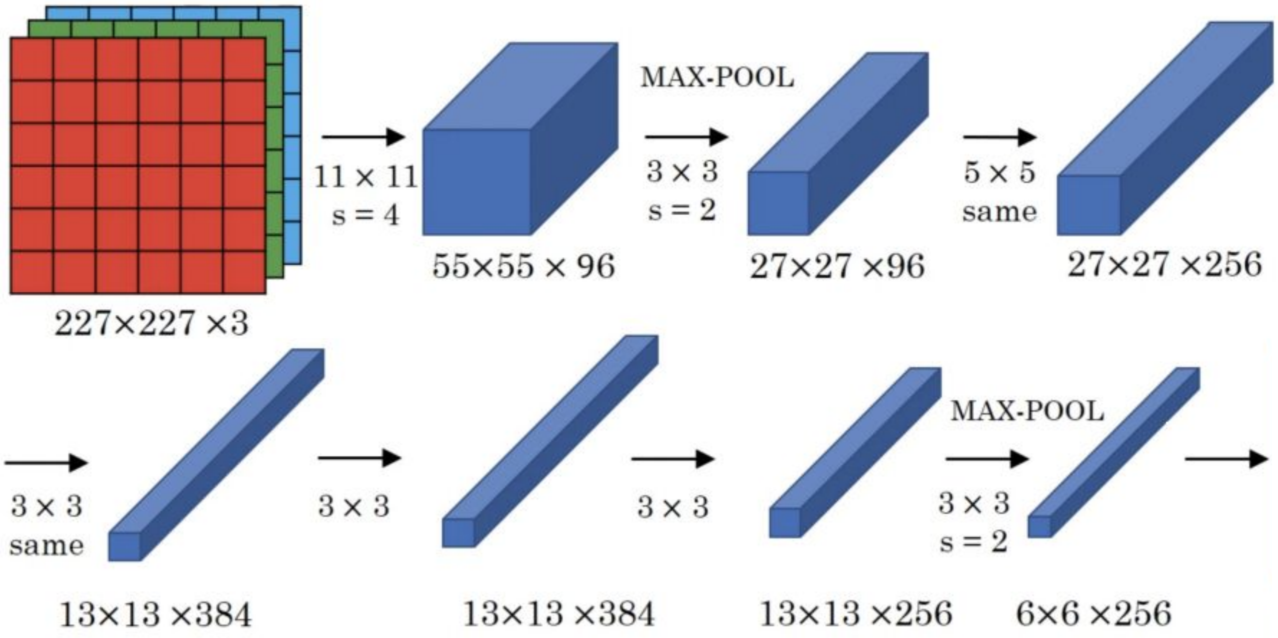


Use of same convolutions As with LeNet: Width, Height Number of Filters

Alex Net how does it work?

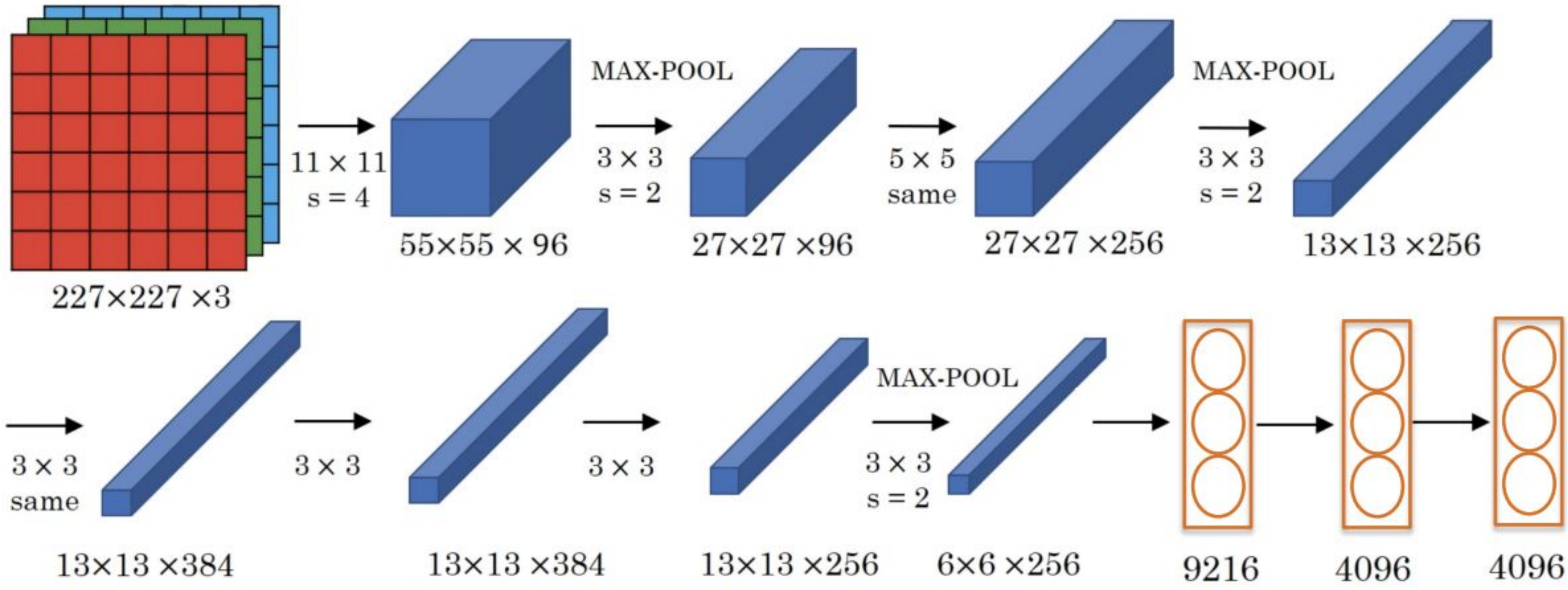


Alex Net how does it work?



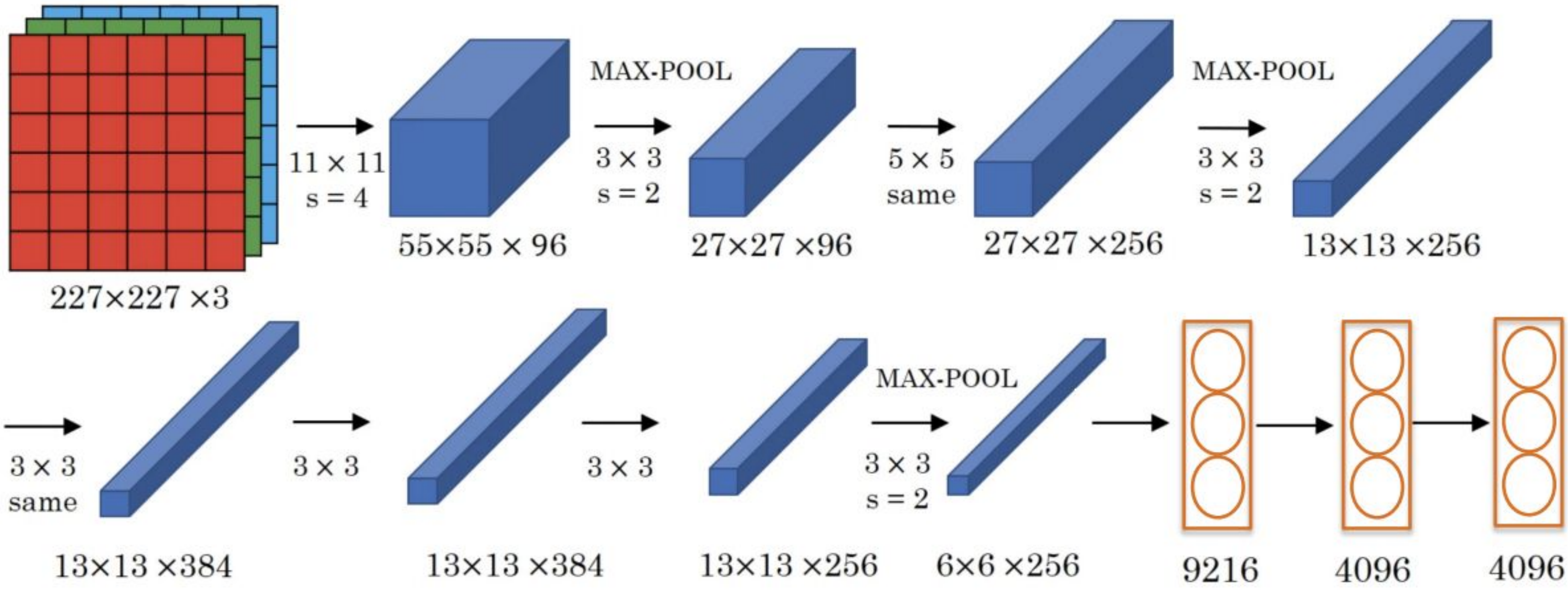
What is here?

Alex Net how does it work?



Softmax for 1000 classes

Alex Net how does it work?



60M parameters

VGGNet

(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 conv3-64	conv3-64 conv3-64	cc cc
maxpool			
conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	co co
maxpool			
conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
FC-4096			
FC-4096			
FC-1000			
soft-max			

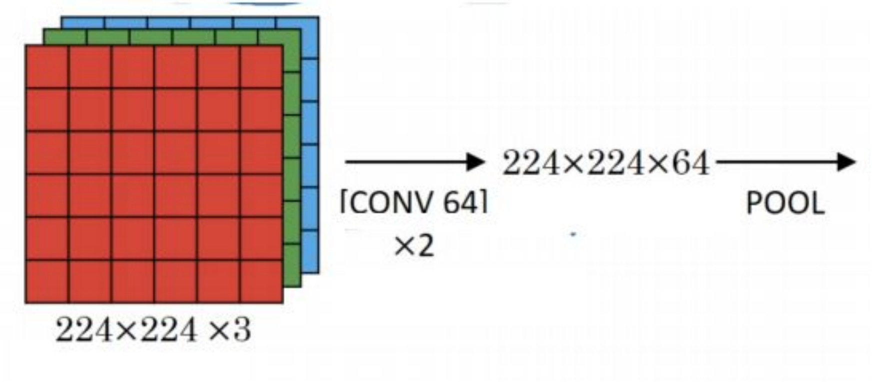
VGGNet

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 conv3-64	conv3-64 conv3-64	cc cc
maxpool			
conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	co co
maxpool			
conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
FC-4096			
FC-4096			
FC-1000			
soft-max			

7.3% top 5 error

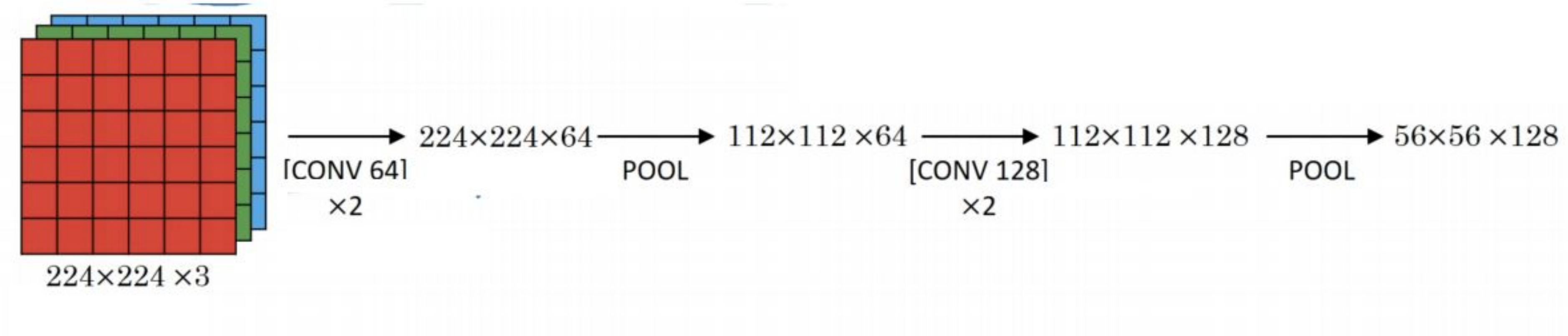
TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

VGGNet



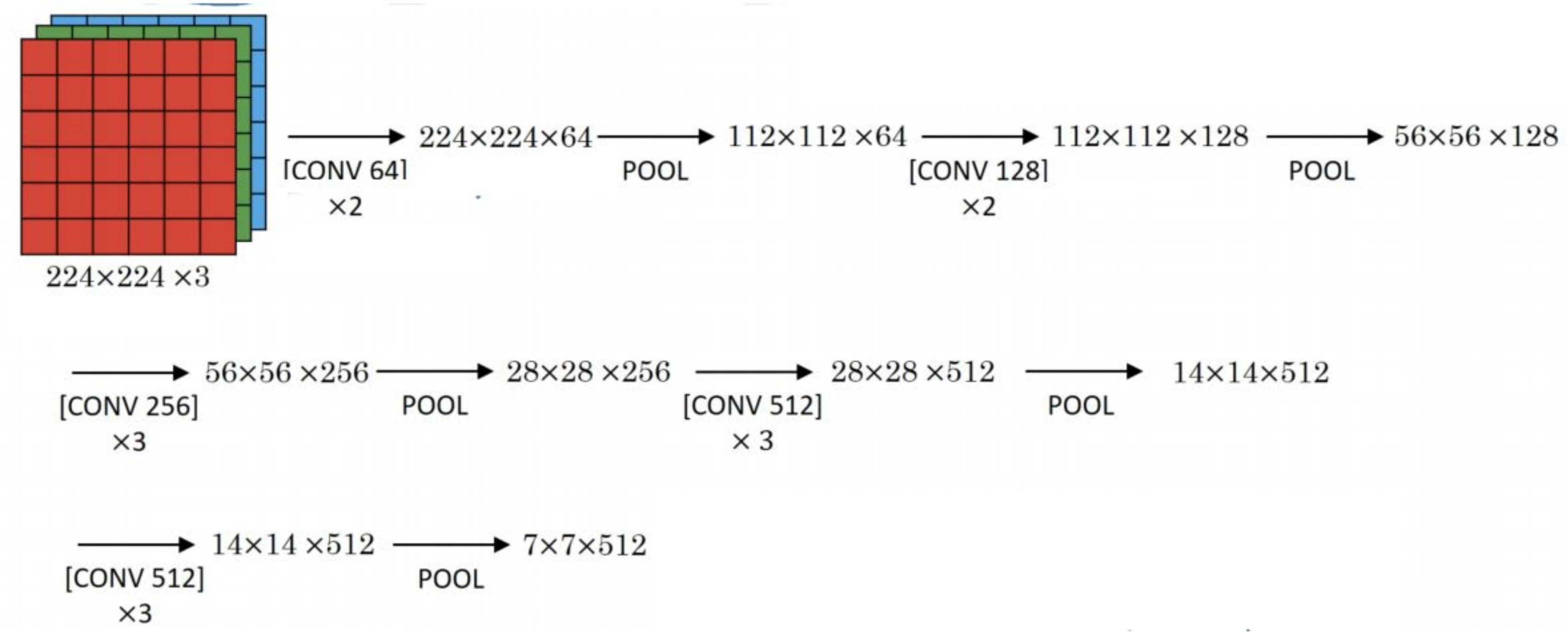
[Simonyan and Zisserman 2014]

VGGNet



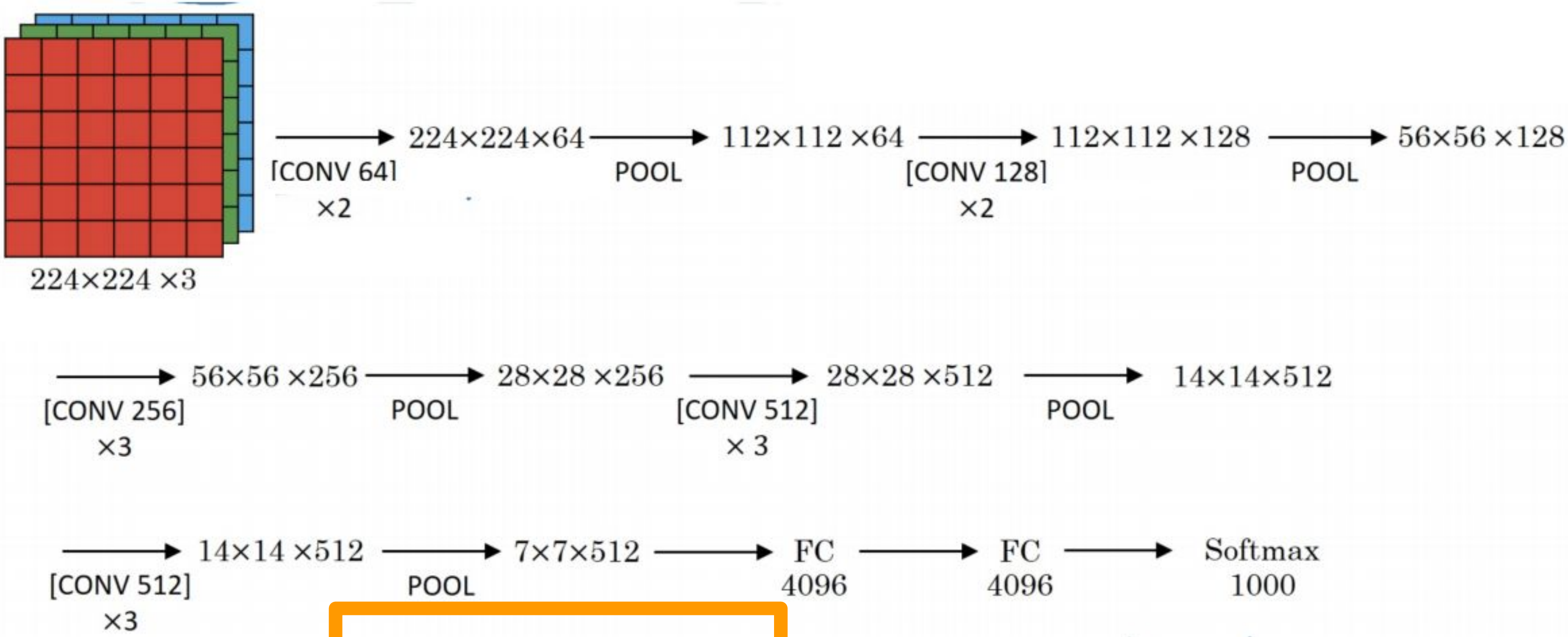
[Simonyan and Zisserman 2014]

VGGNet



[Simonyan and Zisserman 2014]

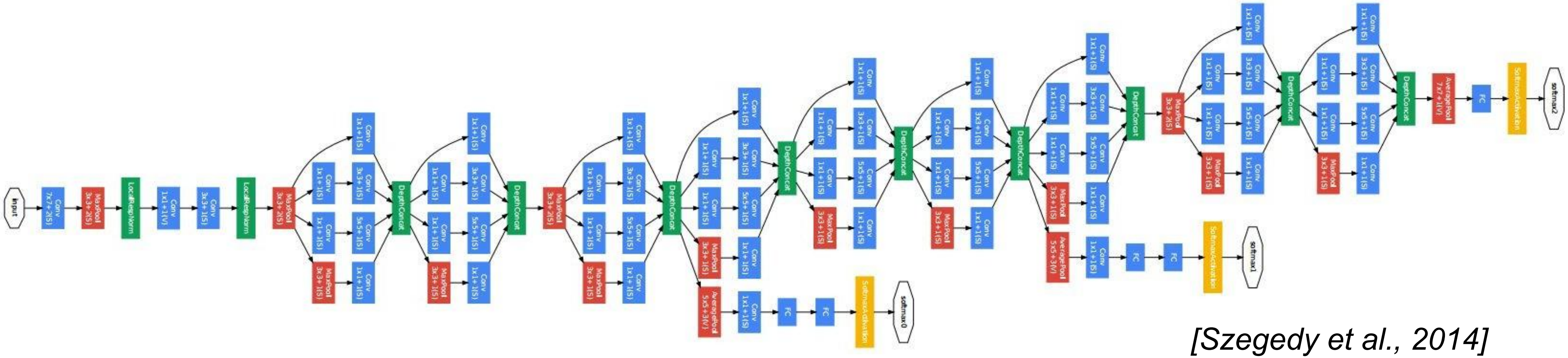
VGGNet 16



138M parameters

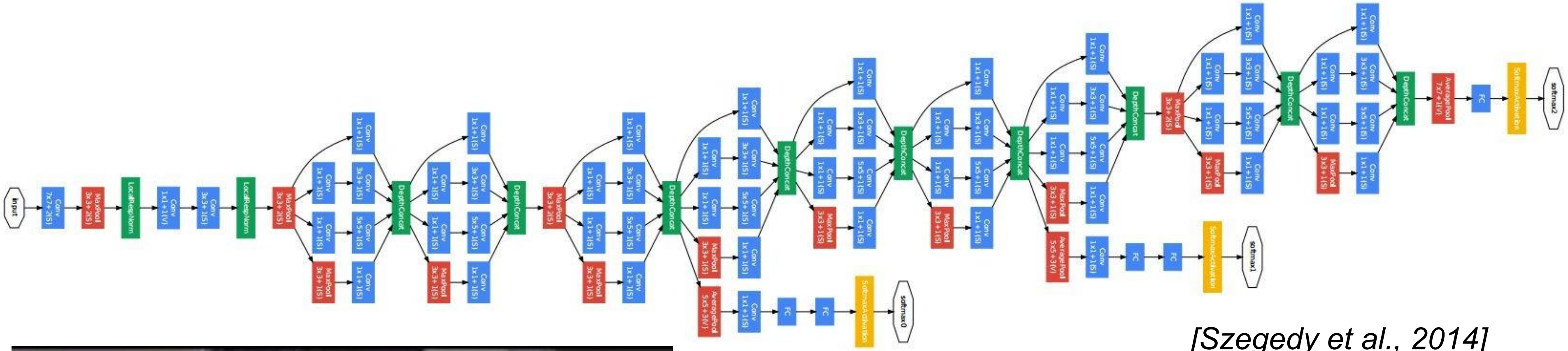
[Simonyan and Zisserman 2014]

GoogLeNet



[Szegedy et al., 2014]

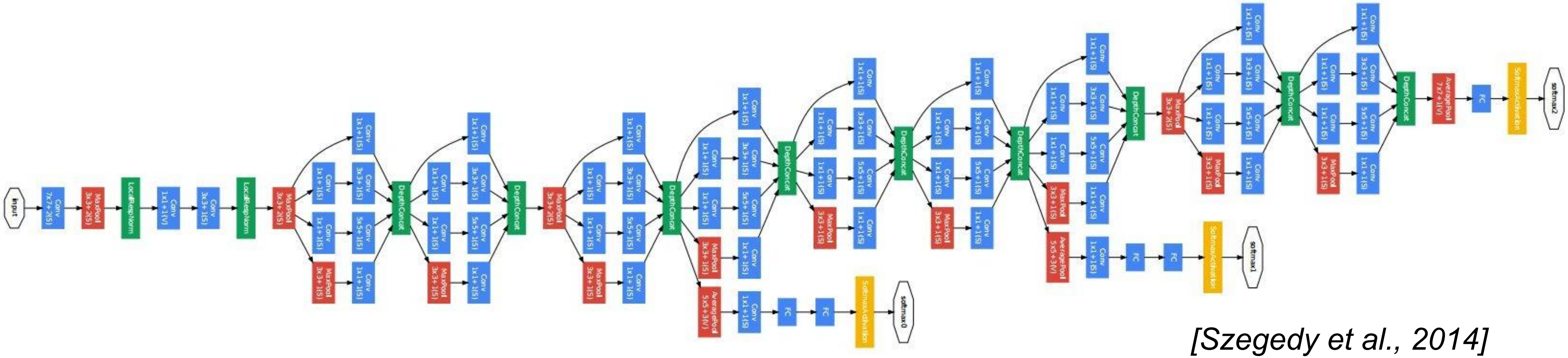
GoogLeNet



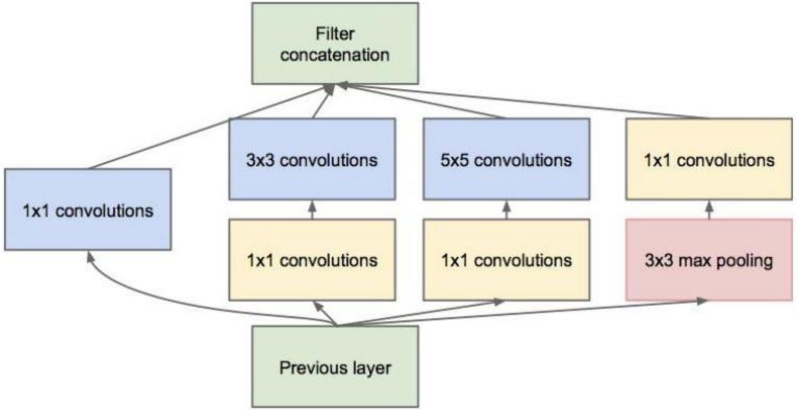
[Szegedy et al., 2014]

Inception module

GoogLeNet

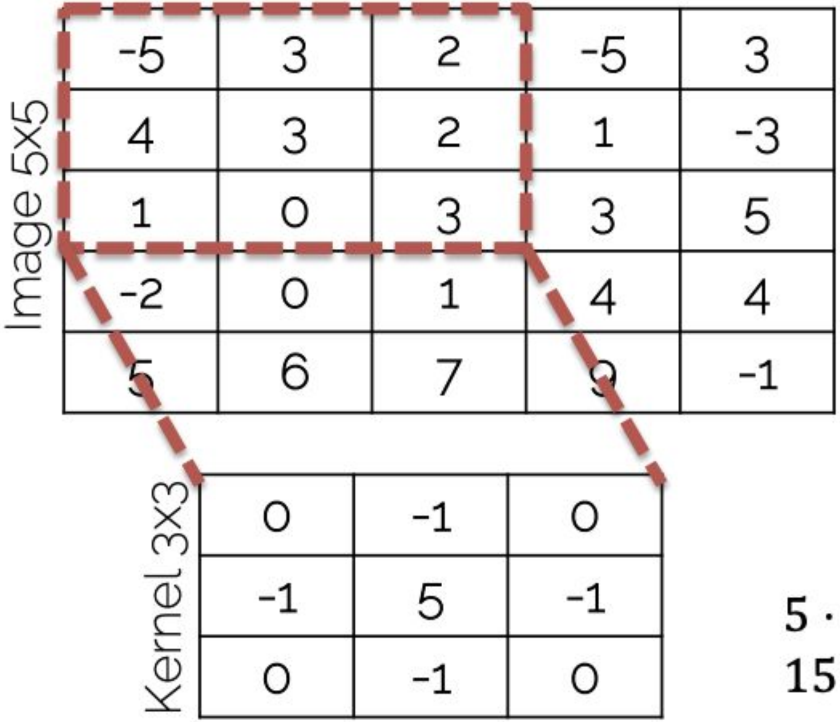


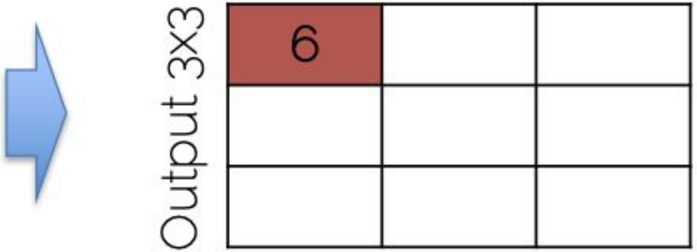
[Szegedy et al., 2014]



Inception module

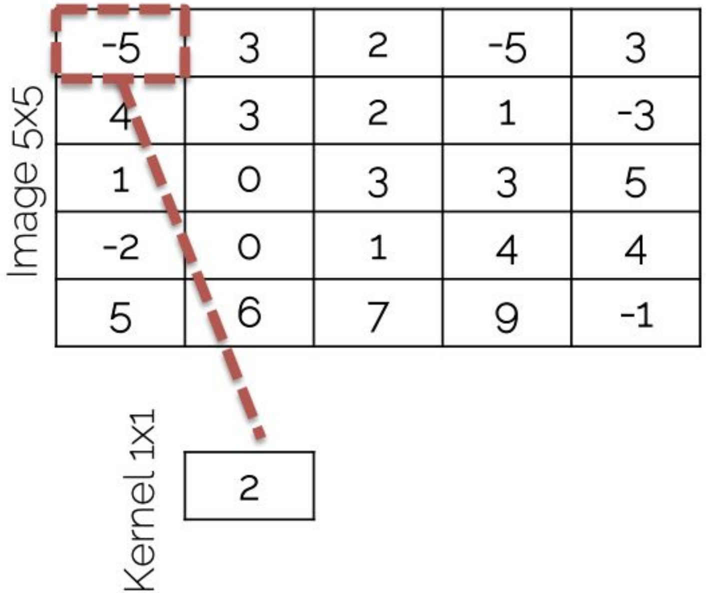
Once again: 1x1 convolutions





$$5 \cdot 3 + (-1) \cdot 3 + (-1) \cdot 2 + (-1) \cdot 0 + (-1) \cdot 4 =$$
$$15 - 9 = 6$$

Once again: 1x1 convolutions



What is the output size?

Once again: 1x1 convolutions

Image 5X5

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

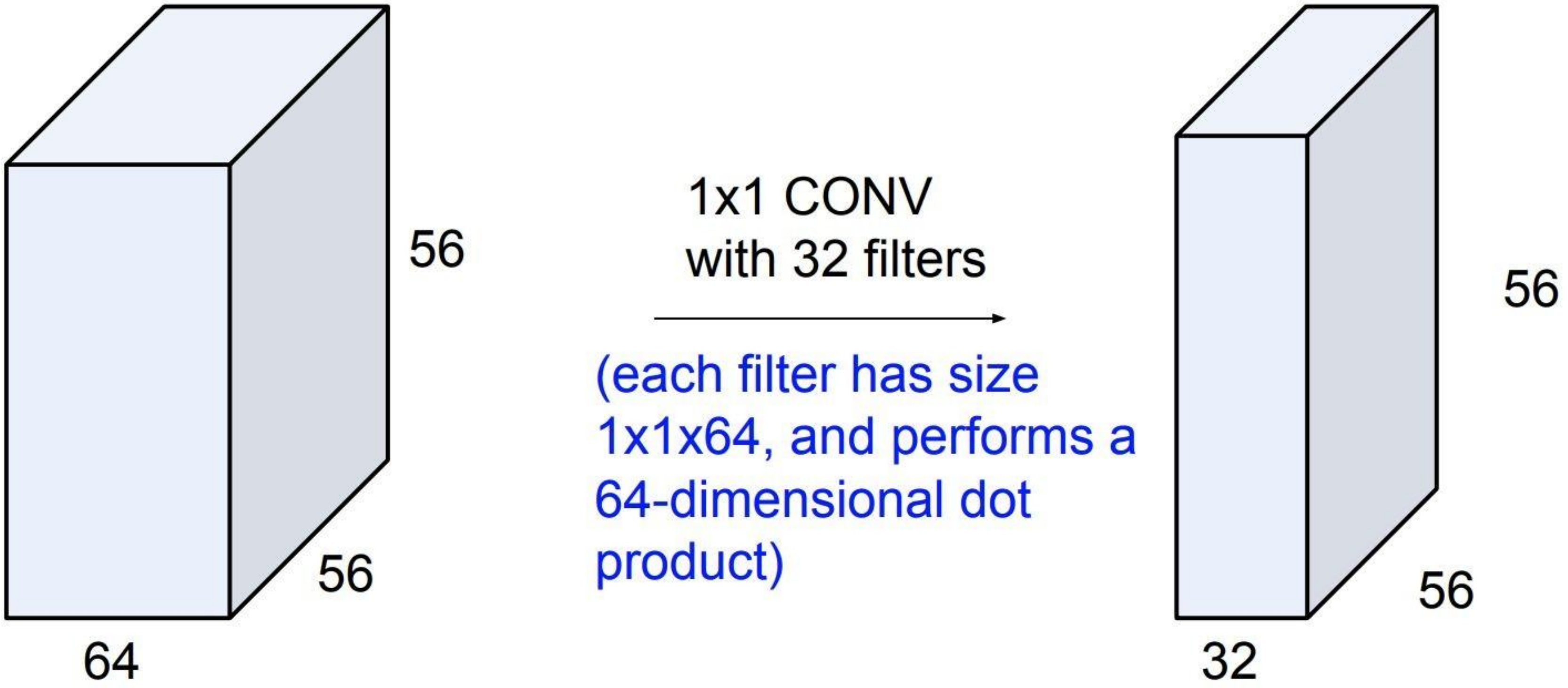
-10	6	4	-10	6
8	6	4	2	-6
2	0	6	6	10
-4	0	2	8	8
10	12	14	18	-2

Kernel 1X1

2

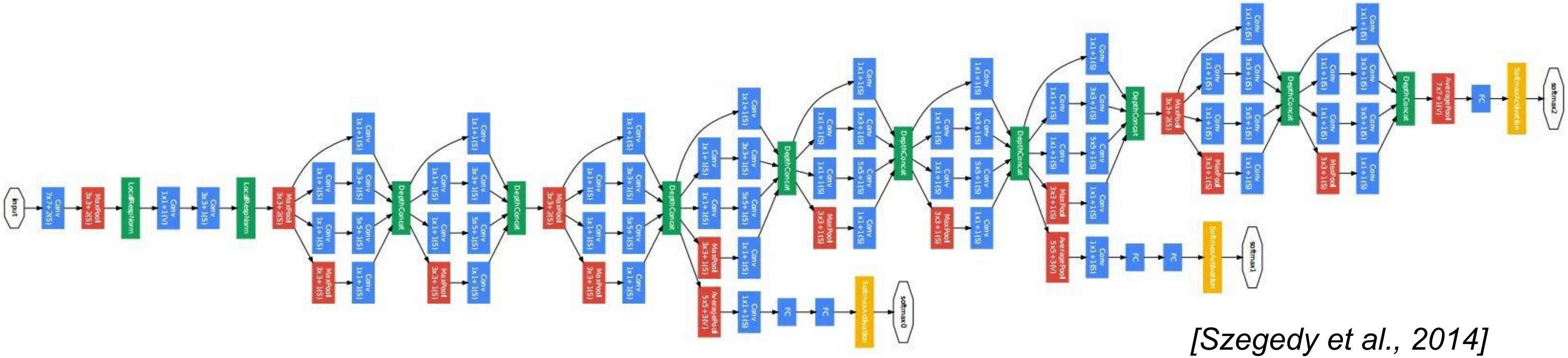
$$-1 * 2 = -2$$

Once again: 1x1 convolutions

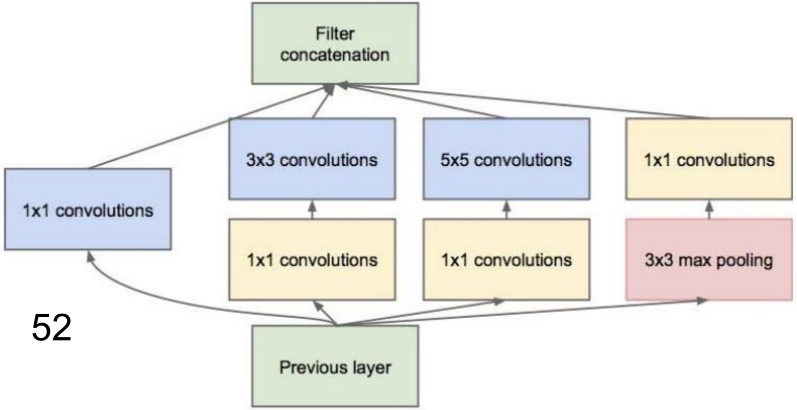


source: http://cs231n.stanford.edu/slides/2016/winter1516_lecture7.pdf

GoogLeNet



[Szegedy et al., 2014]

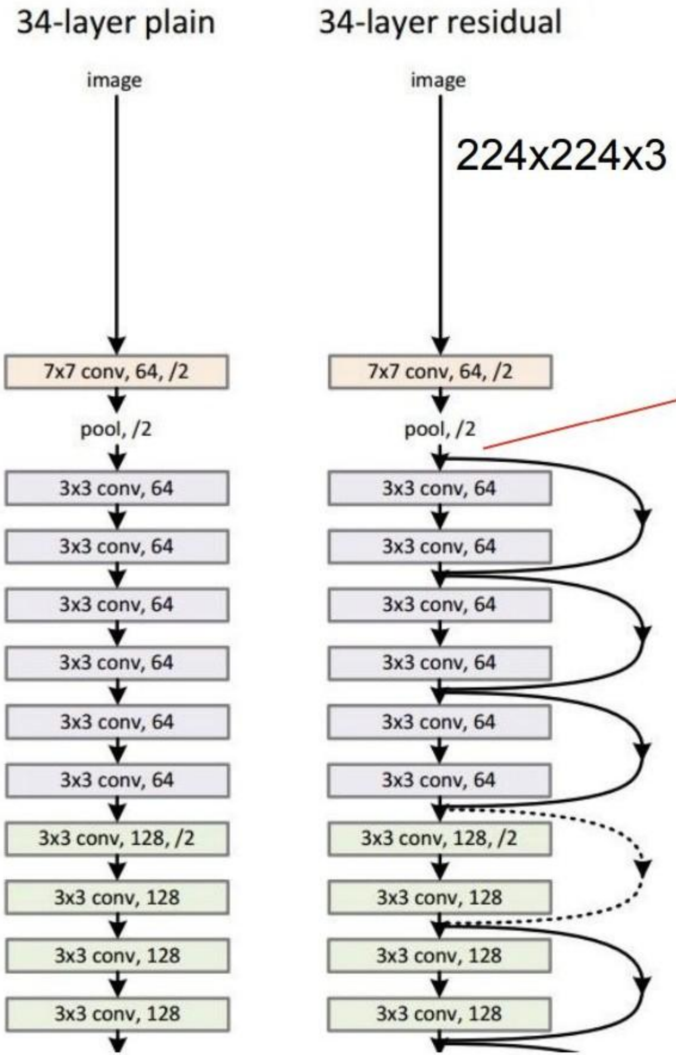


Inception module

ILSVRC 2014 winner (6.7% top 5 error)

ResNet

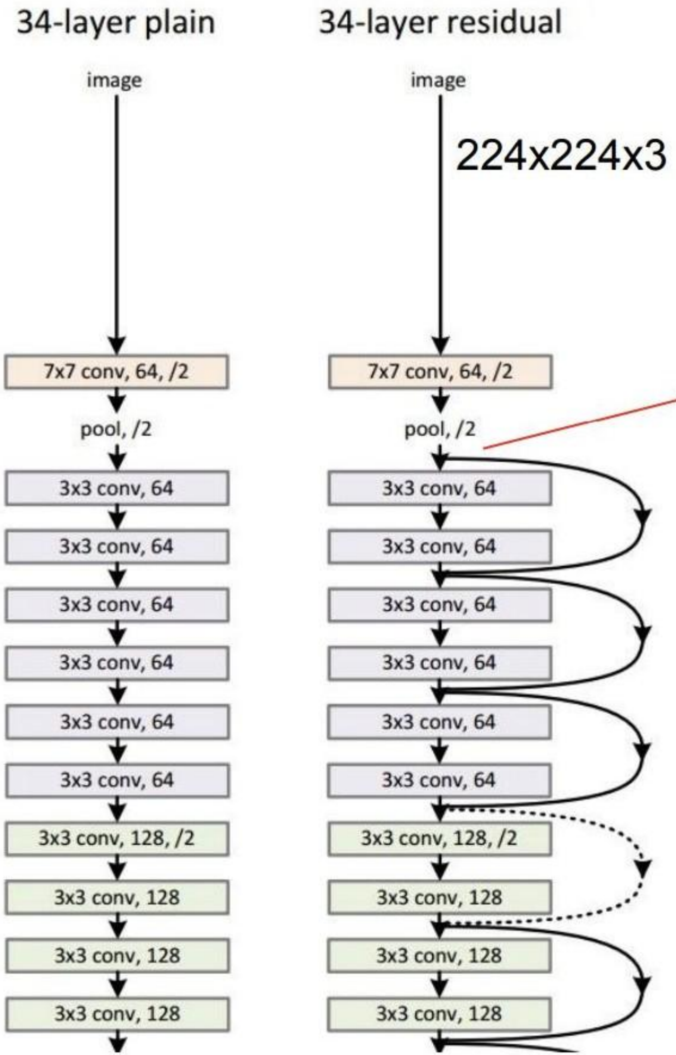
[He et al., 2015]



spatial dimension
only 56x56!

ILSVRC 2015 winner (3.6% top 5 error)

ResNet



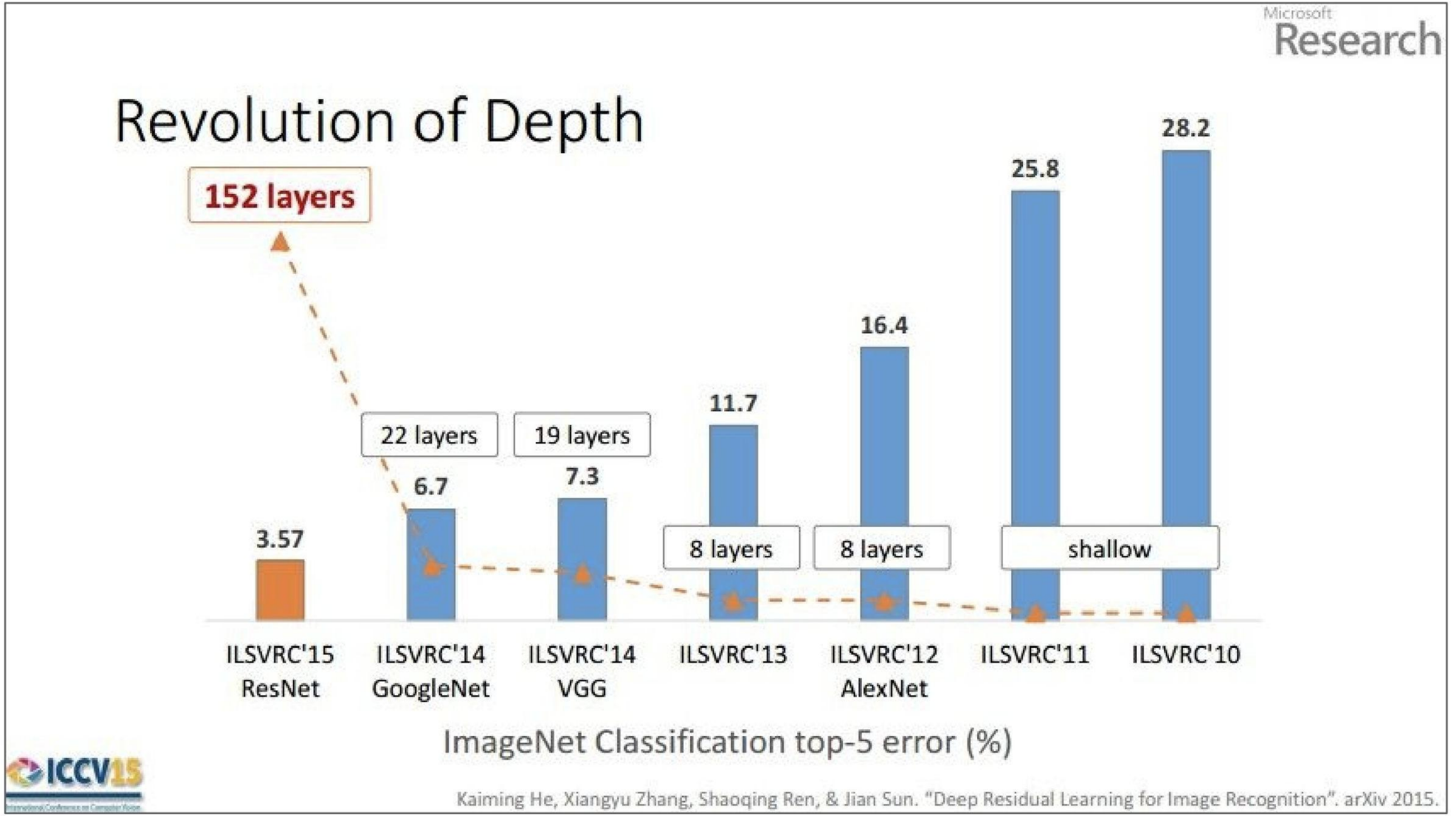
[He et al., 2015]

- Batch Normalization after every CONV layer

spatial dimension only 56x56!

- Xavier/2 initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- ILSVRC 2015 winner (3.6% top 5 error)
- Weight decay of 1e-5

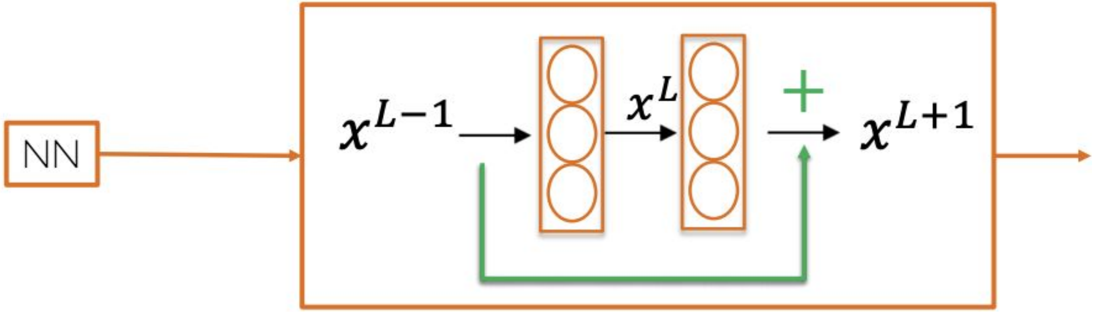
ResNet



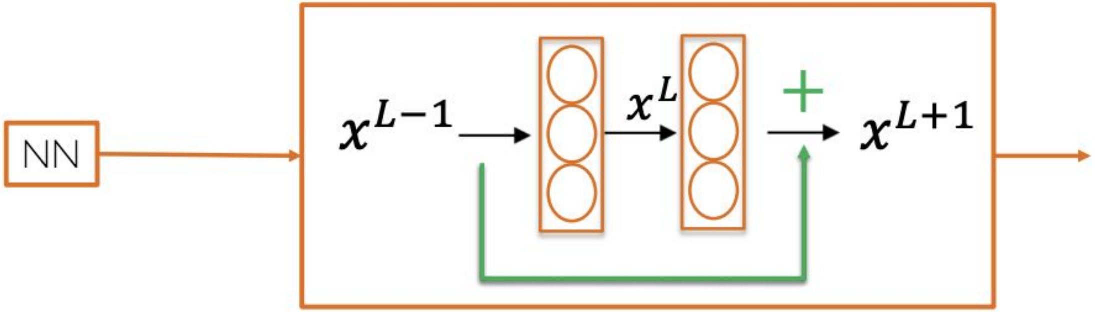
(slide from Kaiming He's recent presentation)

ResNet 152 **60M parameters**

Why do ResNets Work?



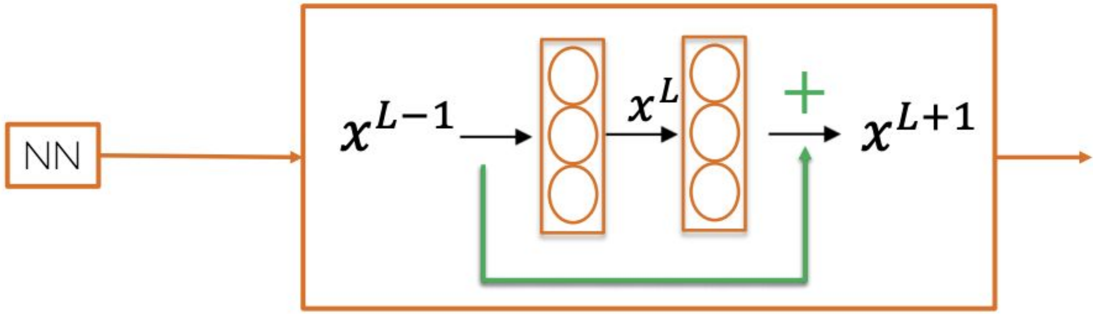
Why do ResNets Work?



$$x^{L+1} = f(\underbrace{W^{L+1}}_{\sim\text{zero}} x^{L} + \underbrace{b^{L+1}}_{\sim\text{zero}} + x^{L-1})$$

$$x^{L+1} = f(x^{L-1})$$

Why do ResNets Work?



$$x^{L+1} = f(W^{L+1}x^{L} + b^{L+1} + x^{L-1})$$

~zero ~zero

$$x^{L+1} = f(x^{L-1})$$

- The identity is easy for the residual block to learn
- Guaranteed it will not hurt performance, can only improve

Summary

- ConvNets stack convolutional, pooling and dense layers
- Trend towards smaller filters and deeper architectures
- 1x1 convolutions are meaningful
- Humanity is already beaten on ImageNet.

Links

- Notes on vector and matrix derivatives: http://cs231n.stanford.edu/vecDerivs.pdf
- Stanford notes on backpropagation: http://cs231n.github.io/optimization-2/
- Stanford notes on different activation functions (and just intuition): http://cs231n.github.io/neural-networks-1/
- Great post on Medium by Andrej Karpathy: https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b
- CS231n notes on data preparation (batch normalization over there): http://cs231n.github.io/neural-networks-2/
- CS231n notes on gradient methods: http://cs231n.github.io/neural-networks-3/
- Original paper introducing Batch Normalization: https://arxiv.org/pdf/1502.03167.pdf
- What Every Computer Scientist Should Know About Floating-Point Arithmetic: https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html
- Convolutional Neural Networks: Architectures, Convolution / Pooling Layers: http://cs231n.github.io/convolutional-networks/
- Understanding and Visualizing Convolutional Neural Networks: http://cs231n.github.io/understanding-cnn/
- CS231n notes on data preparation: http://cs231n.github.io/neural-networks-2/