Convolutional Networks

Machine Learning for Image Analysis

-Vedika Agarwal

Overview

1. CNN and some useful applications
    a. Motivation
    b. Other applications
2. Biological Connection
    a. Idea behind it
3. Structure of a CNN
    a. Convolution- the C in CNN
    b. Non-Linearity
    c. Pooling
    d. Fully-Connected Layer
4. Example architectures
5. Beyond Images
6. References

Why Convolutional Networks?

- Wins every computer vision challenge (classification, segmentation, etc.)
- Can be applied in various domains (speech recognition, game prediction, Natural Language Processing etc.)
- Really good performance- Beats human accuracy in some cases

A visual representation

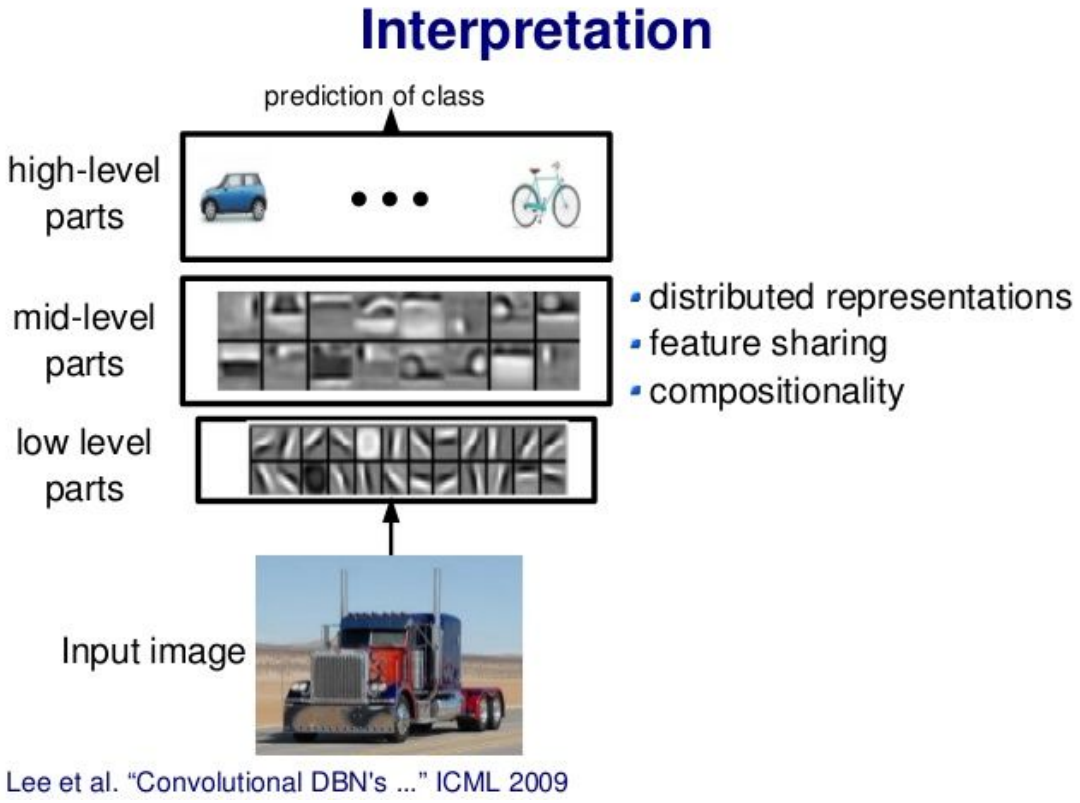
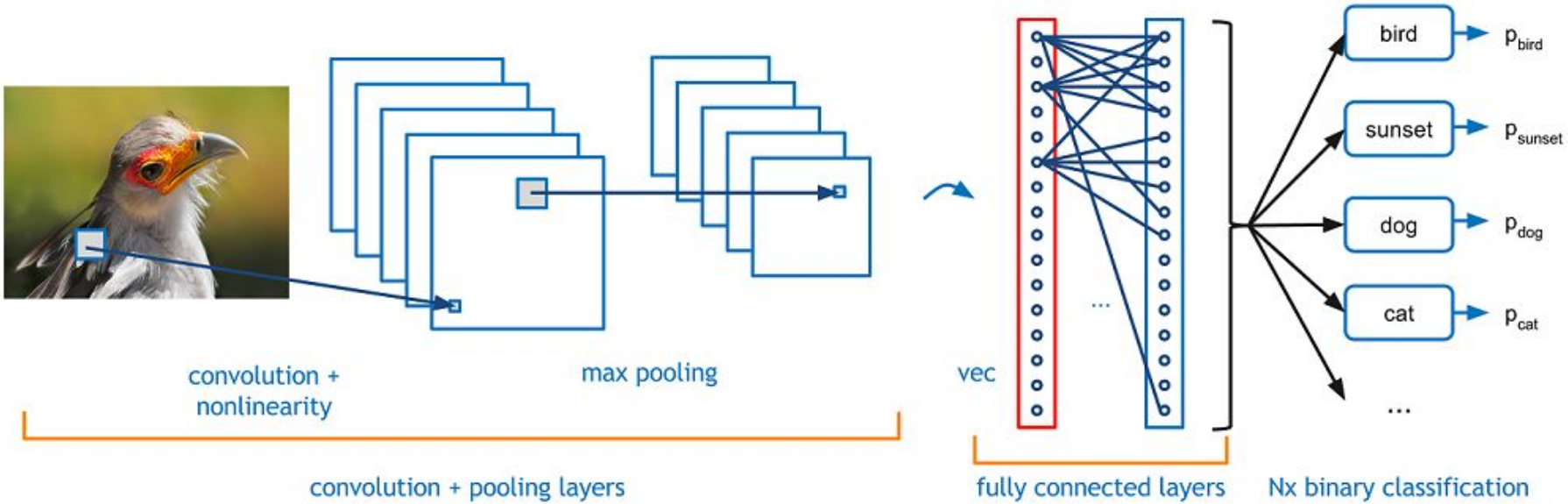
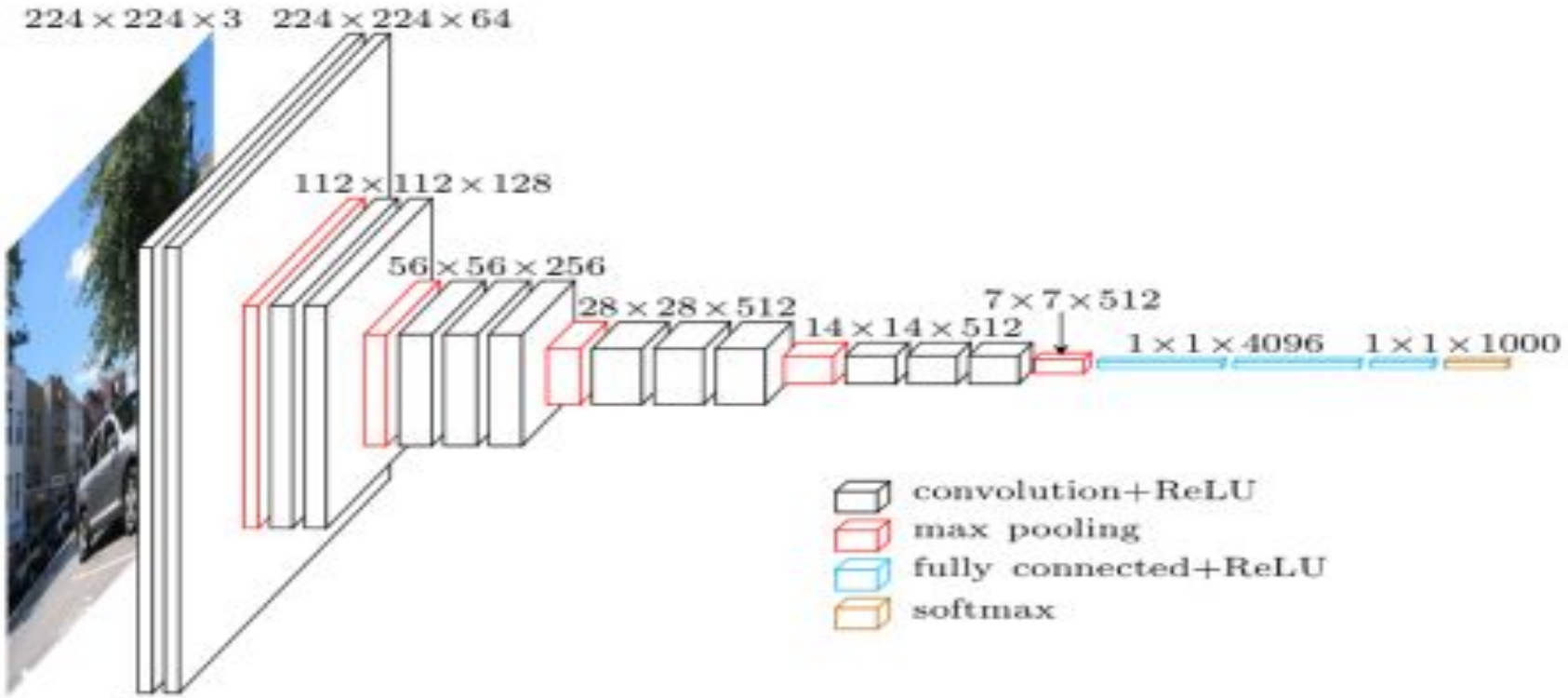


Image Classification



Example- architecture VGGnet



Applications- Visual Recognition

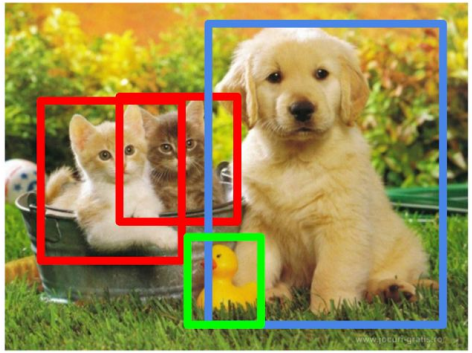
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation

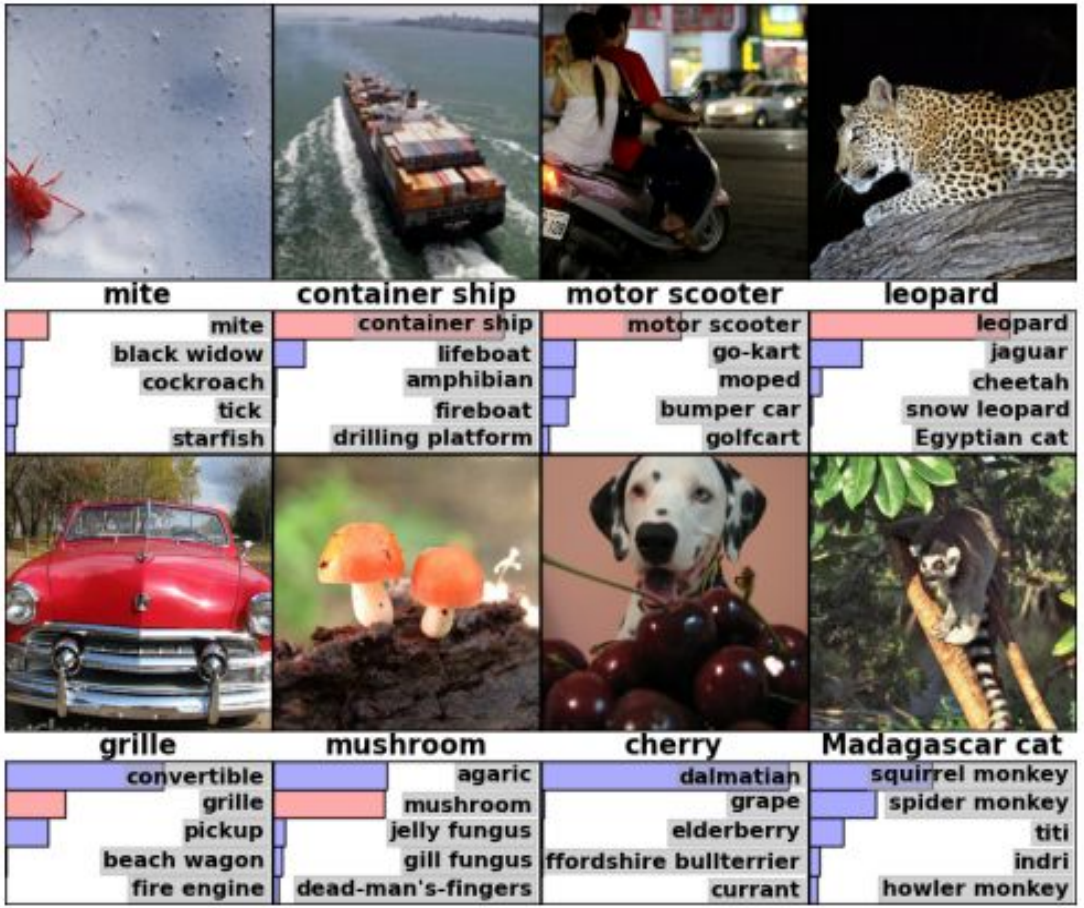
CAT, DOG, DUCK

Single object

Multiple objects

Classification- What?

- Single object, single class
- 2012, **Deep convolutional neural network**, Alex Krizhevsky et al.
- ImageNet- 1 place, top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.



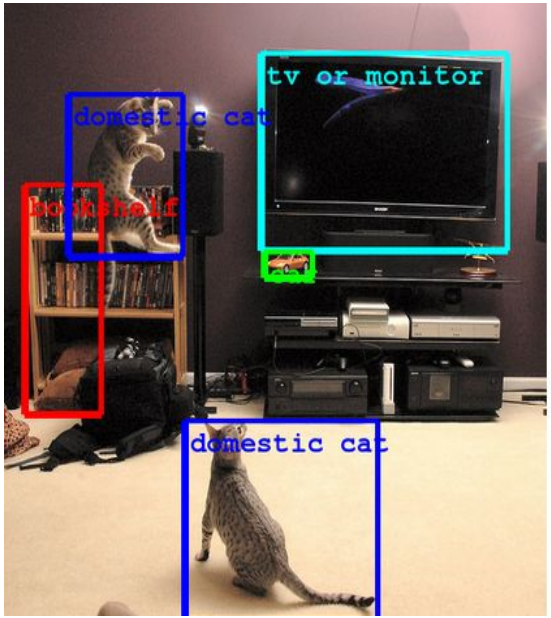
Source: Krizhevsky, A., Sutskever I., and Hinton, G., "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems, 2012.*

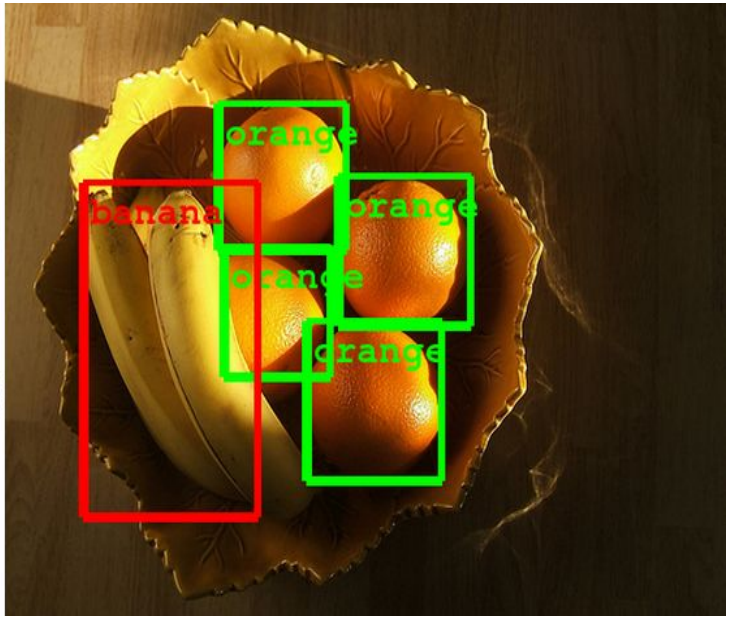
Classification and Localization- What and Where?

- Multiple objects, multiple class
- 2014, **VGG net** by Oxford group

ImageNet challenge:

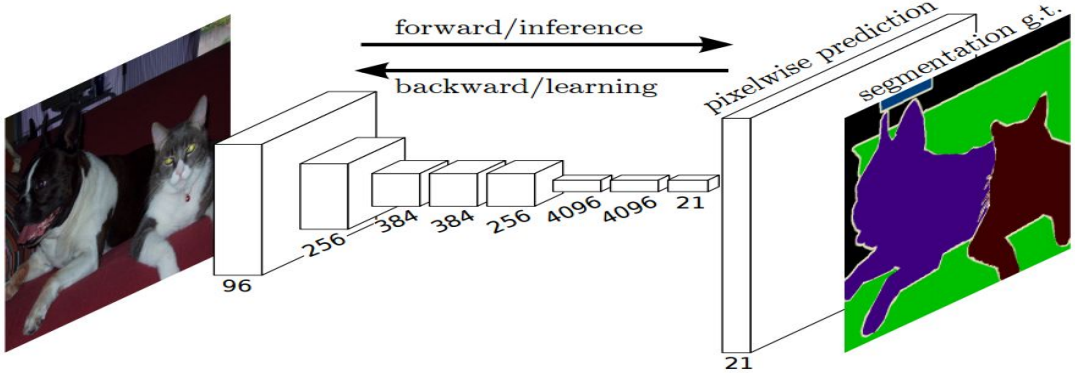
- First in localization
- Second in classification





Source: *K Simonyan and A. Zisserman*, “Very deep convolutional networks for large-scale image recognition,” in ICLR, 2014

Semantic Segmentation- What, Where and Extent?



Given an image- you want to predict the class label for each pixel in the image i.e **pixel-wise semantic labelling**

Source: Fully Convolutional Network, Berkeley group, CVPR 2015

Semantic Segmentation- What, Where and Extent?

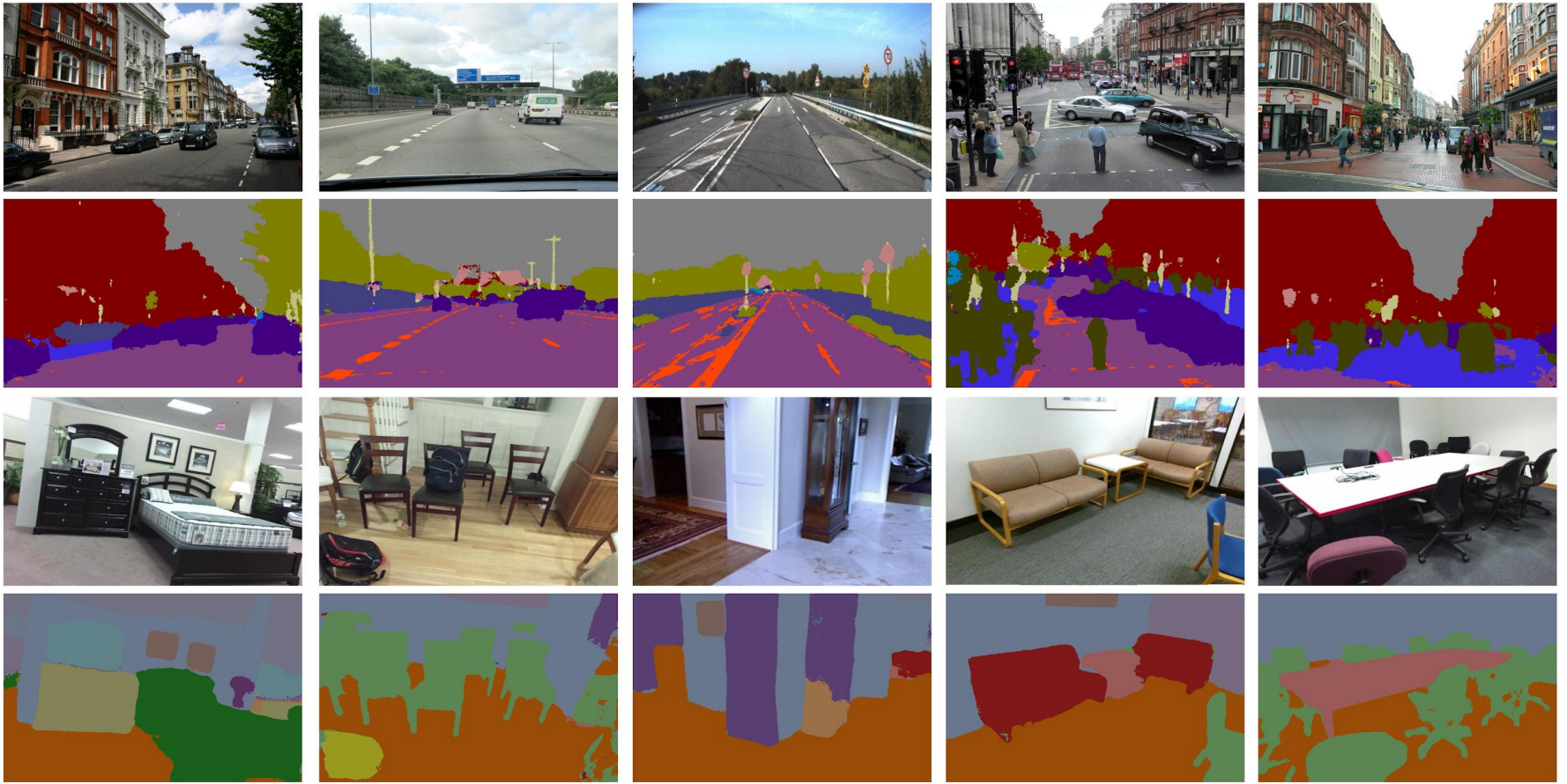
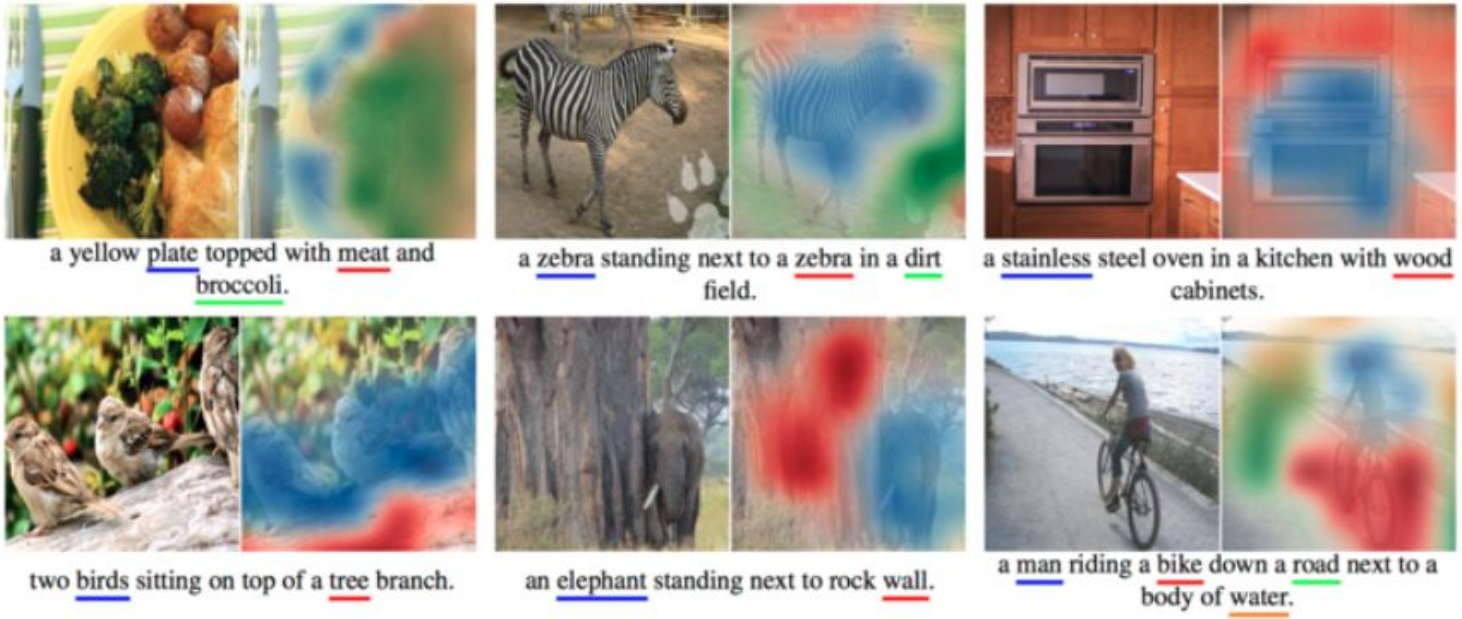


Image Captioning & Visual Attention



Source: *Jiasen Lu^, Caiming Xiong^, Devi Parikh, and Richard Socher. 2016.*
Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning.

Style Transfer

Mona Lisa restyled by Picasso, van Gogh, and Monet

Source: http://genekogan.com/works/style-transfer/

Applications



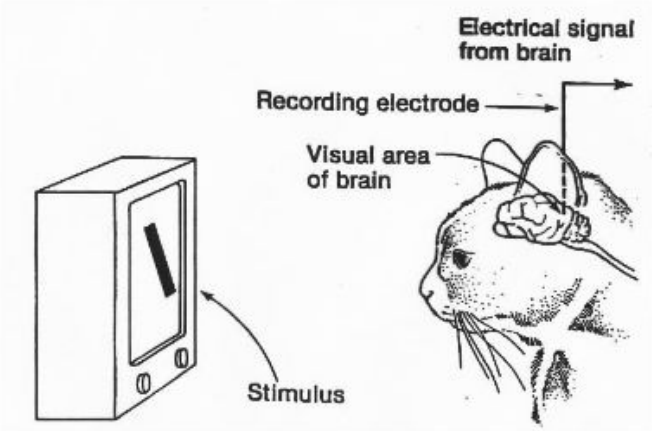
- Autonomous vehicles
- Indoor segmentation for Augmented reality systems
- Learn geometry in the image
- Road scene analysis
- Medical Imaging

Overview

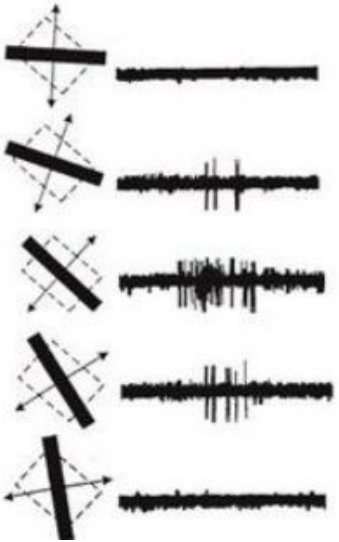
1. CNN and some useful applications
    a. Motivation
    b. Other applications
2. Biological Connection
    a. Idea behind it
3. Structure of a CNN
    a. Convolution- the C in CNN
    b. Non-Linearity
    c. Pooling
    d. Fully-Connected Layer
4. Example architectures
5. Beyond Images
6. References

Biological Connection

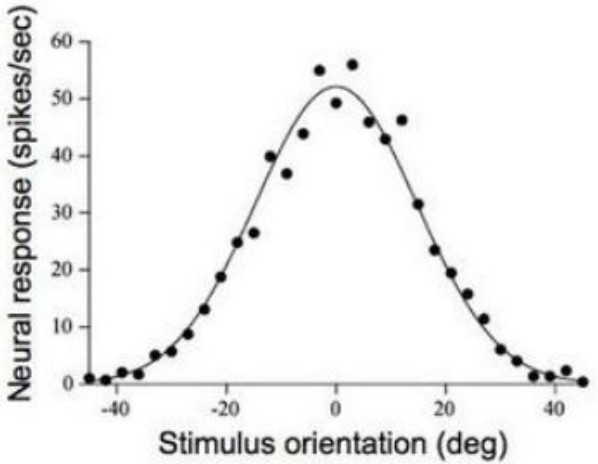
- **Hubel and Wiesel , 1959**



- Some neurons responded only in presence of some edges in certain orientation



- Idea of specialized components having specific tasks is the basis behind CNNs



What is Convolutional Network?

Convolutional network is simply a neural network thats use convolution in place of general matrix multiplication in at least one of their layers

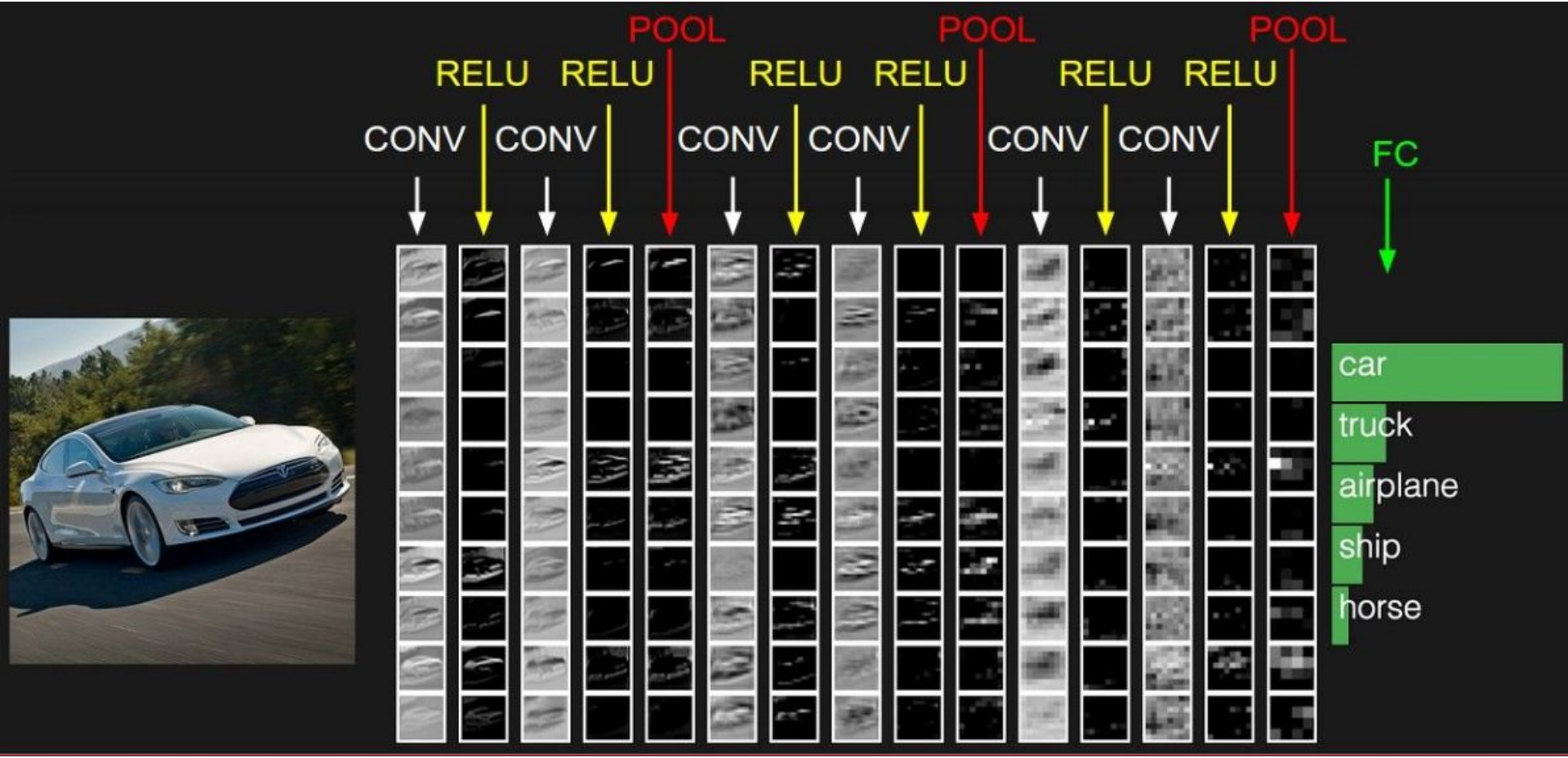
Overview

1. CNN and some useful applications
    a. Motivation
    b. Other applications
2. Biological Connection
    a. Idea behind it
3. Structure of a CNN
    a. Convolution- the C in CNN
    b. Non-Linearity
    c. Pooling
    d. Fully-Connected Layer
4. Example architectures
5. Beyond Images
6. References

Structure

- Convolutional Layer
- Non-linearity (ReLU)
- Pooling
- Fully Connected Layer

preview


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Overview

1. CNN and some useful applications
    a. Motivation
    b. Other applications
2. Biological Connection
    a. Idea behind it
3. Structure of a CNN
    a. Convolution- the C in CNN
    b. Non-Linearity
    c. Pooling
    d. Fully-Connected Layer
4. Example architectures
5. Beyond Images
6. References

Convolution

- Primary purpose - to extract features from the input image
- Discrete-convolution in 1D:

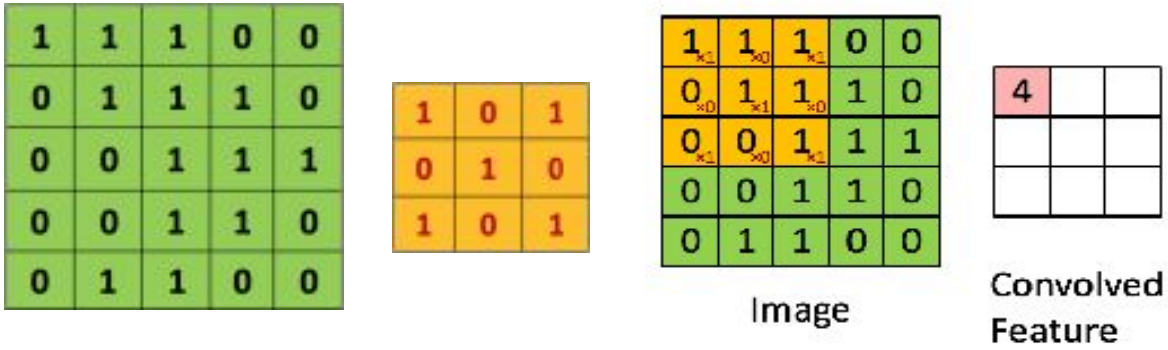
$$(\boldsymbol{f} * \boldsymbol{w})_i \ := \sum_{k=-\infty}^{\infty} f_{i-k}\, w_k$$

- Discrete-convolution in 2D:

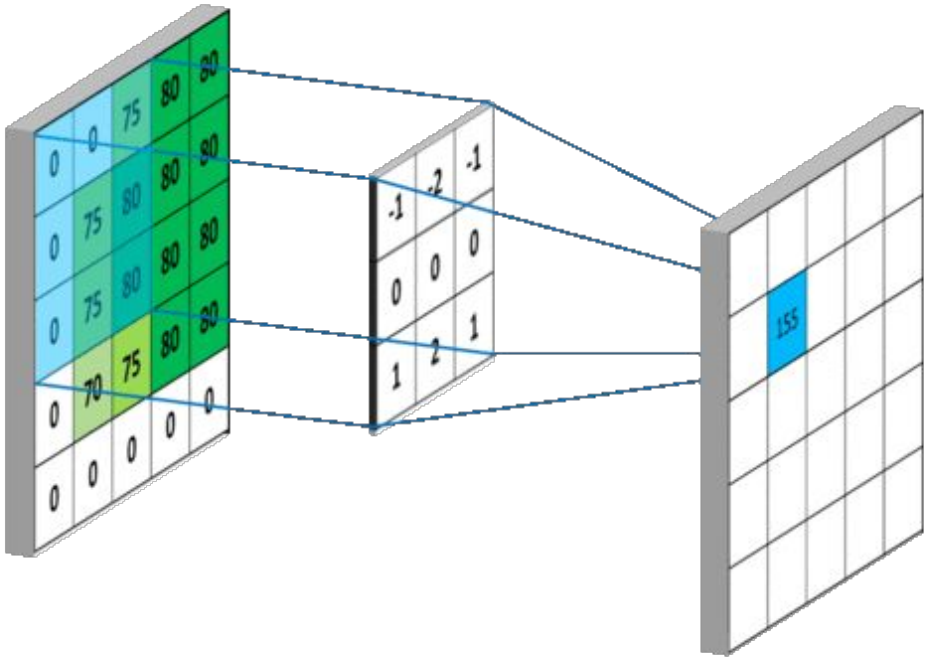
$$(\boldsymbol{f} * \boldsymbol{w})_{i,j} \ := \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} f_{i-k,\, j-\ell}\, w_{k,\ell}$$

Convolution

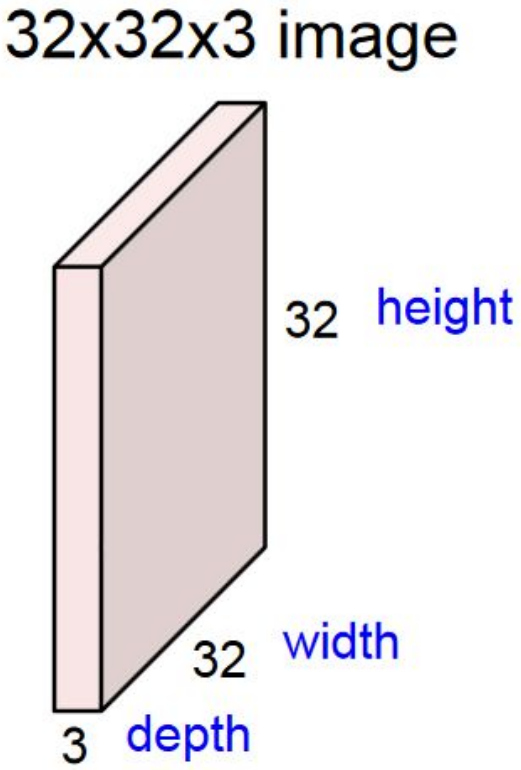
Convolution of the 5 x 5 image and the 3 x 3 matrix can be computed as shown below:



Convolution

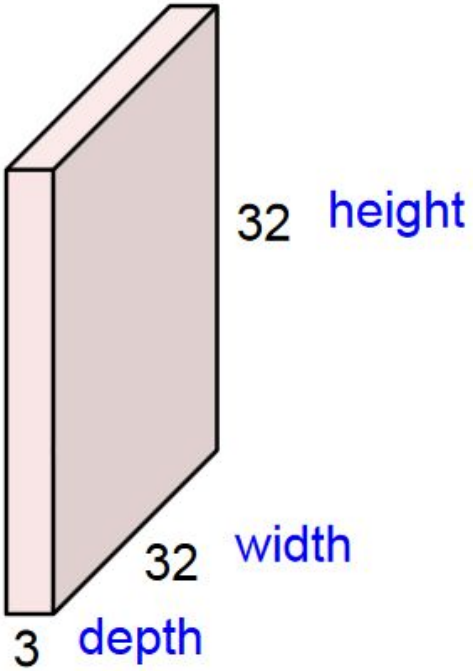


Convolution Layer

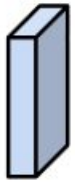


Convolution Layer

32x32x3 image

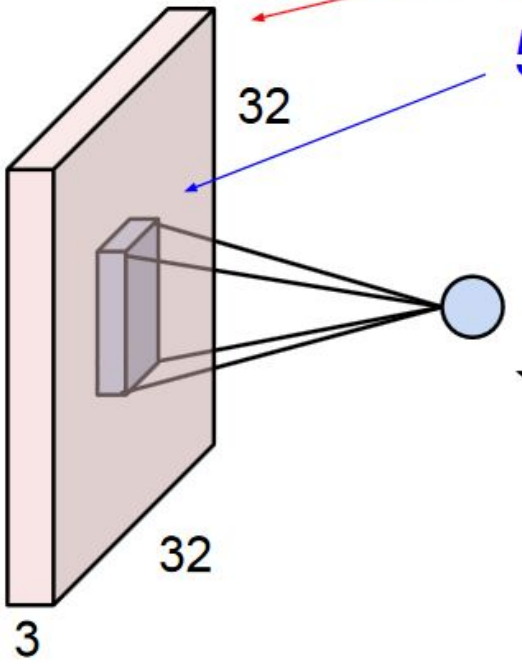


5x5x3 filter



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

Convolution Layer

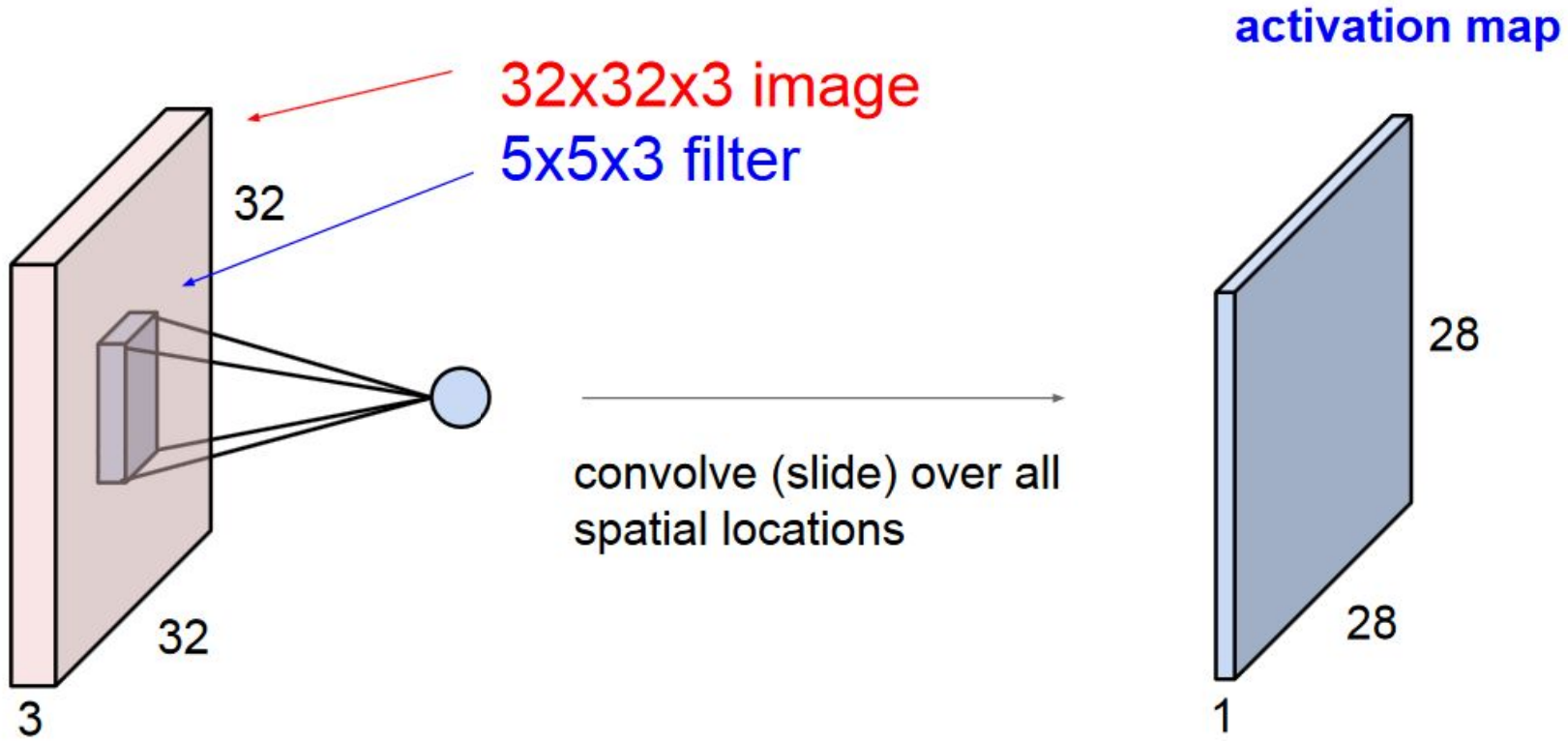


32x32x3 image

5x5x3 filter w

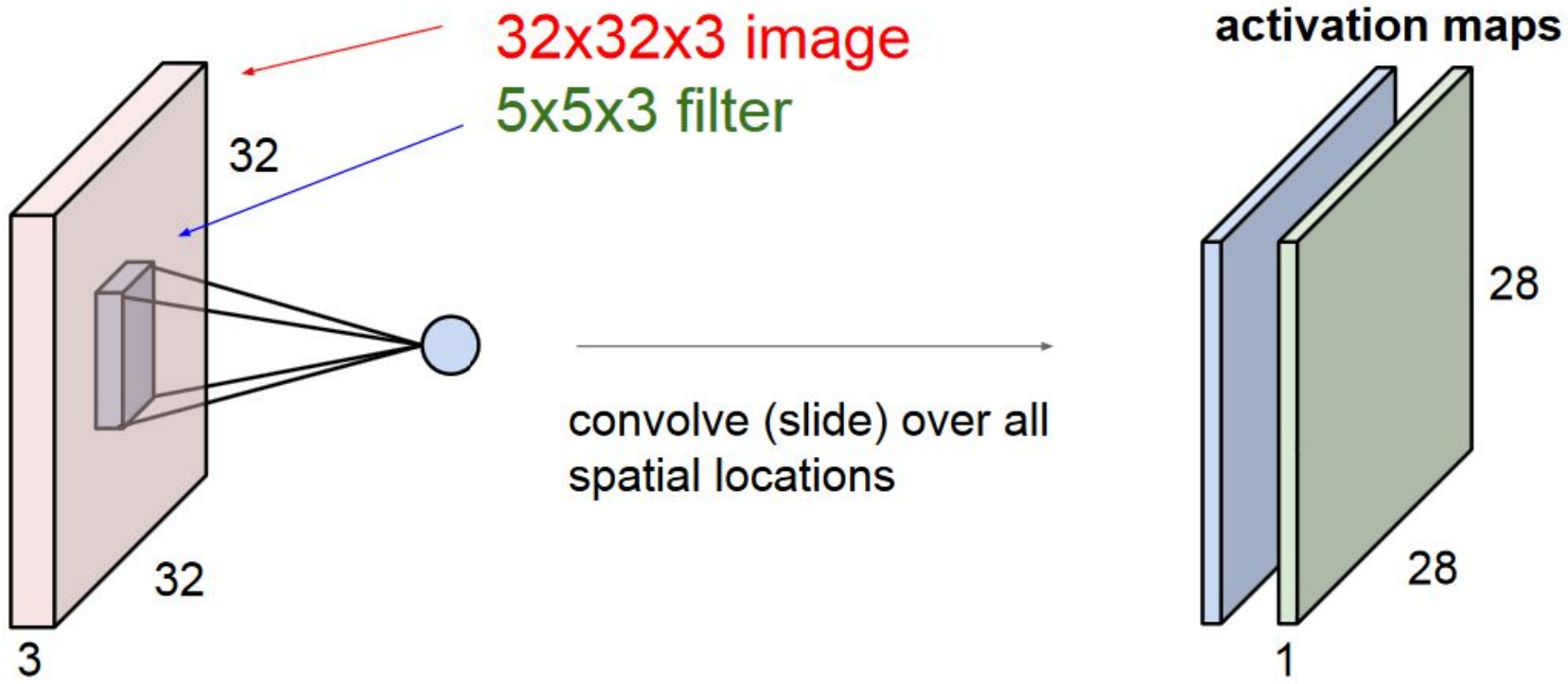
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image

Convolution Layer

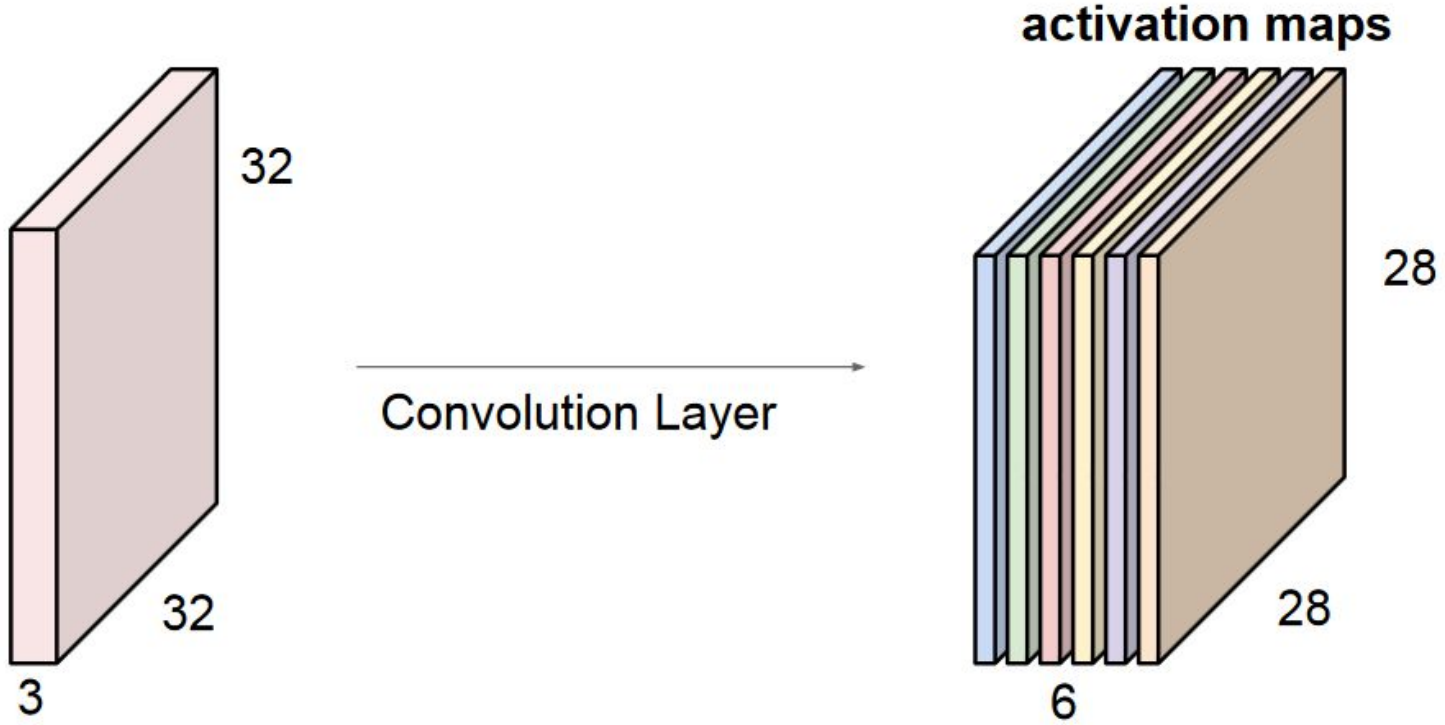


Convolution Layer

consider a second, green filter



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!

Different filters, Different feature maps

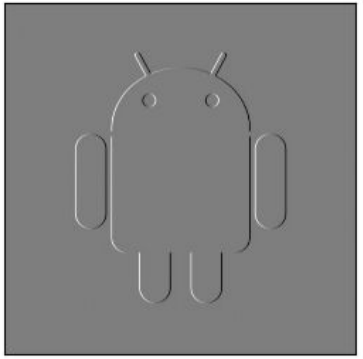
-1	-2	-1
0	0	0
1	2	1

Finds horizontals

-1	0	1
-2	0	2
-1	0	1

Finds verticals

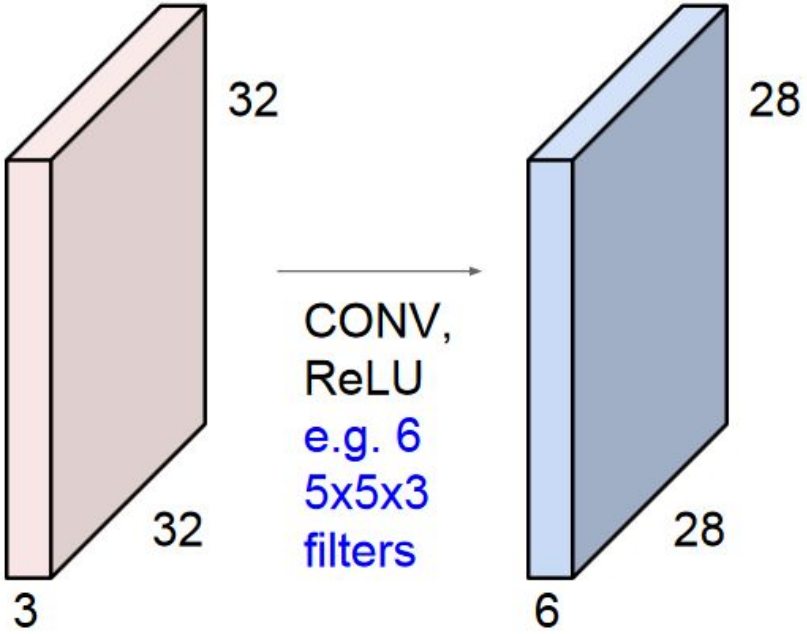
Horizontal Sobel

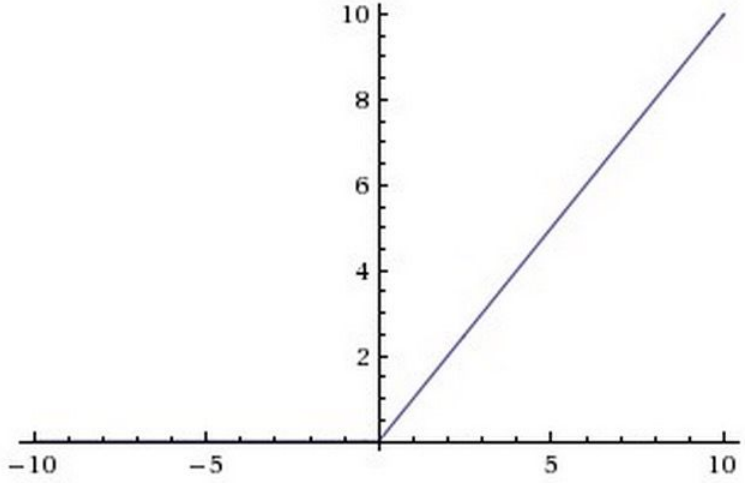


Vertical Sobel

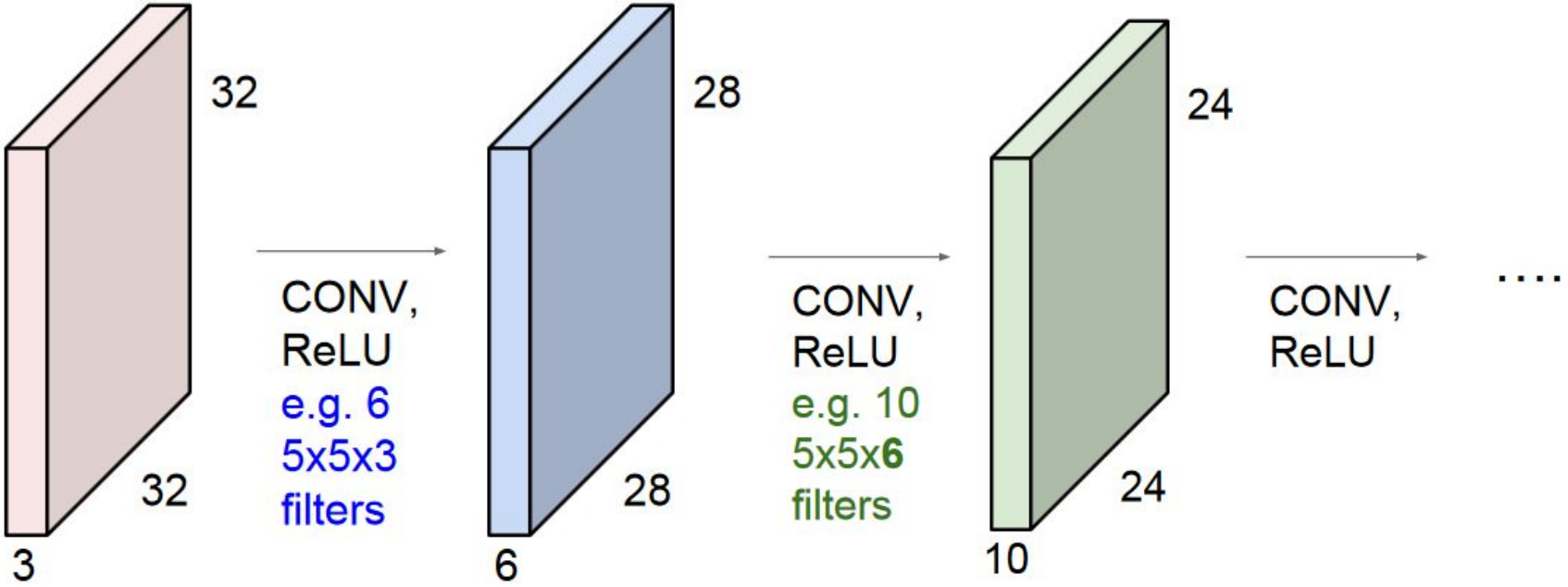
Input

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

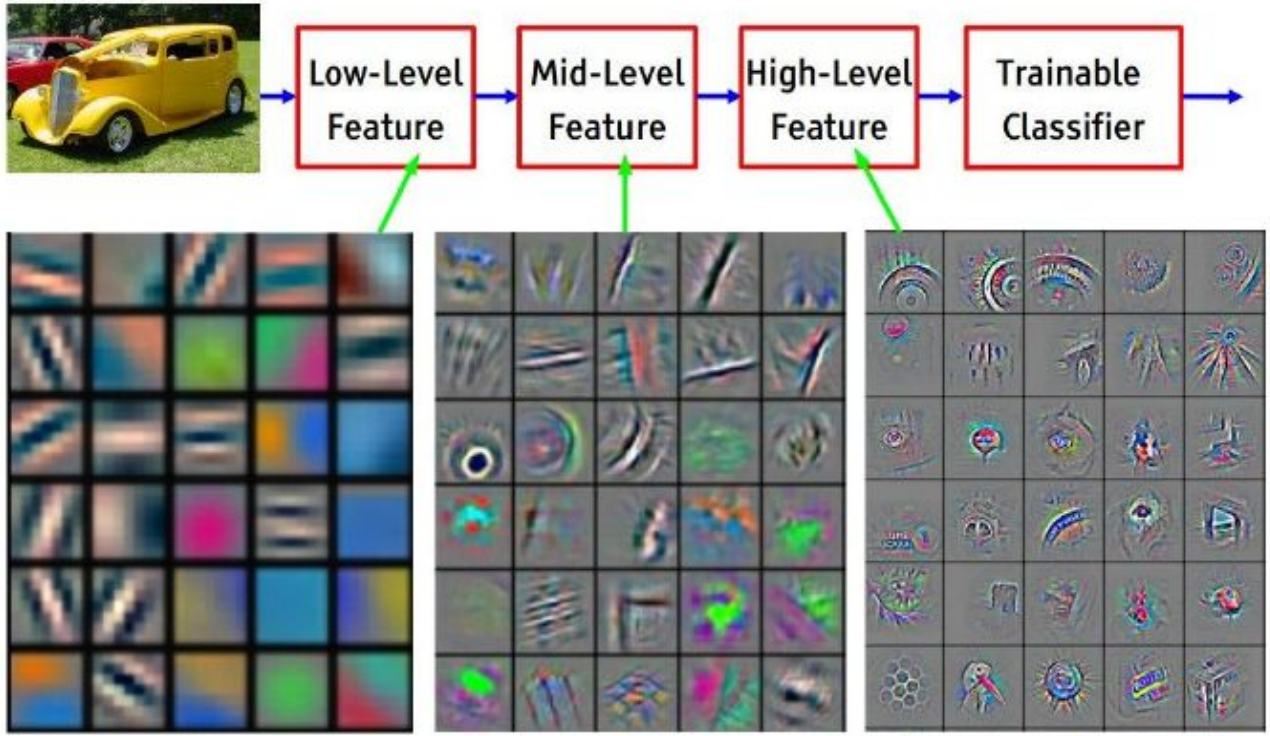




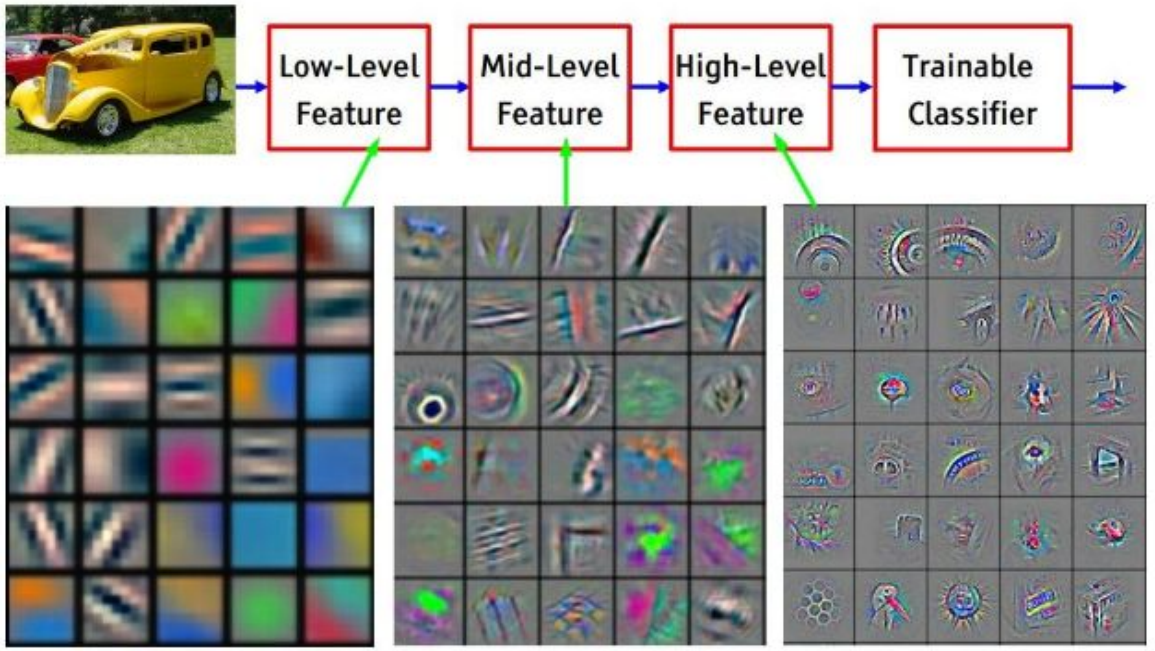
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



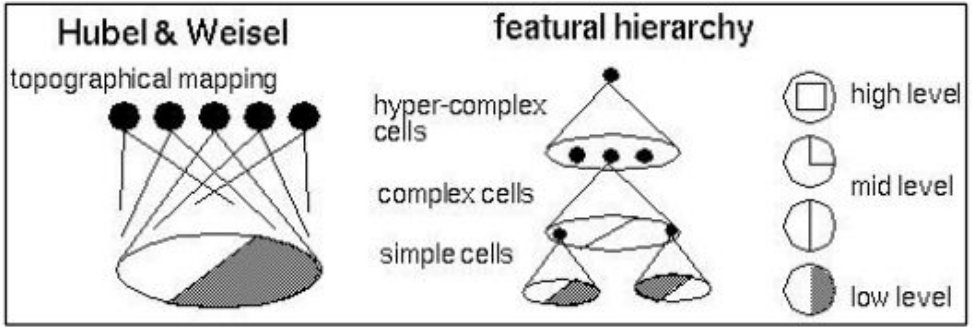
Visualizing Convolutional Networks



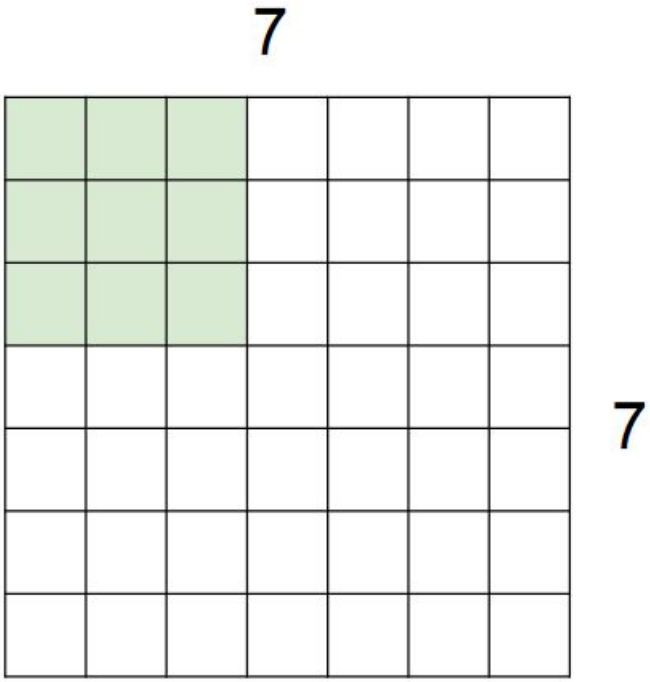
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]



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

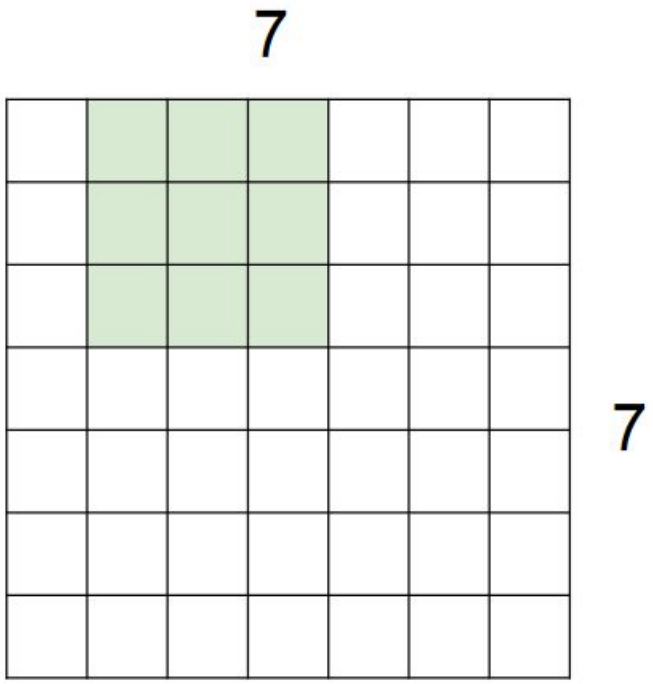


A closer look at spatial dimensions: Stride



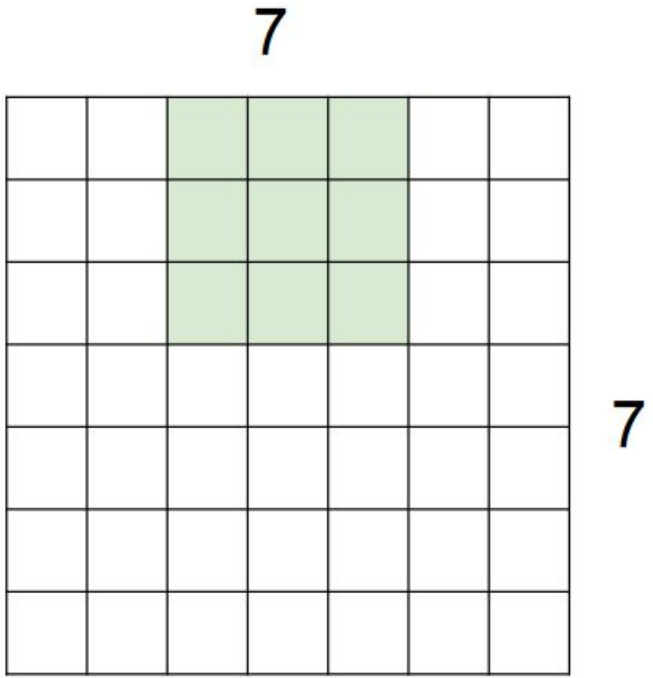
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



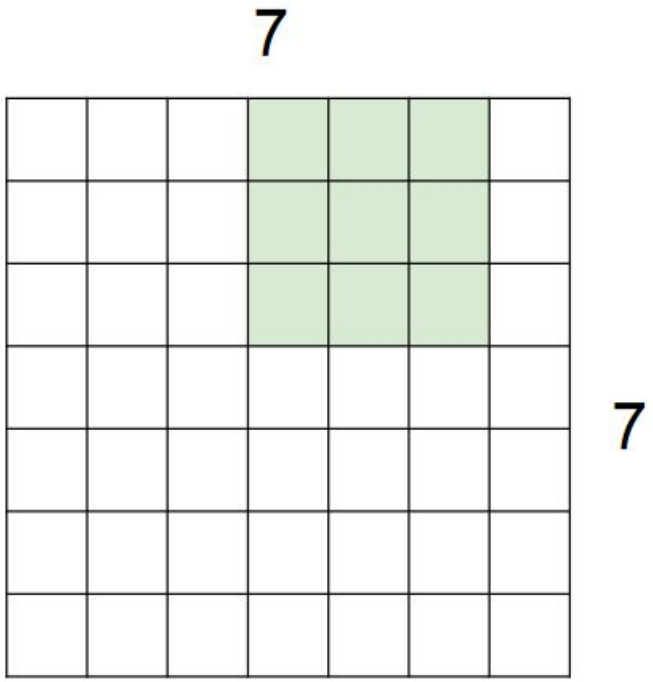
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



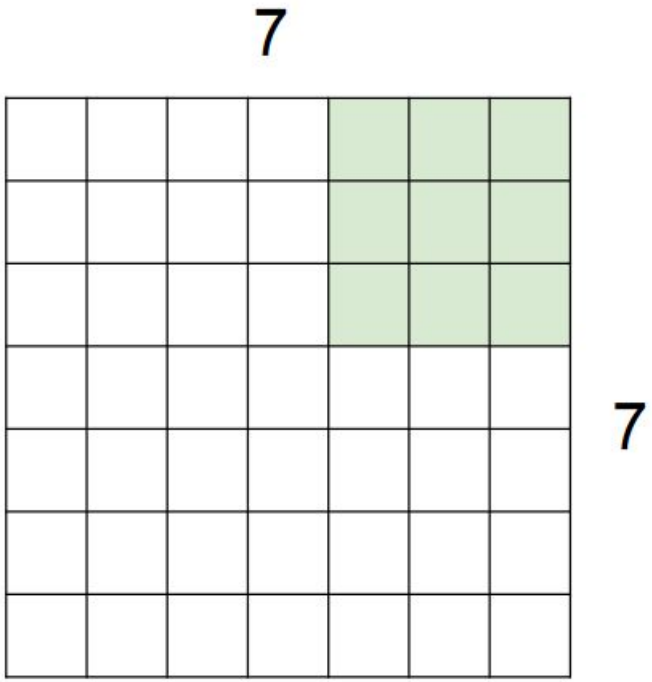
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

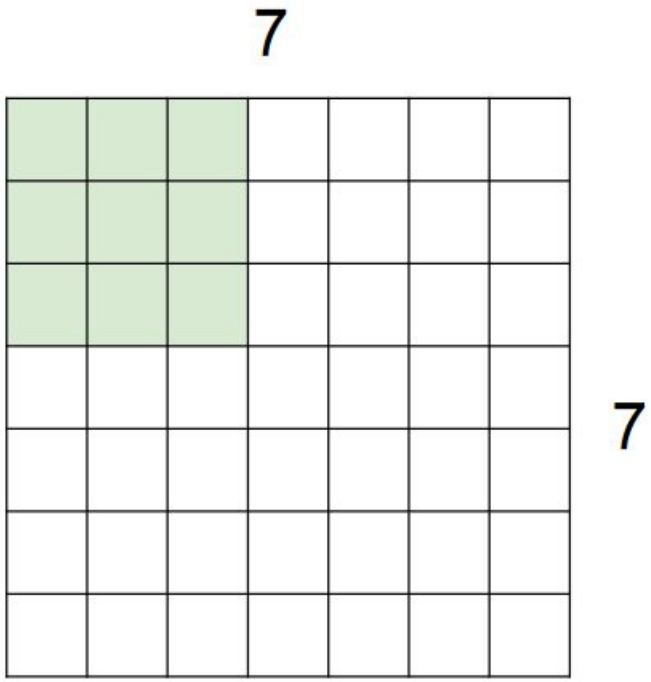
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

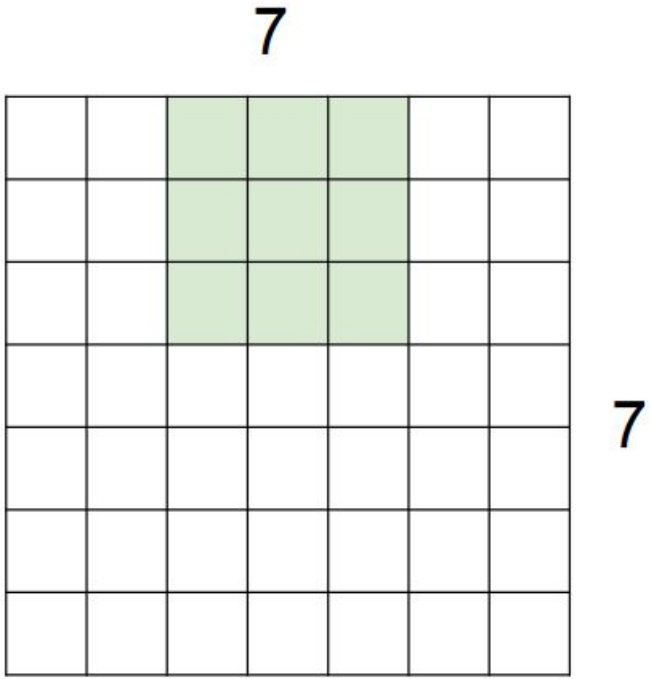
=> 5x5 output

A closer look at spatial dimensions:



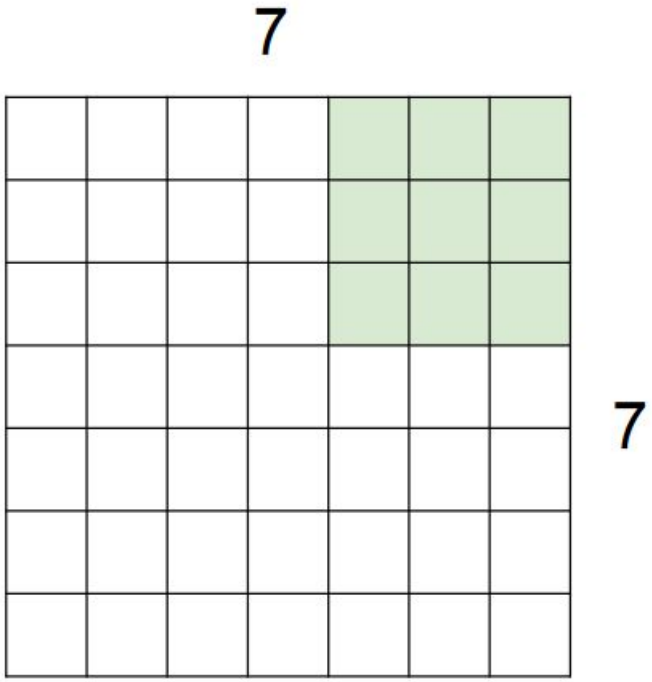
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



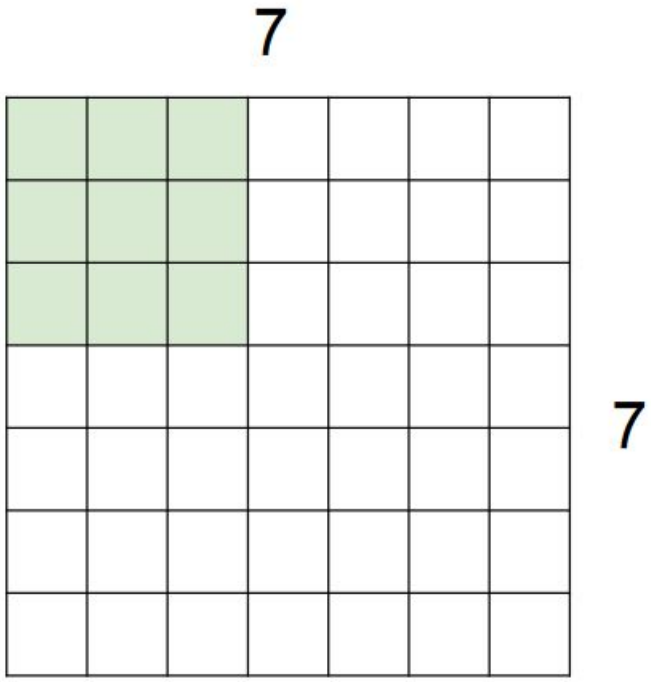
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



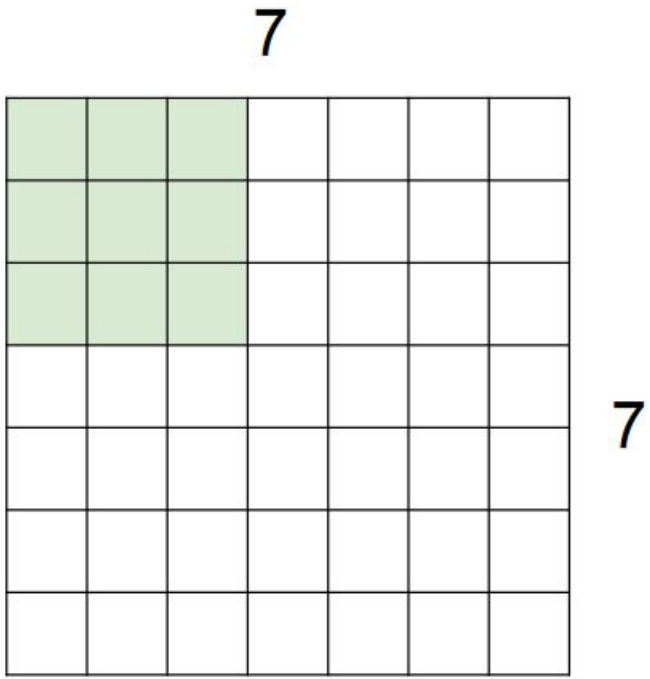
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

A closer look at spatial dimensions:



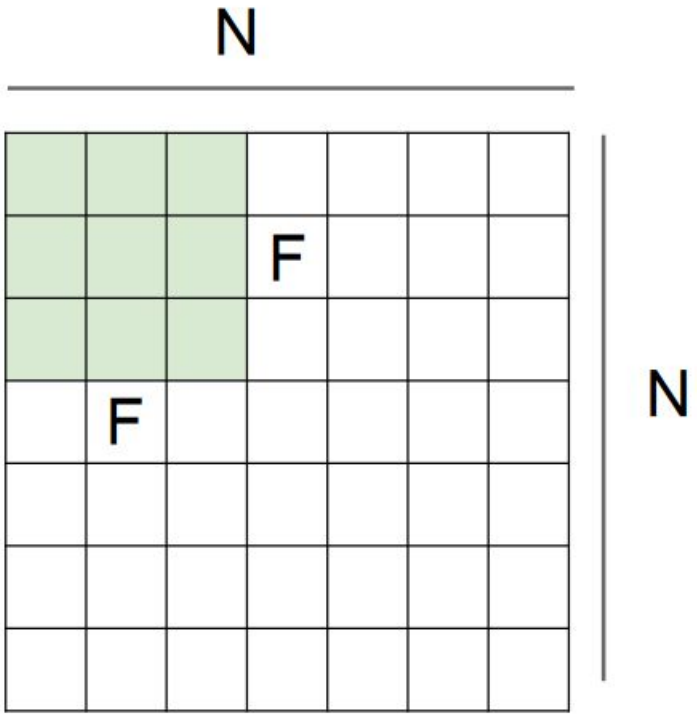
7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

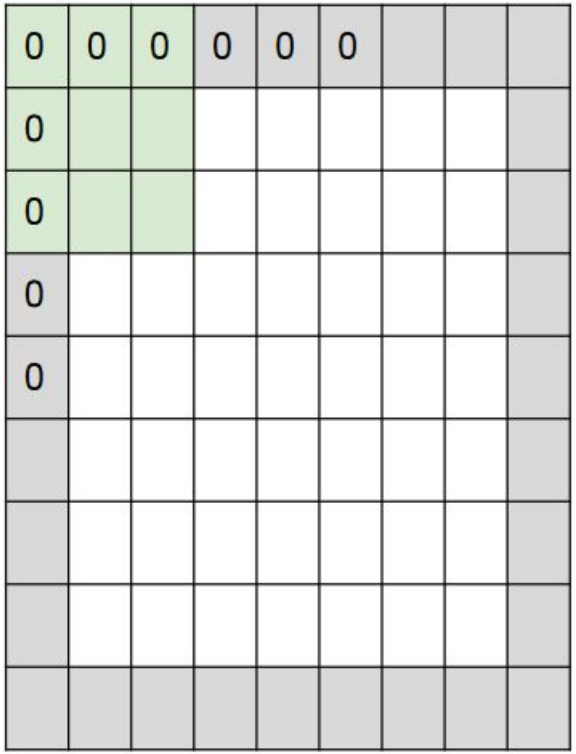
doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

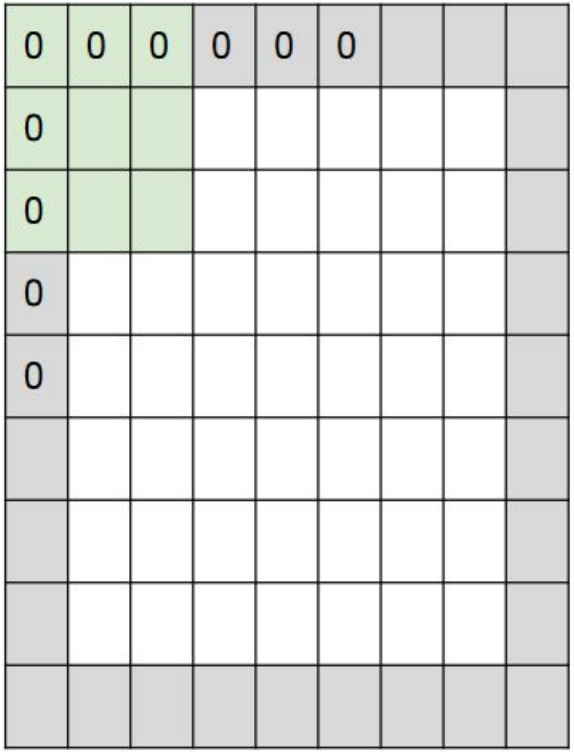
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(N - F) / stride + 1

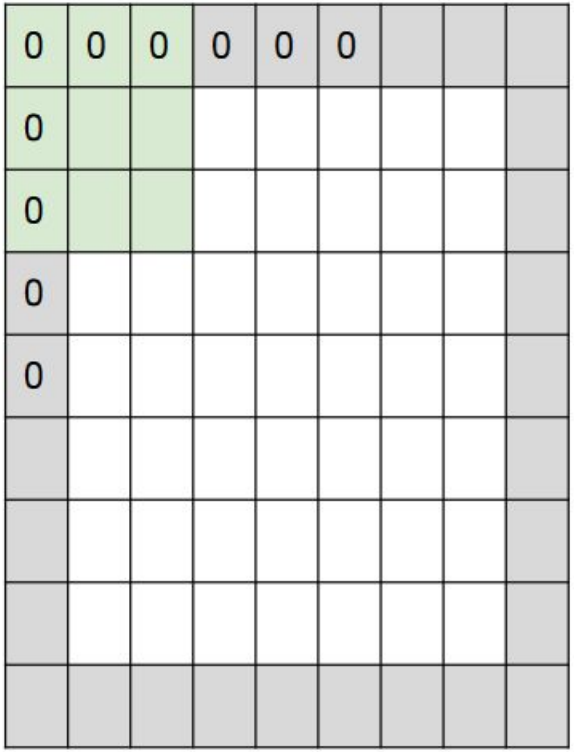
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

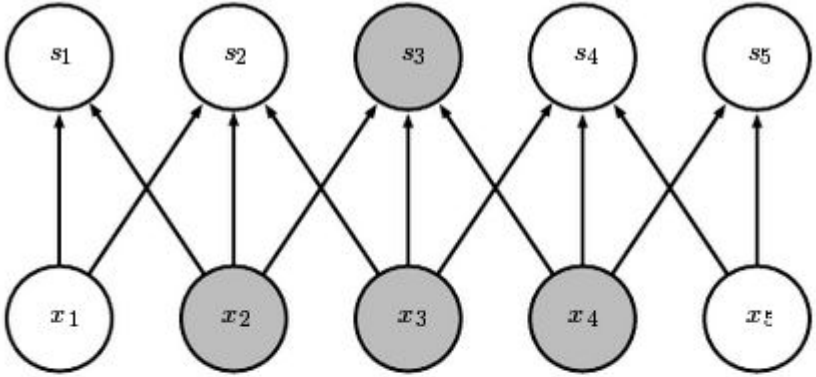
Reasons why convolution is cool

Reason 1: Sparse connectivity

Reason 2: Parameter sharing

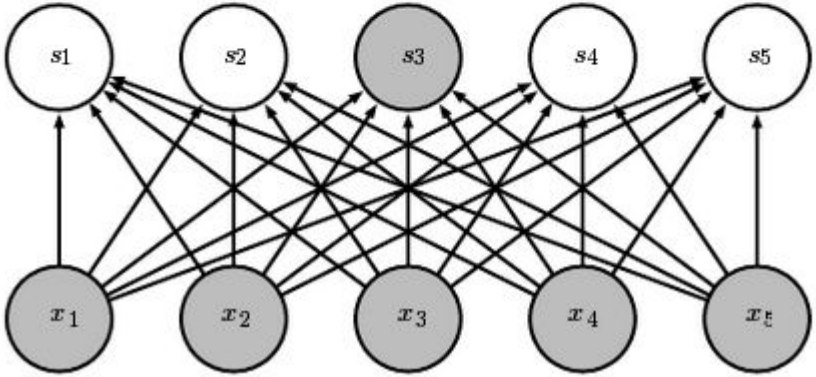
Reason 3: Deeper layers- wider receptive field

Sparse Connectivity



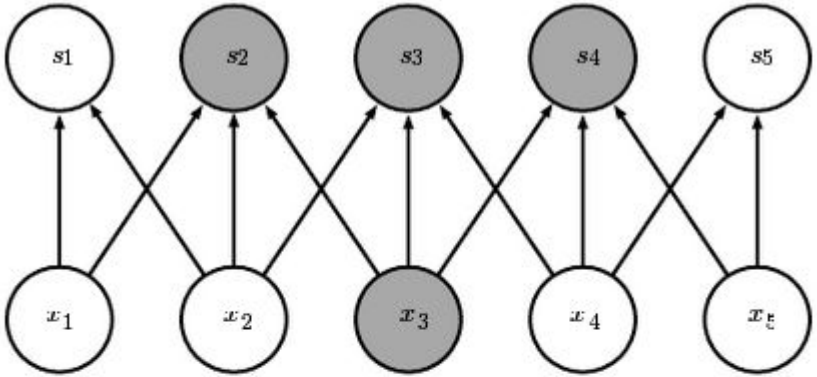
Viewed from above:

Sparse connection => s3 affected by only x1,x21,x3



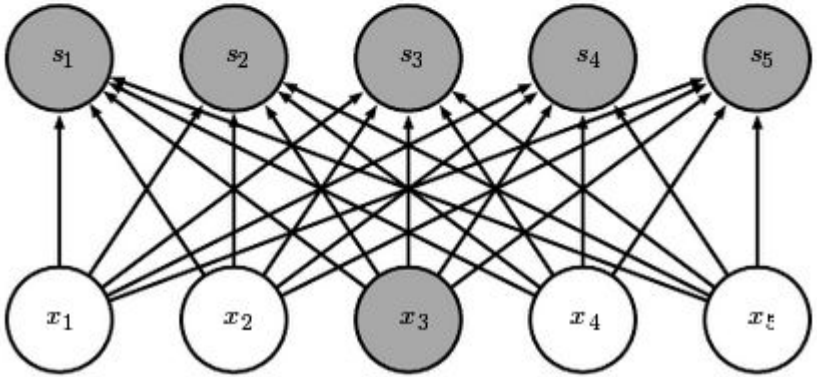
No longer sparse =>s3 affects all the input units

Sparse Connectivity



Viewed from below:

Sparse connection => x3 affects only s2,s3,s4



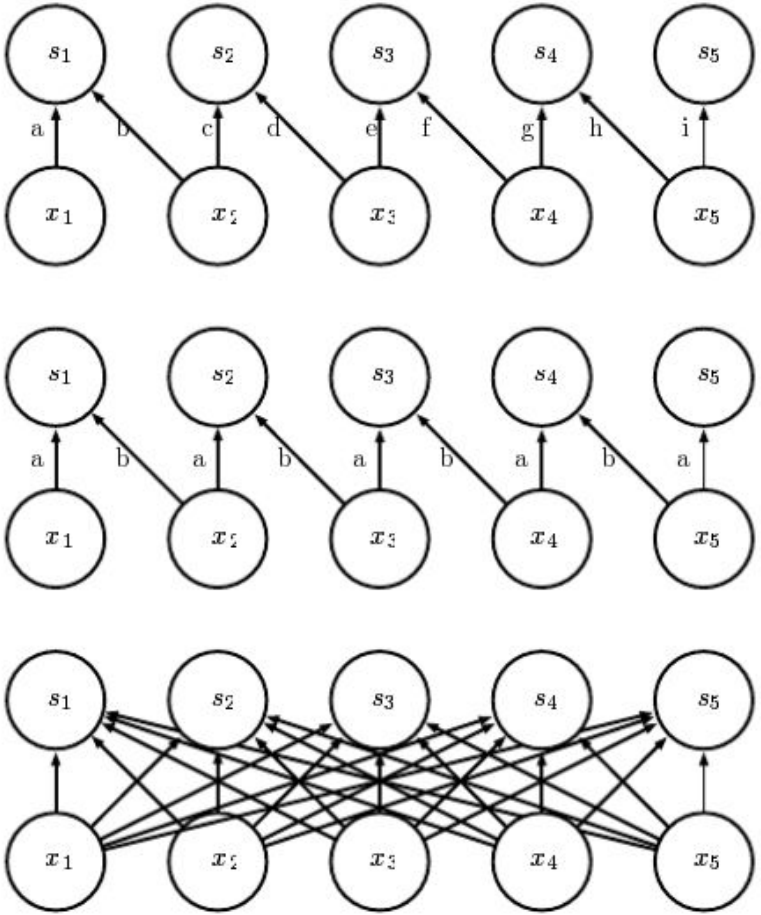
No longer sparse =>x3 affects all the output units

Parameter sharing

- In convolutional net - learn the values of a particular kernel which is used for the entire image

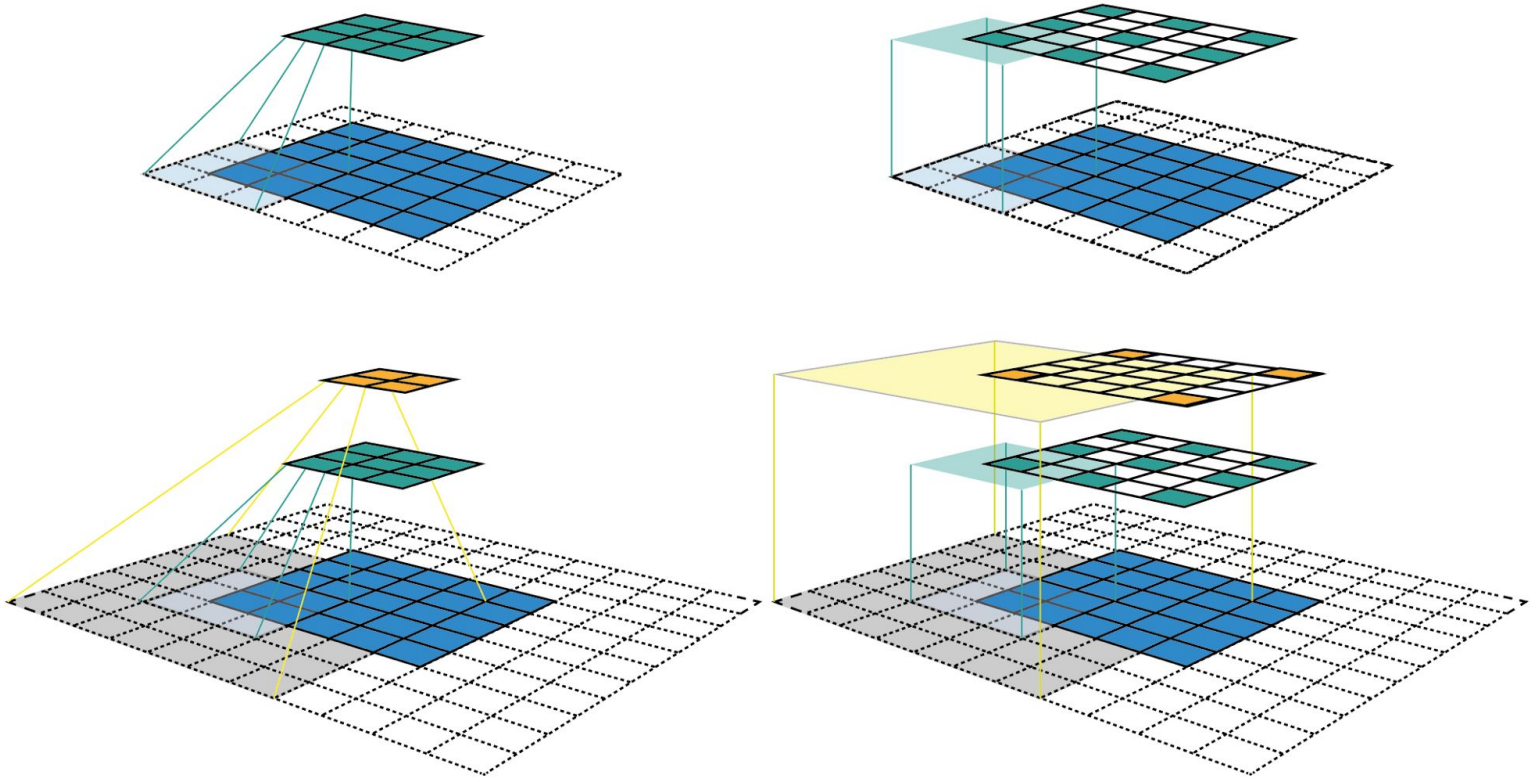
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

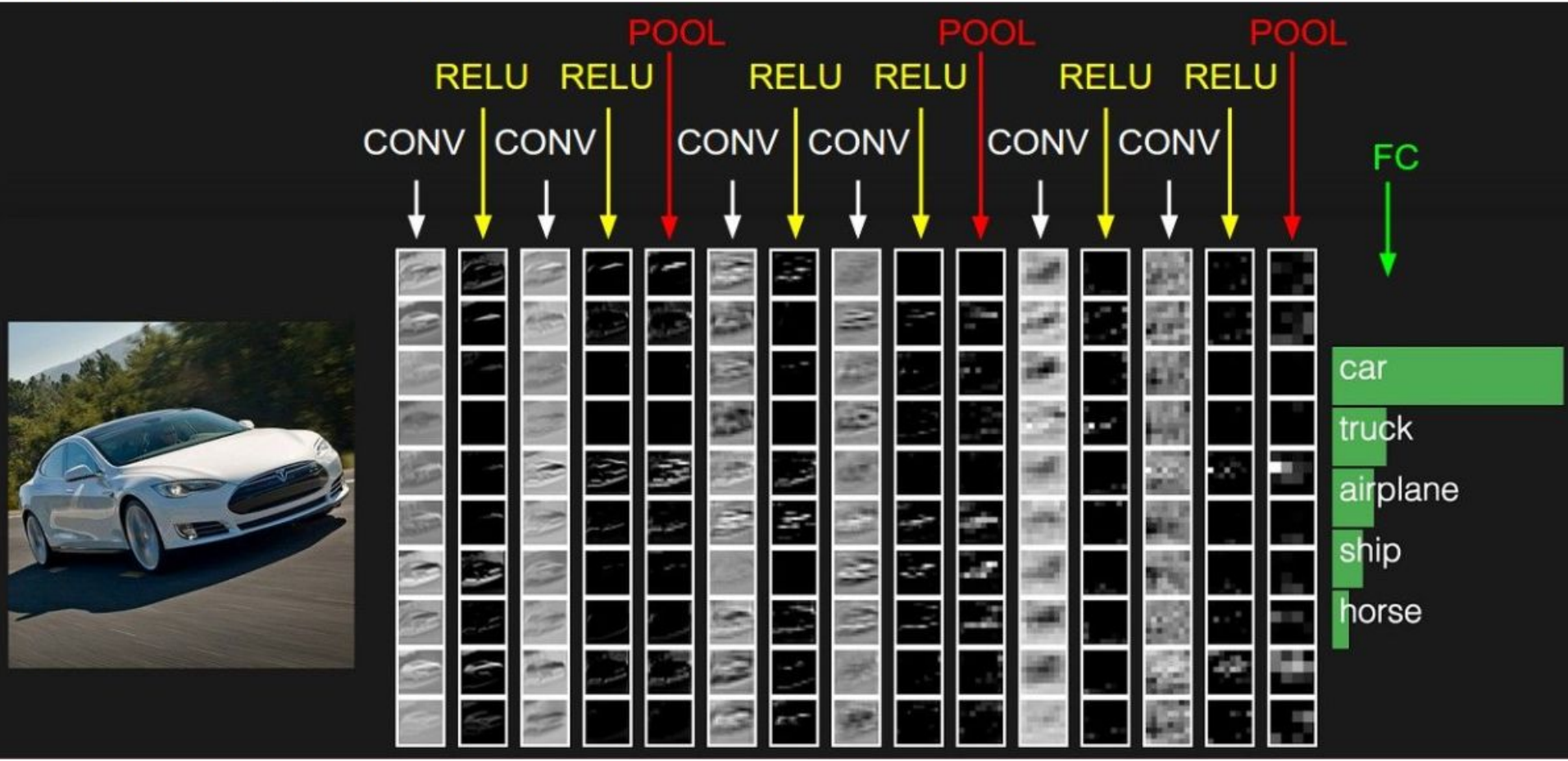


Convolution is thus drastically more efficient than dense matrix multiplication

Deeper layer- wider receptive field



two more layers to go: POOL/FC



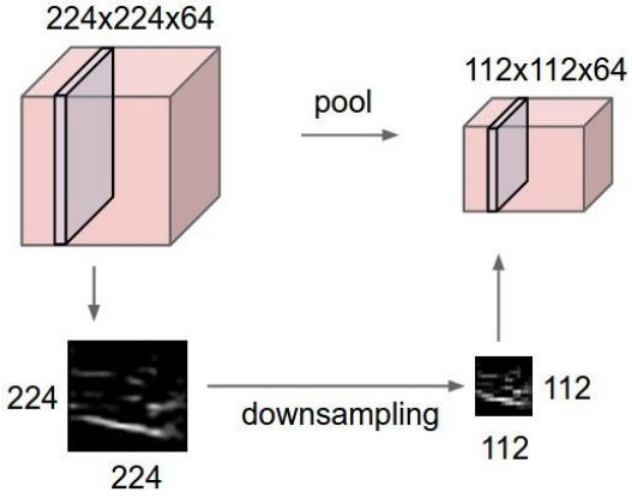
Overview

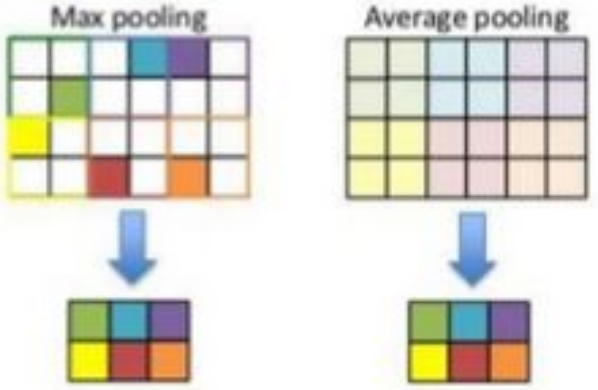
1. CNN and some useful applications
   a. Motivation
   b. Other applications
2. Biological Connection
   a. Idea behind it
3. Structure of a CNN
   a. Convolution- the C in CNN
   b. Non-Linearity
   c. Pooling
   d. Fully-Connected Layer
4. Example architectures
5. Beyond Images
6. References

Pooling

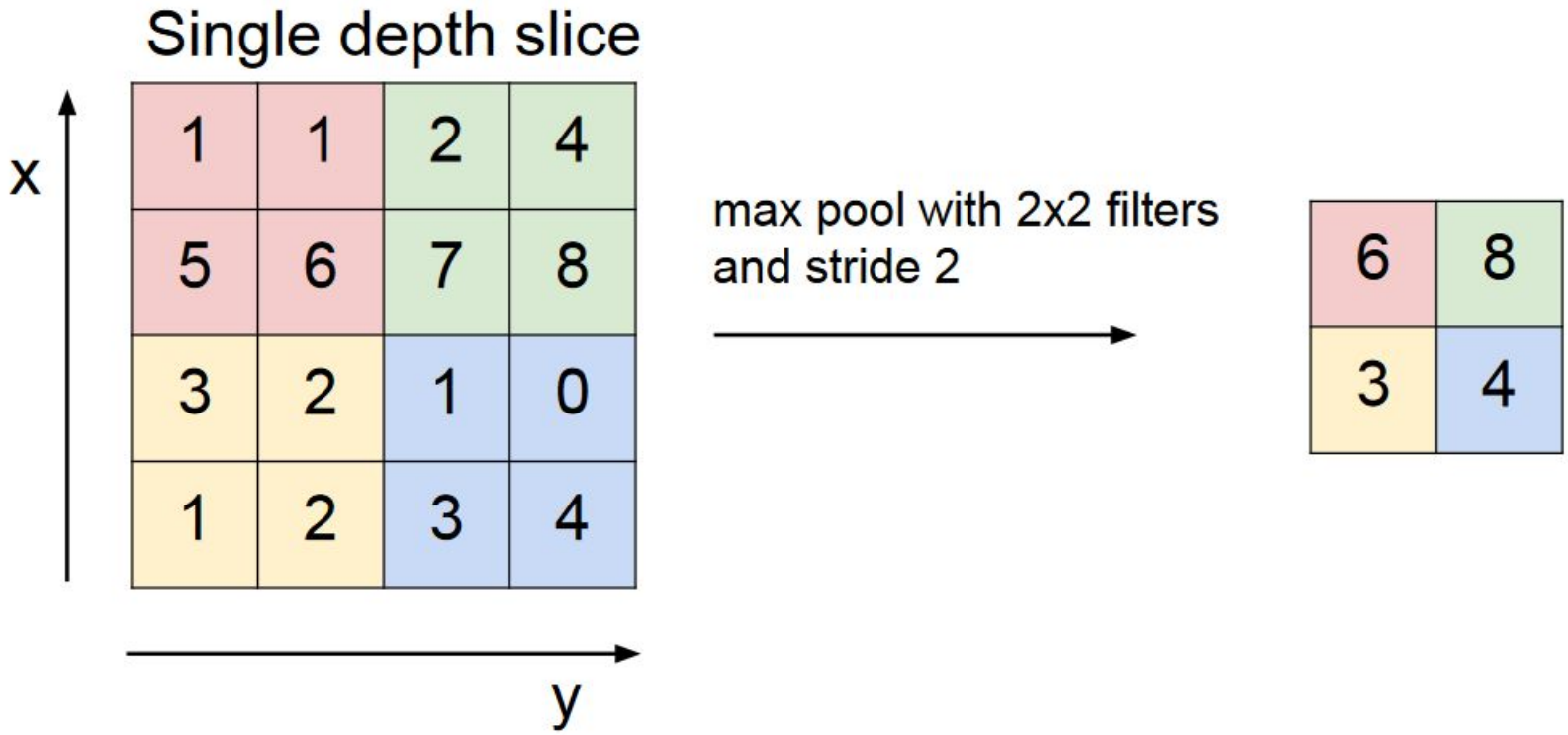
Reduces dimensionality of feature maps- but retains the most important information

Different types: max, average, sum

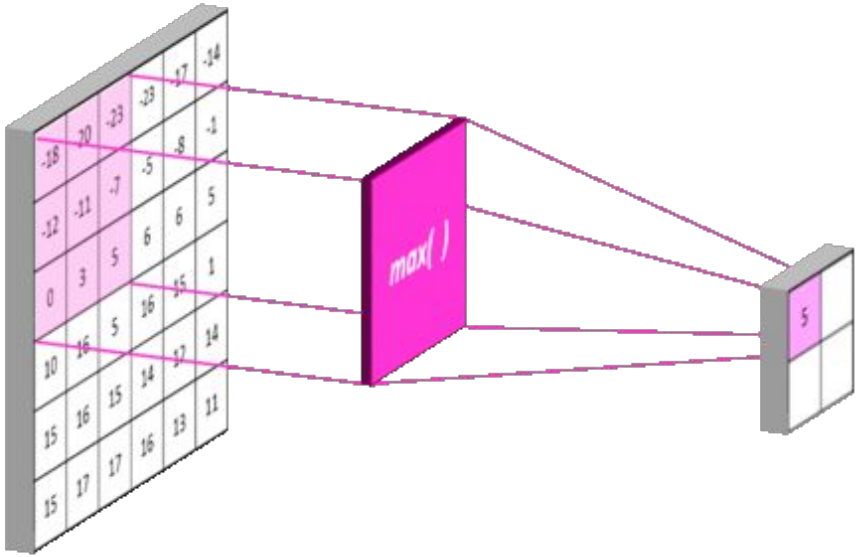




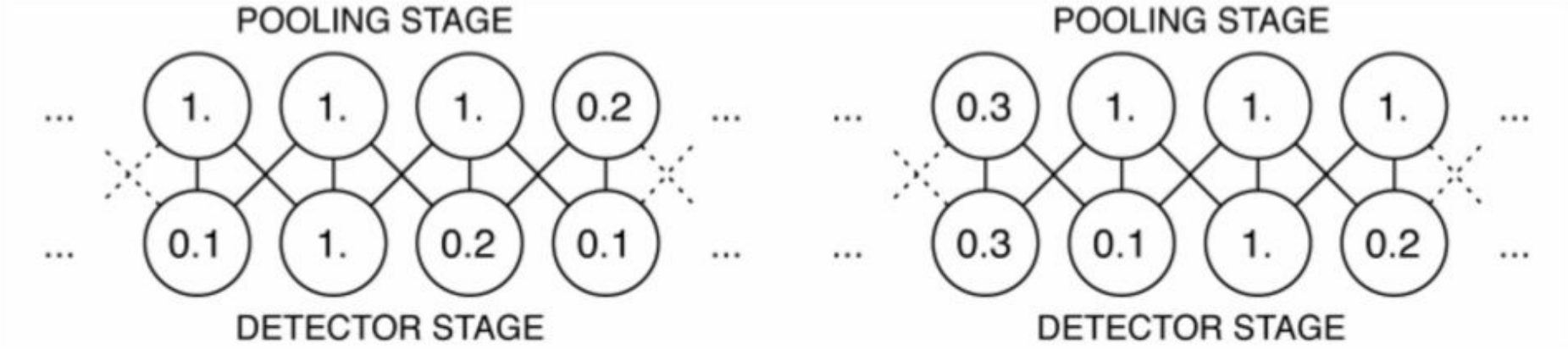
MAX POOLING


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Max Pooling


-16
20
-23
-23
17
-14
-12
-11
-7
-5
-8
-1
0
3
5
6
6
5
10
16
5
16
15
14
17
1
15
16
15
14
17
14
15
17
17
16
13
11
max( )
5

Invariant to small transformations



Pooling, in particular

- makes the input representations (feature dimension) smaller and more manageable
- reduces the number of parameters and computations in the network
- makes network invariant to small transformations, distortions and translations in the input image

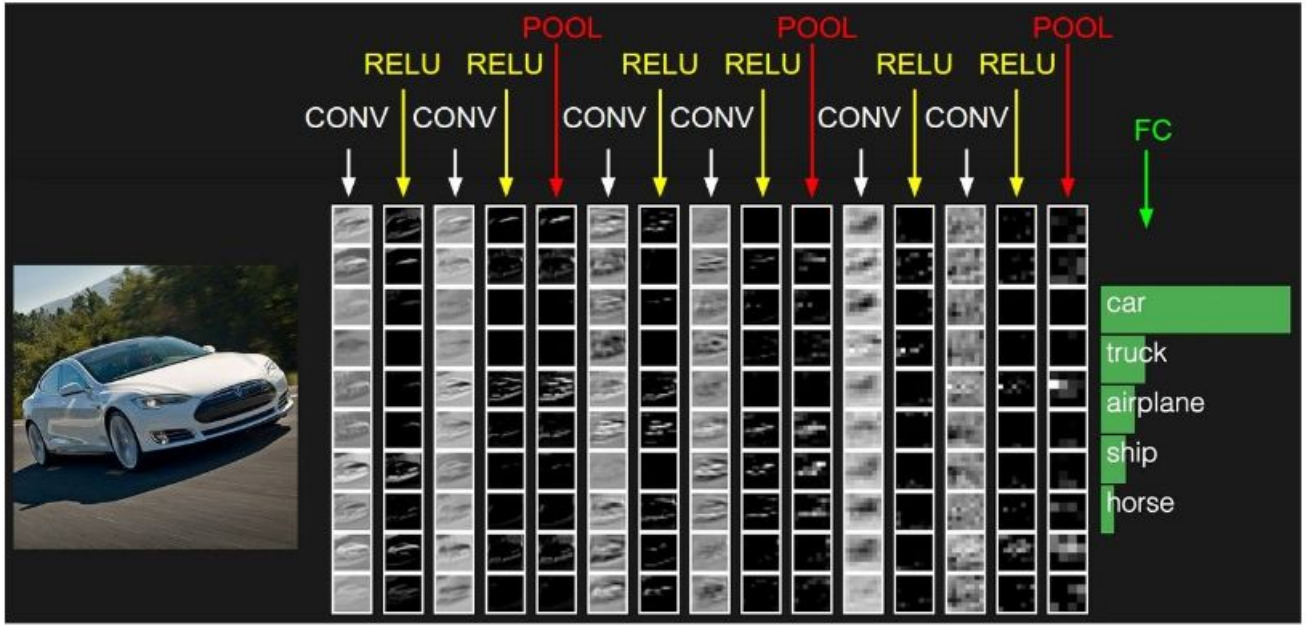
(a) basic (b) *rot* (c) *scale* (d) *trans*

Overview

1. CNN and some useful applications
   a. Motivation
   b. Other applications
2. Biological Connection
   a. Idea behind it
3. Structure of a CNN
   a. Convolution- the C in CNN
   b. Non-Linearity
   c. Pooling
   d. Fully-Connected Layer
4. Example architectures
5. Beyond Images
6. References

Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks

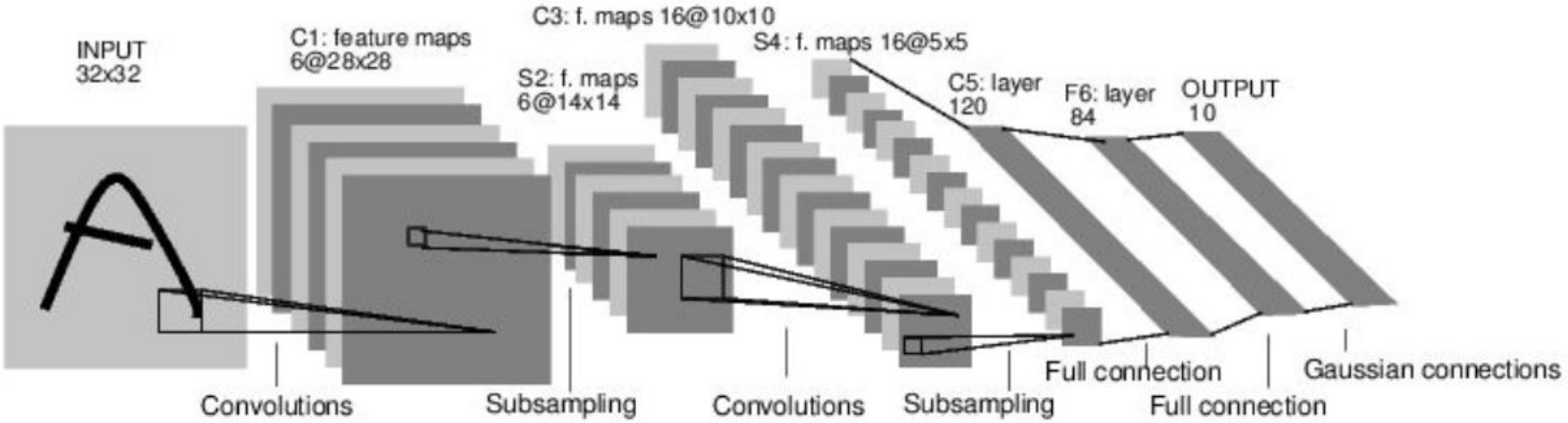


Overview

1. CNN and some useful applications
   a. Motivation
   b. How? -> CNN
   c. Other applications
2. Biological Connection
   a. Idea behind it (where does it come from)
3. Structure of a CNN
   a. Convolution- Different variants of convolution?
   b. Non-Linearity
   c. (Pooling
4. Example architectures
5. Beyond Images
6. References

Case Study: LeNet-5

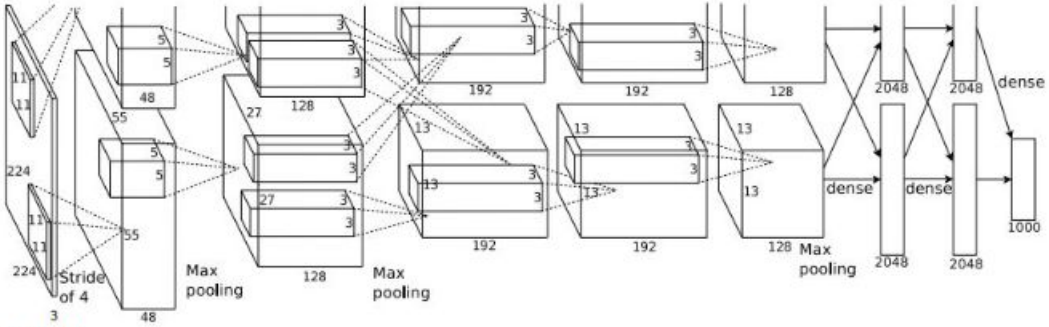
[LeCun et al., 1998]



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

Case Study: AlexNet

[Krizhevsky et al. 2012]

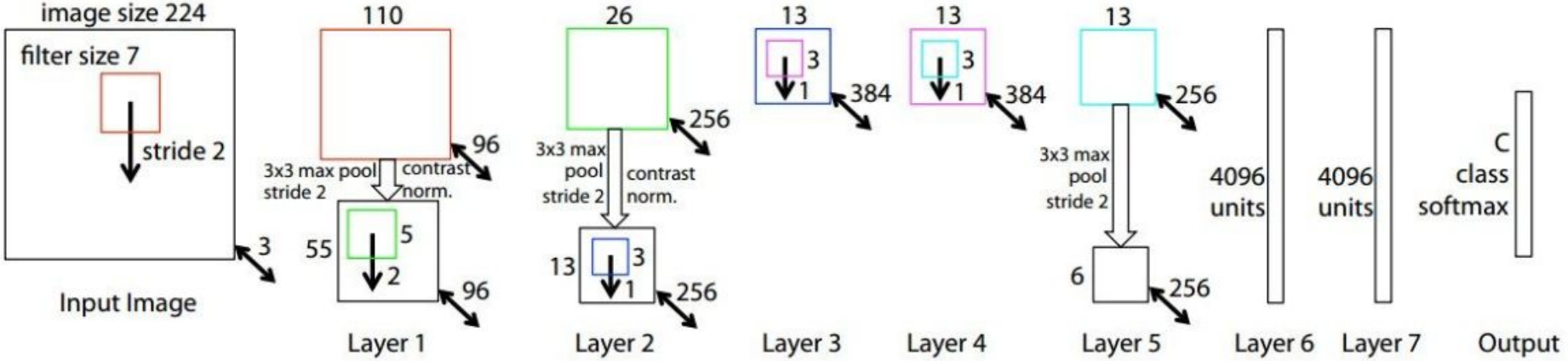


Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] 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%

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

ImageNet top 5 error: 15.4% -> 14.8%

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

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

best model

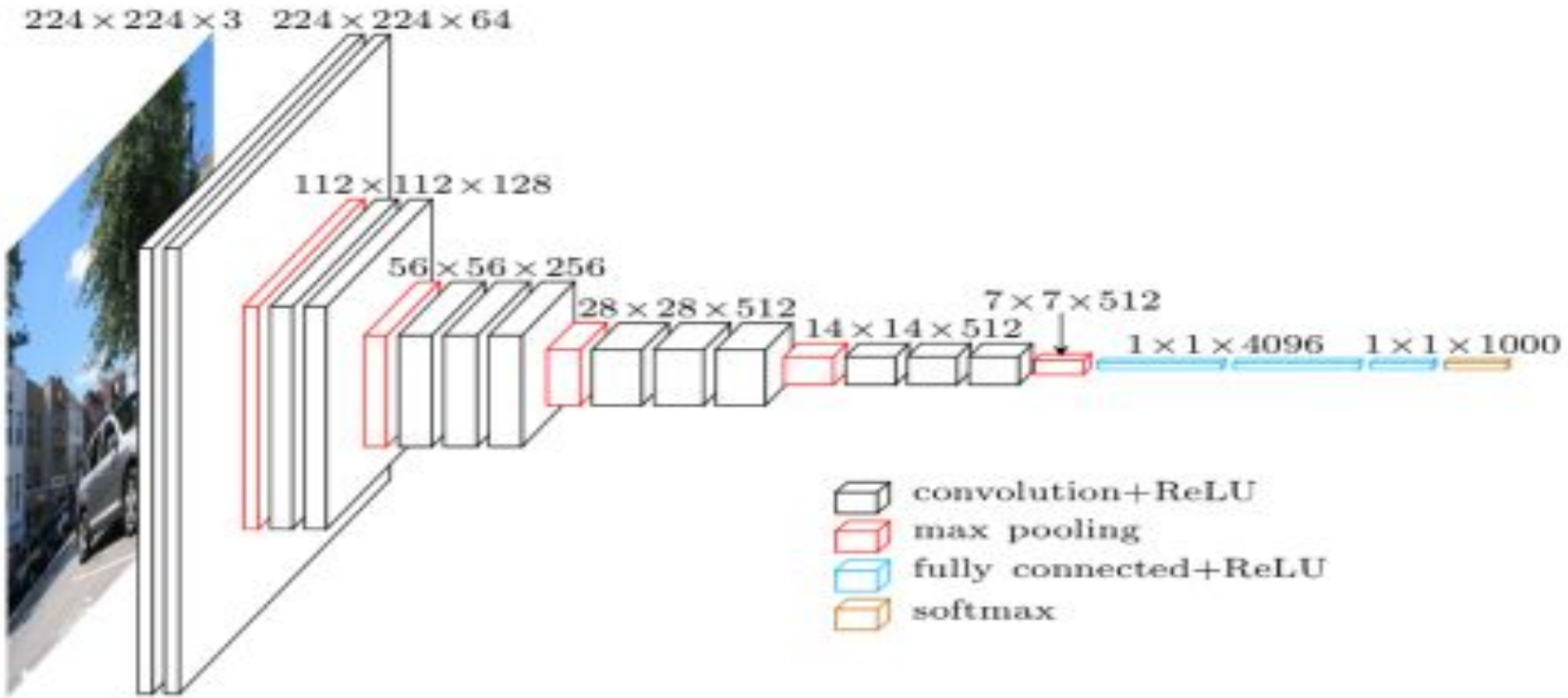
11.2% top 5 error in ILSVRC 2013
->
7.3% top 5 error

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).

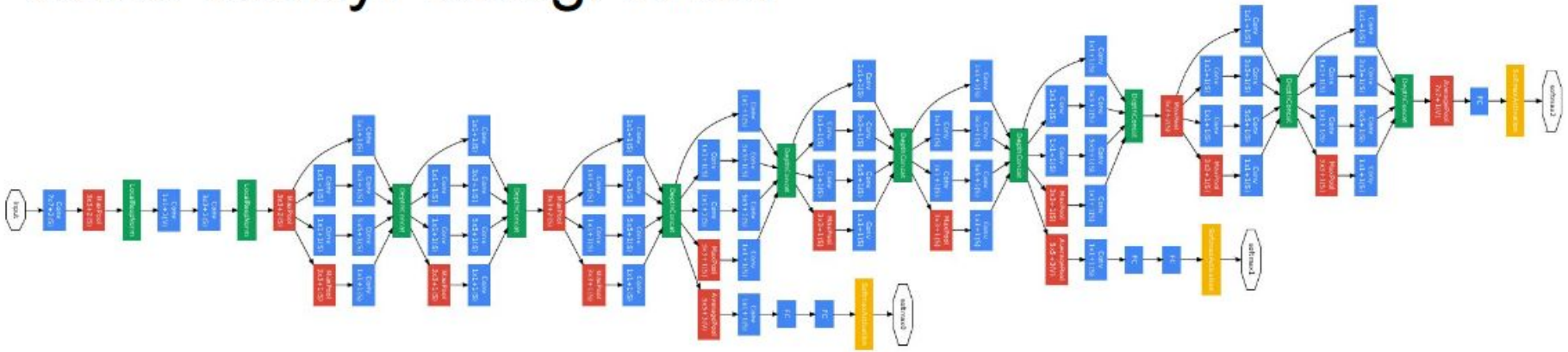
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

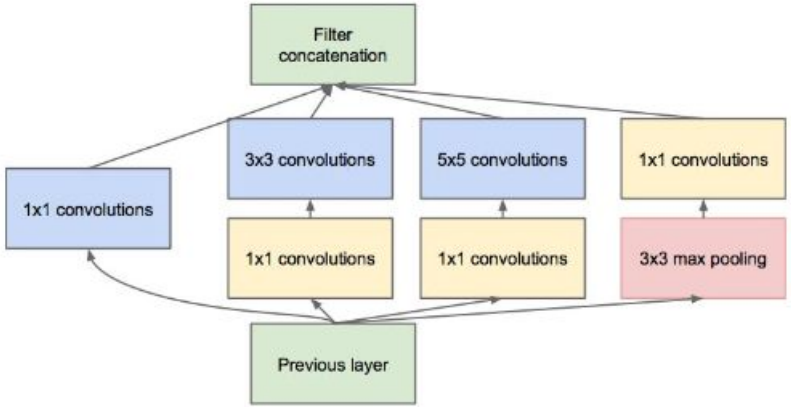
Case Study: VGGNet



Case Study: GoogLeNet

[Szegedy et al., 2014]



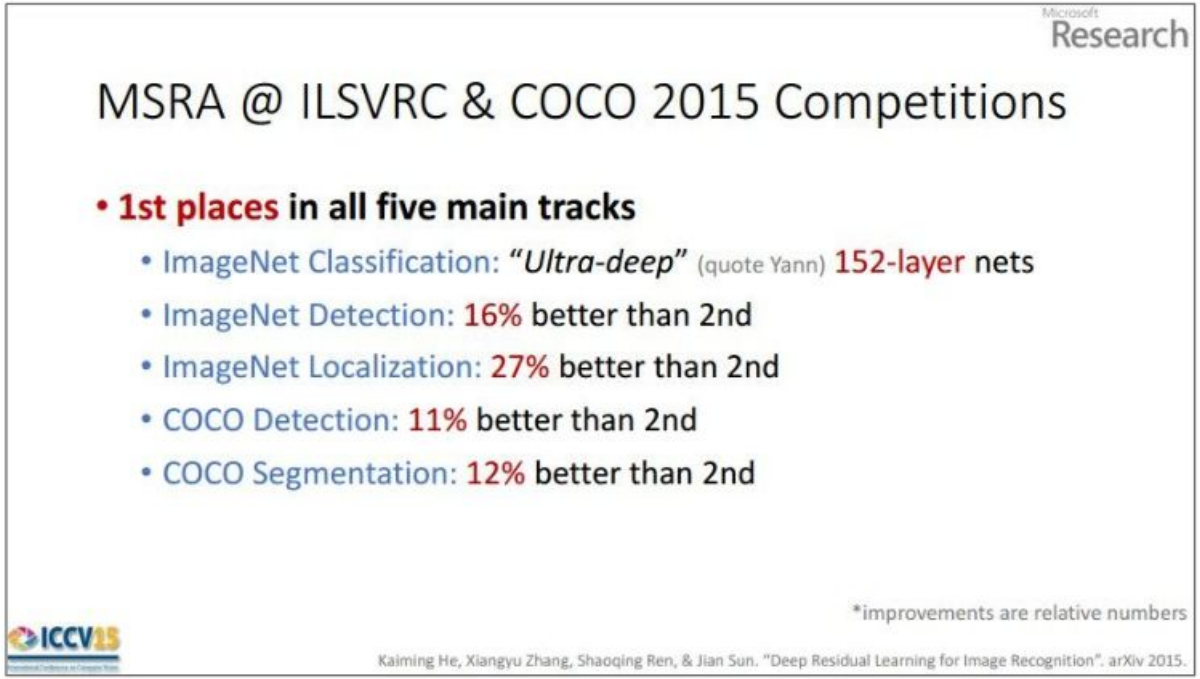


Inception module

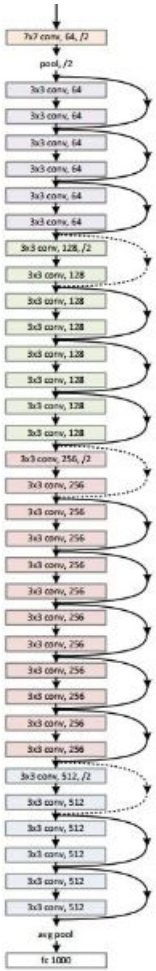
ILSVRC 2014 winner (6.7% top 5 error)

Case Study: ResNet *[He et al., 2015]*

ILSVRC 2015 winner (3.6% top 5 error)



Slide from Kaiming He's recent presentation https://www.youtube.com/watch?v=1PGLj-uKT1w



Case Study: ResNet *[He et al., 2015]*

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

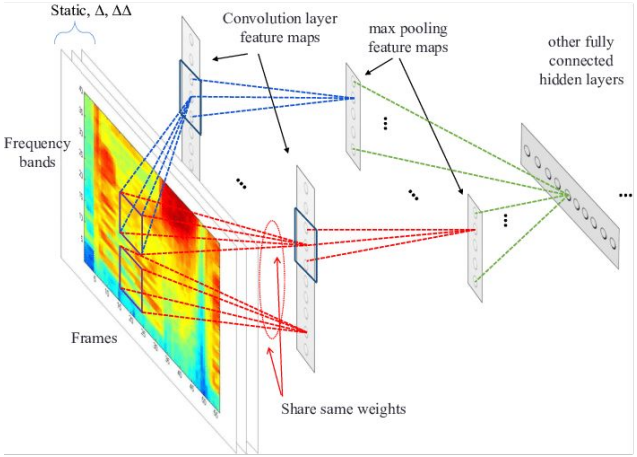
Overview

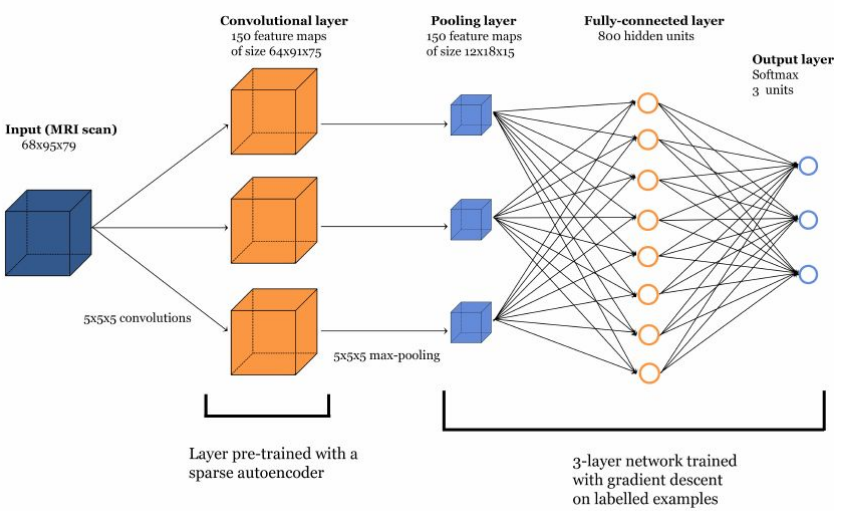
1. CNN and some useful applications
   a. Motivation
   b. How? -> CNN
   c. Other applications
2. Biological Connection
   a. Idea behind it (where does it come from)
3. Structure of a CNN
   a. Convolution- Different variants of convolution?
   b. Non-Linearity
   c. (Pooling
4. Example architectures
5. Beyond Images
6. References

Beyond Images

Convolution can also be used to process:

- 1D: Audio waveform; skeleton animation data
- 2D: Audio data with Fourier transform; Images!
- 3D: Volumetric data, video data





Conclusion

- ConvNets - super important tool, especially for computer vision tasks
- ConvNets stack CONV,POOL,FC layers
- Understanding of CONV and POOL layers
- Visualizing Convolutional network and neuroscientific basis
- Some standard architectures

References

1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, “Deep Learning”, MIT Press, 2016
2. Vijay Badarinarayan, Alex Kendall, Roberto Cipolla, “SegNet -A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” arXiv preprint arXiv:1511.00561, 2015
3. K Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in ICLR, 2014
4. Andrej Karpathy, CS231n Convolutional Neural Networks for Visual Recognition, 2015
5. Yann LeCun, Ranzato, Deep Learning Tutorial, ICML 2013
6. Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks
7. Feature extraction using convolution, Stanford
8. A Beginner’s Guide To Understanding Convolutional Neural Networks
9. Ujjwal Karn, An Intuitive Explanation of Convolutional Neural Networks, 2016
10. Deep Learning Methods for Vision, CVPR 2012 Tutorial
11. Neural Networks by Rob Fergus, Machine Learning Summer School 2015

Thank You!