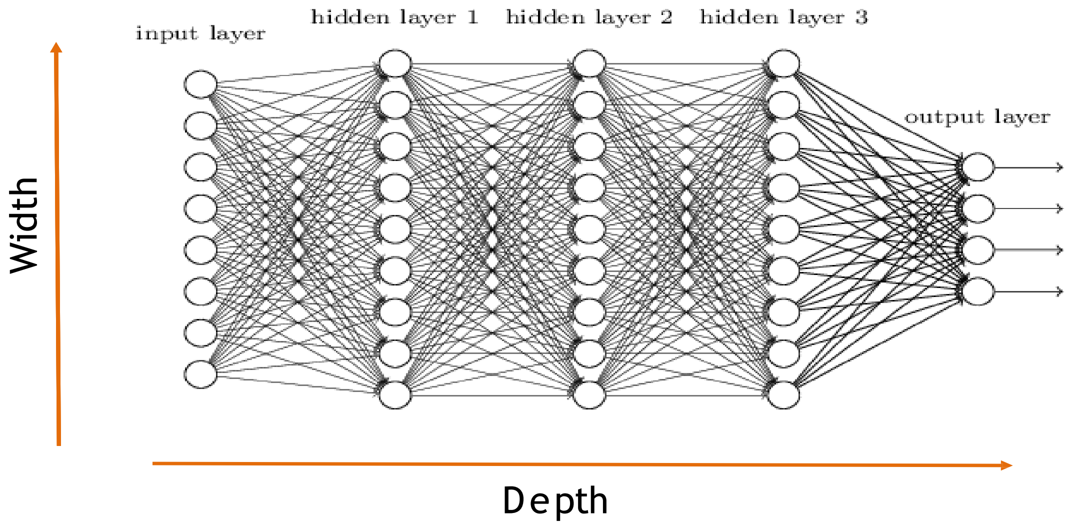
DATA MINING & MACHINE LEARNING

Convolutional Neural Networks

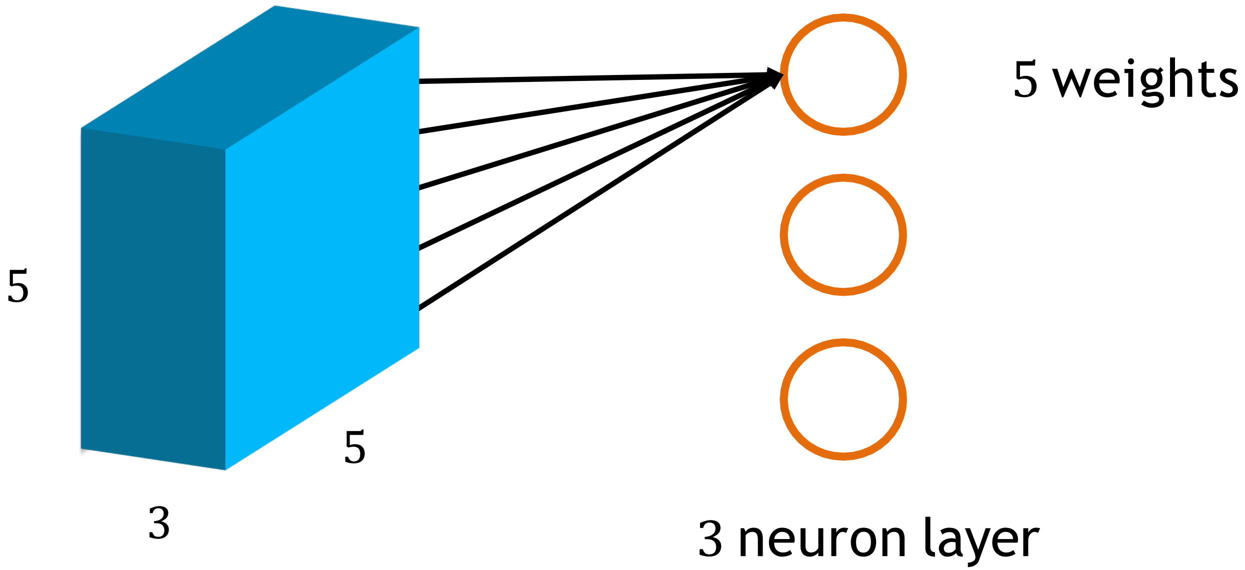
Fully Connected NeuralNetwork

- We know it is good to learn a small model.
- From this fully connected model, do we really need all the edges?
- Can some of these be shared?



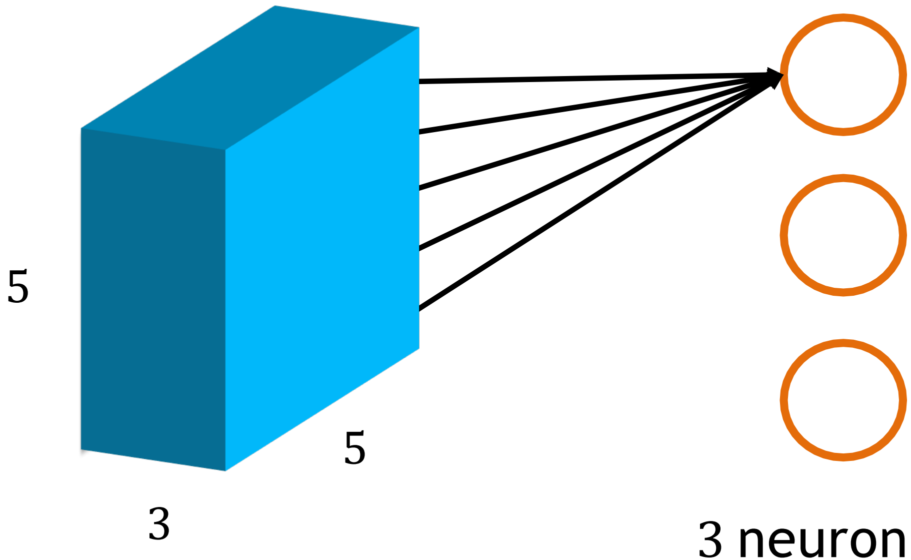
Problems using FC Layers on Images

- How to process a tiny image with FC layers



Problems using FC Layers on Images

- How to process a tiny image with FC layers



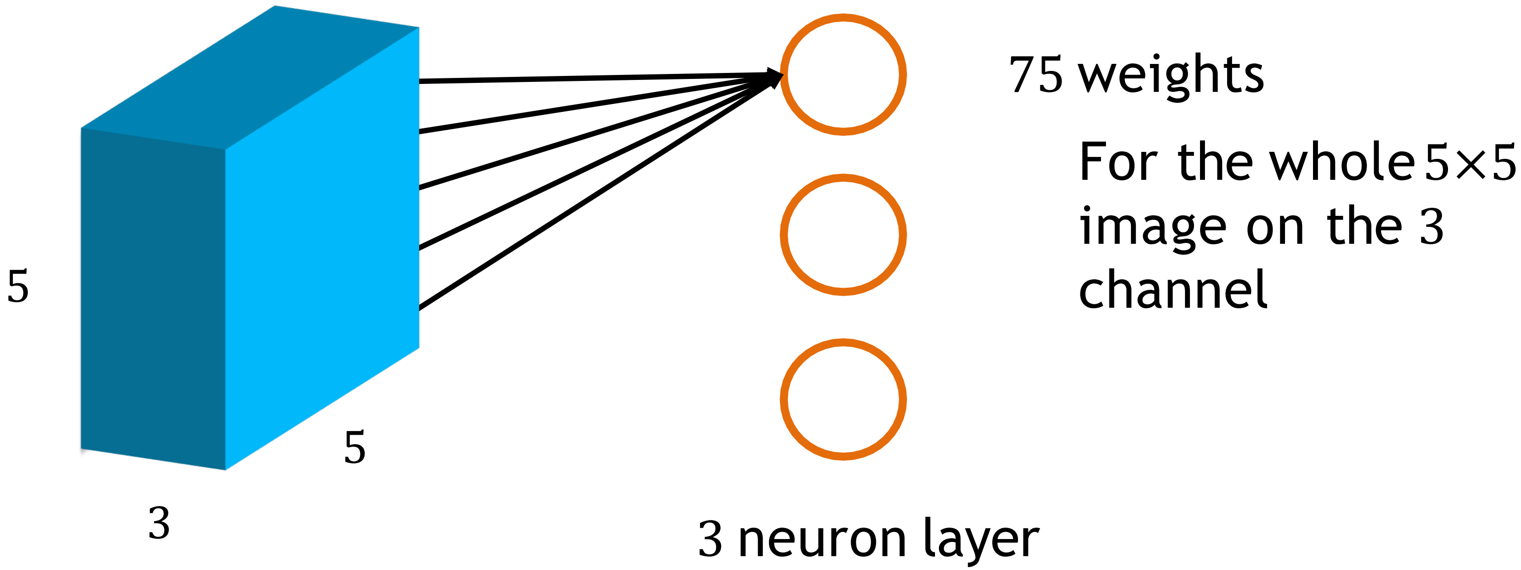
3 neuron layer

25 weights

For the whole 5×5 image on 1 channel

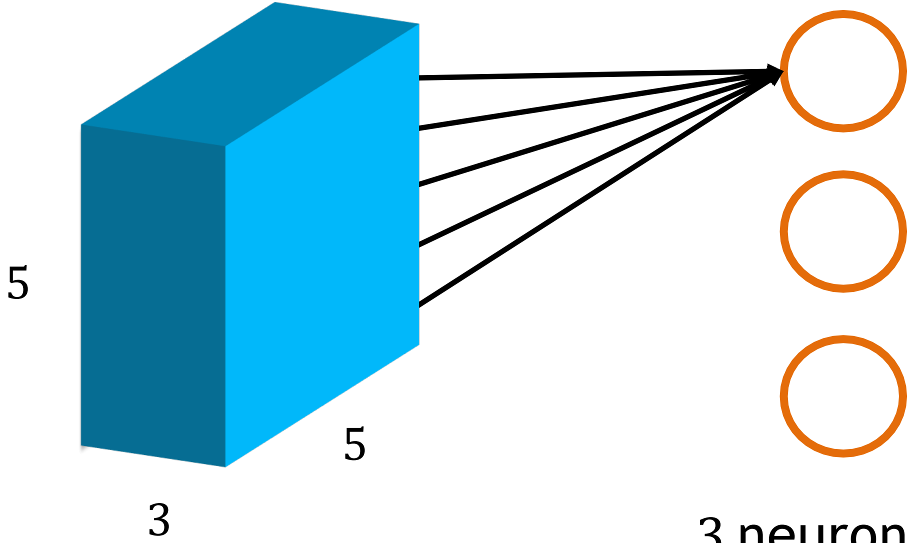
Problems using FC Layers on Images

- How to process a tiny image with FC layers



Problems using FC Layers on Images

- How to process a tiny image with FC layers



75 weights

75 weights

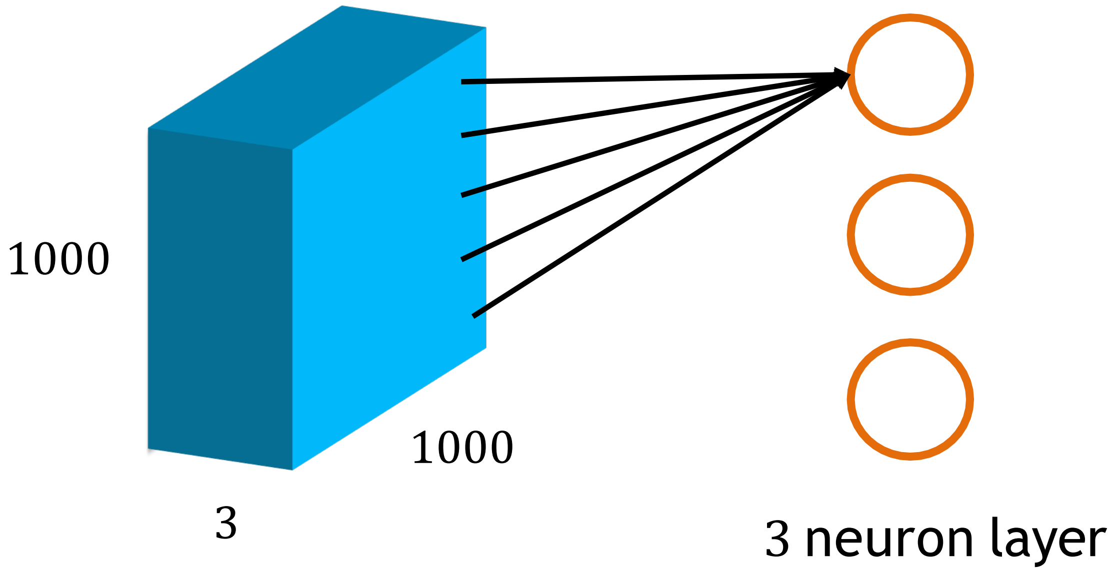
75 weights

3 neuron layer

For the whole 5×5 image on the three channels per neuron

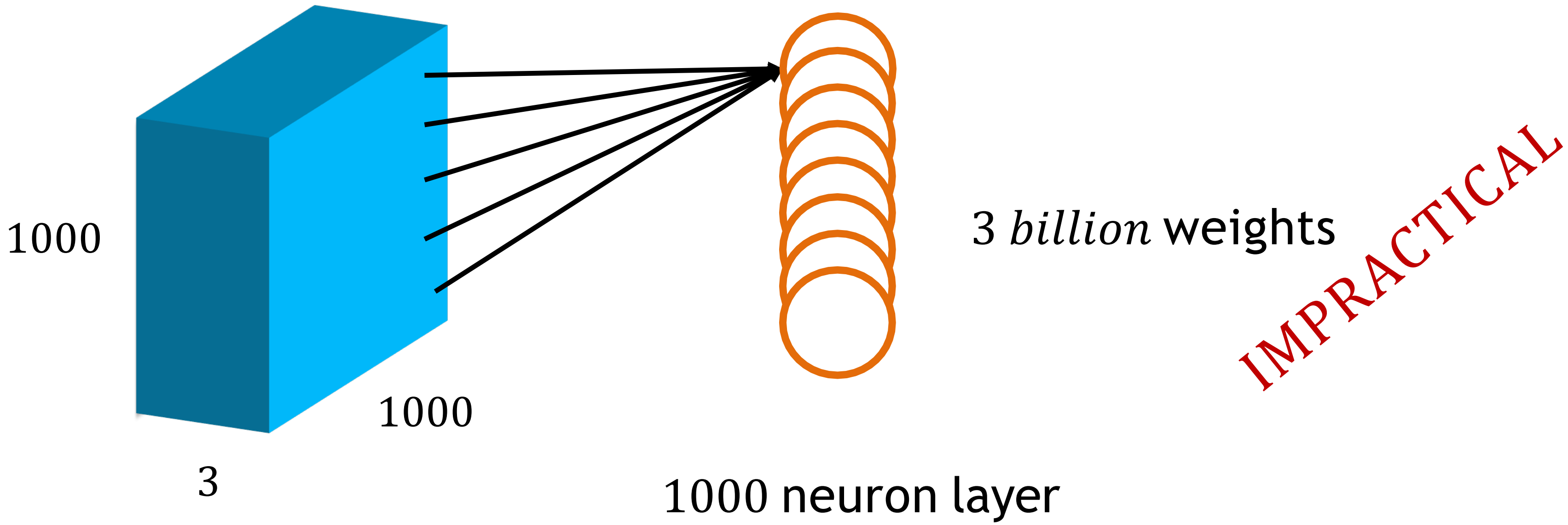
Problems using FC Layers on Images

- How to process a normal image with FC layers



Problems using FC Layers on Images

- How to process a normal image with FC layers



Why not simply more FC Layers?

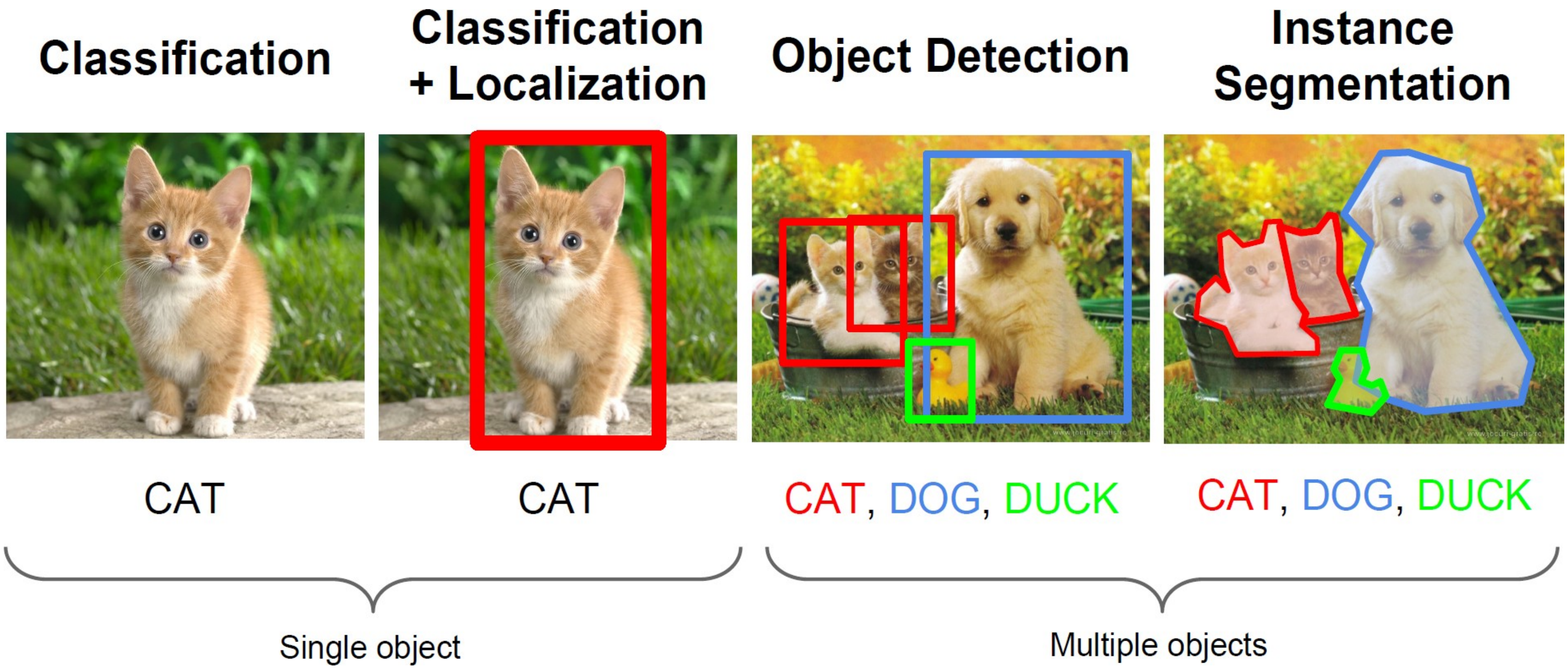
We cannot make networks arbitrarily complex

- Why not just go deeper and get better?
  - No structure!!
  - It is just brute force!
  - Optimization becomes hard
  - Performance plateaus / drops!

Better Way than FC ?

- We want to restrict the degrees of freedom
  - We want a layer with structure
  - Weight sharing → using the same weights for different parts of the image

Using CNNs in Computer Vision



Convolutional NN

In 1995, Yann LeCun and Yoshua Bengio introduced the concept of convolutional neural networks.

Convolutional Neural Networks is extension of traditional Multi-layer Perceptron, based on 3 ideas:

1. Local receive fields
2. Shared weights
3. Spatial / temporal sub-sampling

See LeCun paper (1998) on text recognition:

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

About CNN’s

- CNN’s Were neurobiologically motivated by the findings of locally sensitive and orientation-selective nerve cells in the visual cortex.

- They designed a network structure that implicitly extracts relevant features.

- Convolutional Neural Networks are a special kind of multi-layer neural networks.

About CNN’s

- CNN is a feed-forward network that can extract topological properties from an image.
- Like almost every other neural networks they are trained with a version of the back-propagation algorithm.
- Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing.
- They can recognize patterns with extreme variability (such as handwritten characters).

Convolutions

What are Convolutions?

Discrete case: box filter

f	4	3	2	-5	3	5	2	5	5	6
g	1/3	1/3	1/3							

'Slide' filter kernel from left to right; at each position, compute a single value in the output data

What are Convolutions?

Discrete case: box filter

f	4	3	2	-5	3	5	2	5	5	6
g	1/3	1/3	1/3							
$f * g$		3								

$$4 * (1/3) + 3 * (1/3) + 2 * (1/3) = 3$$

What are Convolutions?

Discrete case: box filter

f	**4**	**3**	**2**	**-5**	**3**	**5**	**2**	**5**	**5**	**6**
g		**1/3**	**1/3**	**1/3**						
$f * g$		**3**	**0**							

3* (1/3)+2*(1/3)+ (-5)*(1/3)=0

What are Convolutions?

Discrete case: box filter

f	4	3	2	-5	3	5	2	5	5	6
g			1/3	1/3	1/3					
$f * g$		3	0	0						

2* (1/3)+(-5)*(1/3)+ 3*(1/3)=0

What are Convolutions?

Discrete case: box filter

f	**4**	**3**	**2**	**-5**	**3**	**5**	**2**	**5**	**5**	**6**
g				**1/3**	**1/3**	**1/3**				
$f * g$		**3**	**0**	**0**	**1**					

-5* (1/3)+3 *(1/3)+ 5*(1/3)=1

What are Convolutions?

Discrete case: box filter

f	**4**	**3**	**2**	**-5**	**3**	**5**	**2**	**5**	**5**	**6**
g					**1/3**	**1/3**	**1/3**			
$f * g$		**3**	**0**	**0**	**1**	**10/3**				

3* (1/3)+ 5*(1/3)+ 2*(1/3)=10/3

What are Convolutions?

Discrete case: box filter

f	**4**	**3**	**2**	**-5**	**3**	**5**	**2**	**5**	**5**	**6**
g						**1/3**	**1/3**	**1/3**		
$f * g$		**3**	**0**	**0**	**1**	**10/3**	**4**			

5* (1/3)+ 2*(1/3)+ 5*(1/3)=4

What are Convolutions?

Discrete case: box filter

f	4	3	2	-5	3	5	2	5	5	6
g							1/3	1/3	1/3	
$f * g$		3	0	0	1	10/3	4	4		

$$2^* (1/3) + 5\ ^*(1/3) + 5^*(1/3) = 4$$

What are Convolutions?

Discrete case: box filter

f	4	3	2	-5	3	5	2	5	5	6
g								1/3	1/3	1/3
$f * g$		3	0	0	1	10/3	4	4	16/3	

$$5 * (1/3) + 5 * (1/3) + 6 * (1/3) = 16/3$$

What are Convolutions?

Discrete case: box filter

4	3	2	-5	3	5	2	5	5	6

1/3	1/3	1/3

??	3	0	0	1	10/3	4	4	16/3	??

What to do at boundaries?

What are Convolutions?

Discrete case: box filter

4	3	2	-5	3	5	2	5	5	6

1/3	1/3	1/3

??	3	0	0	1	10/3	4	4	16/3	??

What to do at boundaries?

Option 1: Shrink

3	0	0	1	10/3	4	4	16/3

What are Convolutions?

Discrete case: box filter

0	4	3	2	-5	3	5	2	5	5	6	0

1/3	1/3	1/3

??	3	0	0	1	10/3	4	4	16/3	??

$$0 \cdot \frac{1}{3} + 4 \cdot \frac{1}{3} + 3 \cdot \frac{1}{3} = \frac{7}{3}$$

What to do at boundaries?

Option 2: Pad (often 0's)

7/3	3	0	0	1	10/3	4	4	16/3	11/3

Convolutions on Images

Image 5×5

-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

Kernel 3×3

0	-1	0
-1	5	-1
0	-1	0

Output 3×3

6		

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

Convolutions on Images

Image 5×5

-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

Kernel 3×3

0	-1	0
-1	5	-1
0	-1	0

Output 3×3

6	1	

$$2*(-1)+3*(-1)+2*(5)+1*(-1)+3*(-1)$$
$$=10-9=1$$

Convolutions on Images

Image 5×5

-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

Kernel 3×3

0	-1	0
-1	5	-1
0	-1	0

Output 3×3

6	1	8

-5* (-1)+ 2* (-1) + 1* (5) + (-3)* (-1) + 3 * (-1)
= 5 + 3 = 8

Convolutions on Images

Image 5×5

-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

Kernel 3×3

0	-1	0
-1	5	-1
0	-1	0

Output 3×3

6	1	8
-7		

$$3*(-1) + 1*(-1) + 0*(5) + 3*(-1) + 0*(-1)$$
$$= 0 - 7 = -7$$

Convolutions on Images

Image 5×5

-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

Kernel 3×3

0	-1	0
-1	5	-1
0	-1	0

Output 3×3

6	1	8
-7	9	

$$2 * (-1) + 0 * (-1) + 3 * (5) + 3 * (-1) + 1 * (-1)$$
$$= 15 - 6 = 9$$

Convolutions on Images

Image 5×5

-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

Kernel 3×3

0	-1	0
-1	5	-1
0	-1	0

Output 3×3

6	1	8
-7	9	2

$$=1*(-1)+3*(-1)+3*(5)+5*(-1)+4*(-1)$$
$$=15-13=2$$

Convolutions on Images

Image 5×5

-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

Kernel 3×3

0	-1	0
-1	5	-1
0	-1	0

Output 3×3

6	1	8
-7	9	2
-5		

$0^{*}\ (-1) + (-2)^{*}\ (-1) + 0^{*}\ (5) + 1^{*}\ (-1) + 6\ ^{*}\ (-1)$

$= 2-7 = -5$

Convolutions on Images

Image 5×5

-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

Kernel 3×3

0	-1	0
-1	5	-1
0	-1	0

Output 3×3

6	1	8
-7	9	2
-5	-9	

$$3 * (-1) + 0 * (-1) + 1 * (5) + 4 * (-1) + 7 * (-1)$$
$$= 5 - 14 = -9$$

Convolutions on Images

Image 5×5

-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

Kernel 3×3

0	-1	0
-1	5	-1
0	-1	0

Output 3×3

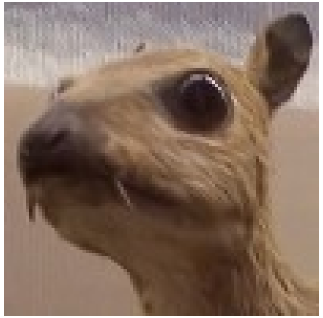
6	1	8
-7	9	2
-5	-9	3

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

Image Filters

- Each kernel gives us a different image filter

Input



Edge detection

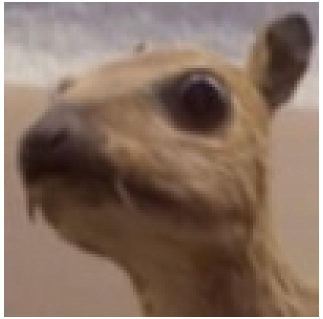
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Sharpen

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

Box mean

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



Gaussian blur

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

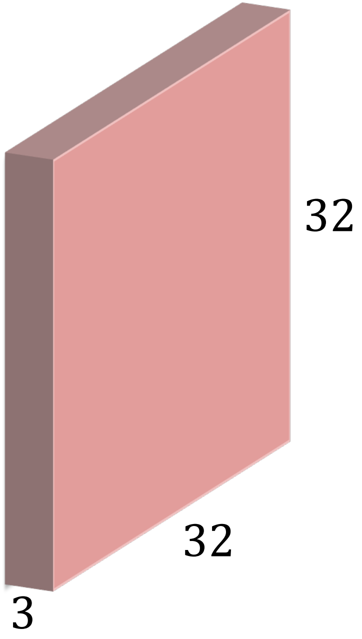
LET'S LEARN THESE FILTERS!

Convolutions on RGBImages

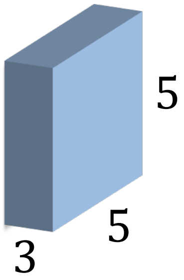
width height depth

image 32×32×3

Depth dimension *must* match;
i.e., filter extends the full depth of the input



filter 5×5×3



Convolve filter with image
i.e., ‘slide’ over it and:

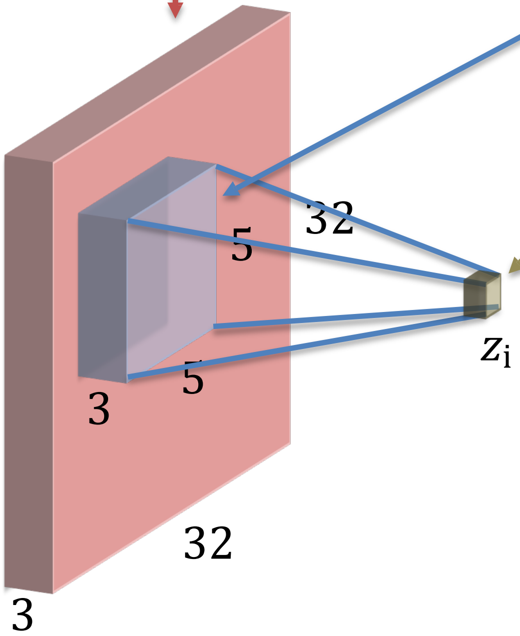
- apply filter at each location
- dot products

Images have depth: e.g. RGB -> 3 channels

Convolutions on RGB Images

32×32×3 image (pixels $\boldsymbol{X}$)

5×5×3 filter (weights vector $\boldsymbol{w}$)



1 number at a time:

equal to dot product between filter weights $\boldsymbol{w}$ and $\boldsymbol{x_i} - th$ chunk of the image. Here: $5 \times 5 \times 3 = 75$-dim dot product + bias

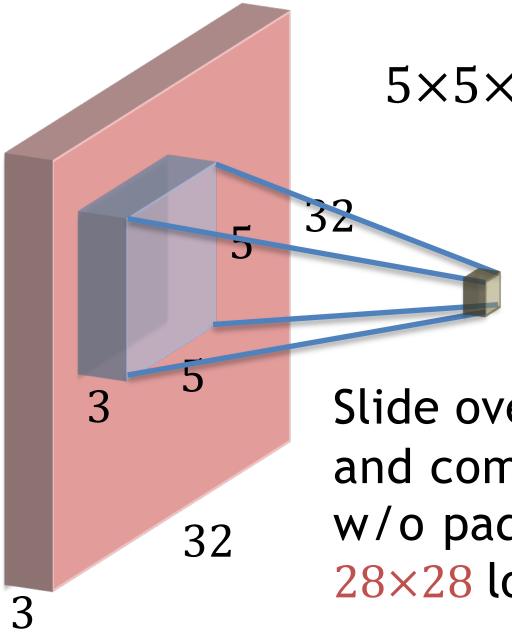
$$\boxed{z_i = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b}$$

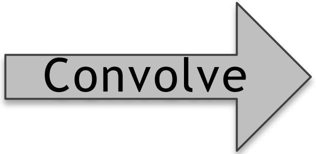
(5×5×3)×1 (5×5×3)×1 1

Convolutions on RGB Images

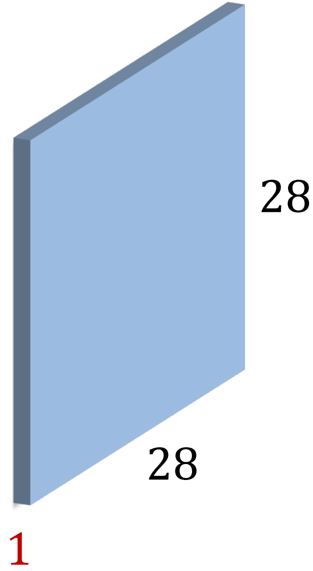
32×32×3 image

5×5×3 filter





Activation map
(also feature map)

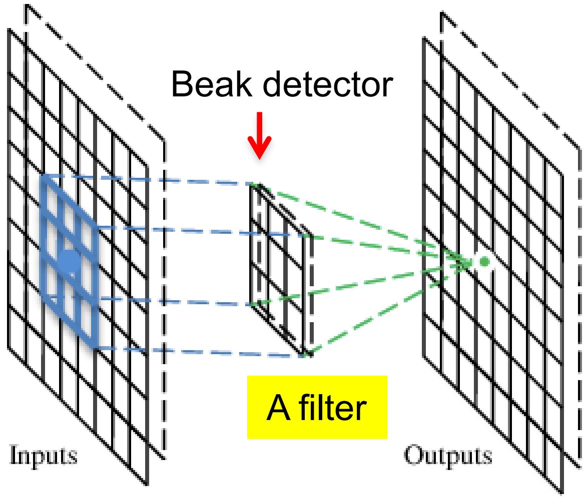


Slide over all spatial locations x_i
and compute all output z_i;
w/o padding, there are
28×28 locations

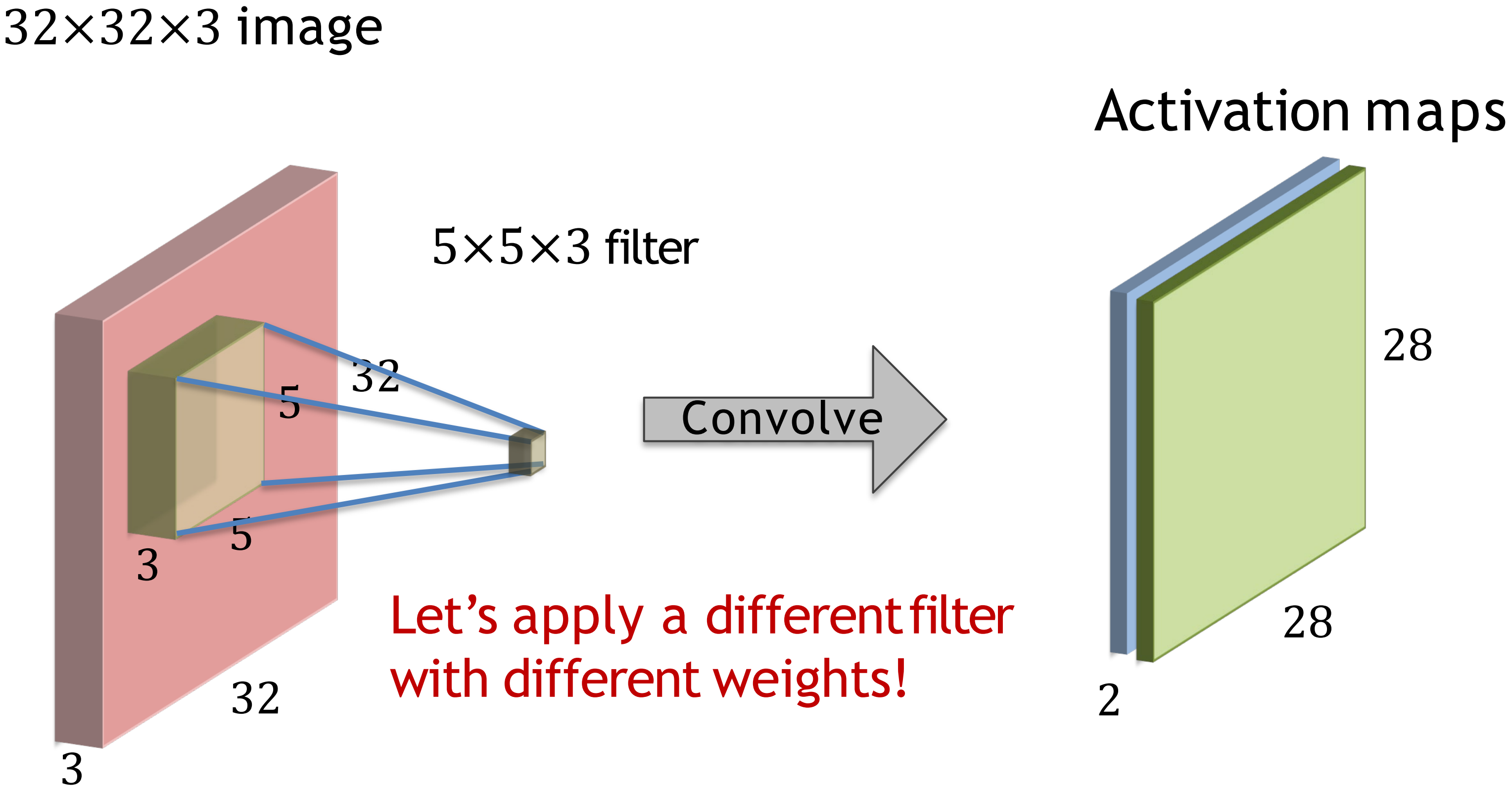
Convolution Layer

A convolutional layer

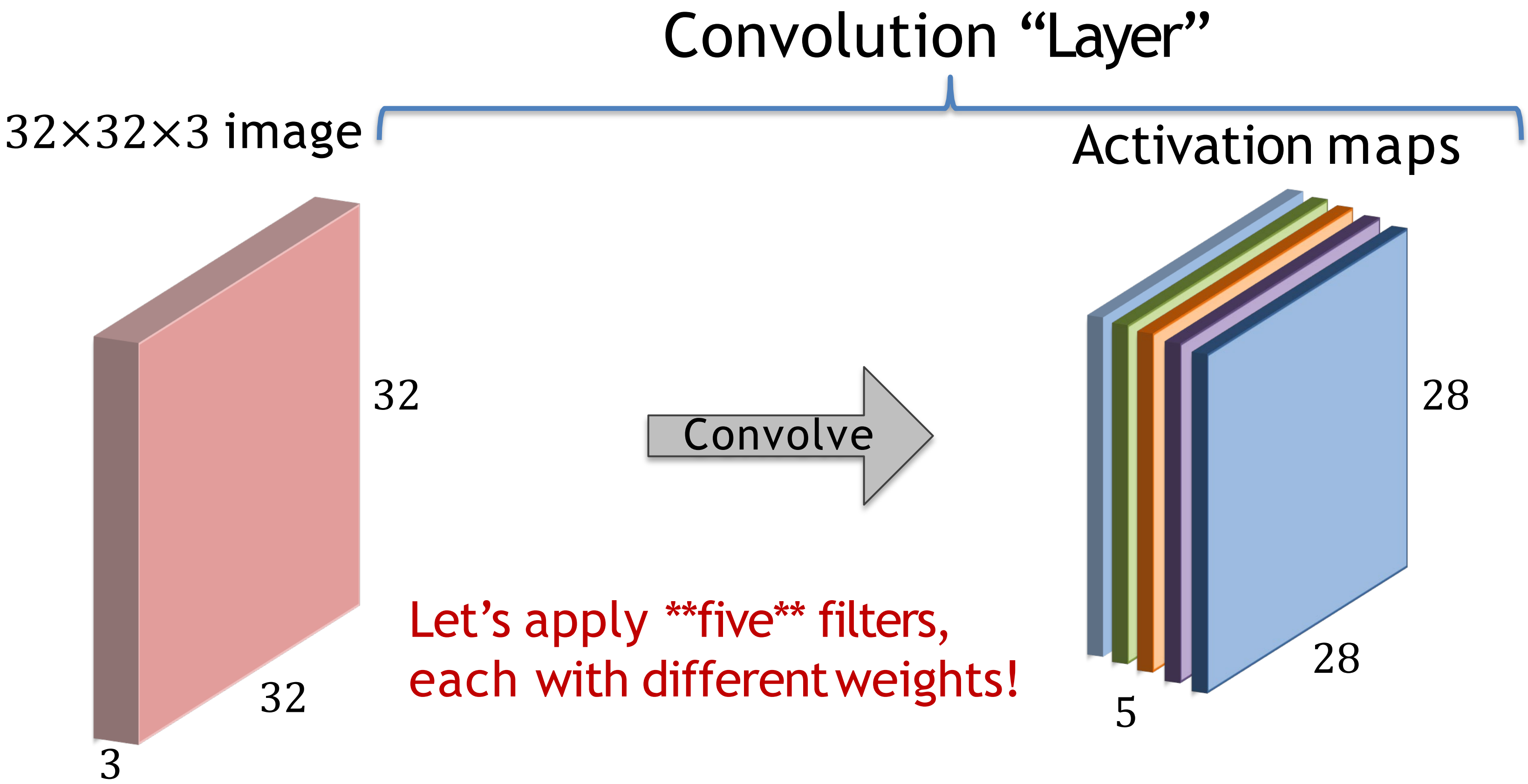
A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



Convolution Layer



Convolution Layer



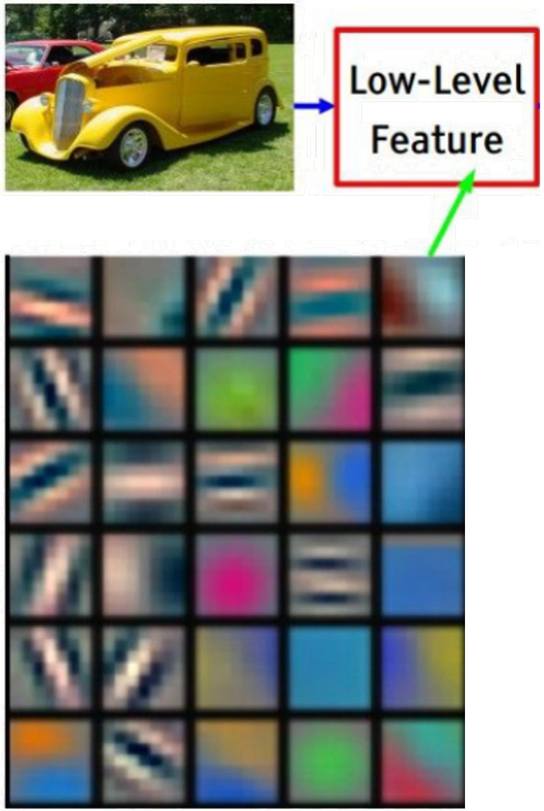
Let's apply **five** filters, each with different weights!

Convolution Layer

- A basic layer is defined by
  - Filter width and height (depth is implicitly given)
  - Number of different filter banks (#weight sets)

- Each filter captures a different image characteristic

Different Filters

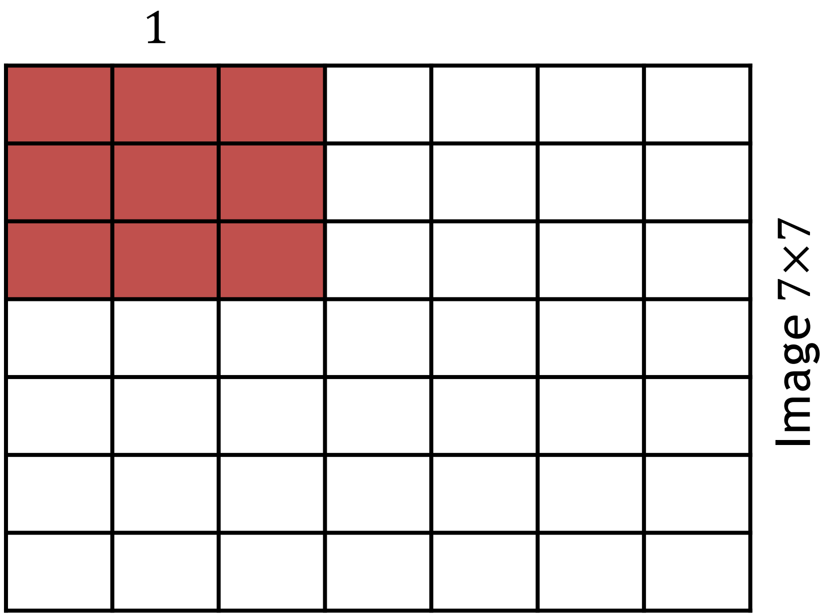


- Each filter captures different image characteristics:
  - Horizontal edges
  - Vertical edges
  - Circles
  - Squares
  - …

[Zeiler & Fergus, ECCV'14] Visualizing and Understanding Convolutional Networks

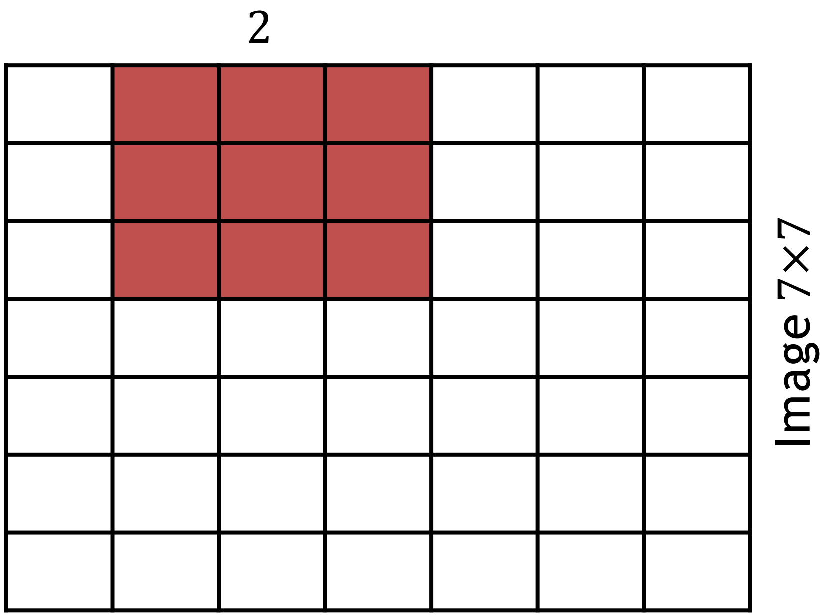
Dimensions of a Convolution Layer

Convolution Layers: Dimensions



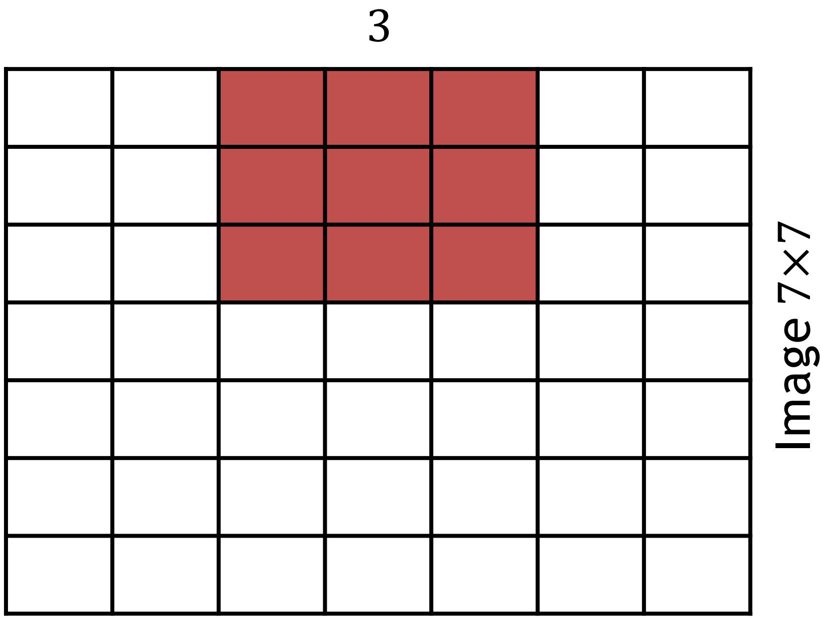
Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Dimensions



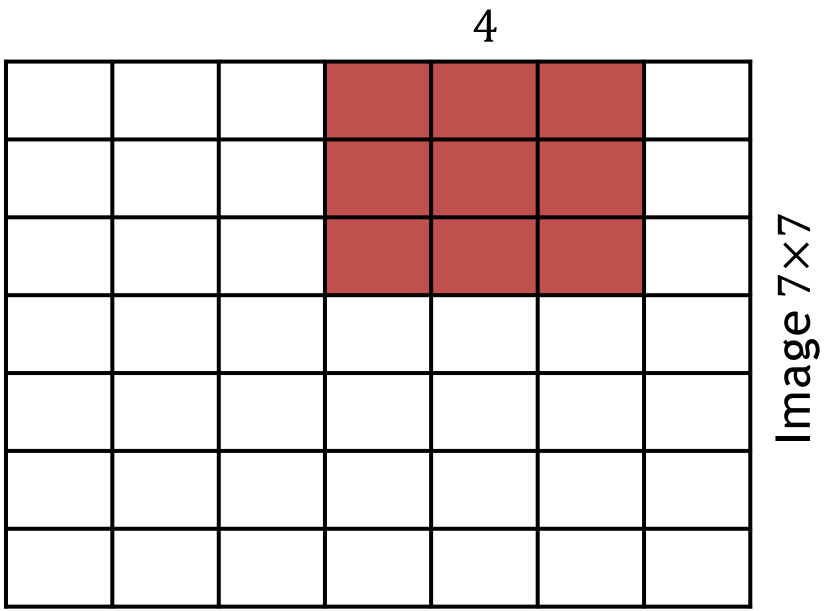
Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Dimensions



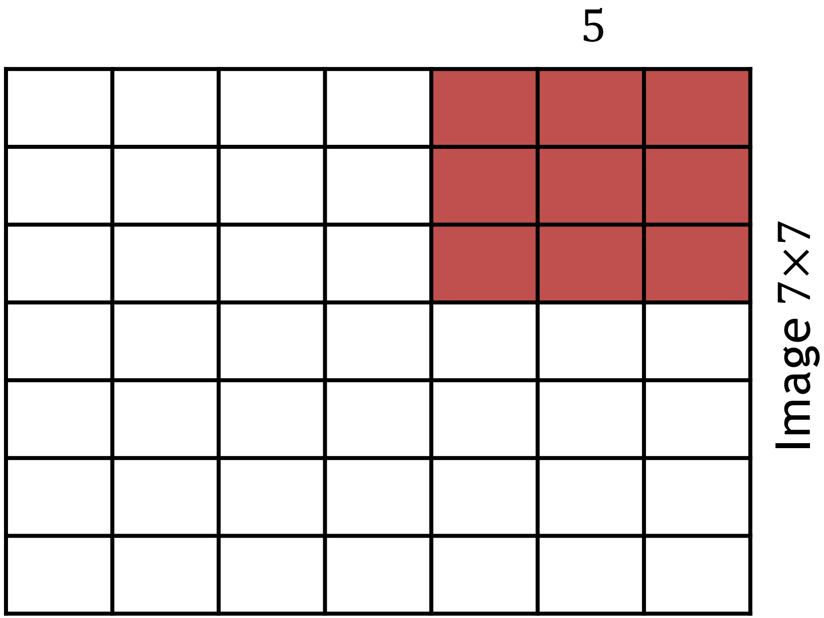
Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Dimensions



Input: 7×7
Filter: 3×3
Output: 5×5

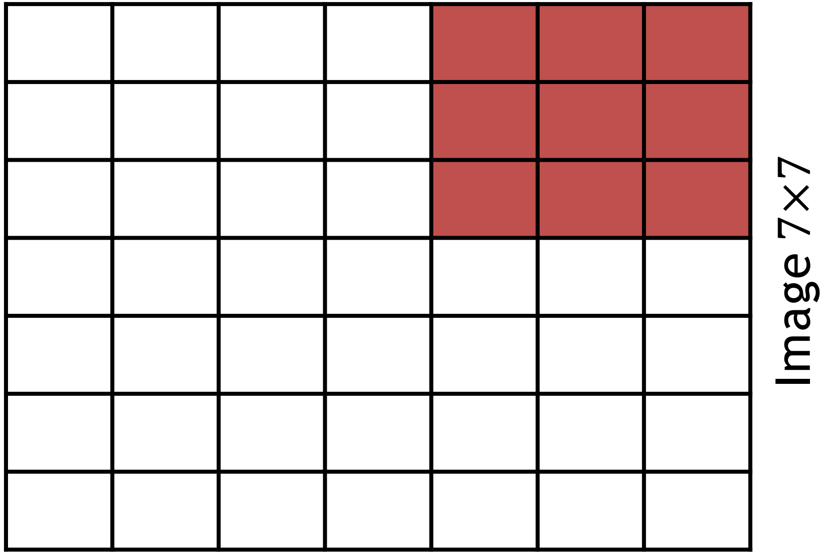
Convolution Layers: Dimensions



Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Stride

With a stride of 1

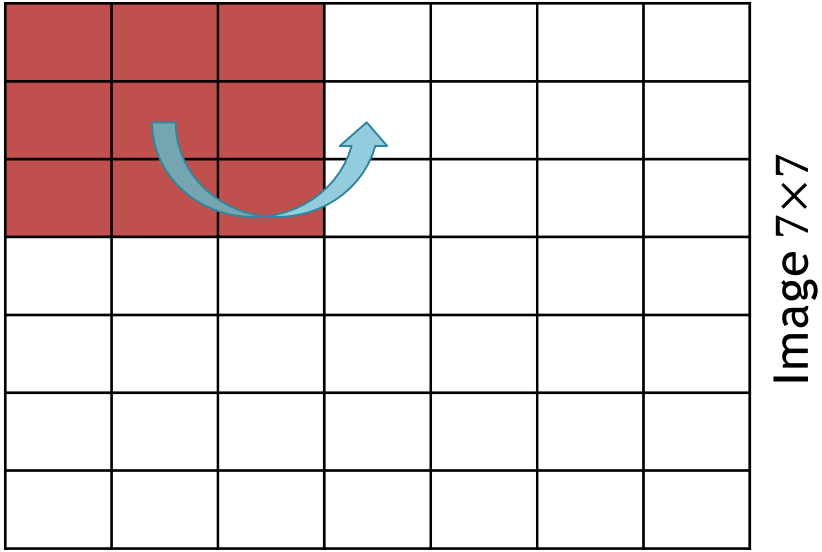


Input:	7×7
Filter:	3×3
Stride:	1
Output:	5×5

Stride of S: apply filter every S-th spatial location; i.e. subsample the image

Convolution Layers: Stride

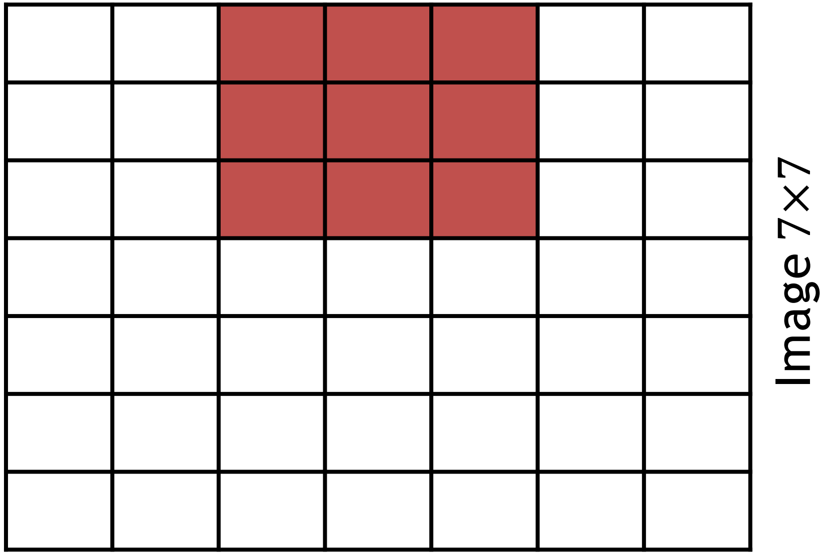
With a stride of 2



Input:	7×7
Filter:	3×3
Stride:	2
Output:	3×3

Convolution Layers: Stride

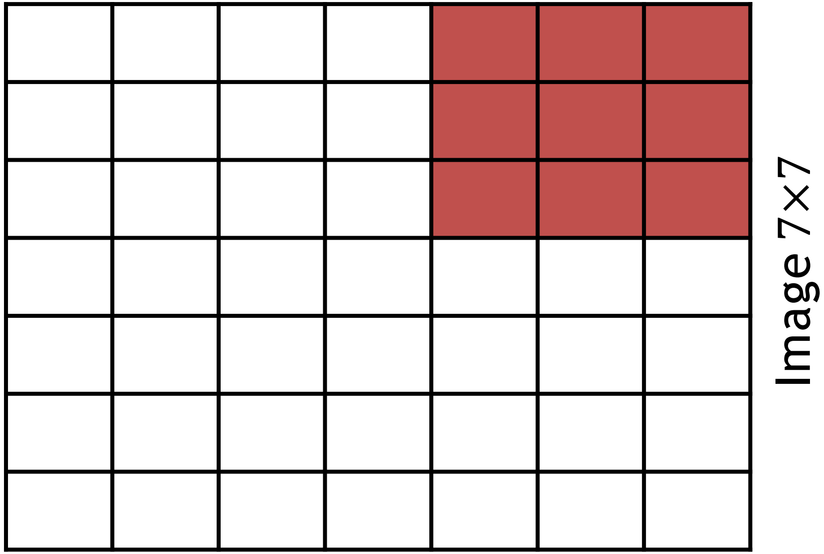
With a stride of 2



Input: 7×7
Filter: 3×3
Stride: 2
Output: 3×3

Convolution Layers: Stride

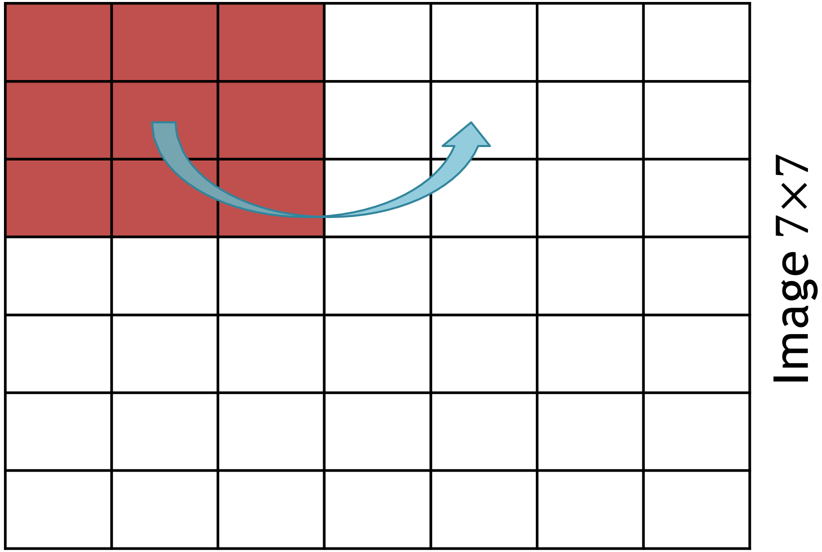
With a stride of 2



Input: 7×7
Filter: 3×3
Stride: 2
Output: 3×3

Convolution Layers: Stride

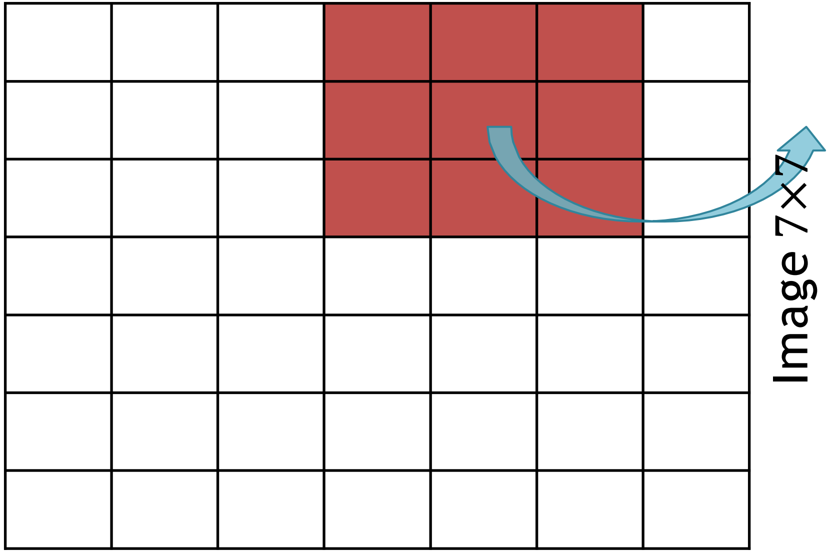
With a stride of 3



Input: 7×7
Filter: 3×3
Stride: 3
Output: $? \times ?$

Convolution Layers: Stride

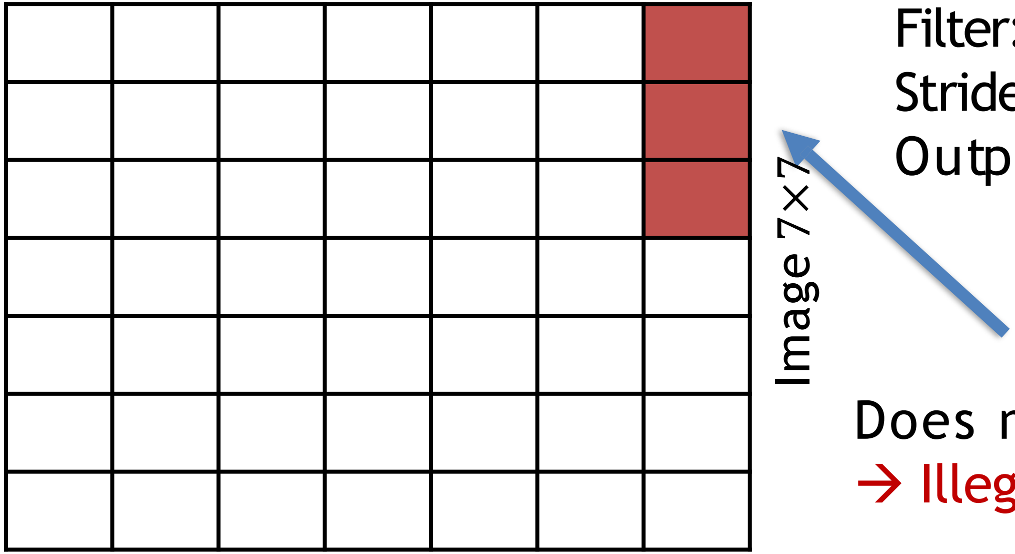
With a stride of 3



Input:	7×7
Filter:	3×3
Stride:	3
Output:	$? \times ?$

Convolution Layers: Stride

With a stride of 3

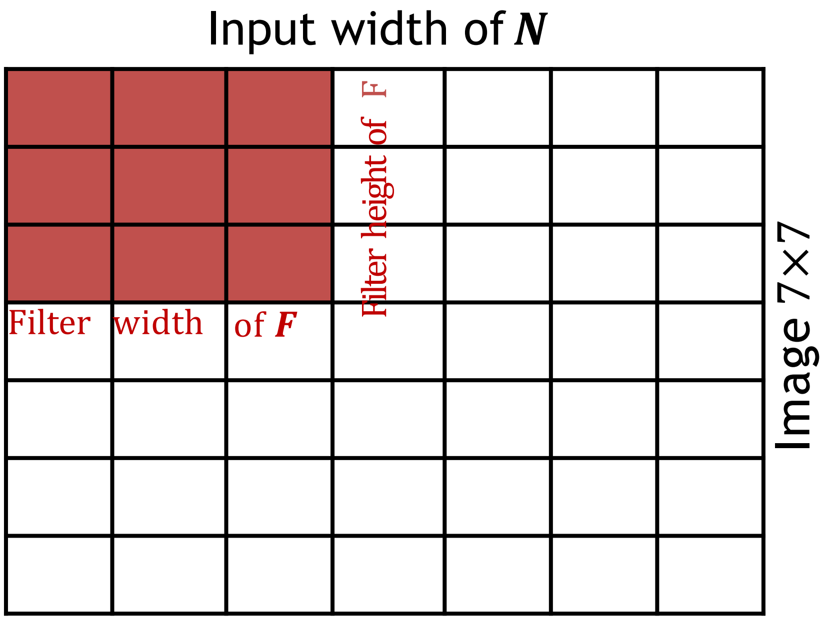


Input:	7×7
Filter:	3×3
Stride:	3
Output:	$? \times ?$

Does not really fit (remainder left)

→ Illegal stride for input & filter size!

Convolution Layers: Dimensions



Input: $N \times N$

Filter: $F \times F$

Stride: S

Output: $\left(\frac{N-F}{S} + 1\right) \times \left(\frac{N-F}{S} + 1\right)$

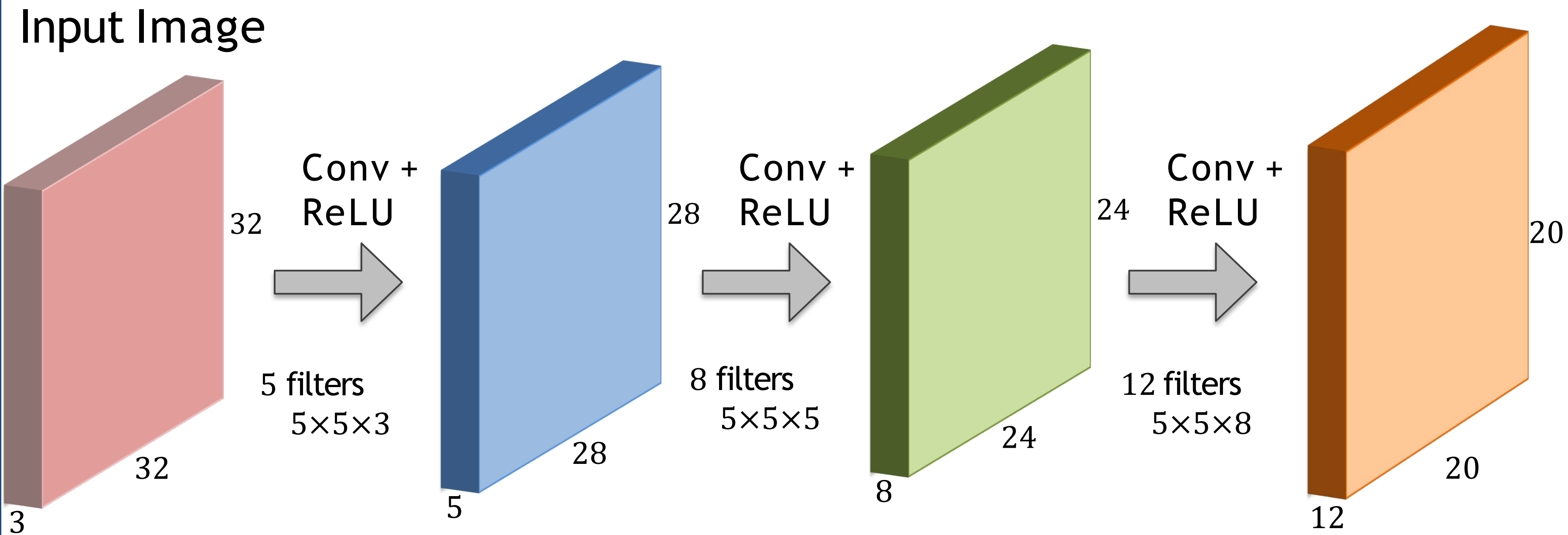
$$N = 7, F = 3, S = 1: \frac{7-3}{1} + 1 = 5$$

$$N = 7, F = 3, S = 2: \frac{7-3}{2} + 1 = 3$$

$$N = 7, F = 3, S = 3: \frac{7-3}{3} + 1 = 2.3$$

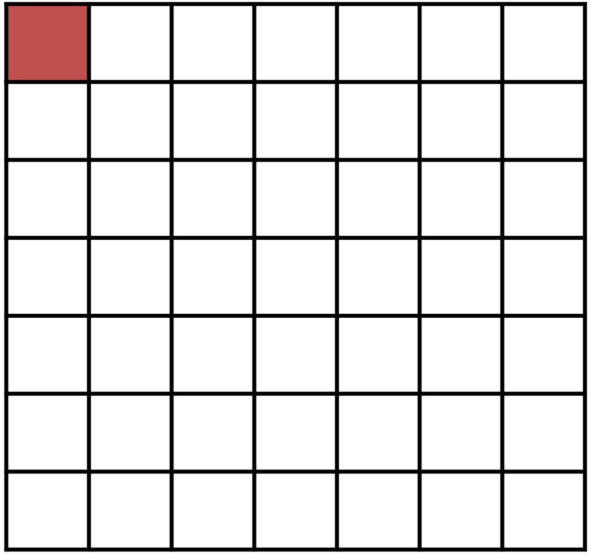
Fractions are illegal

Convolution Layers: Dimensions



Shrinking down so quickly (32→28→24→20) is typically not a good idea…

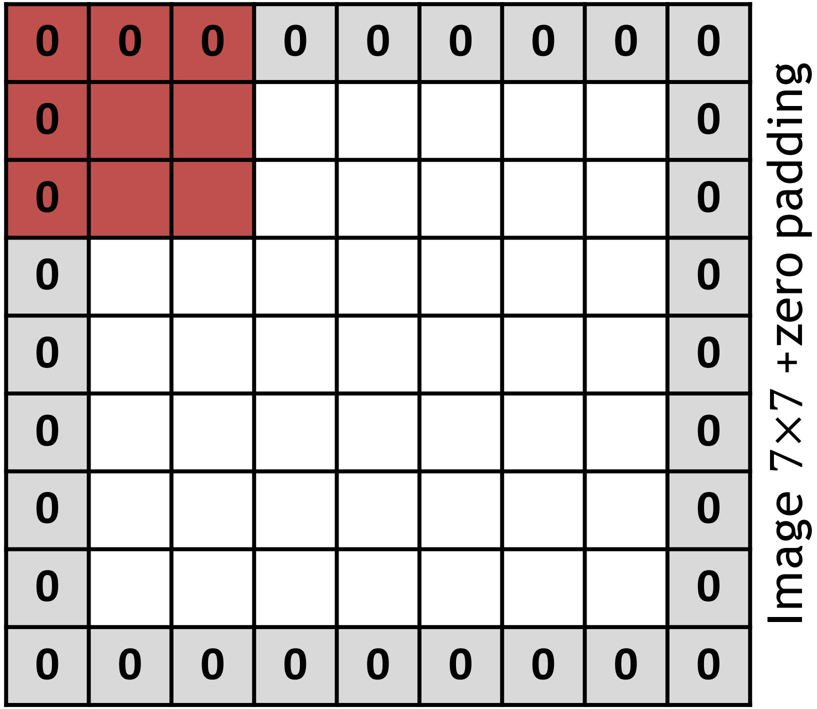
Convolution Layers: Padding



Why padding?

- Sizes get small too quickly
- Corner pixel is only used once

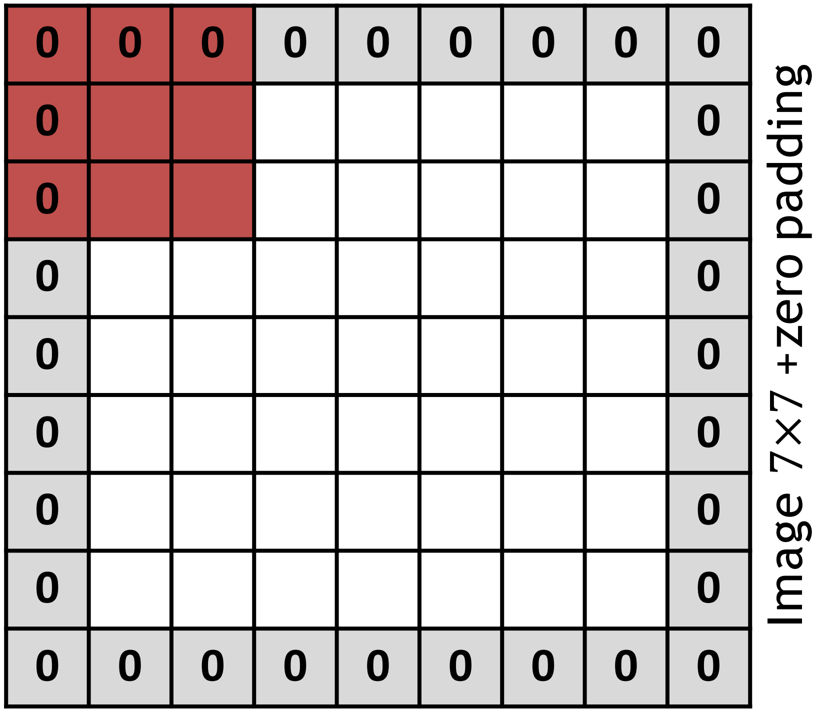
Convolution Layers: Padding



Why padding?

- Sizes get small too quickly
- Corner pixel is only used once

Convolution Layers: Padding



Input ($N \times N$): 7×7
Filter ($F \times F$): 3×3
Padding (P): 1
Stride (S): 1
Output 7×7

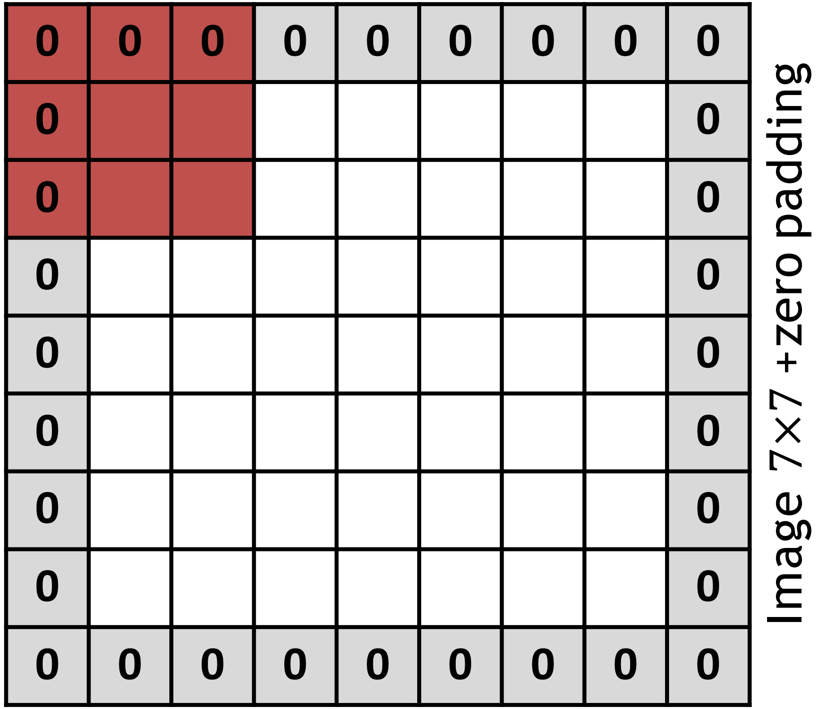
Most common is '*zero*' padding

Output Size:

$$\left(\left\lfloor \frac{N + 2 * P - F}{S} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{N + 2 * P - F}{S} \right\rfloor + 1\right)$$

$\lfloor \quad \rfloor$ denotes the floor operator (as in practice an integer division is performed)

Convolution Layers: Padding



Types of convolutions:

- Valid convolution: using no padding
- Same convolution: output=input size

  Set padding to $P = \frac{F\text{-}1}{2}$

Convolution Layers: Dimensions

Example

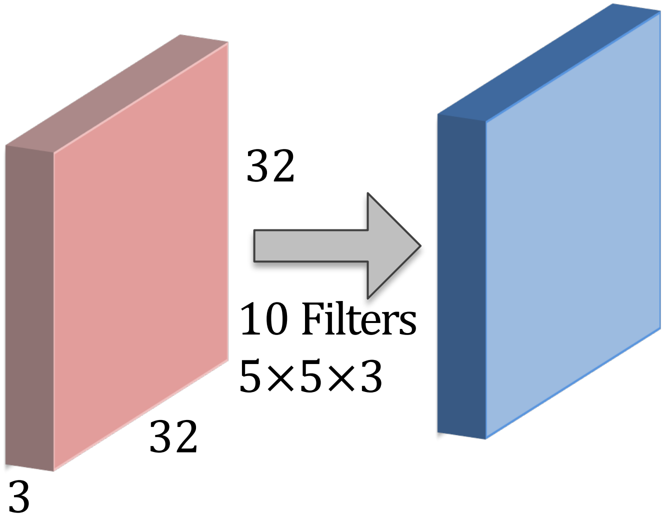
Input image: $32 \times 32 \times 3$

10 filters 5×5

Stride 1

Pad 2

Depth of 3 is implicitly given



Output size is:

$$\frac{32 + 2 \cdot 2 - 5}{1} + 1 = 32$$

i.e. $32 \times 32 \times 10$

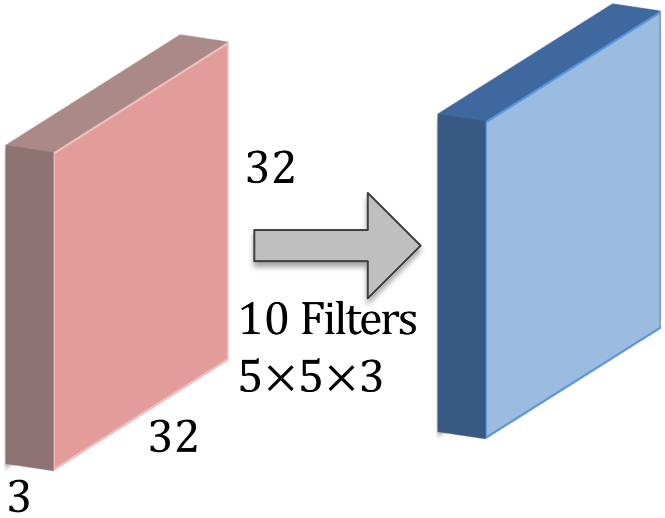
Remember

Output: $\left(\left\lfloor \frac{\mathrm{N} + 2 * \mathrm{P} - \mathrm{F}}{\mathrm{S}} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{\mathrm{N} + 2 * \mathrm{P} - \mathrm{F}}{\mathrm{S}} \right\rfloor + 1\right)$

Convolution Layers: Dimensions

Example

Input image: $32 \times 32 \times 3$
10 filters 5×5
Stride 1
Pad 2



Output size is:

$$\frac{32 + 2 \cdot 2 - 5}{1} + 1 = 32$$

i.e. $32 \times 32 \times 10$

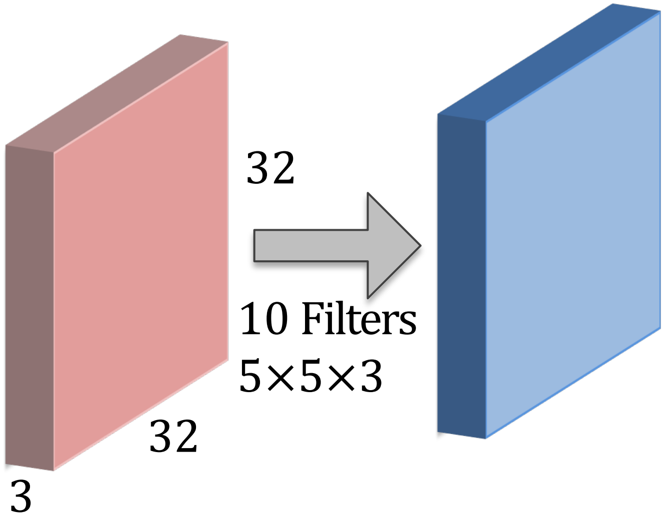
Remember

Output: $\left(\left\lfloor \frac{N + 2 * P - F}{S} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{N + 2 * P - F}{S} \right\rfloor + 1\right)$

Convolution Layers: Dimensions

Example

Input image: $32 \times 32 \times 3$
10 filters 5×5
Stride 1
Pad 2



Number of parameters (weights):
Each filter has $5 \times 5 \times 3 + 1 = 76$ params (+1 for bias)
-> $76 * 10 = 760$ parameters in layer

Example

- You are given a convolutional layer with 4 filters, kernel size 5, stride 1, and no padding that operates on an RGB image.

- Q1: What are the dimensions and the shape of its weight tensor?

  - ☐ A1: $(3, 4, 5, 5)$
  - ☐ A2: $(4, 5, 5)$
  - ☐ A3: depends on the width and height of the image

Example

- You are given a convolutional layer with 4 filters, kernel size 5, stride 1, and no padding that operates on an RGB image.

- Q1: What are the dimensions and the shape of its weight tensor?

A1: (3, 4, 5, 5)

Input channels (RGB = 3)

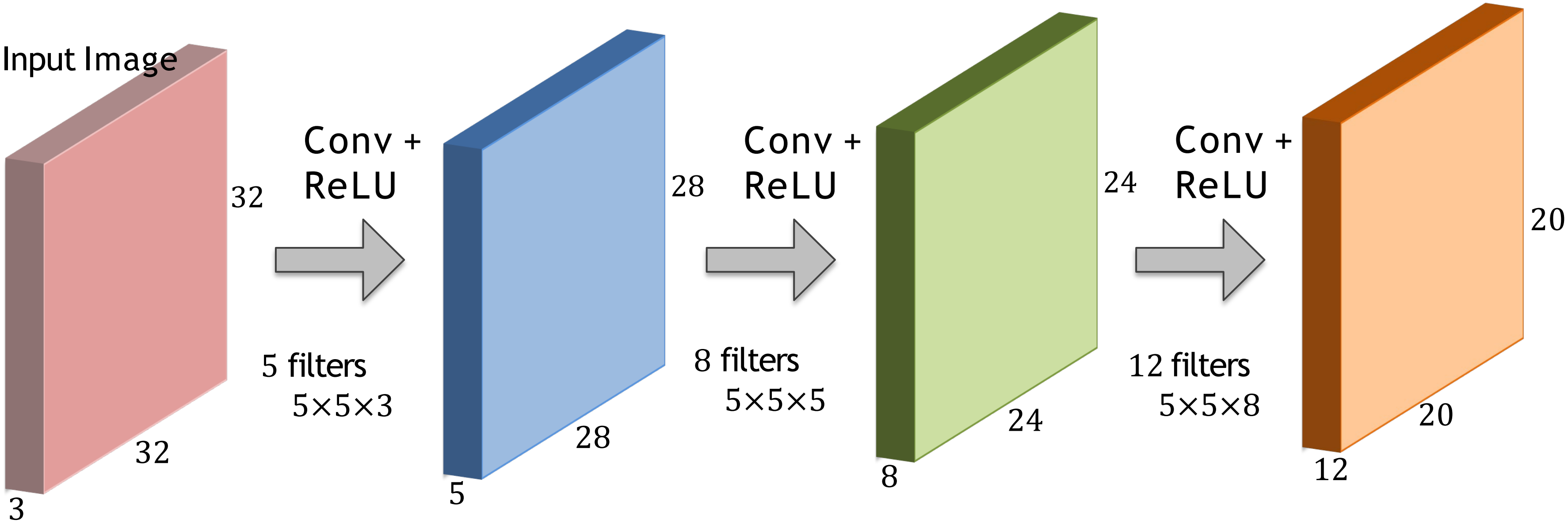
Output size = 4 filters

Filter size = 5×5

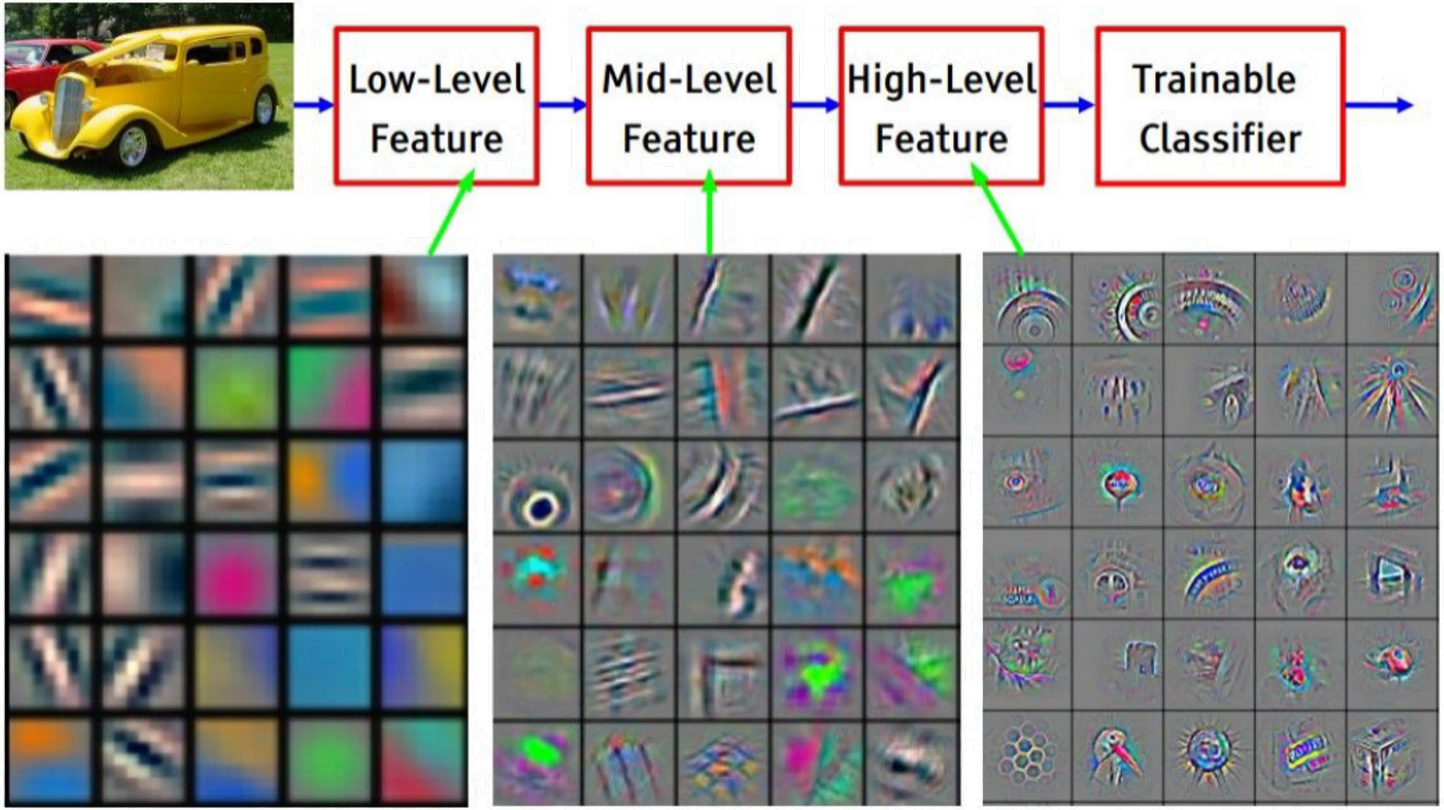
Convolutional Neural Network (CNN)

CNN Prototype

ConvNet is concatenation of Conv Layers and activations



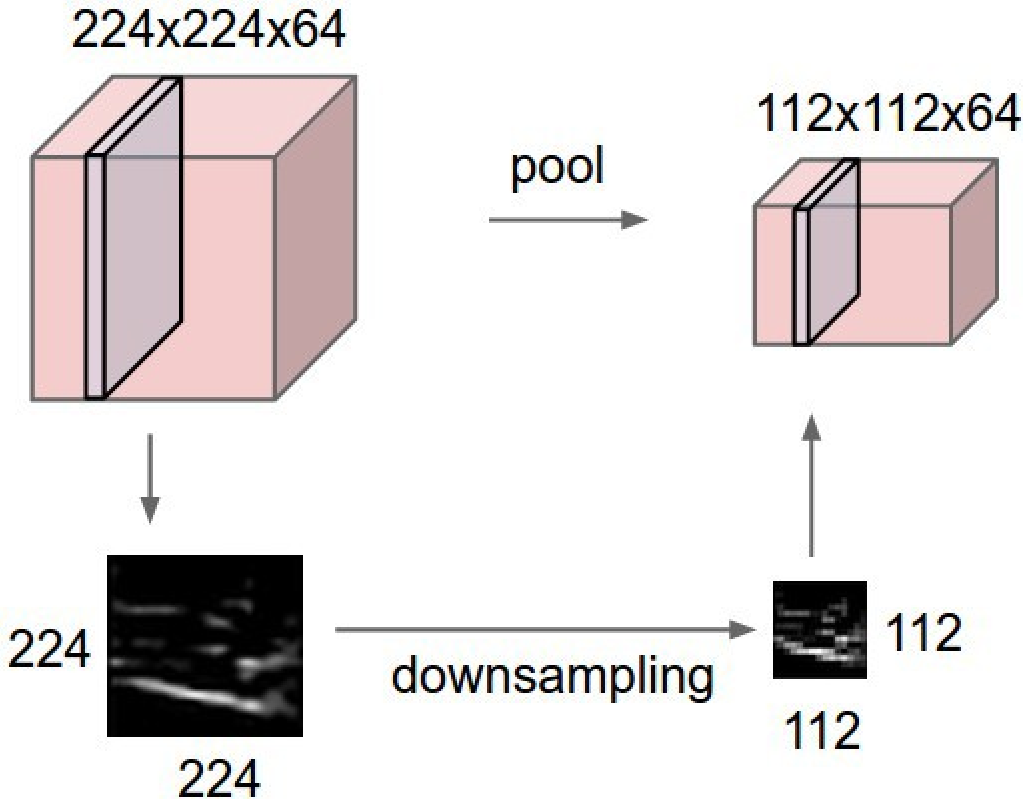
CNN Learned Filters



[Zeiler & Fergus, ECCV'14] Visualizing and Understanding Convolutional Networks

Pooling

Pooling Layer



[Li et al., CS231n Course Slides] Lecture 5: Convolutional Neural Networks

Pooling Layer: Max Pooling

Single depthslice of input

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

Max pool with
2×2 filters and stride 2

‘Pooled’ output

6	9
3	4

Pooling Layer

- Conv Layer = 'Feature Extraction'
  - Computes a feature in a given region
- Pooling Layer = 'Feature Selection'
  - Picks the strongest activation in a region

Pooling Layer

- Input is a volume of size $W_{\text{in}} \times H_{\text{in}} \times D_{\text{in}}$
- Two hyperparameters
  - Spatial filter extent F
  - Stride S

  Filter count K and padding P make no sense here
- Output volume is of size $W_{\text{out}} \times H_{\text{out}} \times D_{\text{out}}$
  - $W_{\text{out}} = \dfrac{W_{\text{in}} - \text{F}}{\text{S}} + 1$
  - $H_{\text{out}} = \dfrac{H_{\text{in}} - \text{F}}{\text{S}} + 1$
  - $D_{\text{out}} = D_{\text{in}}$
- Does not contain parameters; e.g. it's fixed function

Pooling Layer

Common settings:
$F = 2, S = 2$
$F = 3, S = 2$

- Input is a volume of size $W_{in} \times H_{in} \times D_{in}$
- Two hyperparameters
  - Spatial filter extent F
  - Stride S

Filter count K and padding P make no sense here

- Output volume is of size $W_{out} \times H_{out} \times D_{out}$
  - $W_{out} = \dfrac{W_{in} - F}{S} + 1$
  - $H_{out} = \dfrac{H_{in} - F}{S} + 1$
  - $D_{out} = D_{in}$
- Does not contain parameters; e.g. it's fixed function

Pooling Layer: Average Pooling

Single depthslice of input

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

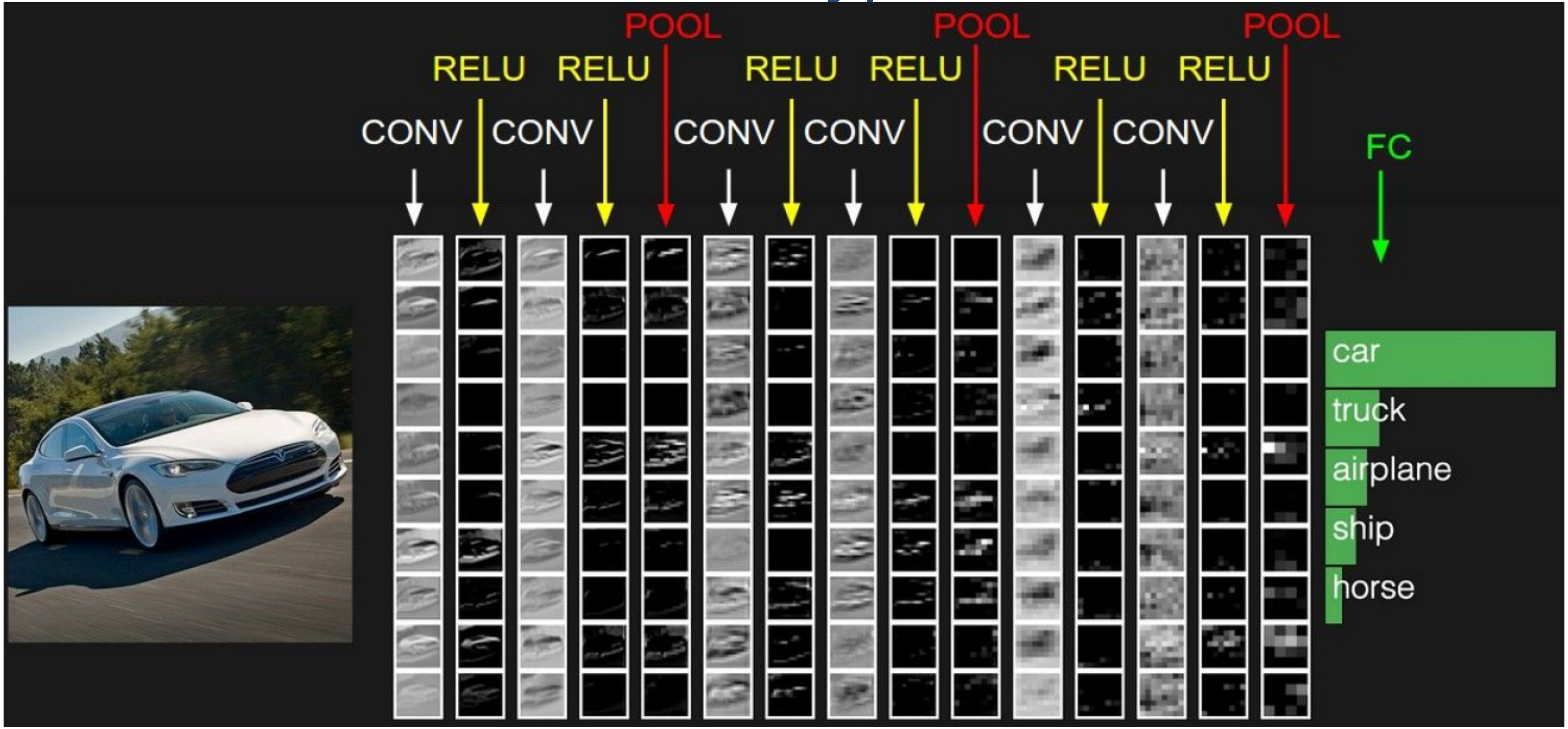
Average pool with 2×2 filters and stride 2

'Pooled' output

2.5	6
1.75	3

- Typically used deeper in the network

CNN Prototype



Final Fully-Connected Layer

- Same as what we had in ‘ordinary’ neural networks
  - Make the final decision with the extracted features from the convolutions
  - One or two FC layers typically

Convolutions vs Fully-Connected

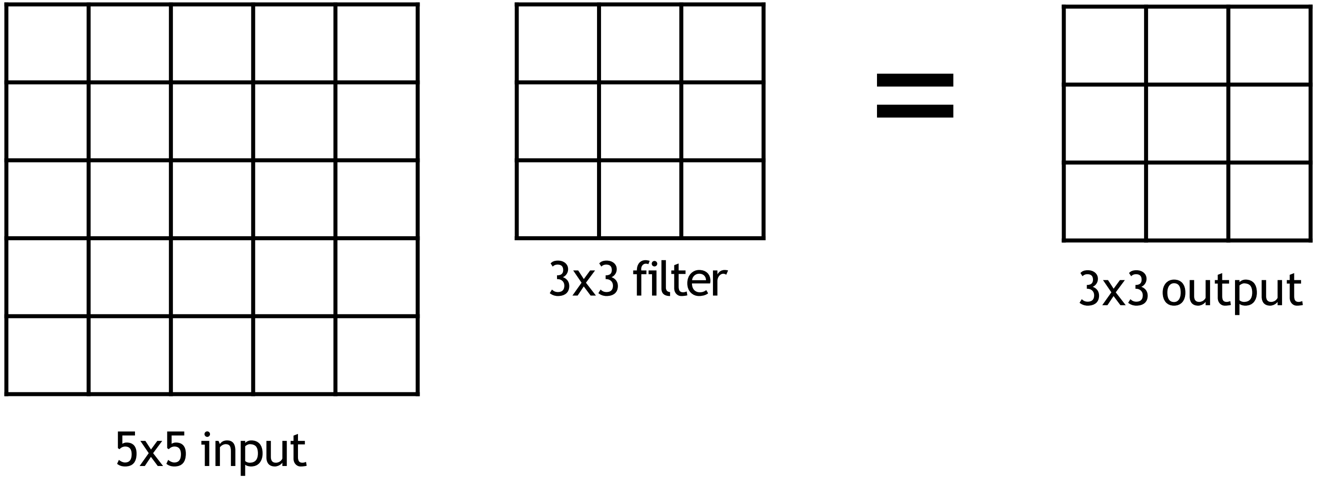
- In contrast to fully-connected layers, we want to restrict the degrees of freedom
  - FC is somewhat brute force
  - Convolutions are structured

- Sliding window to with the same filter parameters to extract image features
  - Concept of weight sharing
  - Extract same features independent of location

Receptive field

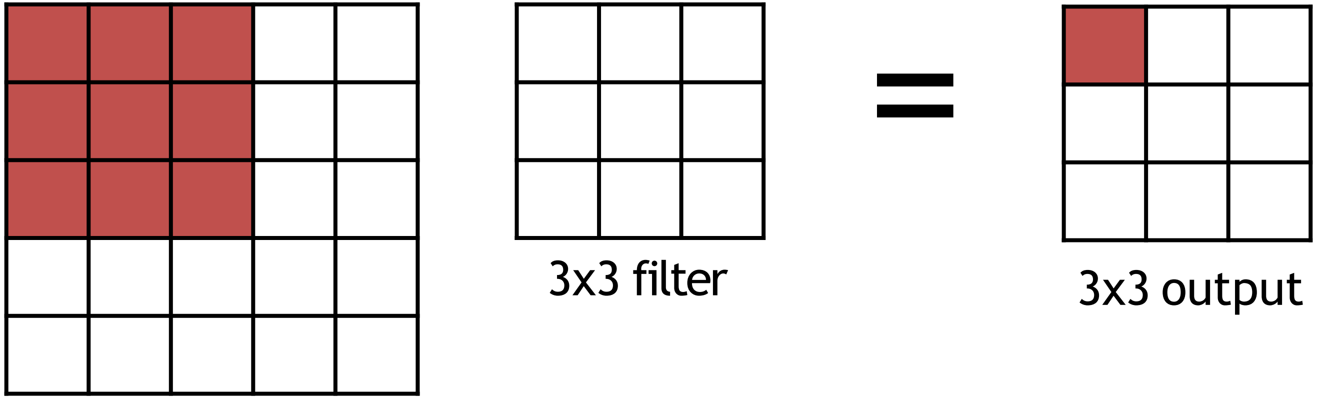
Receptive Field

- Spatial extent of the connectivity of a convolutional filter



Receptive Field

- Spatial extent of the connectivity of a convolutional filter

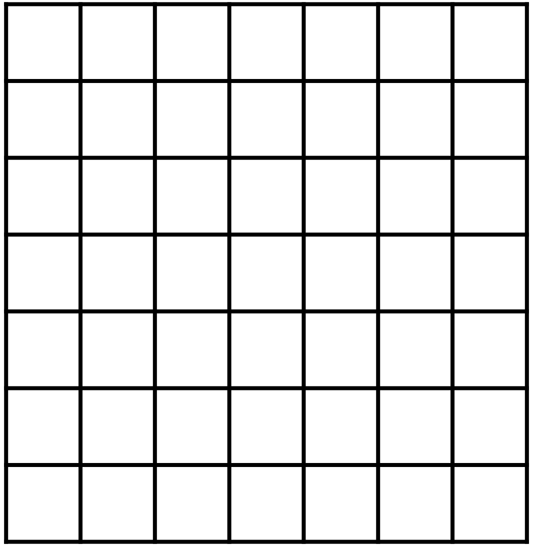


5x5 input

3x3 receptive field = 1 output pixel is connected to 9 input pixels

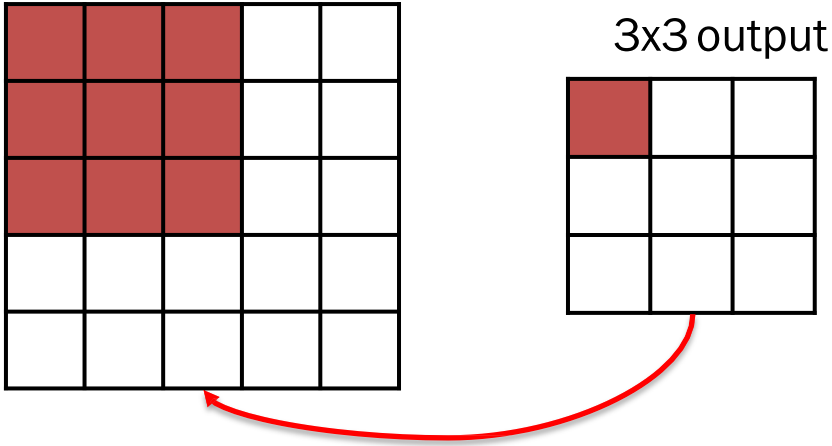
Receptive Field

- Spatial extent of the connectivity of aconvolutional filter



7x7 input

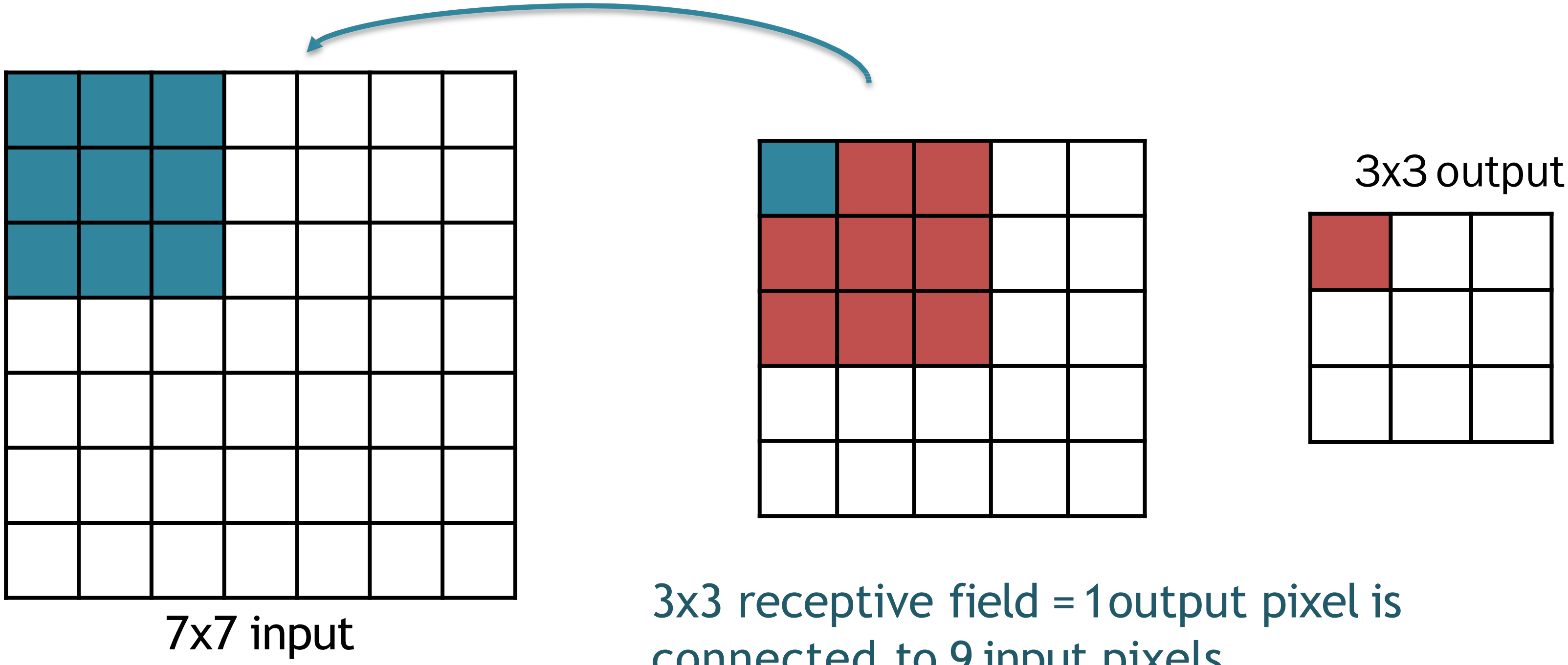
3x3 output



3x3 receptive field = 1output pixel is connected to 9 input pixels

Receptive Field

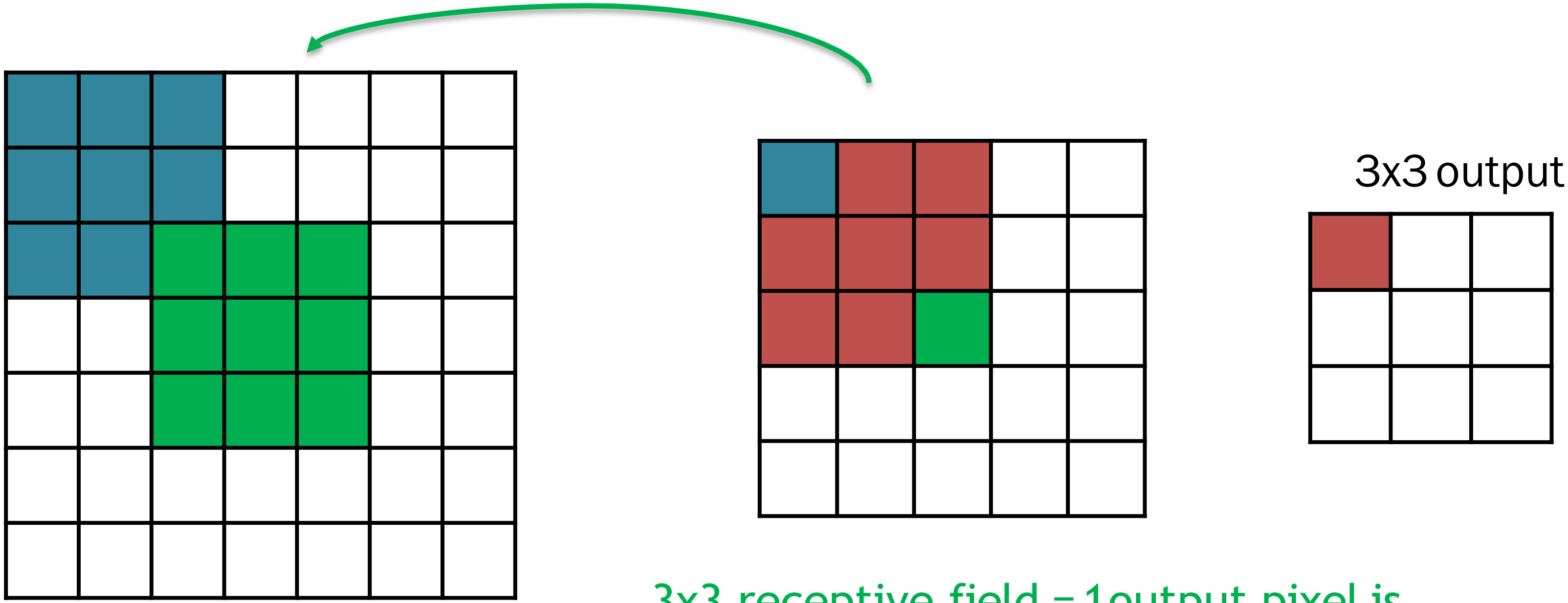
- Spatial extent of the connectivity of aconvolutional filter



3x3 receptive field = 1output pixel is connected to 9 input pixels

Receptive Field

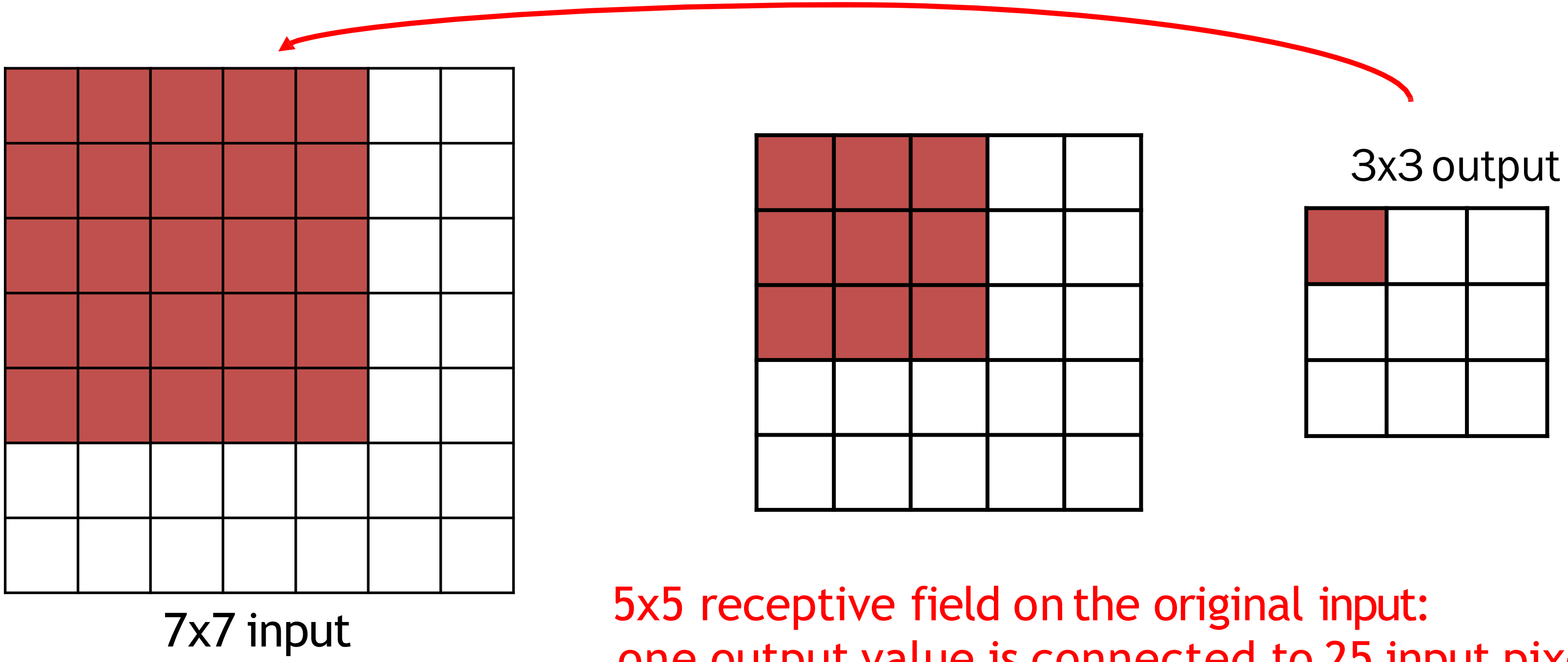
- Spatial extent of the connectivity of aconvolutional filter



3x3 receptive field = 1output pixel is connected to 9 input pixels

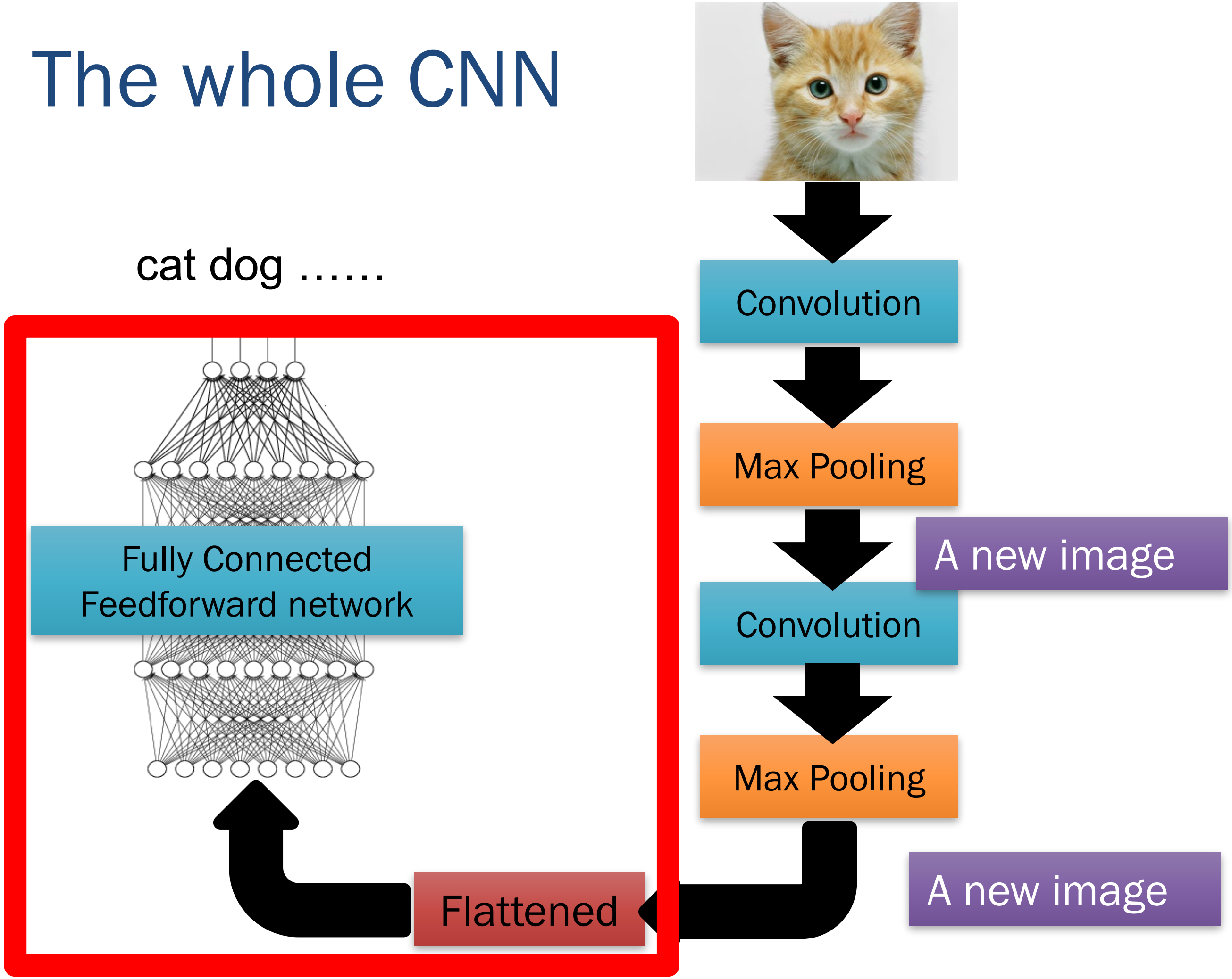
Receptive Field

- Spatial extent of the connectivity of aconvolutional filter

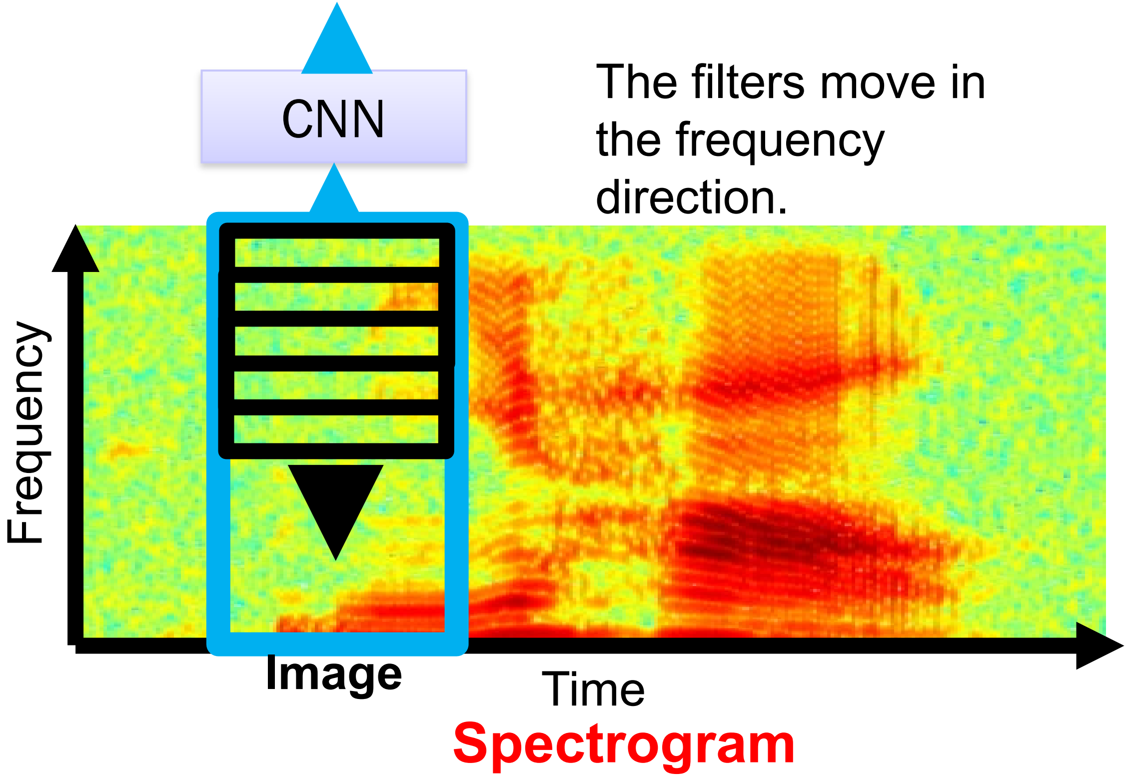


5x5 receptive field on the original input:
one output value is connected to 25 input pixels

The whole CNN

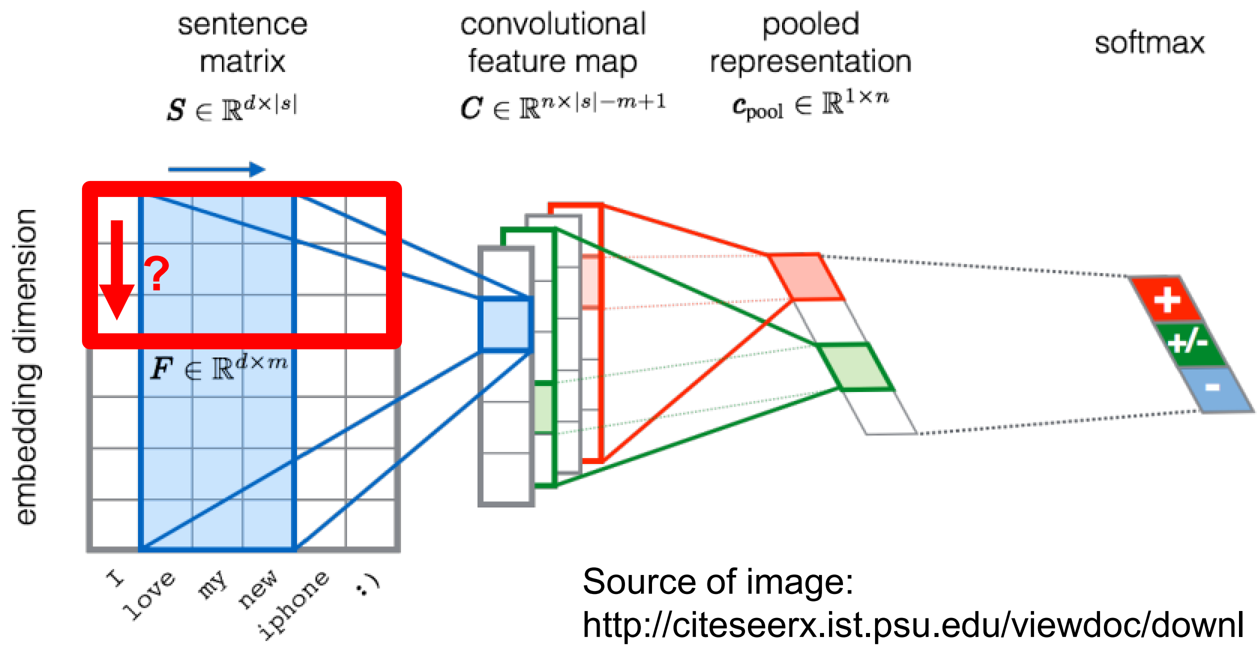


CNN in speech recognition



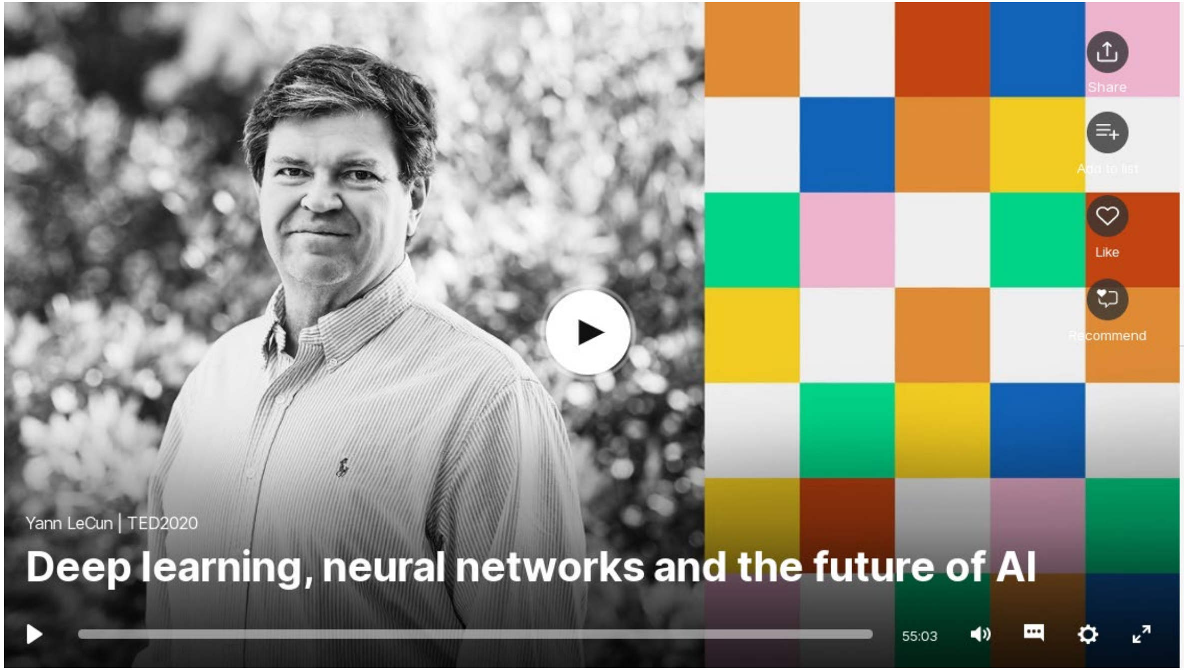
The filters move in the frequency direction.

CNN in text classification



Source of image: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.6858&rep=rep1&type=pdf

A lot of buzz about Deep Learning



Yann LeCun, the chief AI scientist at Facebook, helped develop the deep learning algorithms

References

- Goodfellow et al. “Deep Learning” (2016),
  - Chapter 9: Convolutional Networks
- http://cs231n.github.io/convolutional-networks/

Acknowledgments

Most slides adapted from Visual Computing Group