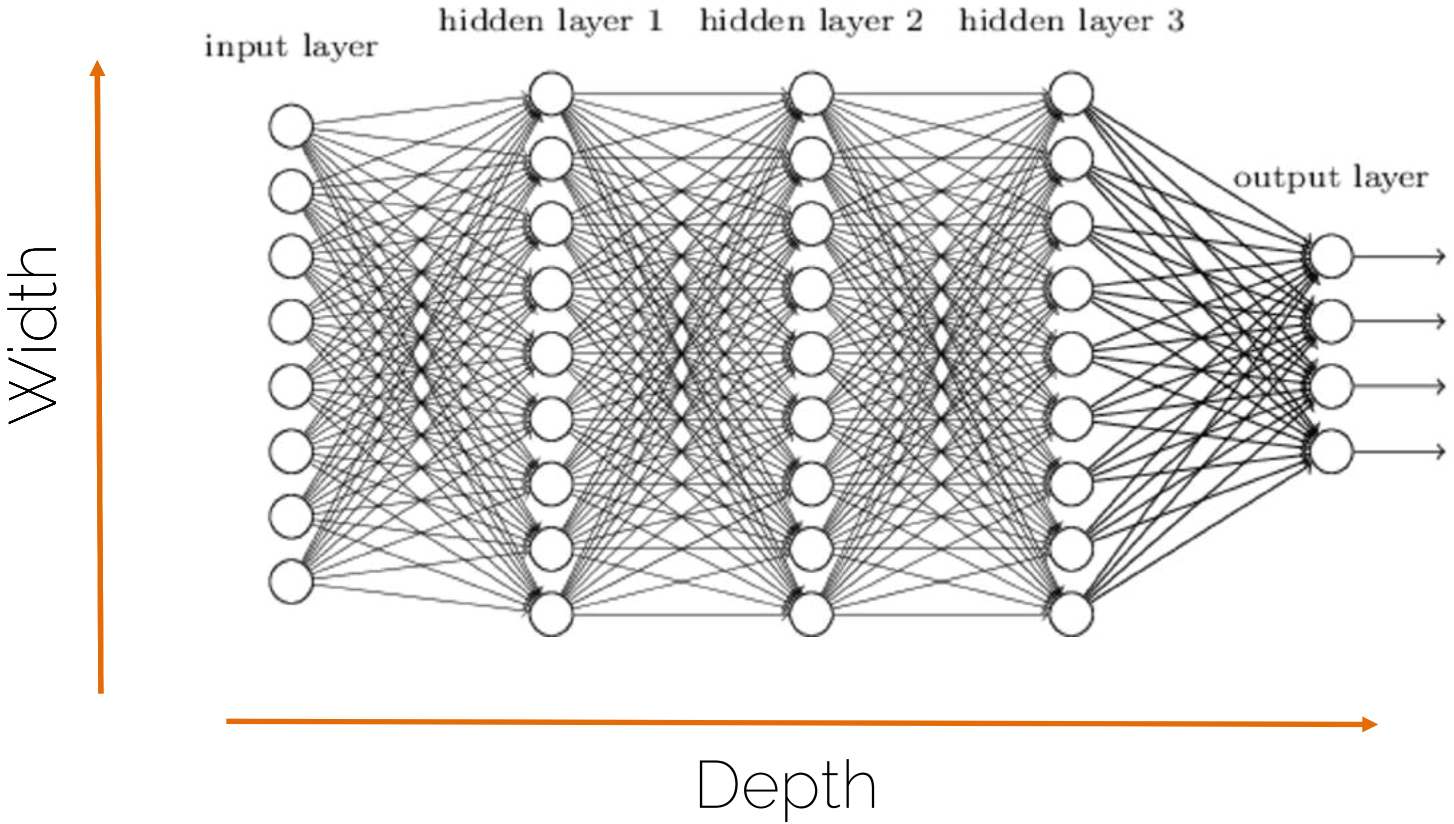
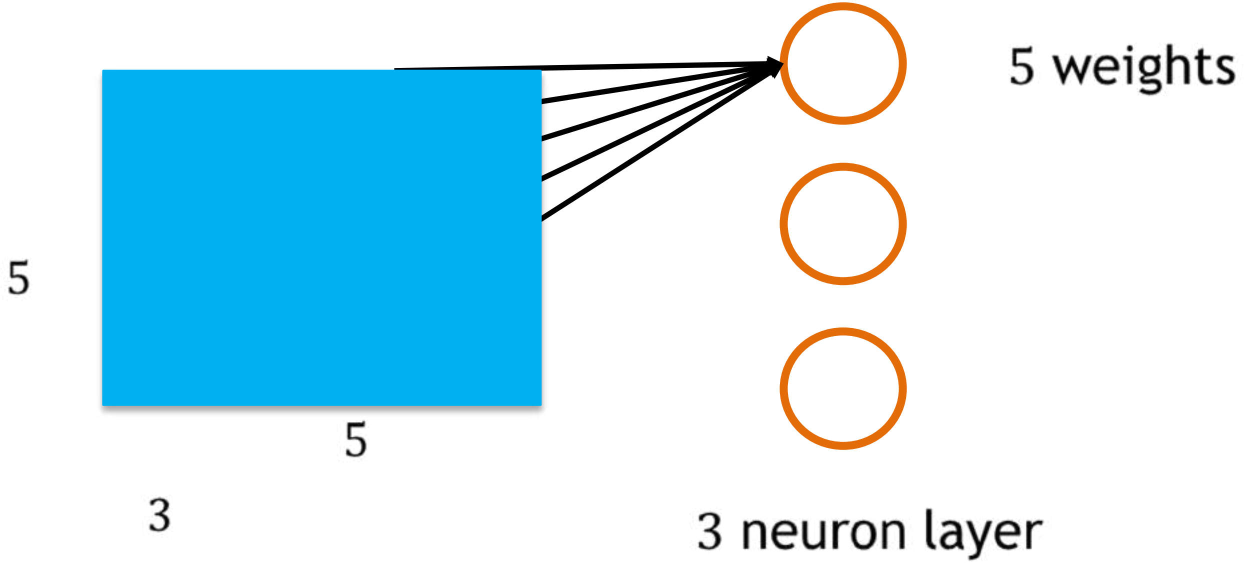
Lecture 9 - Convolutional Neural Networks

Fully Connected Neural Network



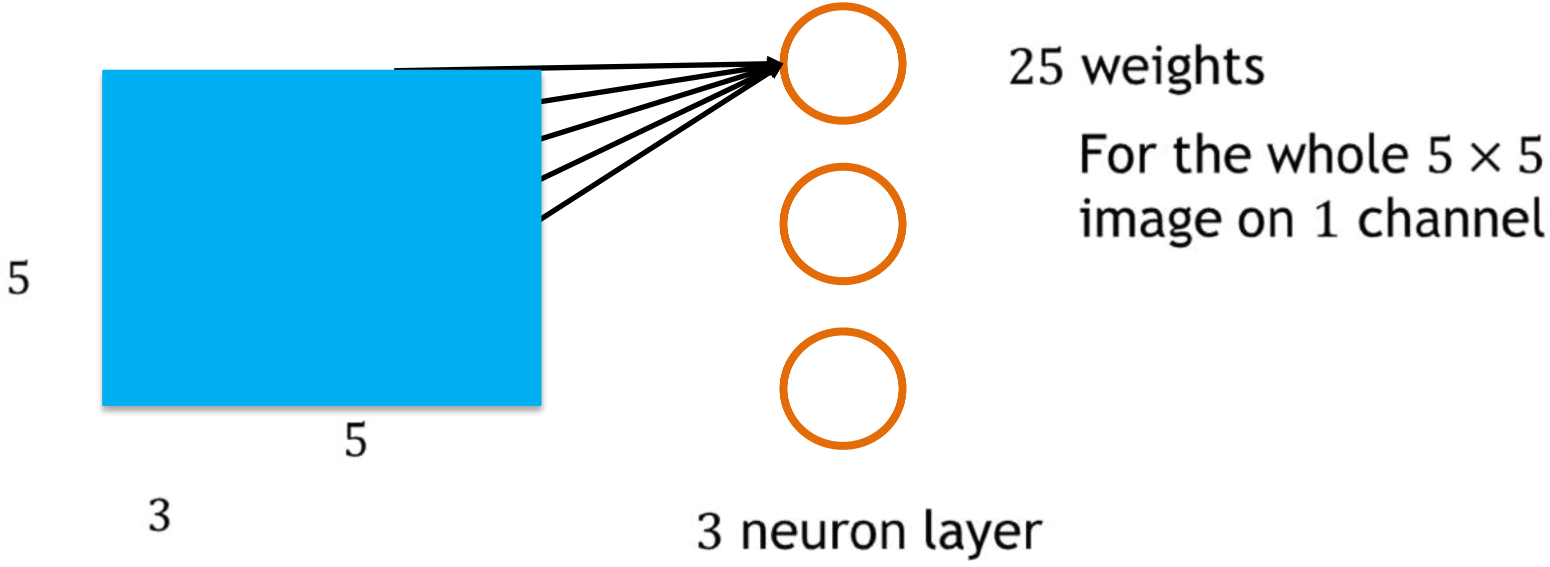
Problems using FC Layers on Images

- How to process a tiny image with FC layers



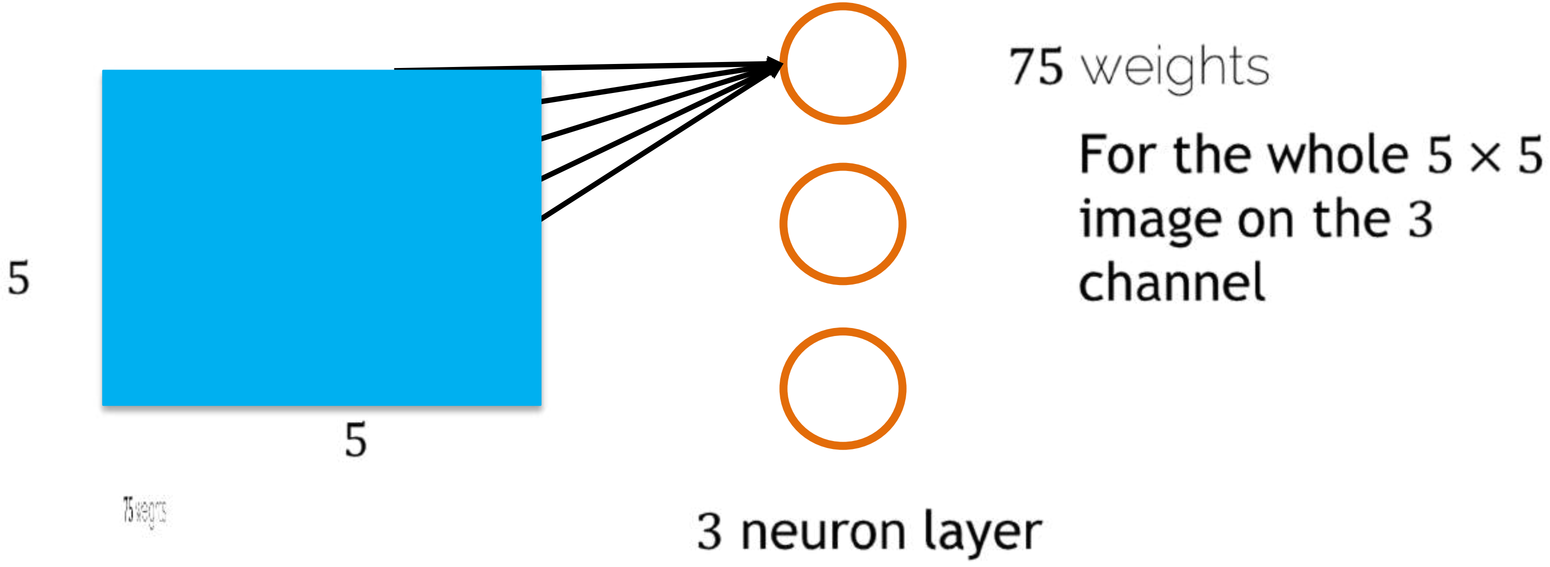
Problems using FC Layers on Images

- How to process a tiny image with FC layers



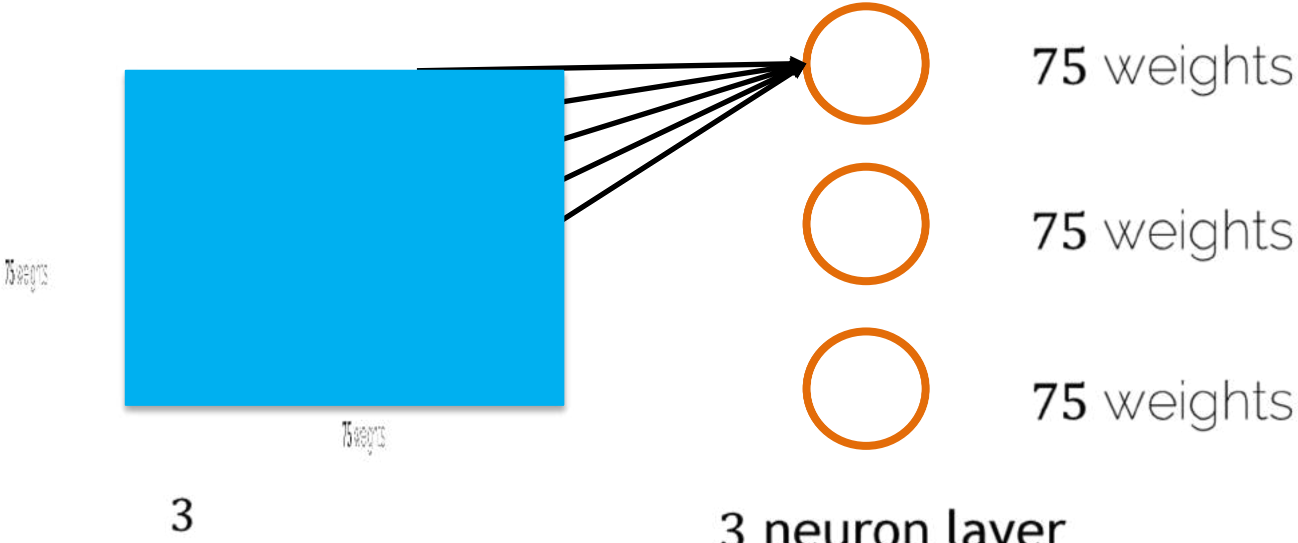
Problems using FC Layers on Images

- How to process a tiny image with FC layers



Problems using FC Layers on Images

- How to process a tiny image with FC layers

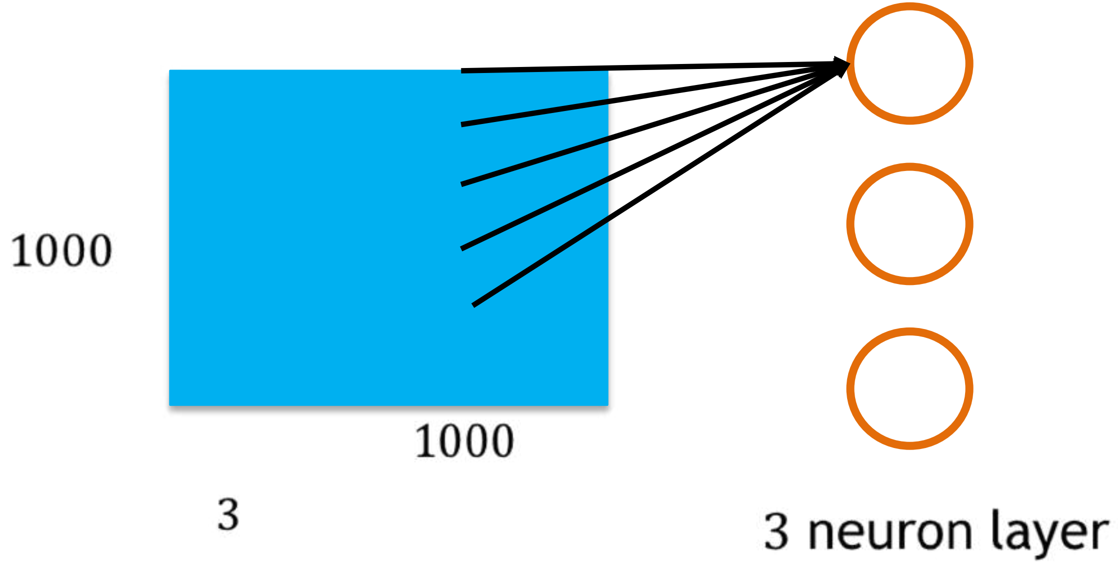


3 neuron layer

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

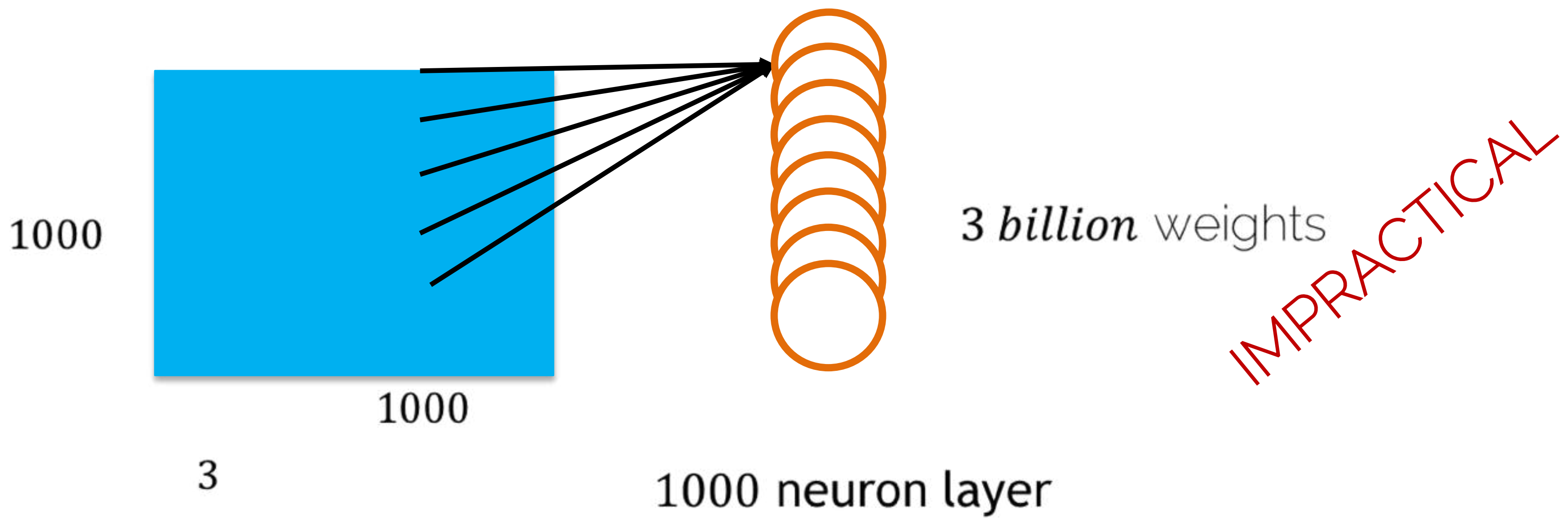
Problems using FC Layers on Images

- 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?

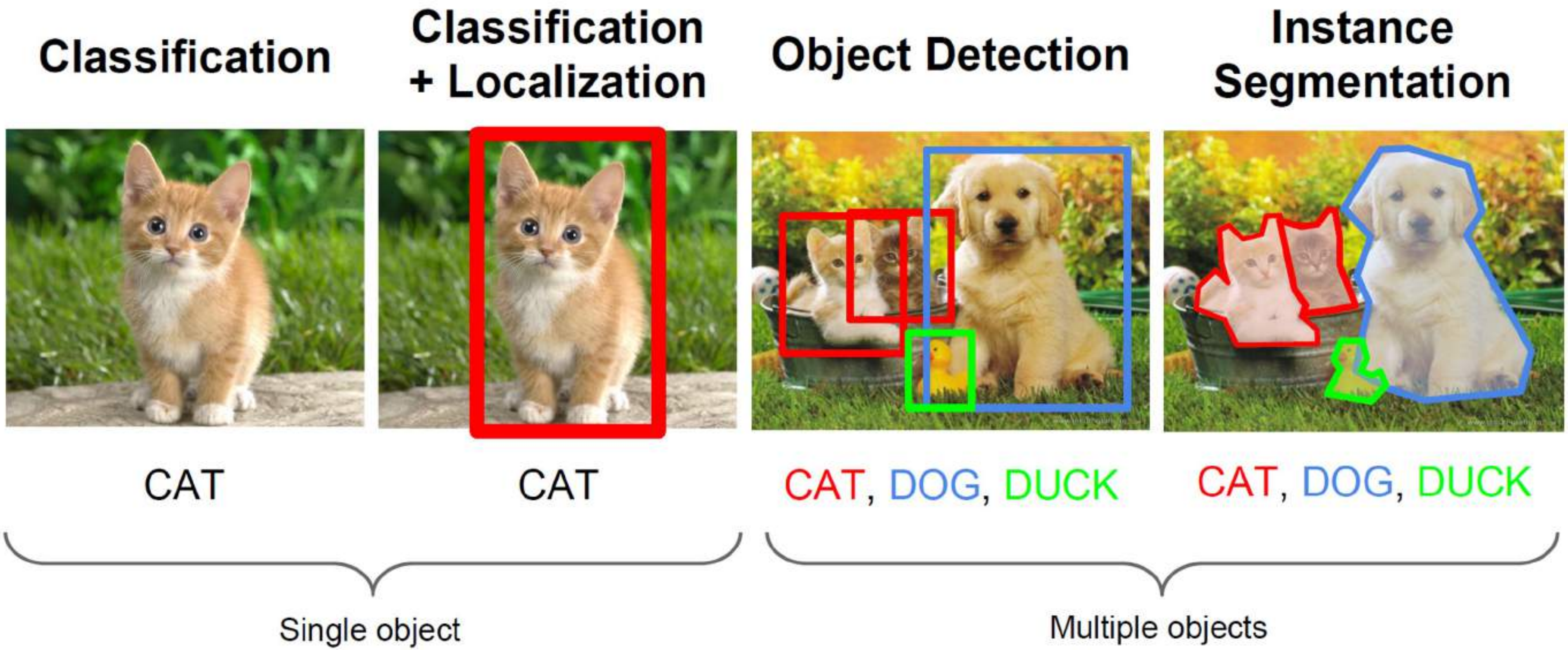
We cannot make networks arbitrarily complex

- 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 (one output unit is connected to only a few input units that are neighbors of each other).
  - Weight sharing, using the same weights for different parts of the image (intuition: if finding some feature is interesting in a part of an image, finding it in other parts is interesting too).

Using CNNs in Computer Vision

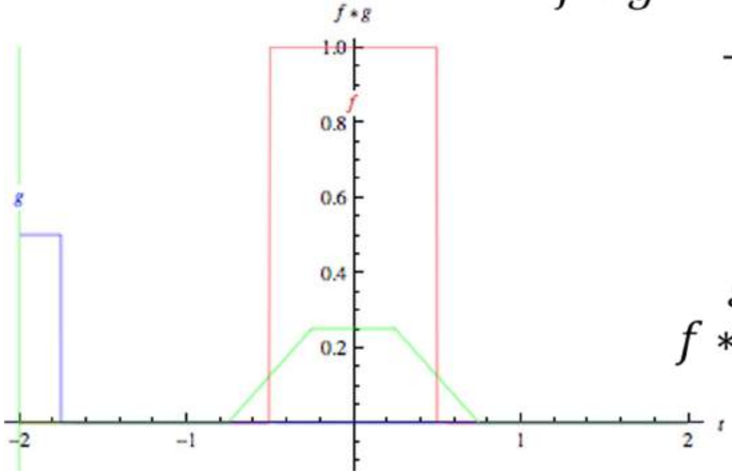


[Li et al., CS231n Course Slides] Lecture 12: Detection and Segmentation

Convolutions

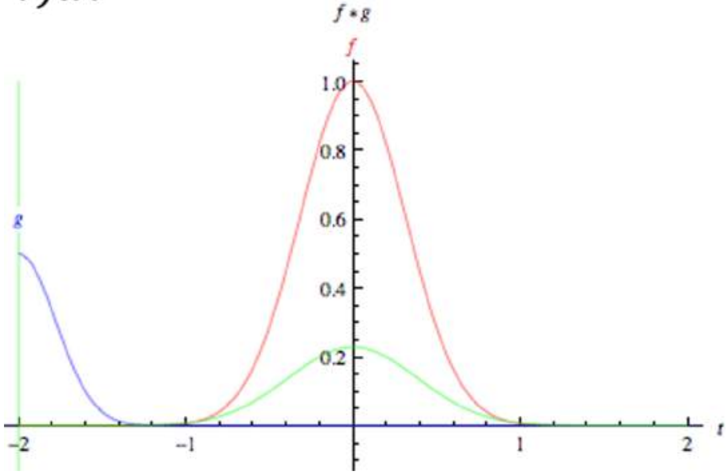
What are Convolutions?

$$f * g = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$



Convolution of two box functions

f = red
g = blue
$f * g$ = green



Convolution of two Gaussians

Application of a filter to a function

— The 'smaller' one is typically called the filter kernel

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								

$$4 \cdot \frac{1}{3} + 3 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} = 3$$

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							

$$3 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} + (-5) \cdot \frac{1}{3} = 0$$

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						

$$2 \cdot \frac{1}{3} + (-5) \cdot \frac{1}{3} + 3 \cdot \frac{1}{3} = 0$$

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					

$$(-5) \cdot \frac{1}{3} + 3 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} = 1$$

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				

$$3 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} = \frac{10}{3}$$

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			

$$5 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} = 4$$

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		

$$2 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} = 4$$

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**	

$$5 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} + 6 \cdot \frac{1}{3} = \frac{16}{3}$$

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	??

What to do at boundaries?

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

Option 2: Pad (often 0's)

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

Convolutions on Images

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

Kernel 3x3

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

Output 3x3

6		

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

Convolutions on Images

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

Kernel 3x3

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

Output 3x3

6	1	

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

Convolutions on Images

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

Kernel 3x3

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

Output 3x3

6	1	8

$$5 \cdot 1 + (-1) \cdot (-5) + (-1) \cdot (-3) + (-1) \cdot 3$$
$$+ (-1) \cdot 2$$
$$= 5 + 3 = 8$$

Convolutions on Images

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

Kernel 3x3

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

Output 3x3

6	1	8
-7		

$$5 \cdot 0 + (-1) \cdot 3 + (-1) \cdot 0 + (-1) \cdot 1 + (-1) \cdot 3$$
$$= 0 - 7 = -7$$

Convolutions on Images

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

Kernel 3x3

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

Output 3x3

6	1	8
-7	9	

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

Convolutions on Images

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

Kernel 3x3

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

Output 3x3

6	1	8
-7	9	2

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

Convolutions on Images

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

Kernel 3x3

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

Output 3x3

6	1	8
-7	9	2
-5		

$$5 \cdot 0 + (-1) \cdot 0 + (-1) \cdot 1 + (-1) \cdot 6 \\ + (-1) \cdot (-2) \\ = -5$$

Convolutions on Images

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

Kernel 3x3

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

Output 3x3

6	1	8
-7	9	2
-5	-9	

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

Convolutions on Images

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

Kernel 3x3

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

Output 3x3

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

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

Image Filters

- Each kernel gives us a different image filter

Input

Edge detection

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

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!

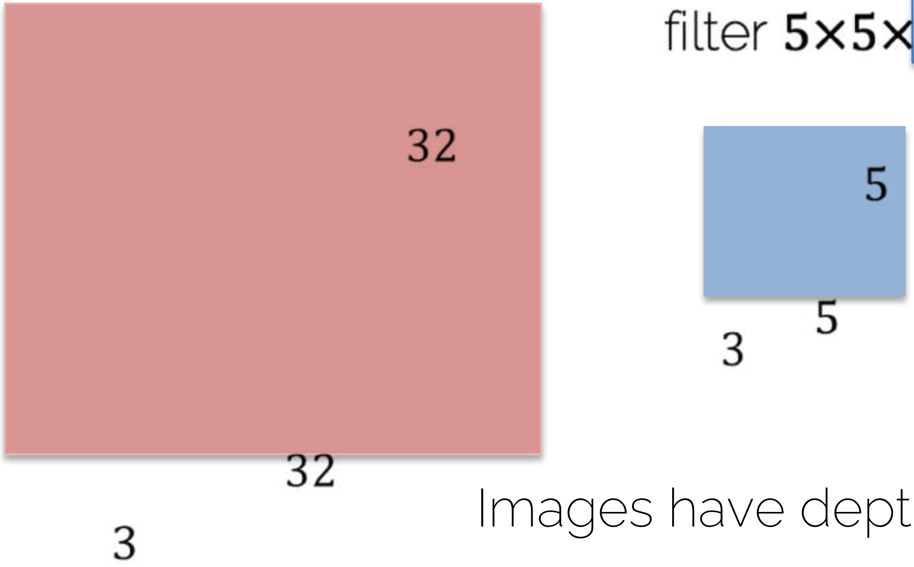
Convolutions on RGB Images

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



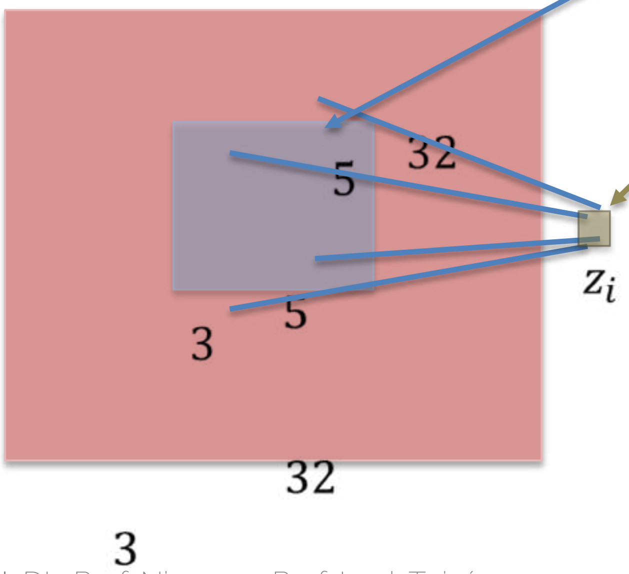
Convolve filter with image
i.e., 'slide' over it and:
- apply filter at each location
- dot products

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

Convolutions on RGB Images

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

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



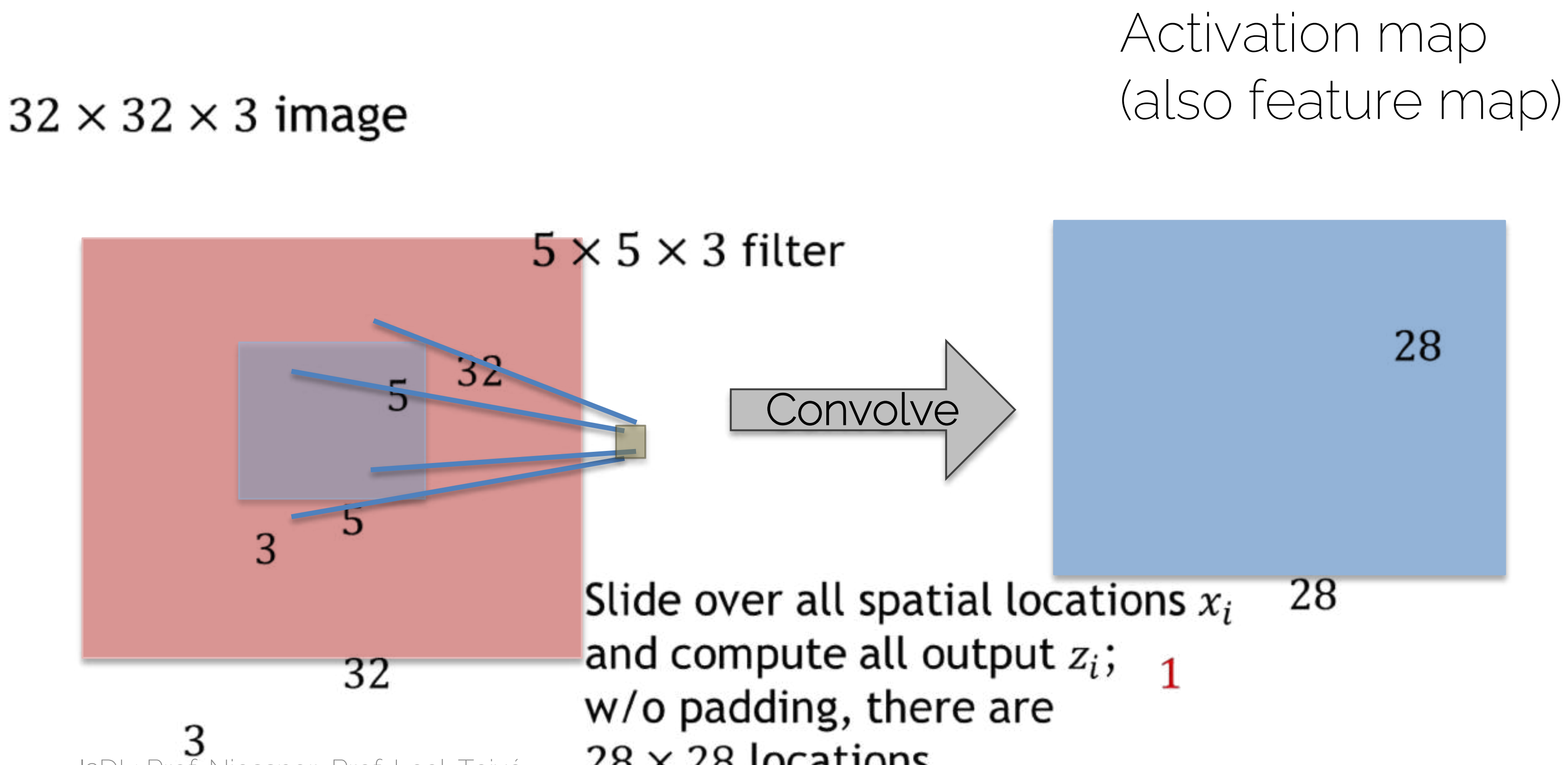
1 number at a time:

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

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

$(5 \times 5 \times 3) \times 1$ $\quad$ $(5 \times 5 \times 3) \times 1$ $\quad$ 1

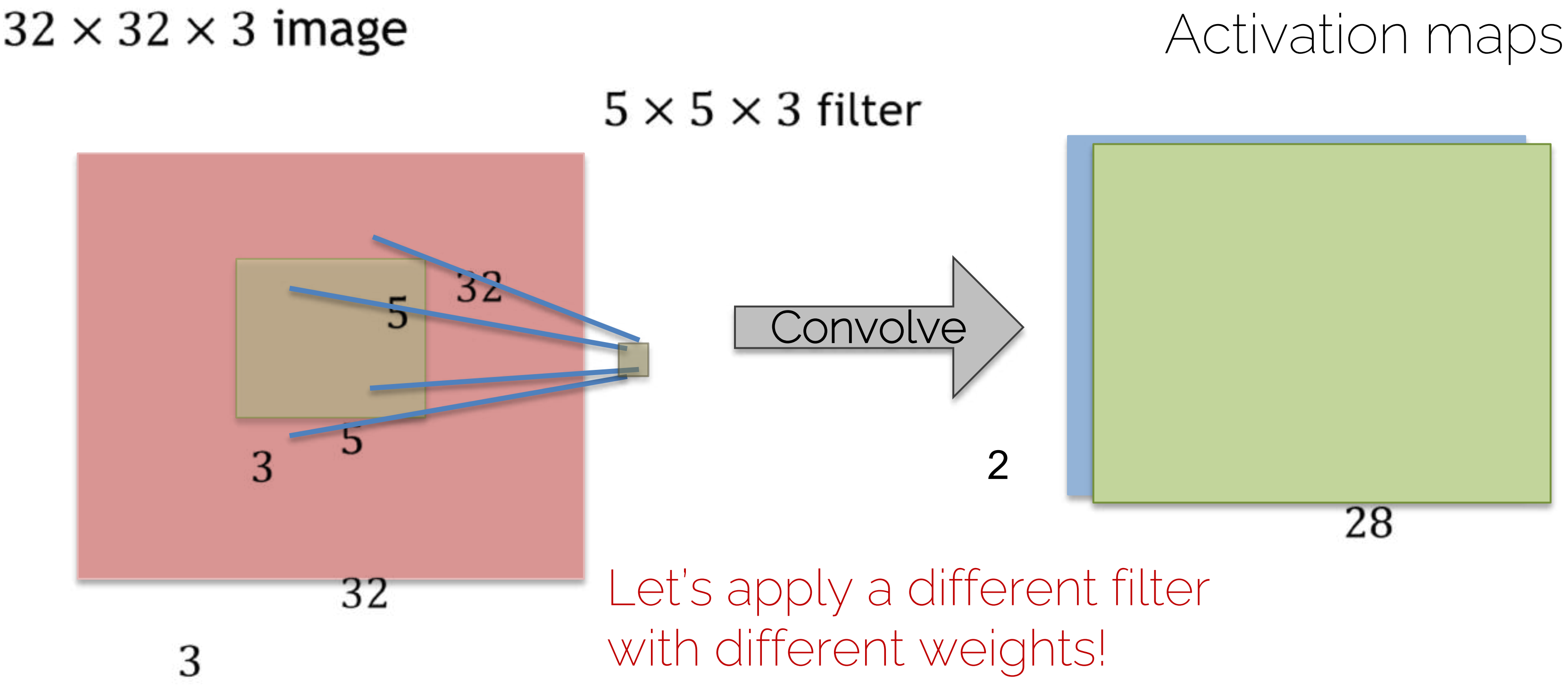
Convolutions on RGB Images



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

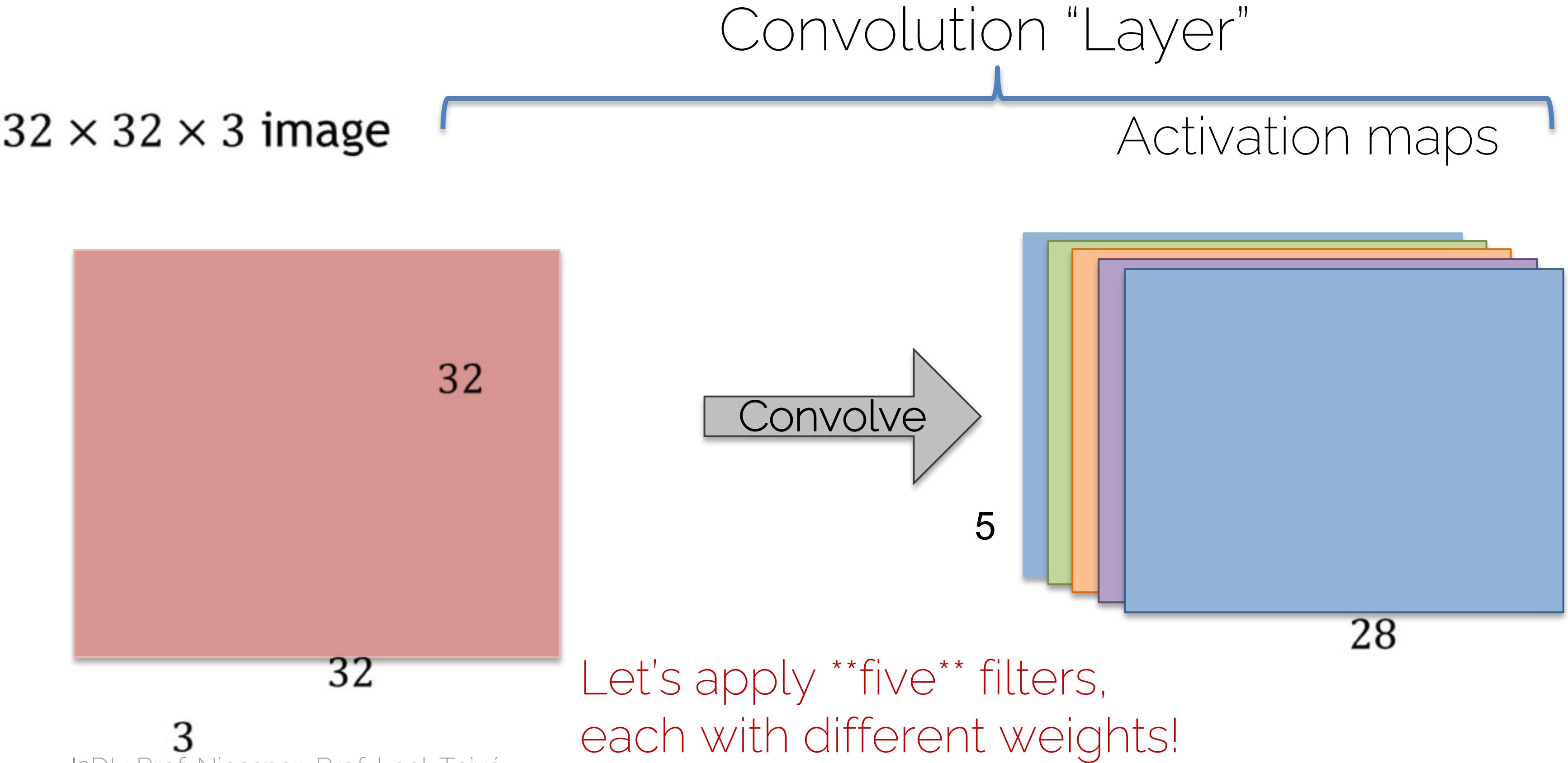
Convolution Layer

Convolution Layer



Let's apply a different filter
with different weights!

Convolution Layer



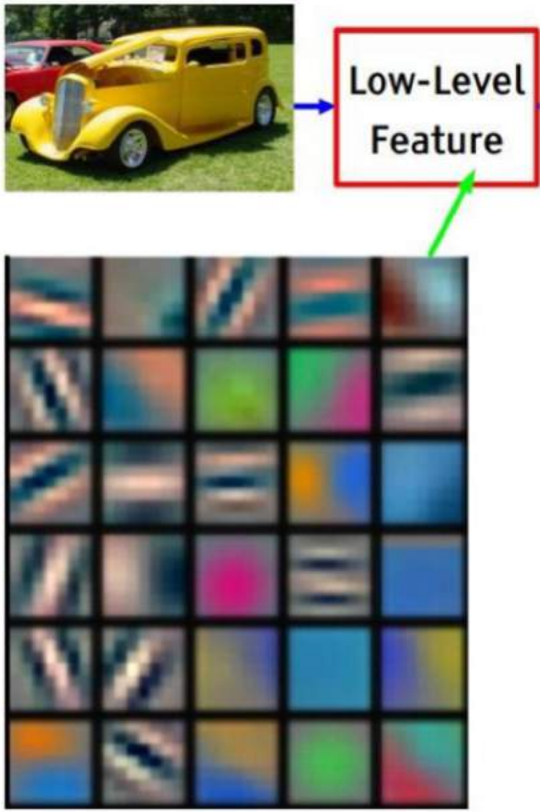
Let's apply **five** filters, each with different weights!

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

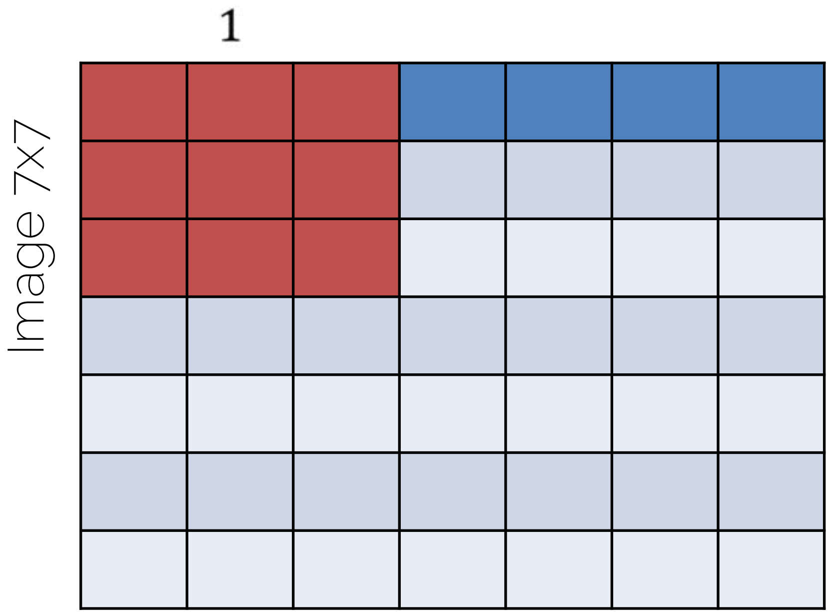


- Each filter captures different image characteristics:
  - Horizontal edges
  - Vertical edges
  - Circles
  - Squares
  - …

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

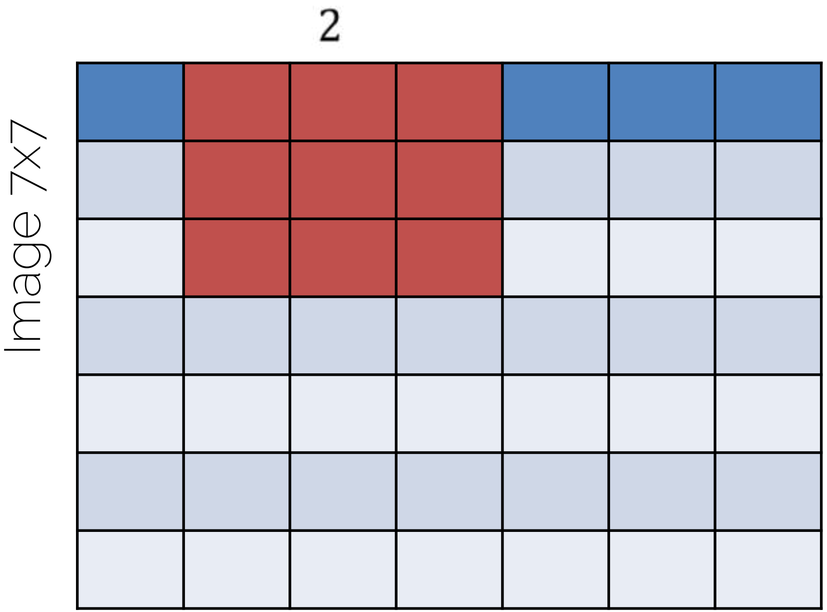
Dimensions of a Convolution Layer

Convolution Layers: Dimensions



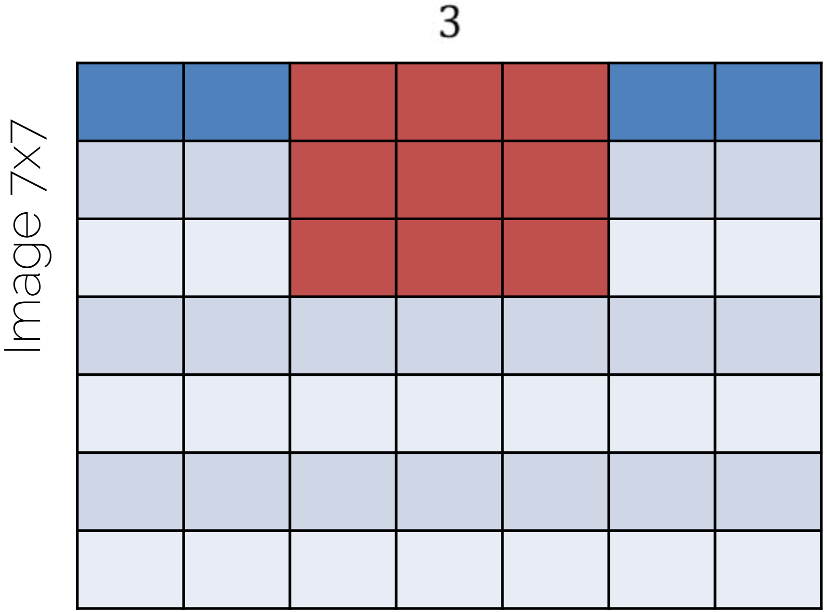
Input: 7x7
Filter: 3x3
Output:

Convolution Layers: Dimensions



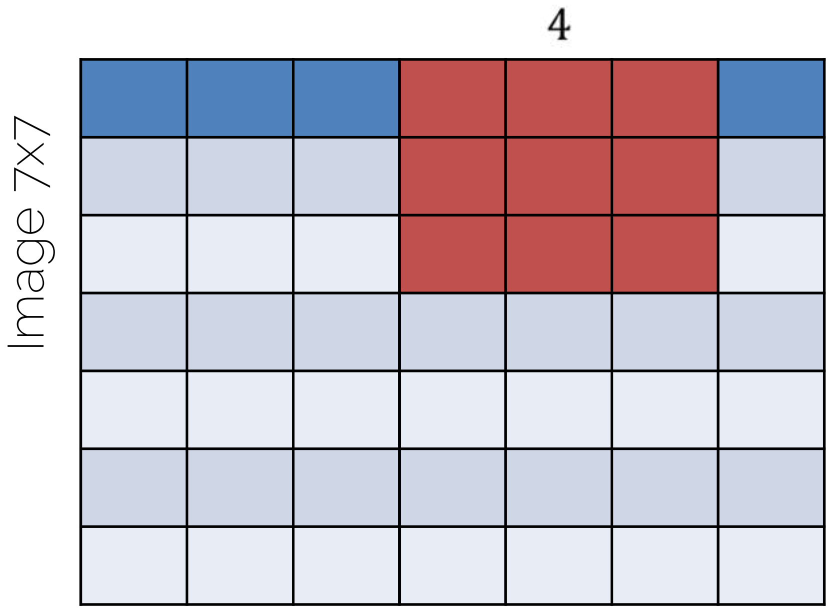
Input: 7x7
Filter: 3x3
Output:

Convolution Layers: Dimensions



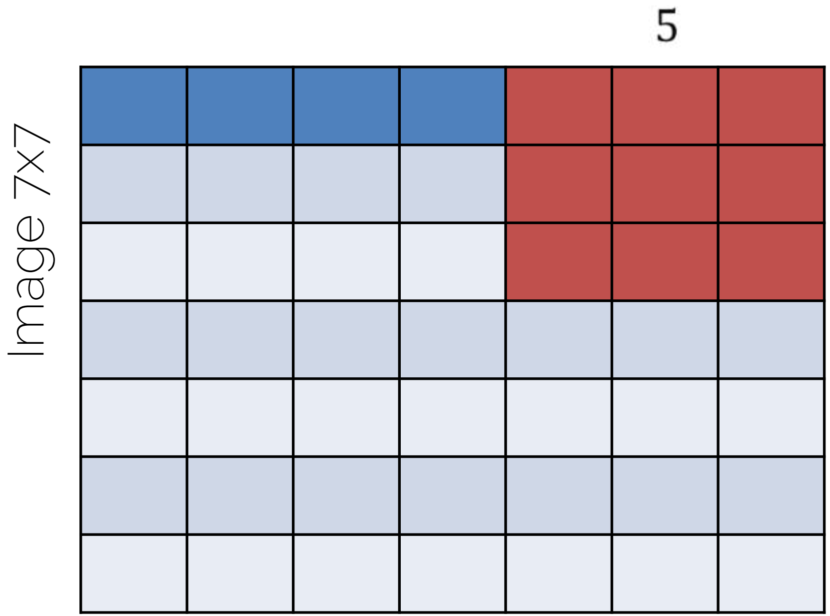
Input: 7x7
Filter: 3x3
Output:

Convolution Layers: Dimensions



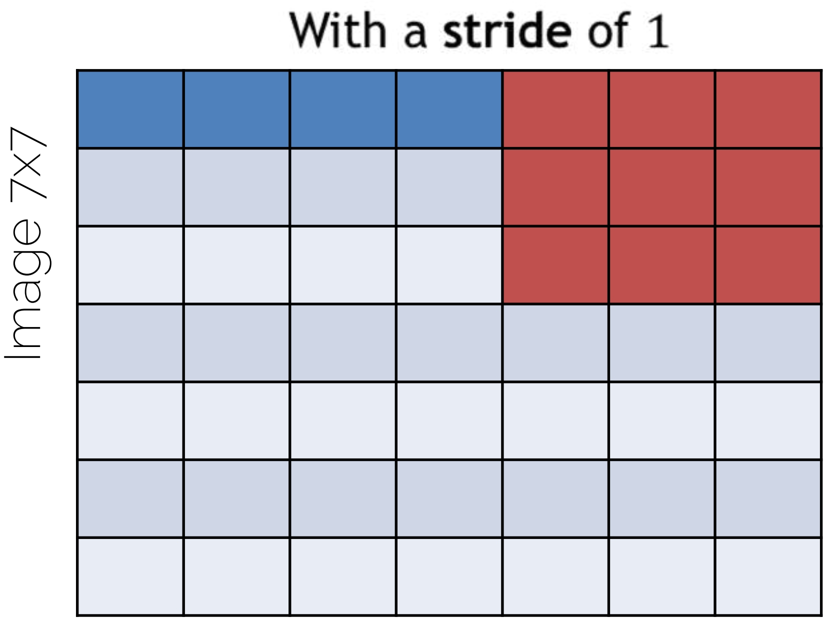
Input: 7x7
Filter: 3x3
Output:

Convolution Layers: Dimensions



Input: 7x7
Filter: 3x3
Output:

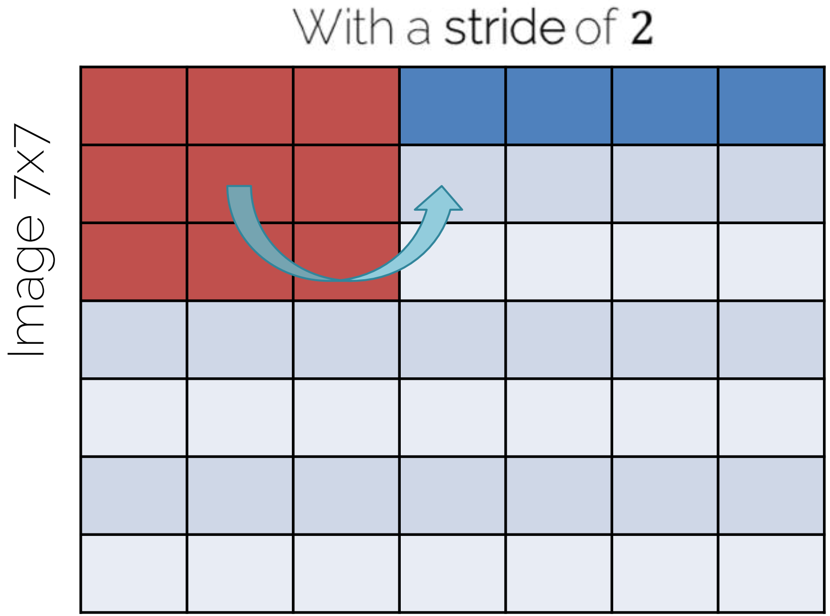
Convolution Layers: Stride



Input: 7x7
Filter: 3x3
Output: 5x5

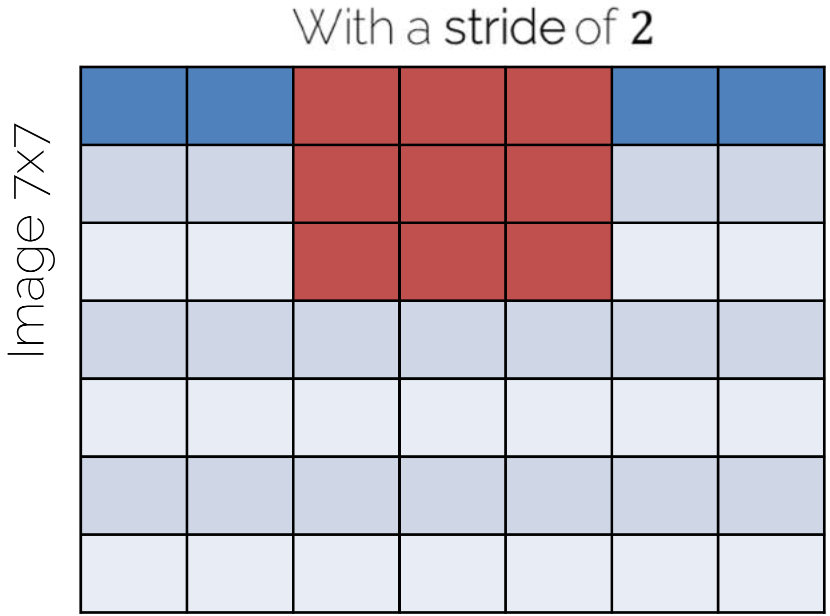
Stride of S: apply filter every S-th spatial location; i.e. subsample the image

Convolution Layers: Stride



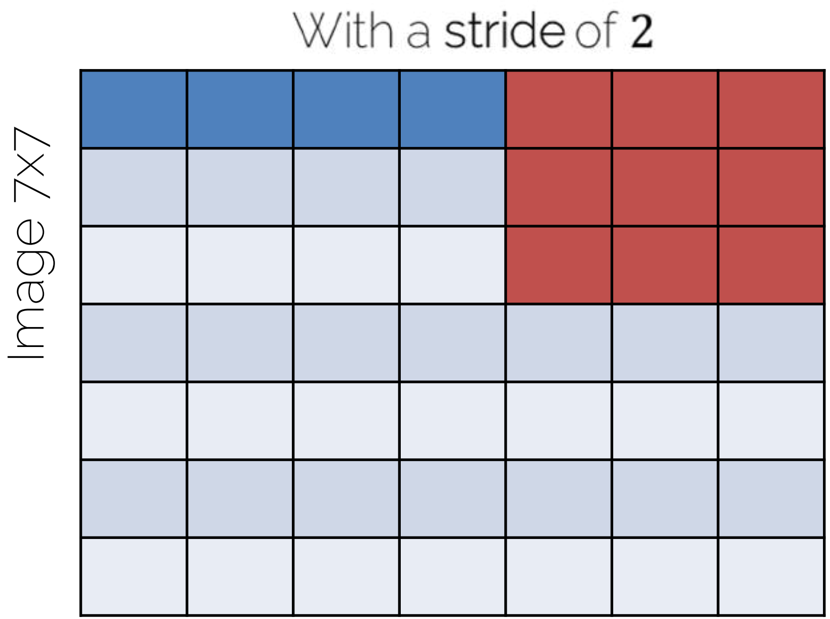
Input: 7x7
Filter: 3x3
Stride: 2
Output:

Convolution Layers: Stride



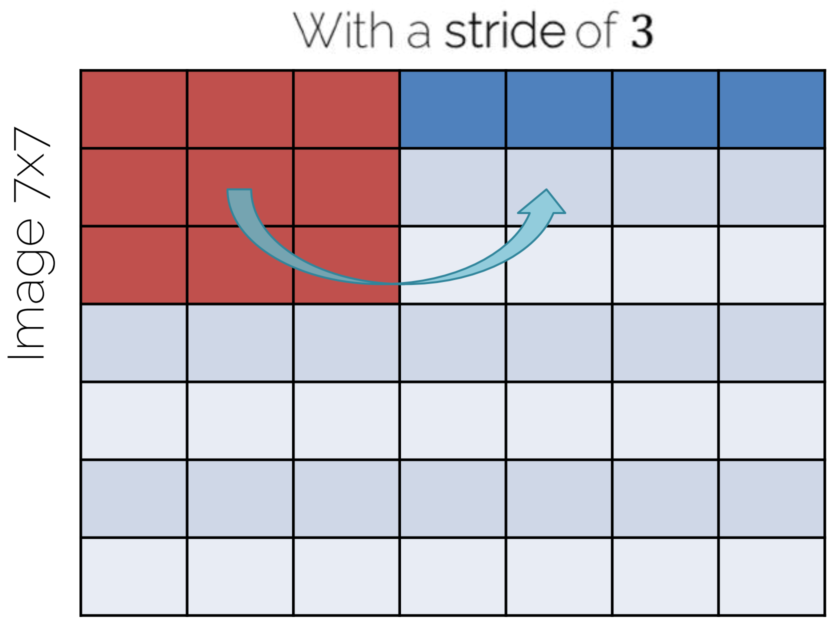
Input: 7x7
Filter: 3x3
Stride: 2
Output:

Convolution Layers: Stride



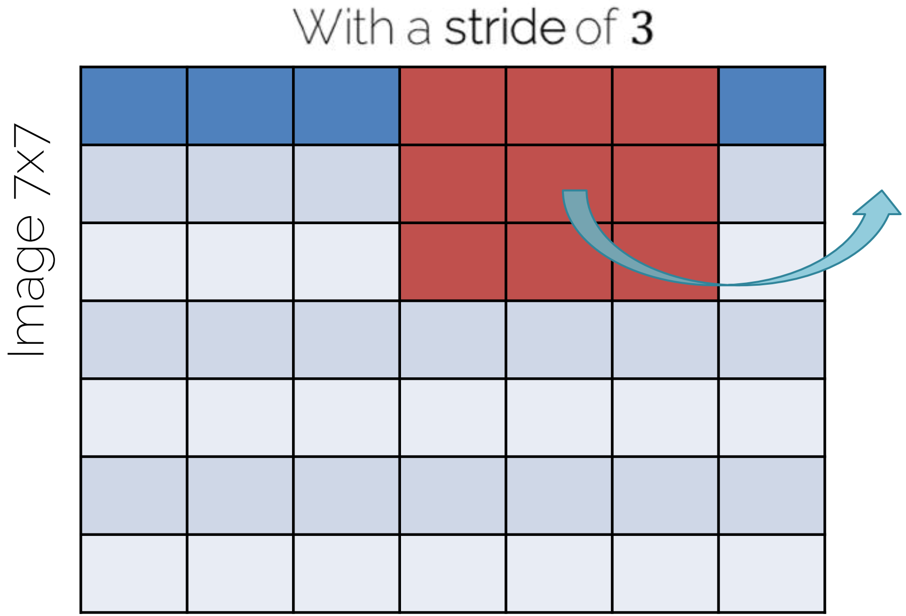
Input: 7x7
Filter: 3x3
Stride: 2
Output: 3x3

Convolution Layers: Stride



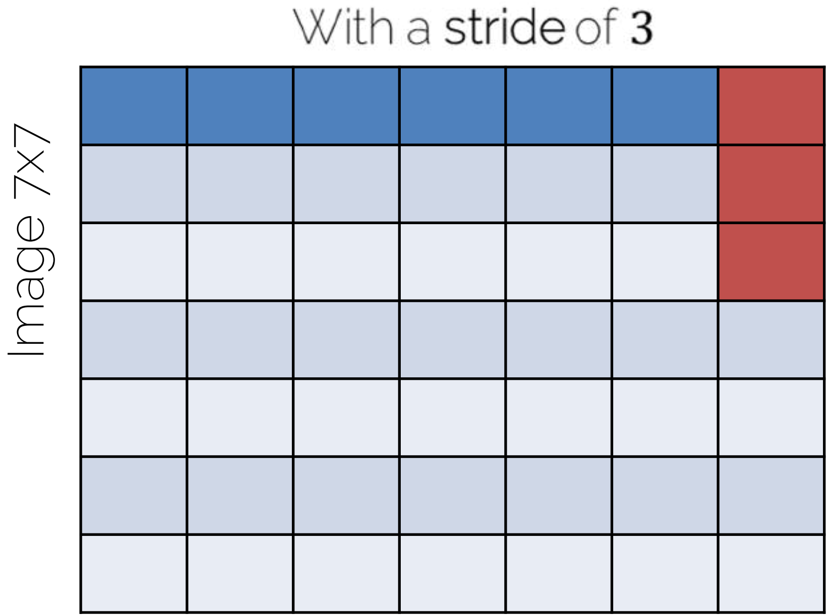
Input: 7x7
Filter: 3x3
Stride: 3
Output:

Convolution Layers: Stride



Input: 7x7
Filter: 3x3
Stride: 3
Output:

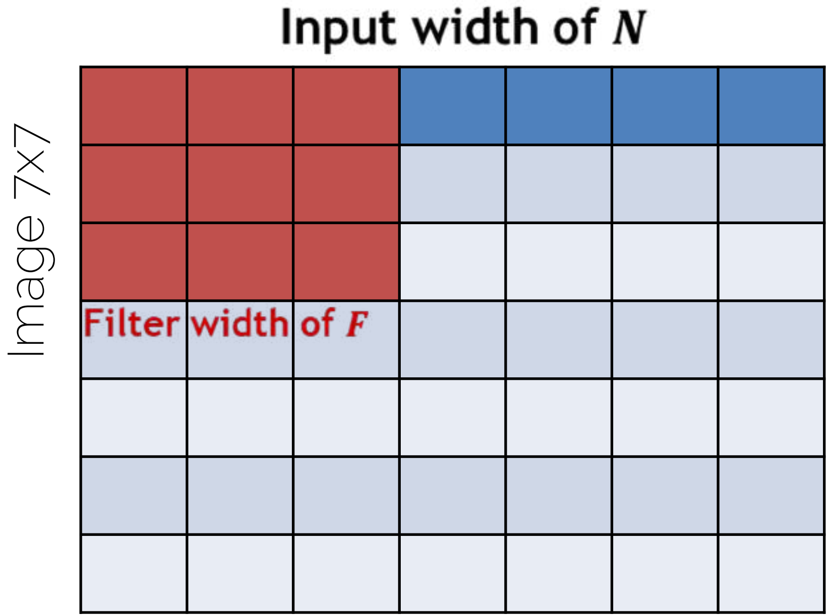
Convolution Layers: Stride



Input: 7x7
Filter: 3x3
Stride: 3
Output: hmm

Does not really fit (remainder left)
Illegal stride for input & filter size!

Convolution Layers: Dimensions



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

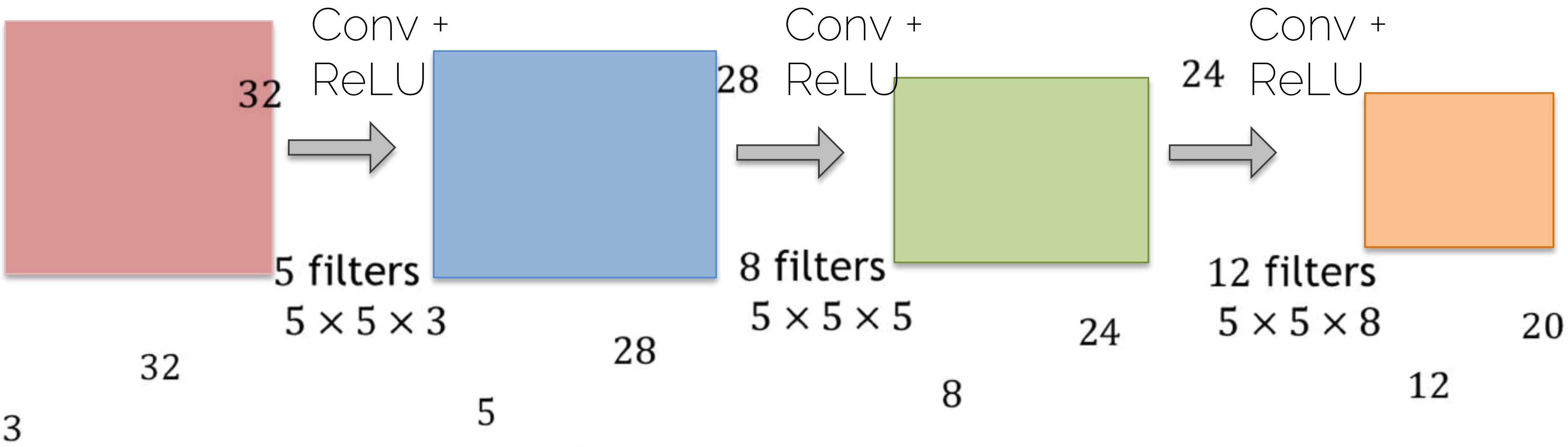
$$N = 7, F = 3, S = 2: \frac{7-3}{2} + 1 = 3$$

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

Fractions are illegal

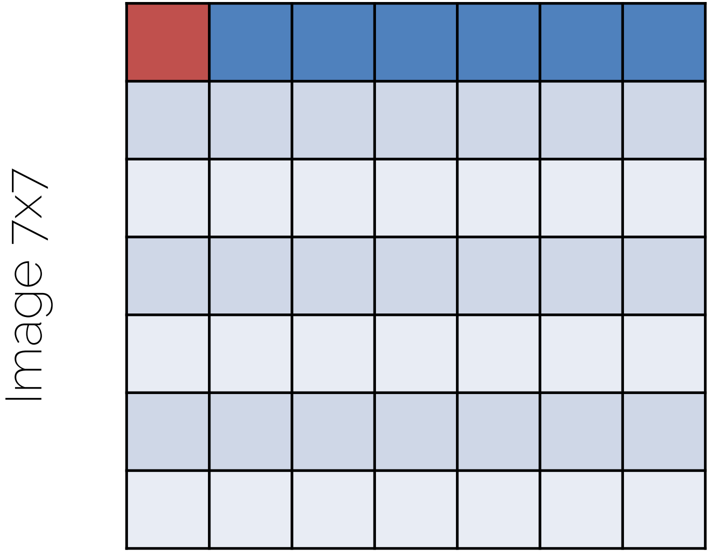
Convolution Layers: Dimensions

Input Image



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

Convolution Layers: Padding

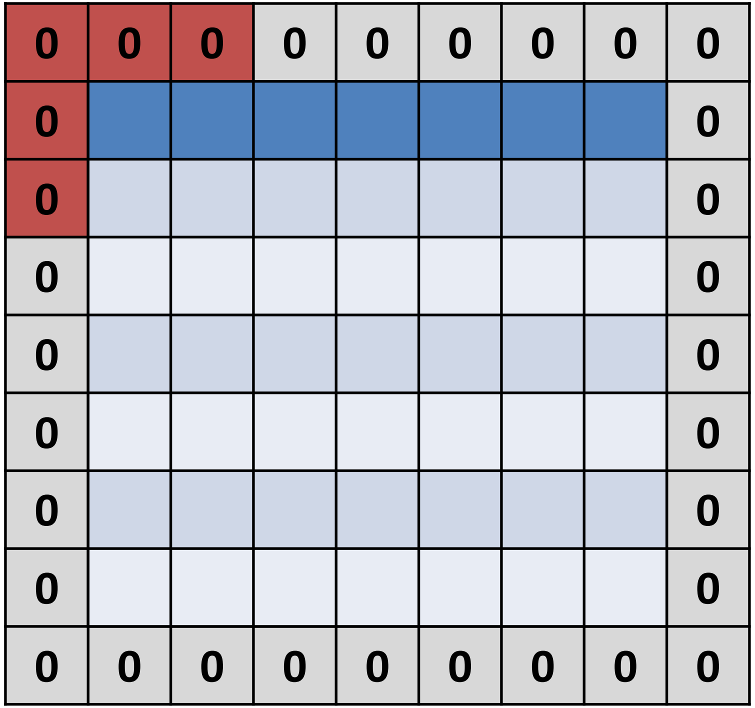


Why padding?

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

Convolution Layers: Padding

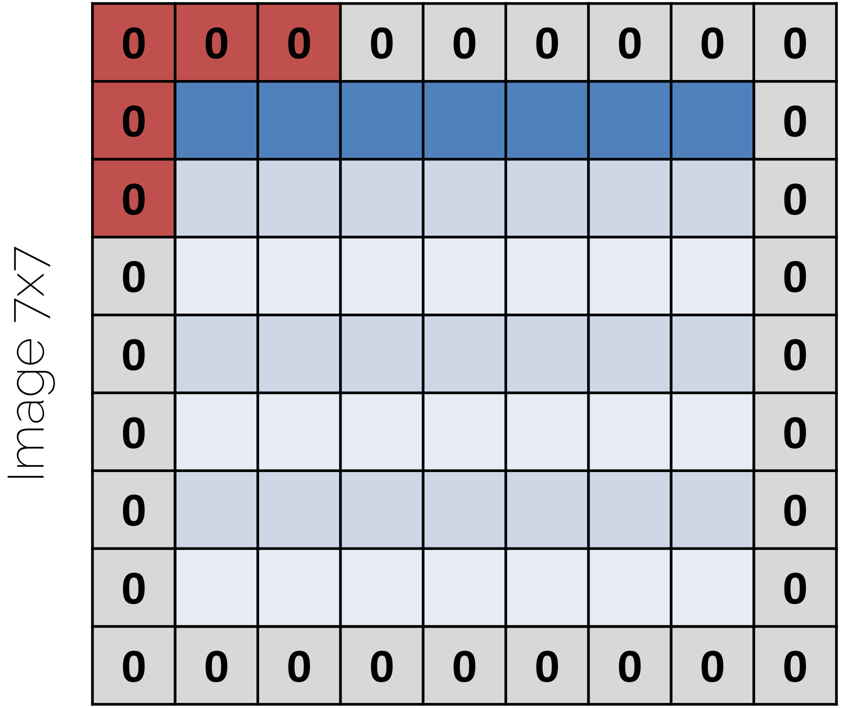
Image 7x7

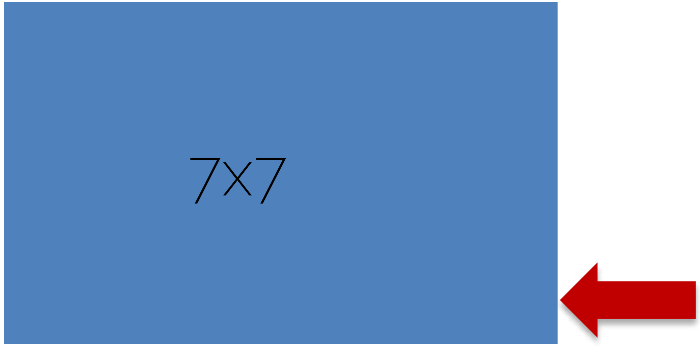


Why padding?

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

Convolution Layers: Padding





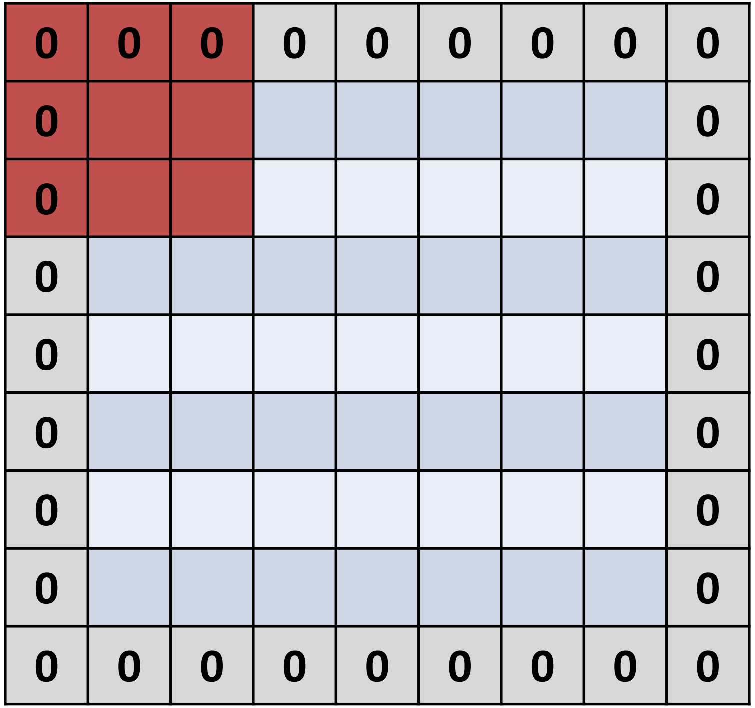
Most common is '*zero*' padding

Output Size:

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

$\lfloor\ \rfloor$ 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

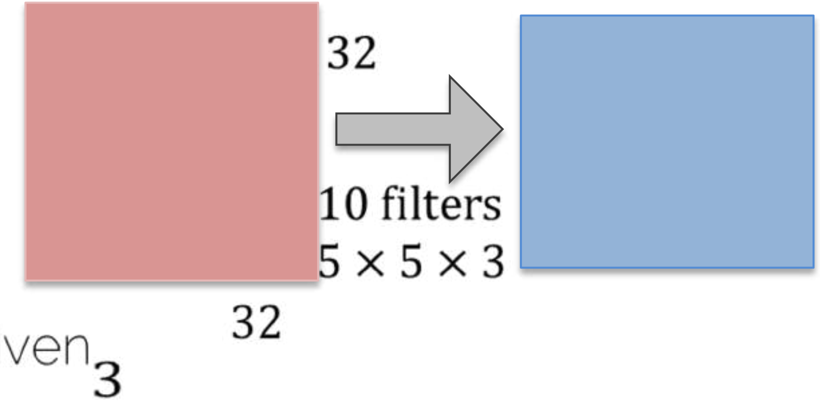
Set padding to $P = \frac{F-1}{2}$

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

Convolution Layers: Dimensions

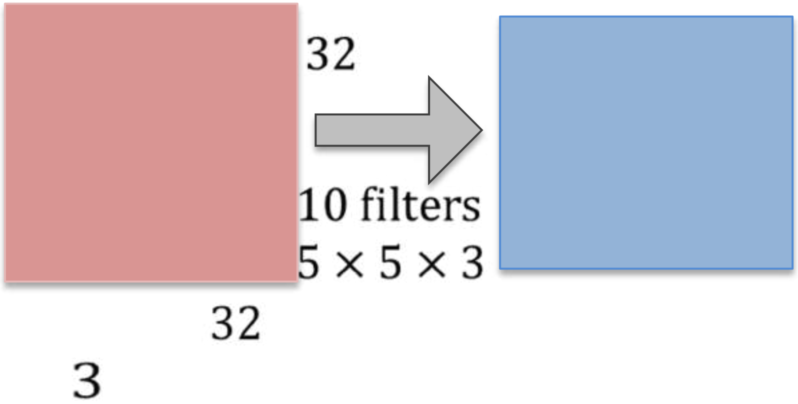
Example

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

Output size is:

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

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



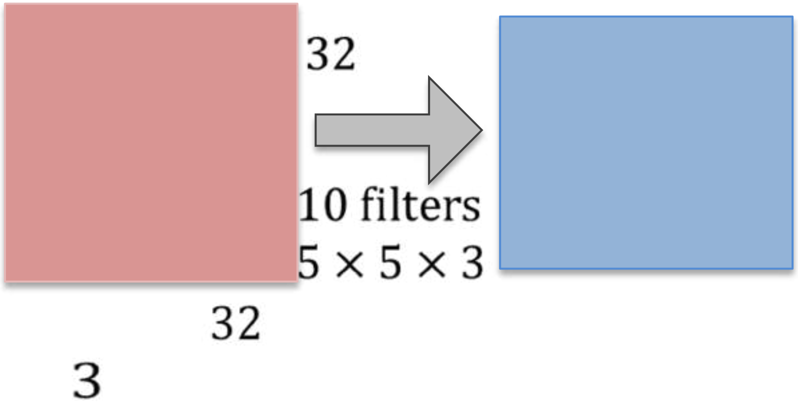
Remember

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

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)
-> $76 \cdot 10 = 760$ parameters in layer

Example

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

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

  - [ ] A1: $(\mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{5})$
  - [ ] A2: $(\mathbf{4}, \mathbf{5}, \mathbf{5})$
  - [ ] 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)

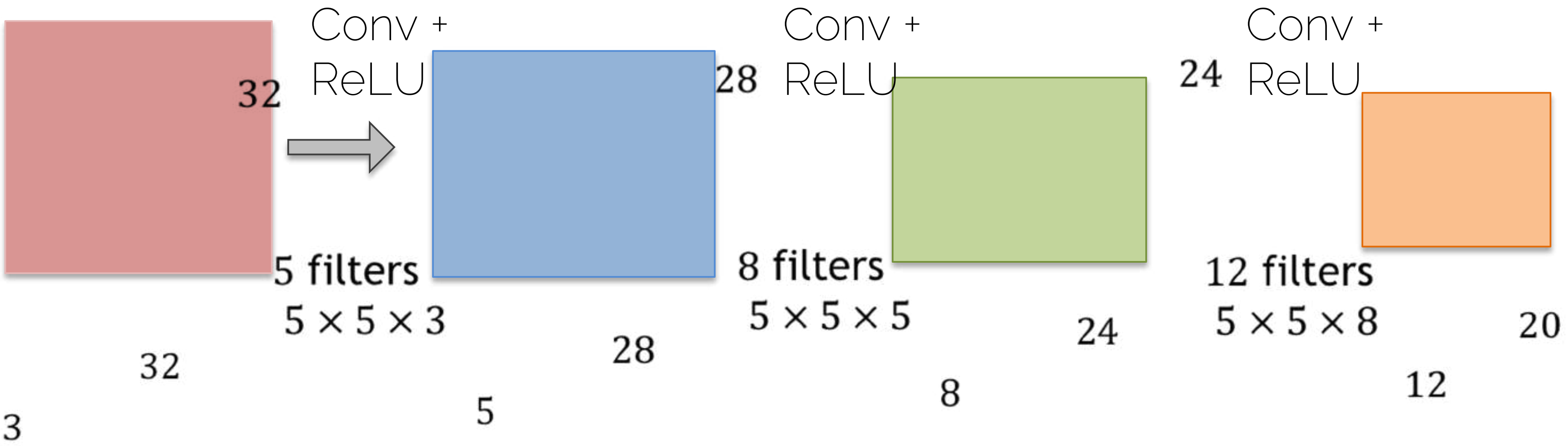
Output size = 4 filters

Filter size = **5×5**

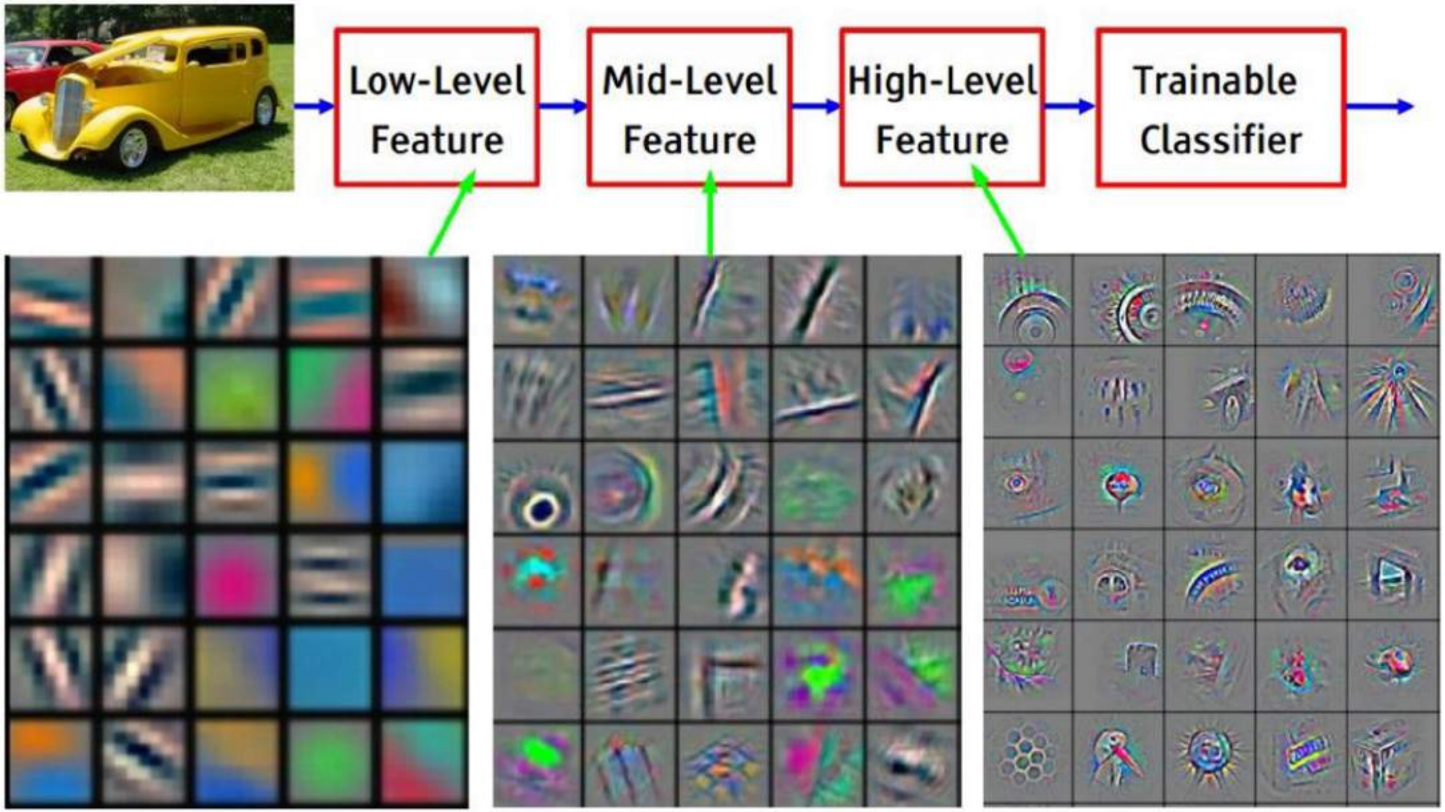
Convolutional Neural Network (CNN)

CNN Prototype

ConvNet is concatenation of Conv Layers and activations



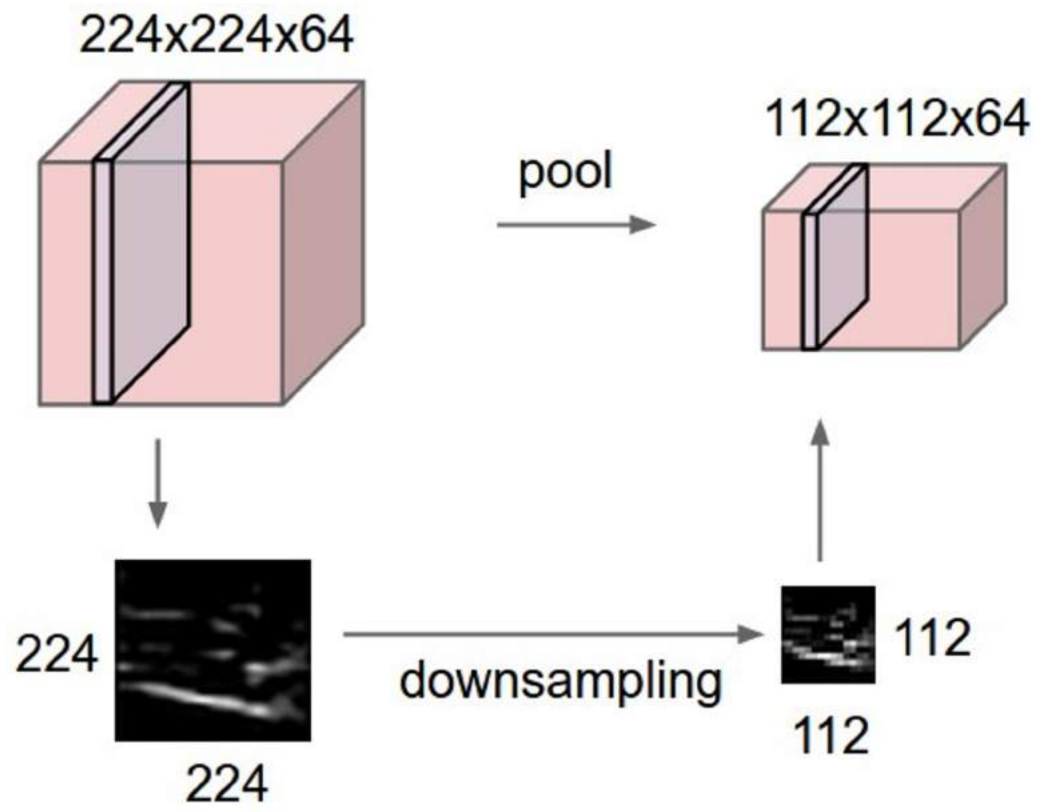
CNN Learned Filters



[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 depth slice 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 $W_{in} \times H_{in} \times D_{in}$
- Two hyperparameters
  - Spatial filter extent F
  - Stride S

Filter count K and padding P make no sense here

- Output volume is of size $W_{out} \times H_{out} \times D_{out}$
  - $W_{out} = \frac{W_{in} - F}{S} + 1$
  - $H_{out} = \frac{H_{in} - F}{S} + 1$
  - $D_{out} = D_{in}$

Pooling Layer

- Input is a volume of size $W_{in} \times H_{in} \times D_{in}$
- Two hyperparameters
  – Spatial filter extent F
  – Stride S
- Output volume is of size $W_{out} \times H_{out} \times D_{out}$
  – $W_{out} = \frac{W_{in} - F}{S} + 1$
  – $H_{out} = \frac{H_{in} - F}{S} + 1$
  – $D_{out} = D_{in}$

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

Pooling Layer: Average Pooling

Single depth slice of input

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

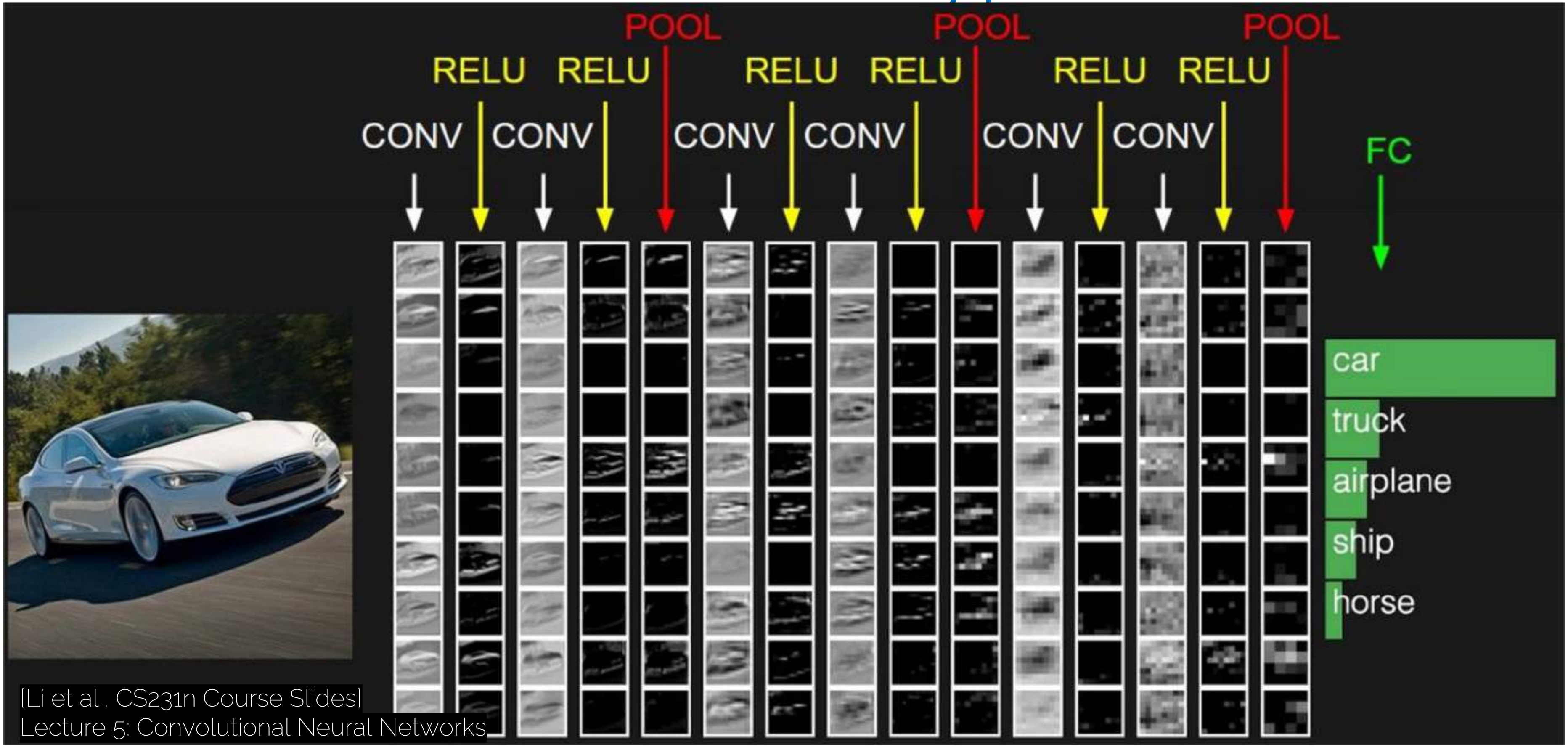
Average pool with
2×2 filters and stride 2

'Pooled' output

2.5	6
1.75	3

- Typically used deeper in the network

CNN Prototype



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

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

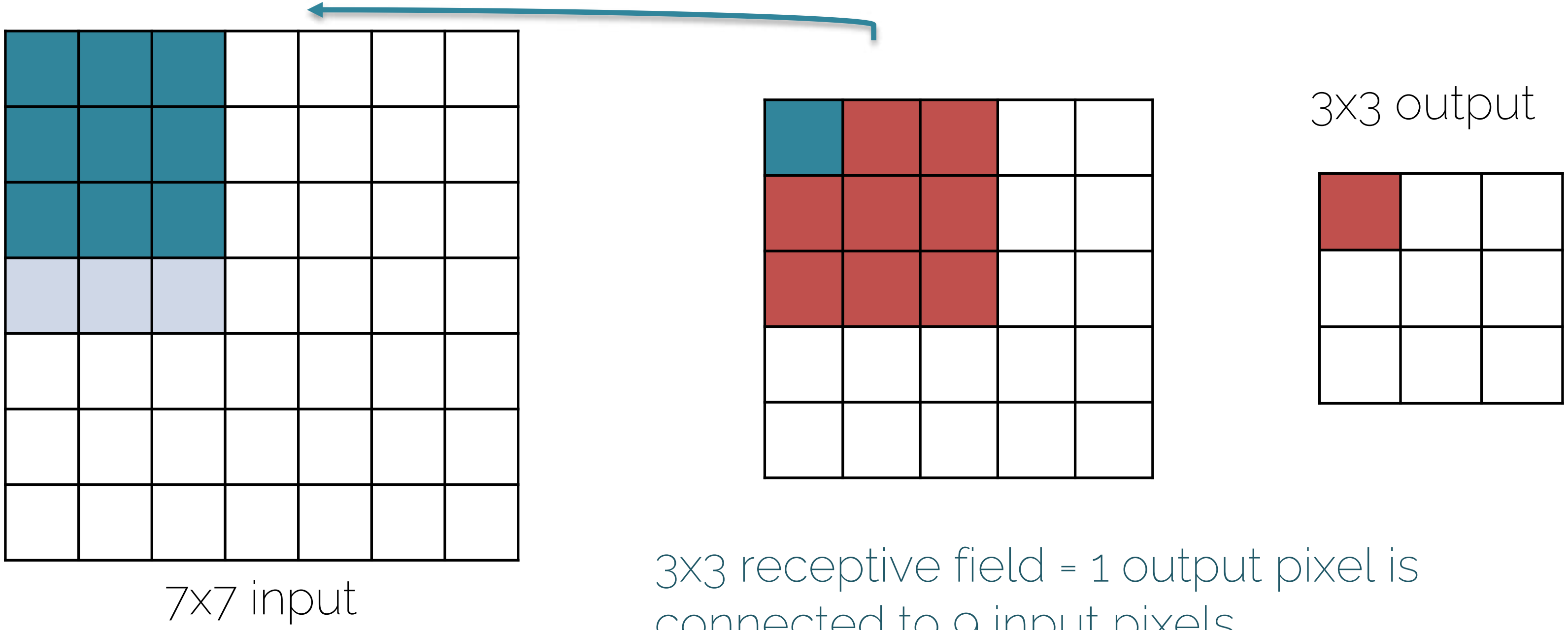
Convolutions vs Fully-Connected

- 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

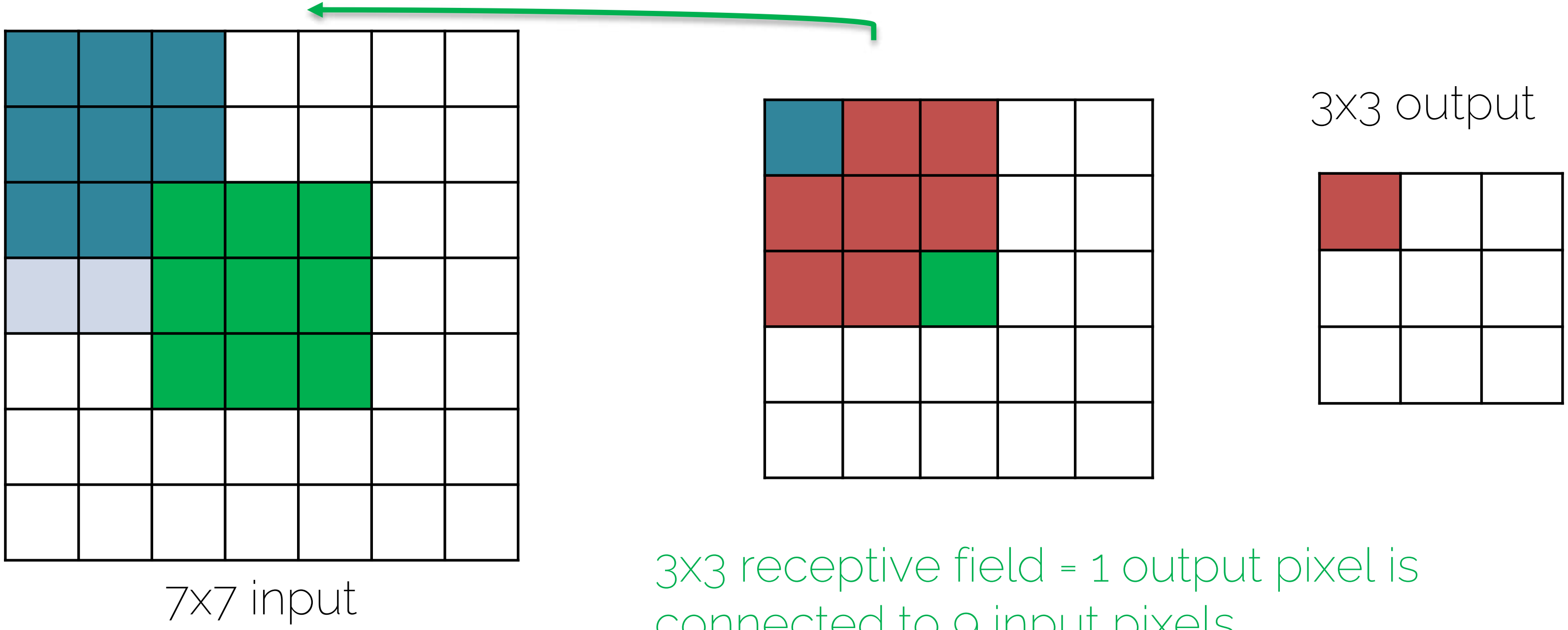
- Spatial extent of the connectivity of a convolutional filter



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

Receptive Field

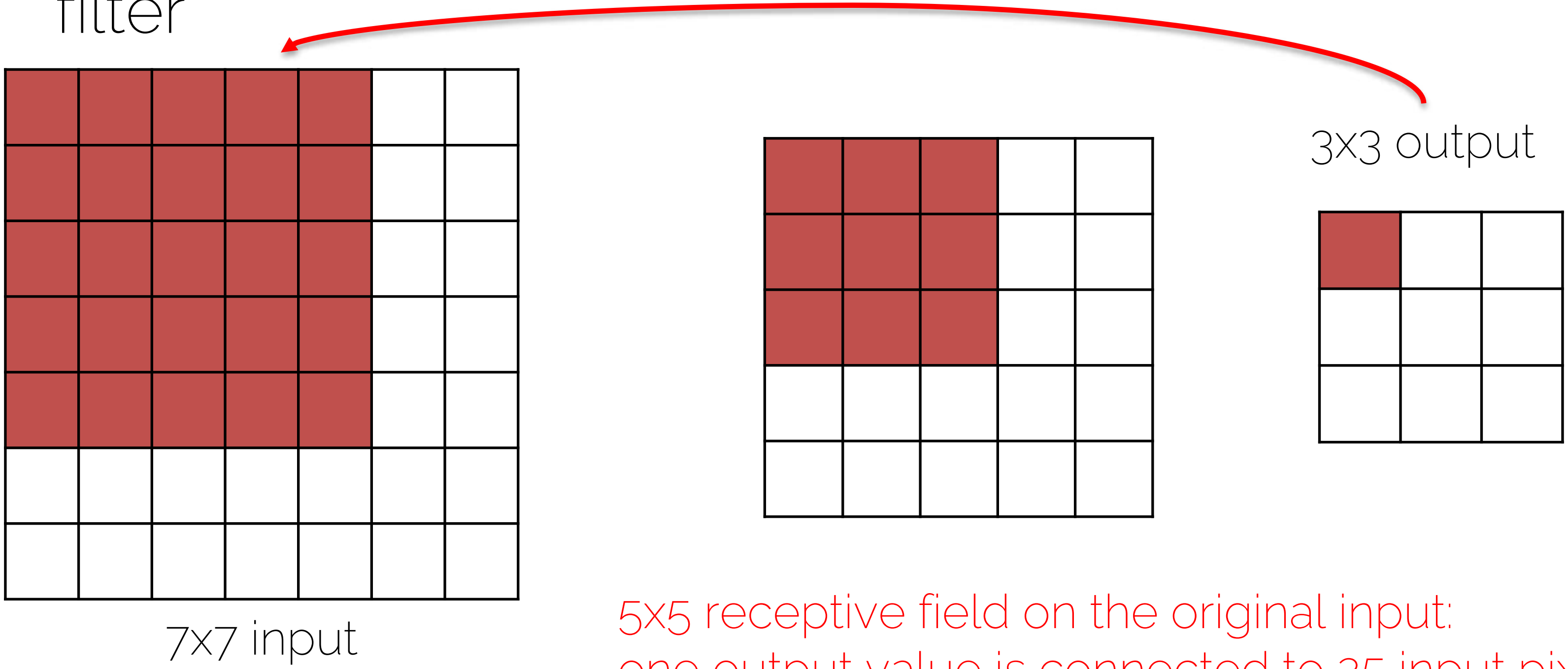
- Spatial extent of the connectivity of a convolutional filter



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

Receptive Field

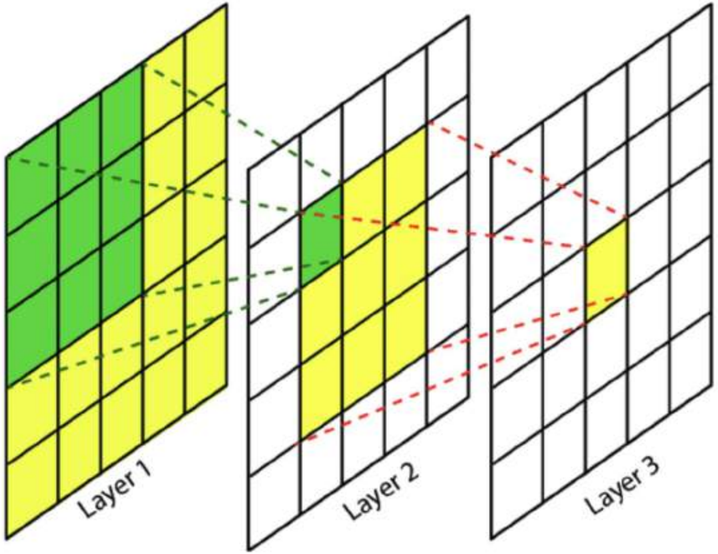
- Spatial extent of the connectivity of a convolutional filter



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

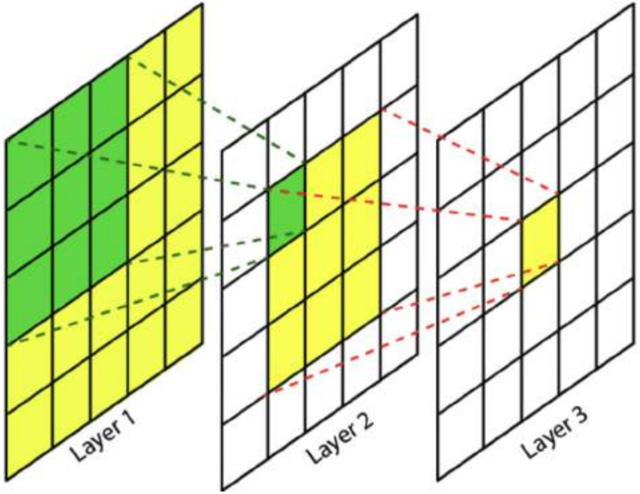
Receptive field

- 2 consecutive kernels of size 3x3, would have a receptive field of size 5x5. 3 consecutive kernels of size 3x3 have a receptive field of size 7x7.



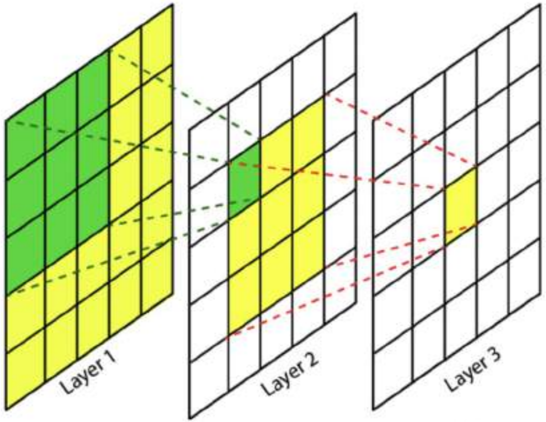
Receptive field

- 2 consecutive kernels of size 3x3, would have a receptive field of size 5x5. 3 consecutive kernels of size 3x3 have a receptive field of size 7x7.
- Question: is it more efficient to use three 3x3 kernels of a single 7x7 kernel?



Receptive field

- 2 consecutive kernels of size 3x3, would have a receptive field of size 5x5. 3 consecutive kernels of size 3x3 have a receptive field of size 7x7.
- Question: is it more efficient to use three 3x3 kernels of a single 7x7 kernel?
- Answer: 3 times 3x3=27 parameters (+ 3 for bias). 1 times 7x7 = 49 parameters (+1 for bias). 30 < 50.
- Bonus: more nonlinearity.



See you next time!

References

- Goodfellow et al. “Deep Learning” (2016),
  – Chapter 9: Convolutional Networks

- http://cs231n.github.io/convolutional-networks/