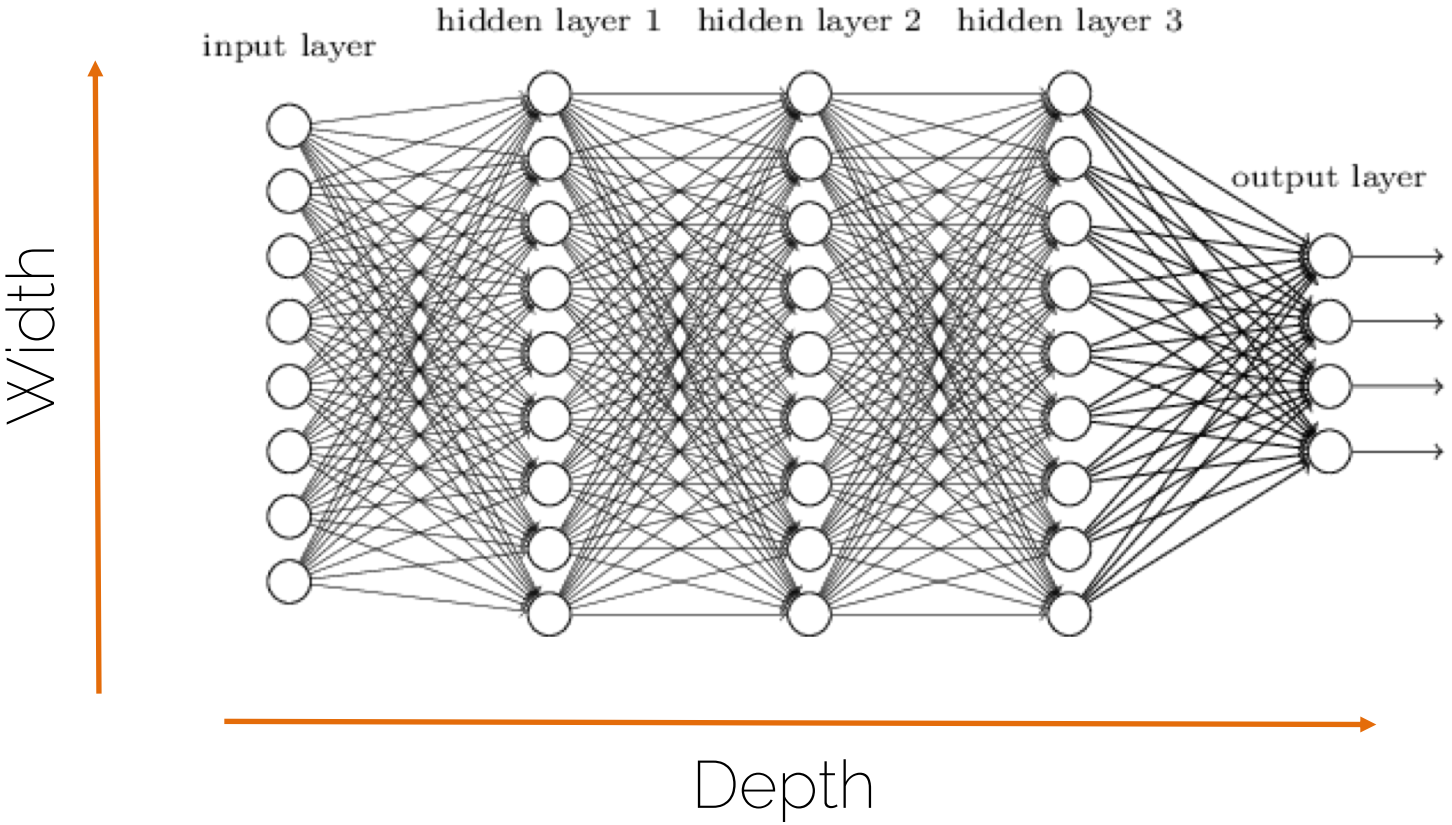
Lecture 8 Recap

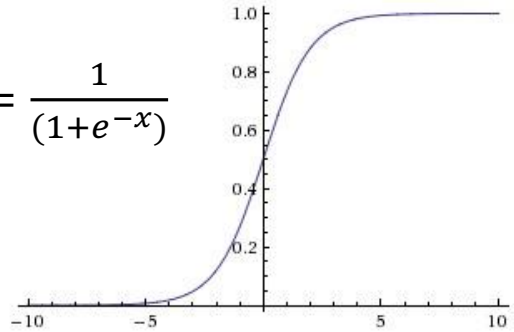
What do we know so far?



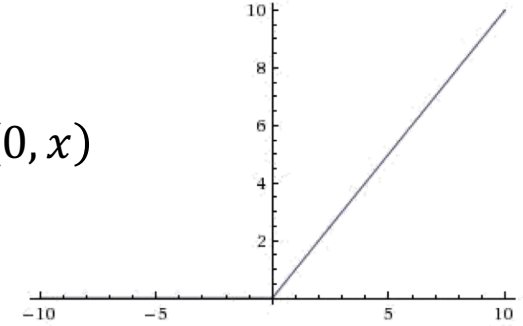
What do we know so far?

Activation Functions (non-linearities)

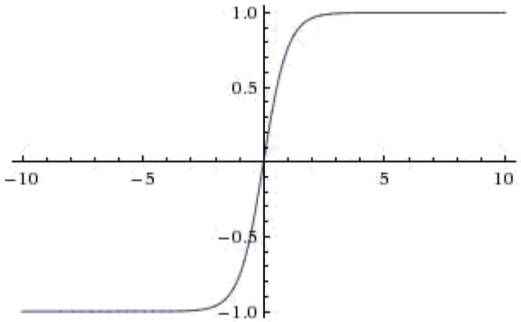
Sigmoid: $\sigma(x) = \frac{1}{(1+e^{-x})}$



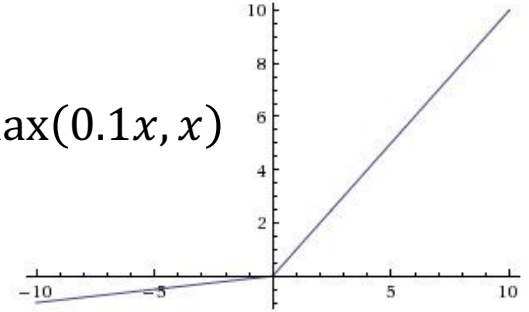
ReLU: $\max(0, x)$



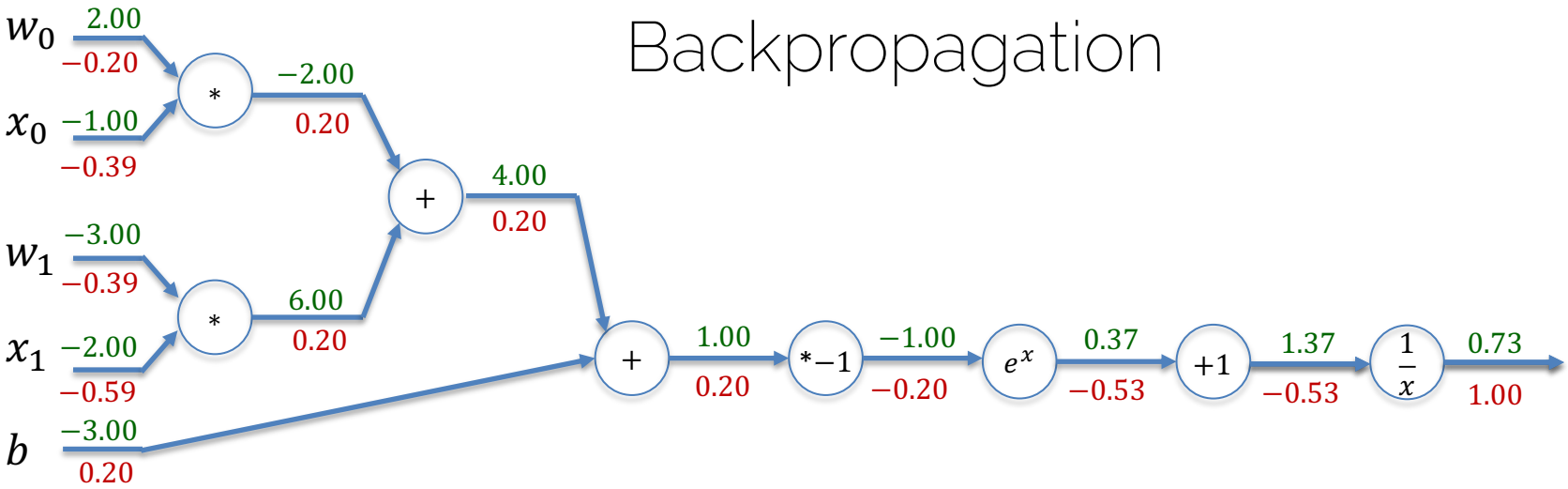
tanh: $\tanh(x)$



Leaky ReLU: $\max(0.1x, x)$

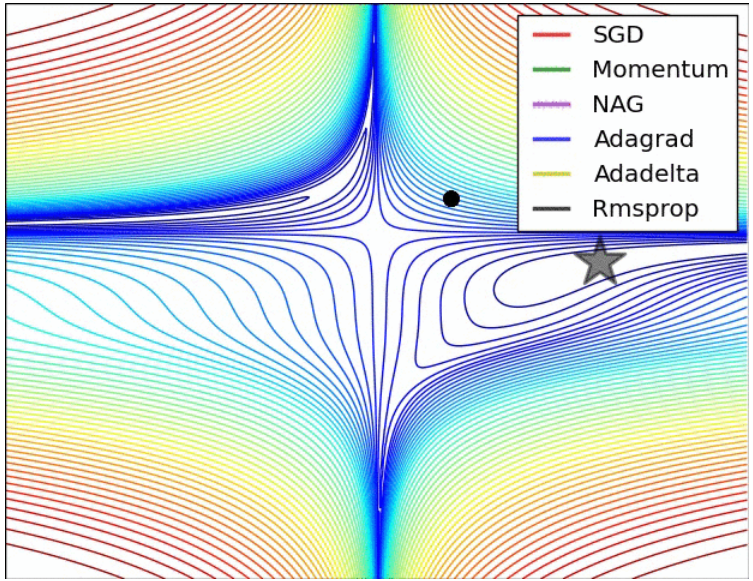


What do we know so far?

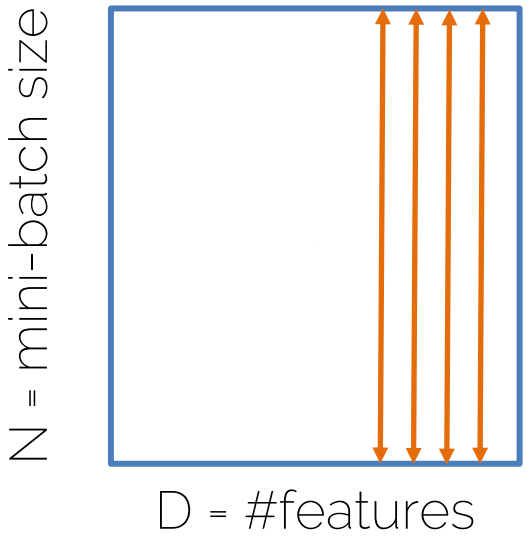


What do we know so far?

SGD Variations (Momentum, etc.)



Batchnorm



Why not only more Layers?

- We can not make networks arbitrarily complex
  - Why not just go deeper and get better?

  - No structure!!
  - It's just brute force!
  - Optimization becomes hard
  - Performance plateaus / drops!

Dealing with images

Using CNNs in Computer Vision

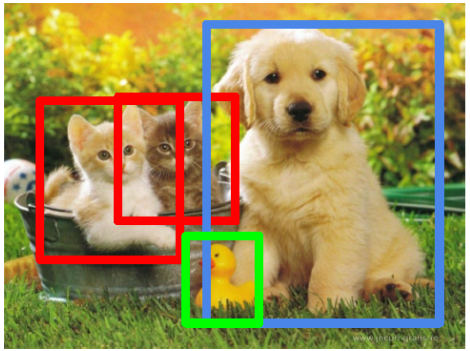
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation

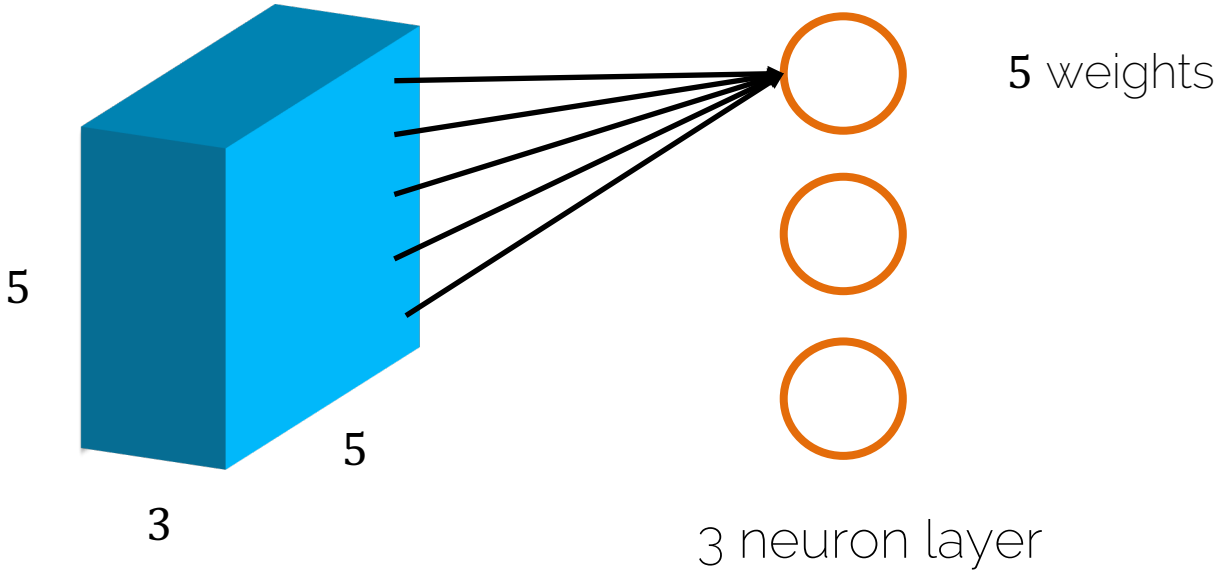
CAT, DOG, DUCK

Single object

Multiple objects

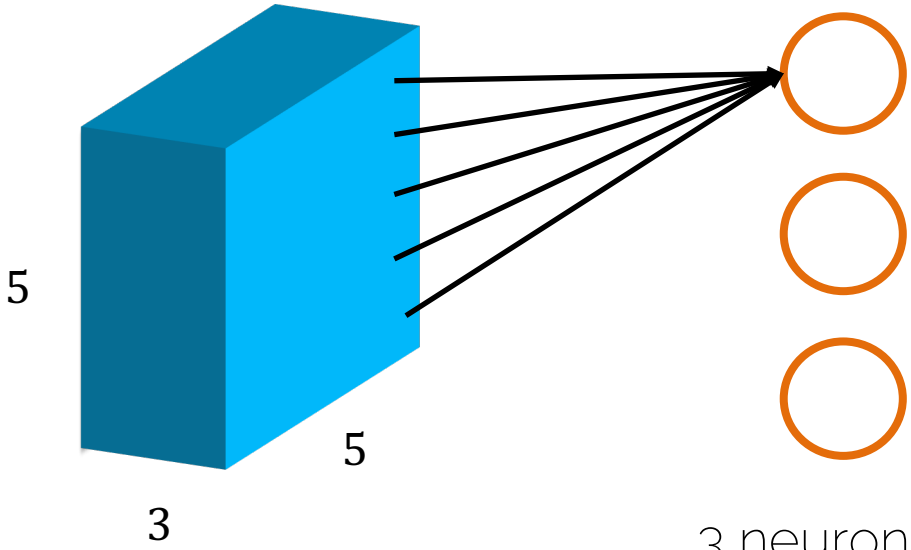
FC layers on images

- How to process a tiny image with FC layers



FC layers on images

- How to process a tiny image with FC layers



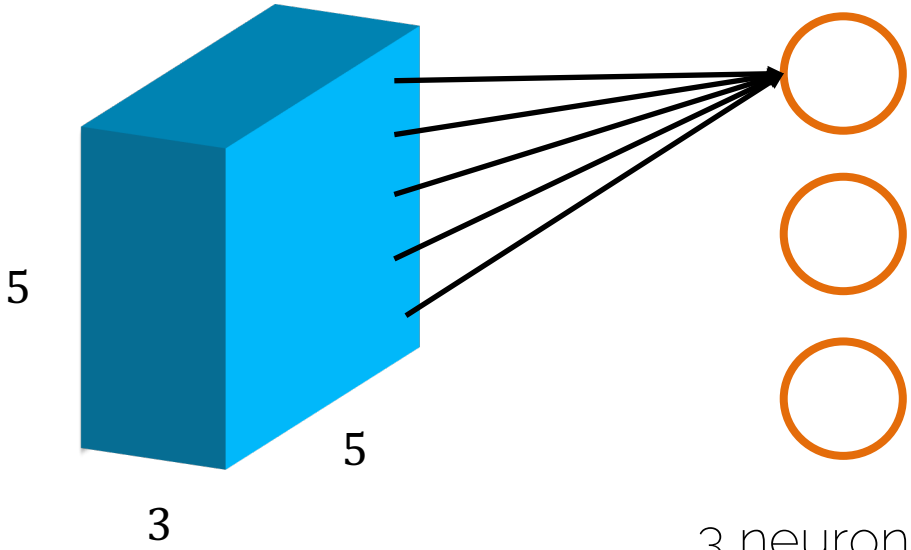
25 weights

3 neuron layer

For the whole 5x5 image on

FC layers on images

- How to process a tiny image with FC layers



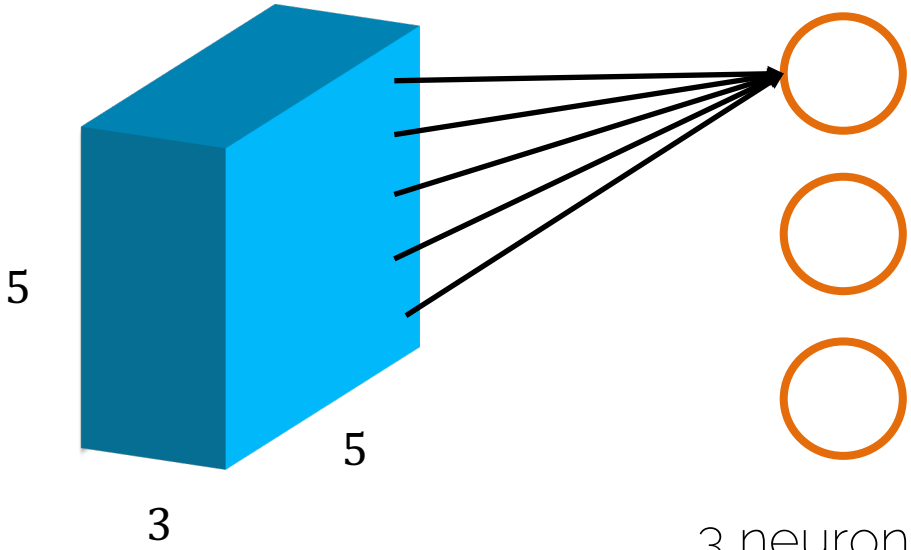
75 weights

3 neuron layer

For the whole 5x5 image on the three channels

FC layers on images

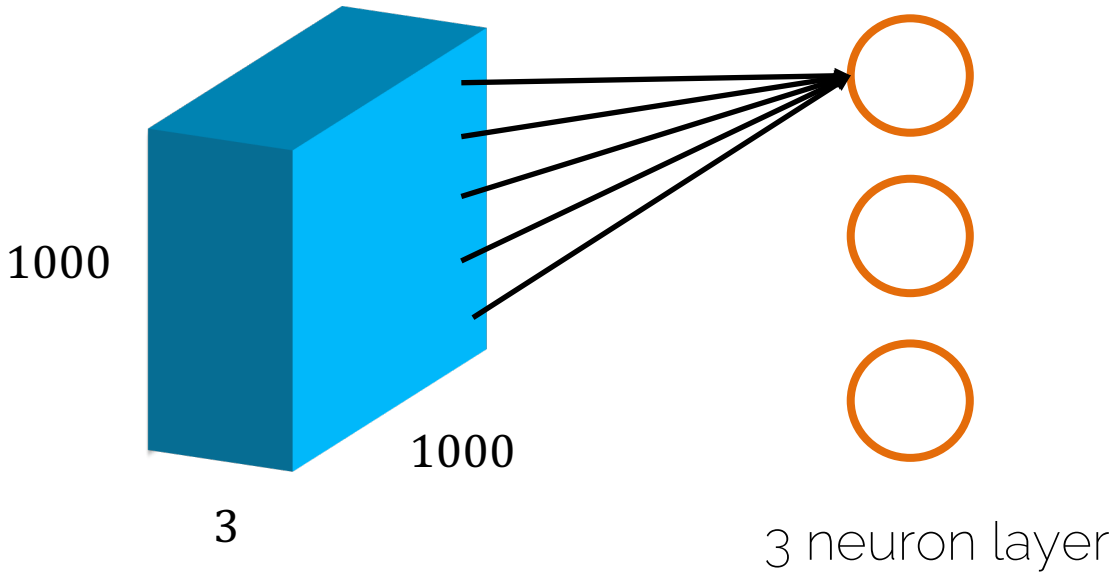
- How to process a tiny image with FC layers



For the whole 5x5 image on the three channels **per neuron**

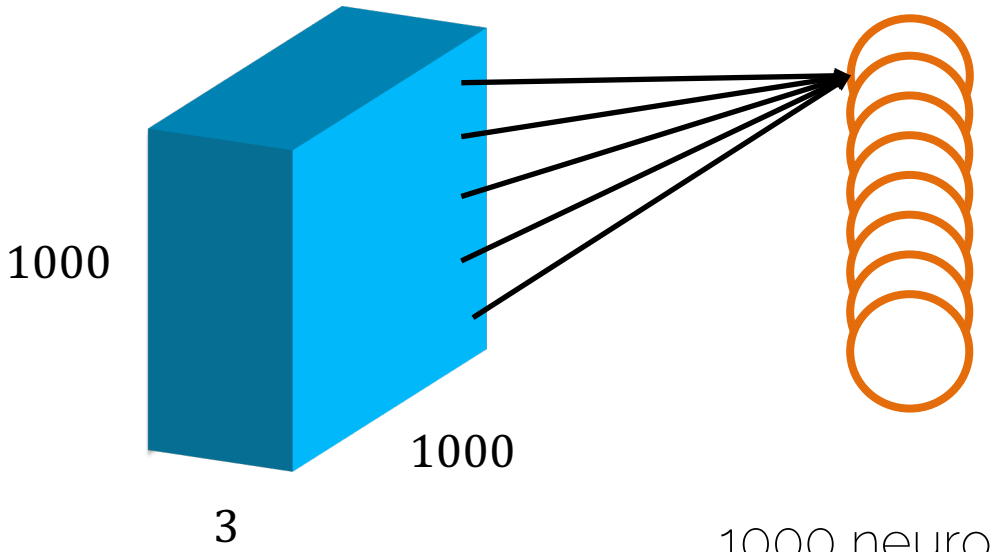
FC layers on images

- How to process a **normal** image with FC layers



FC layers on images

- How to process a **normal** image with FC layers



IMPRACTICAL

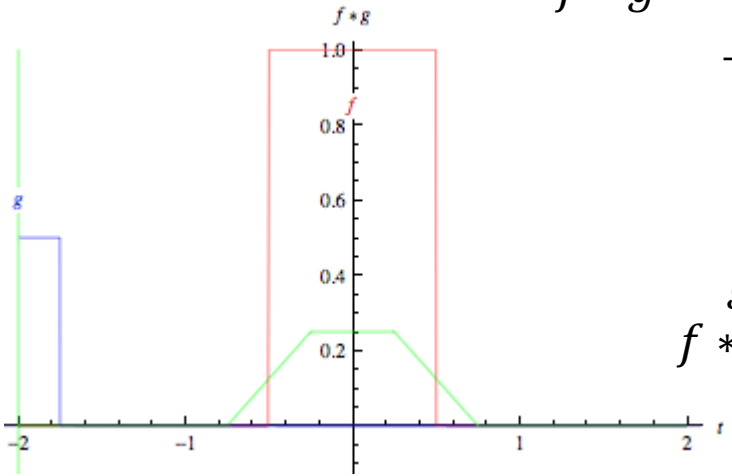
An alternative to Fully-Connected

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

Convolutions

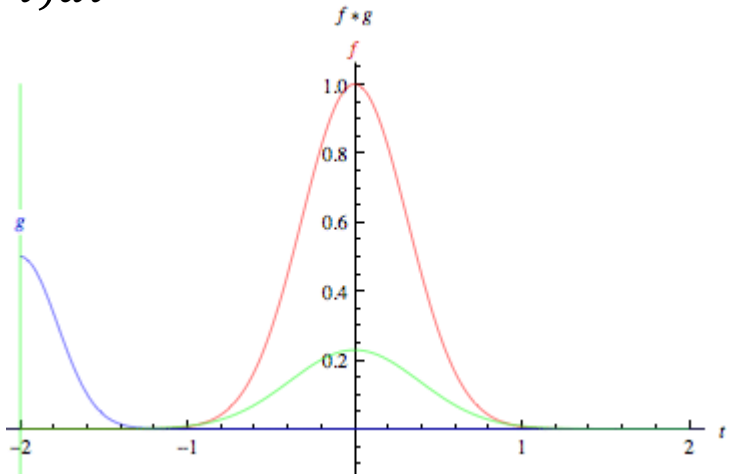
What are Convolutions?

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



Convolution of two box functions

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

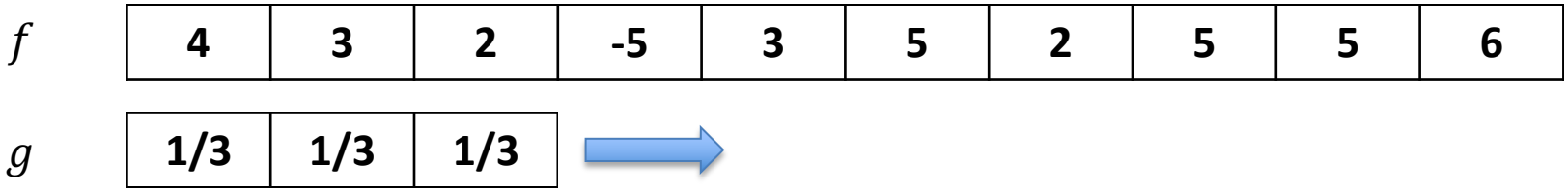


Convolution of two Gaussians

application of a filter to a function
the 'smaller' one is typically called the filter kernel

What are Convolutions?

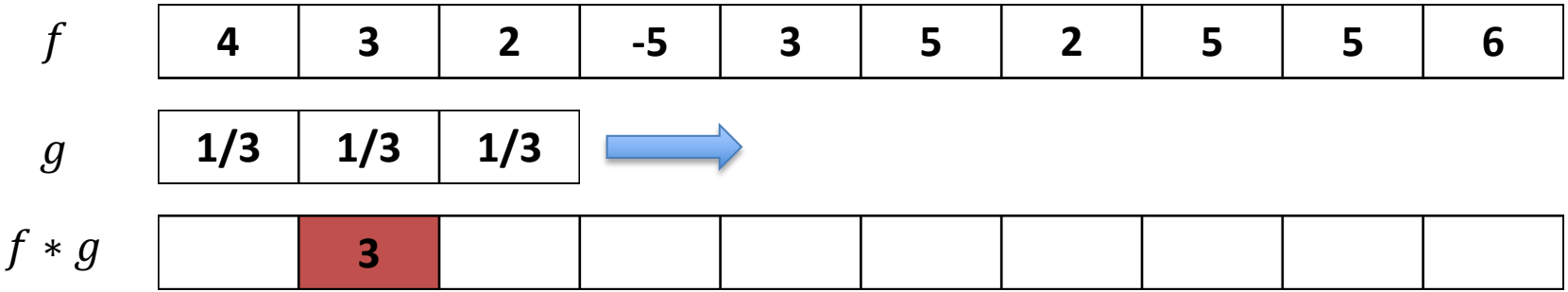
Discrete case: box filter



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

What are Convolutions?

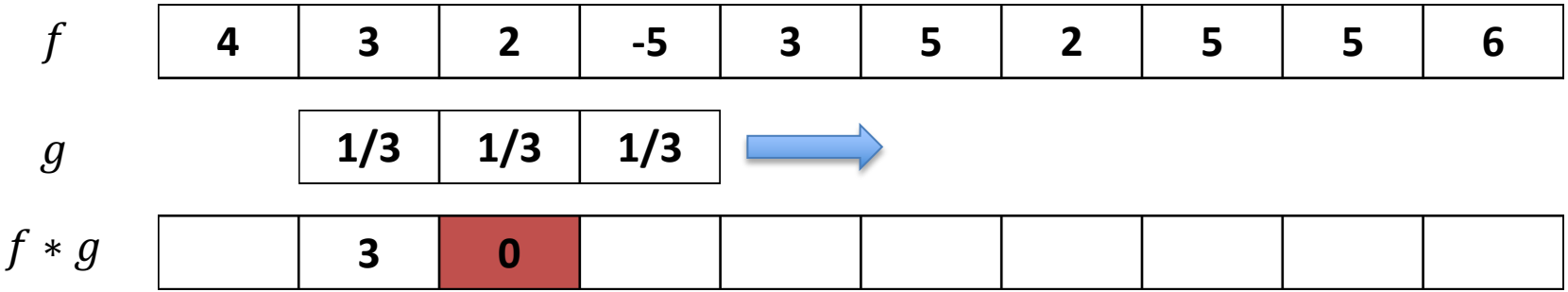
Discrete case: box filter



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

What are Convolutions?

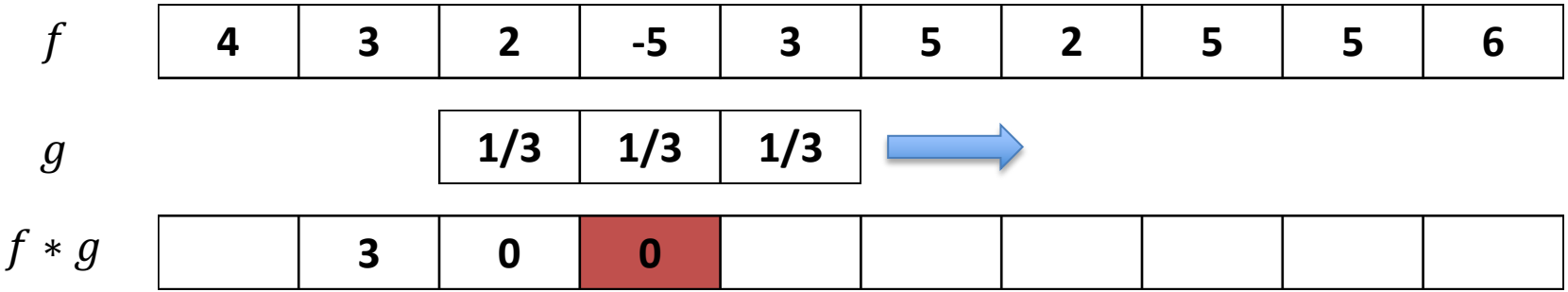
Discrete case: box filter



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

What are Convolutions?

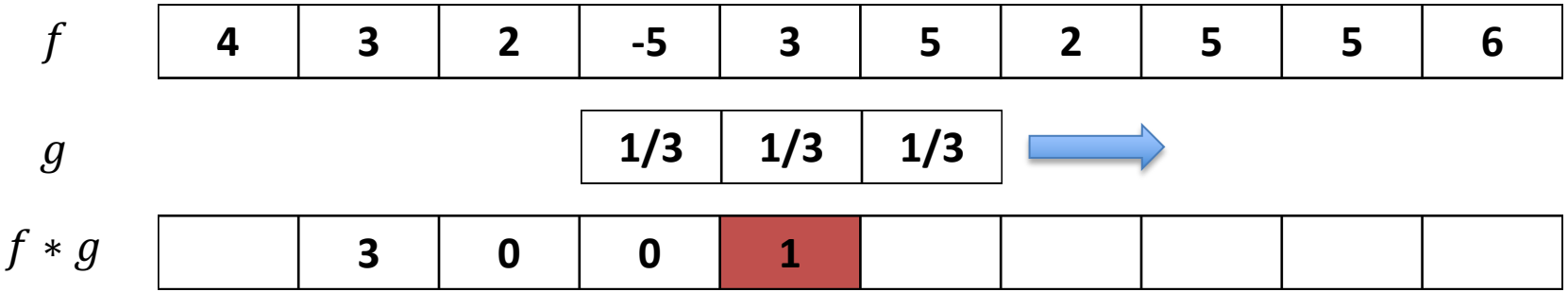
Discrete case: box filter



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

What are Convolutions?

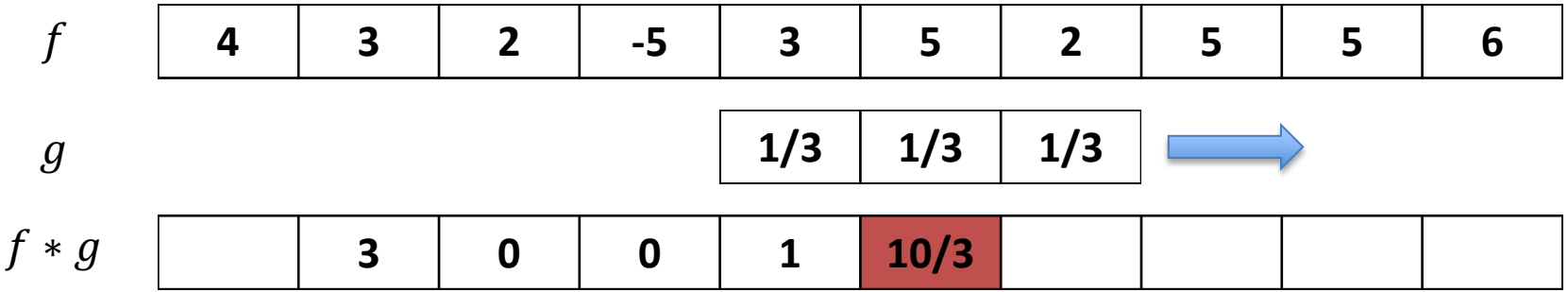
Discrete case: box filter



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

What are Convolutions?

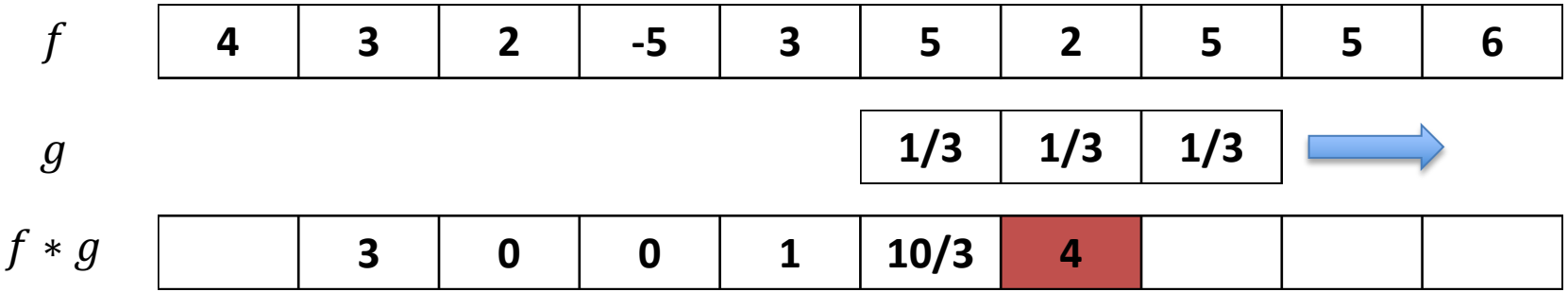
Discrete case: box filter



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

What are Convolutions?

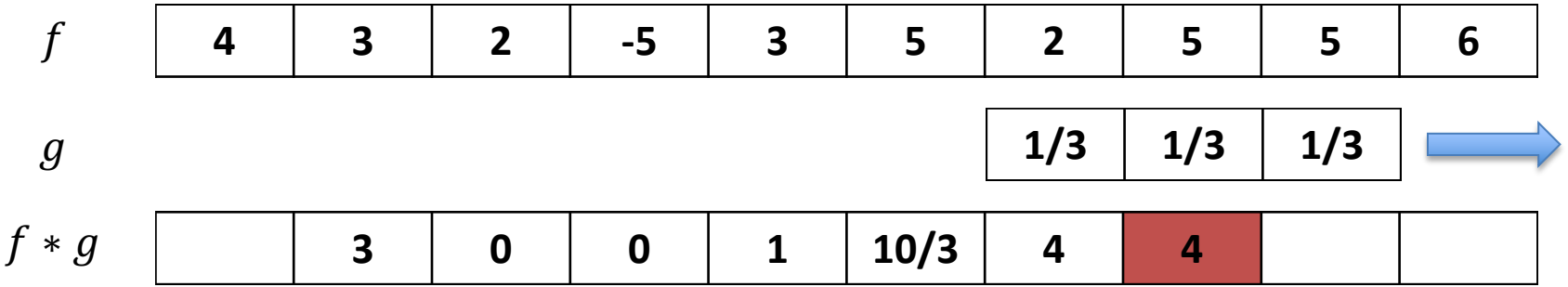
Discrete case: box filter



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

What are Convolutions?

Discrete case: box filter



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

What are Convolutions?

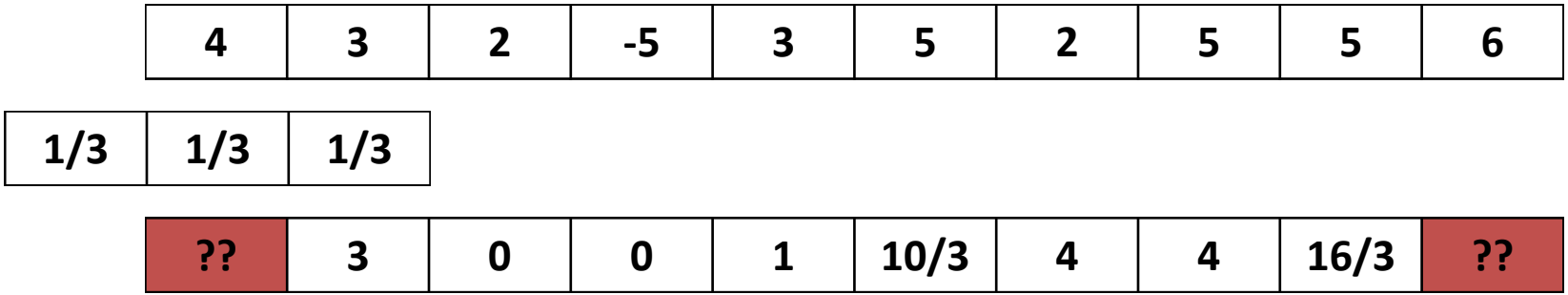
Discrete case: box filter

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?

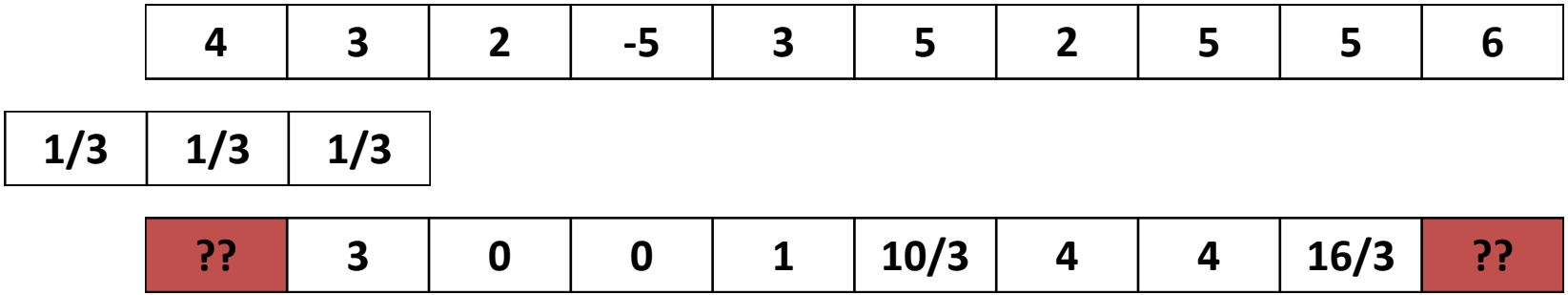
Discrete case: box filter



What to do at boundaries?

What are Convolutions?

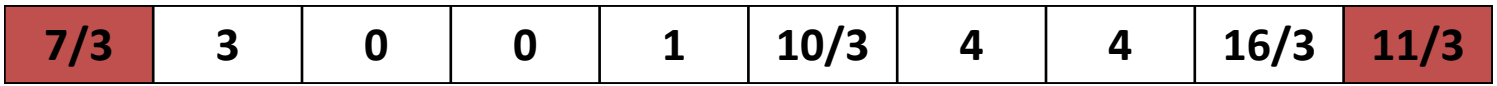
Discrete case: box filter



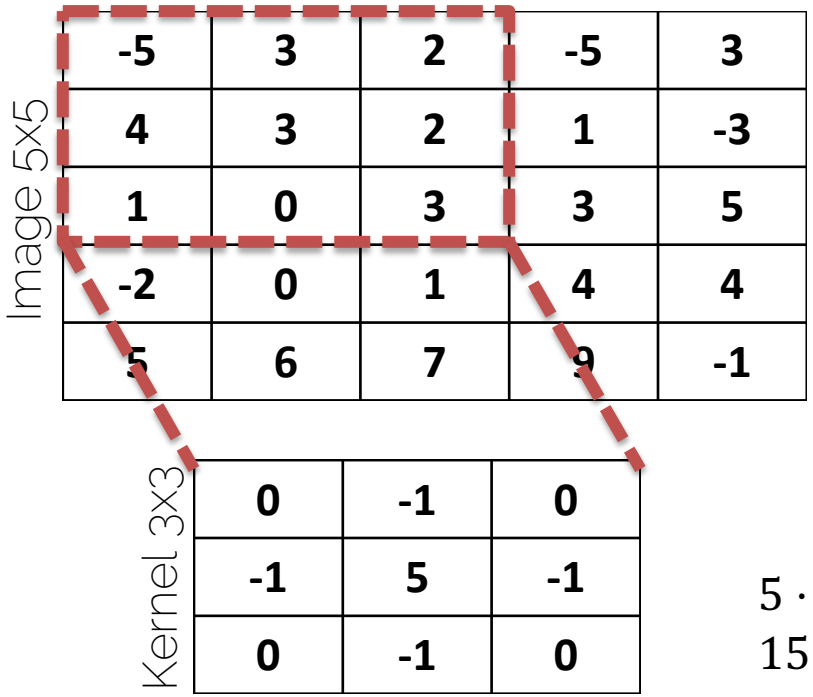
What to do at boundaries?

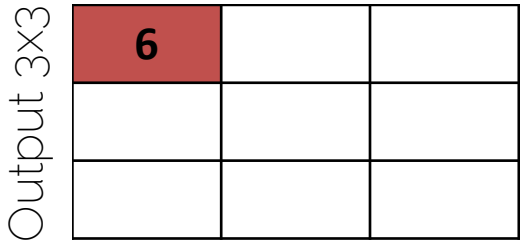
1) Shrink

2) Pad often '0'



Convolutions on Images





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

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

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

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
-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

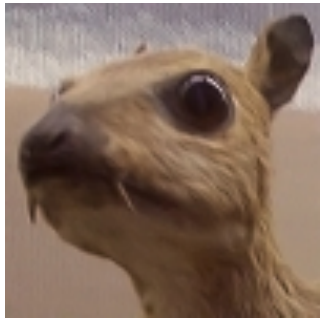
6	1	8
-7	9	2
-5	-9	3

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

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

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

LET'S LEARN THESE FILTERS!

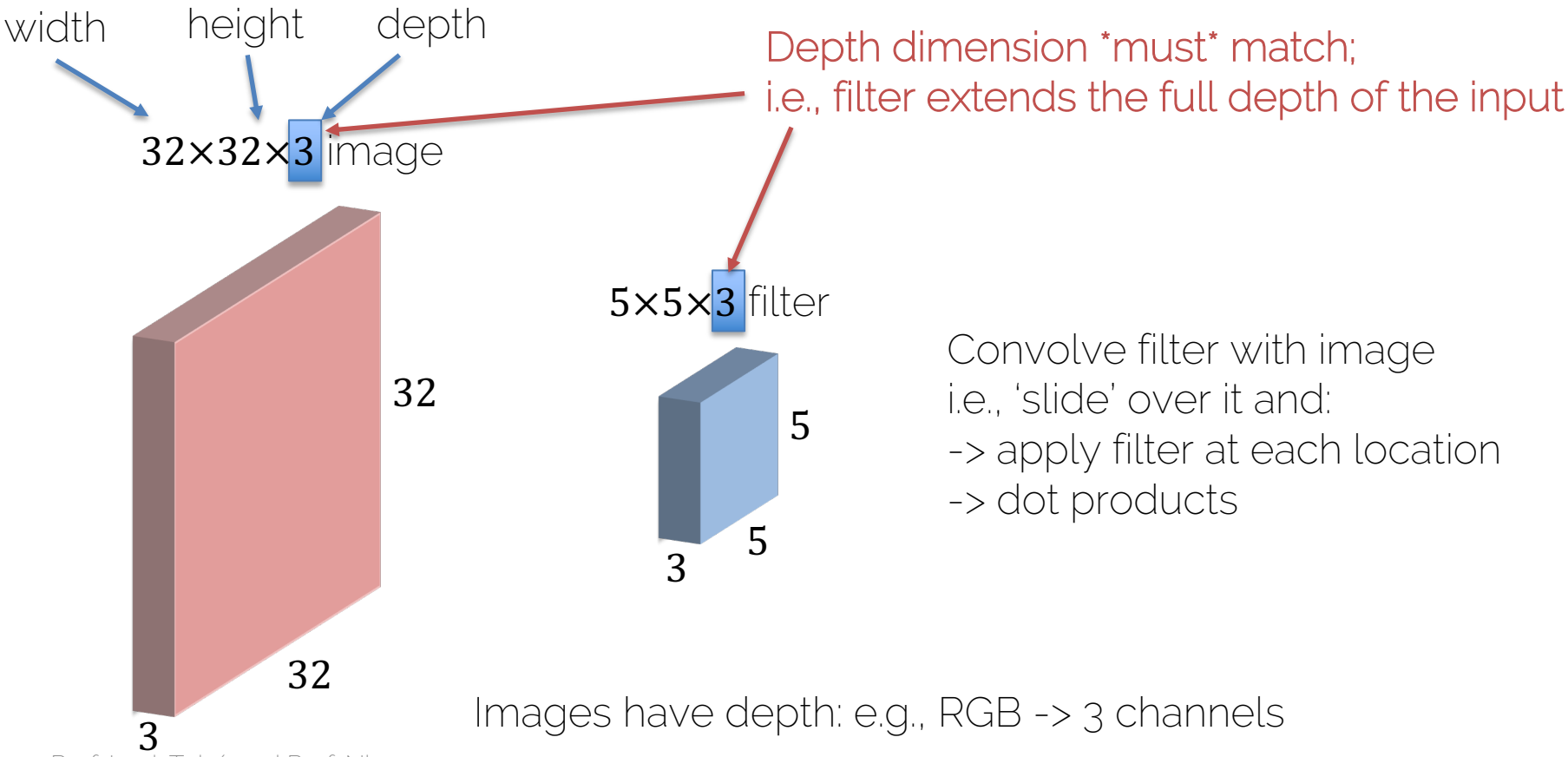
Sharpen

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

Gaussian blur

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

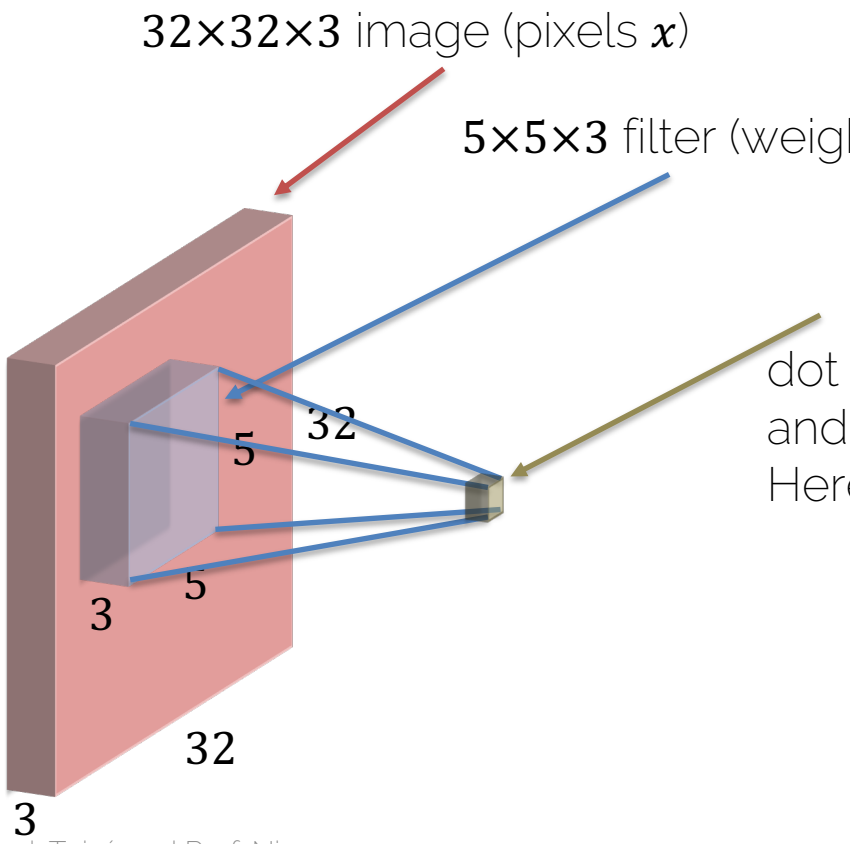
Convolutions on RGB Images



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

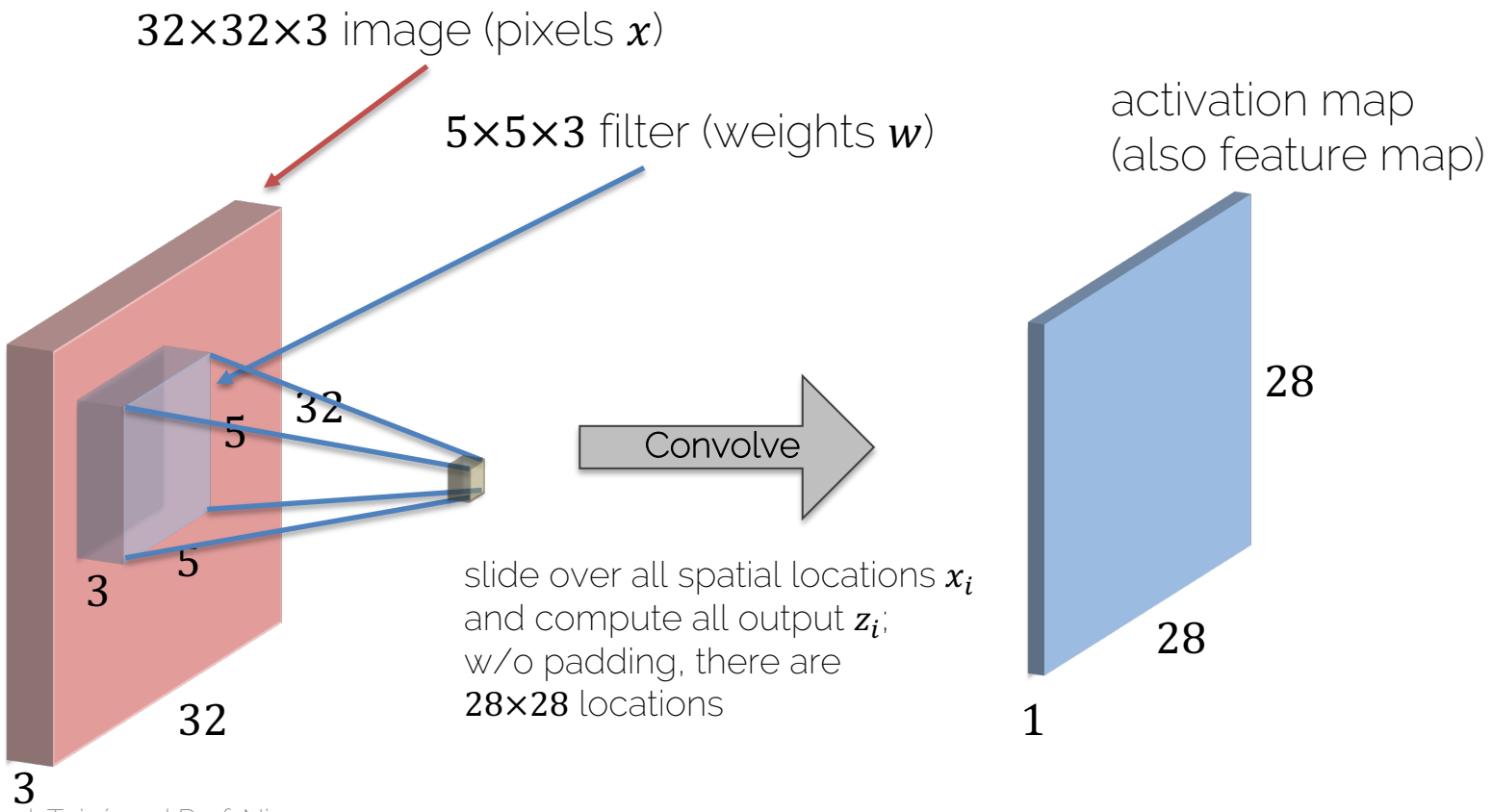


1 number:
dot product between filter weights w
and $x_i - th$ chunk of the image
Here: $5 \cdot 5 \cdot 3 = 75$-dim dot product + bias

$$z_i = w^T x_i + b$$

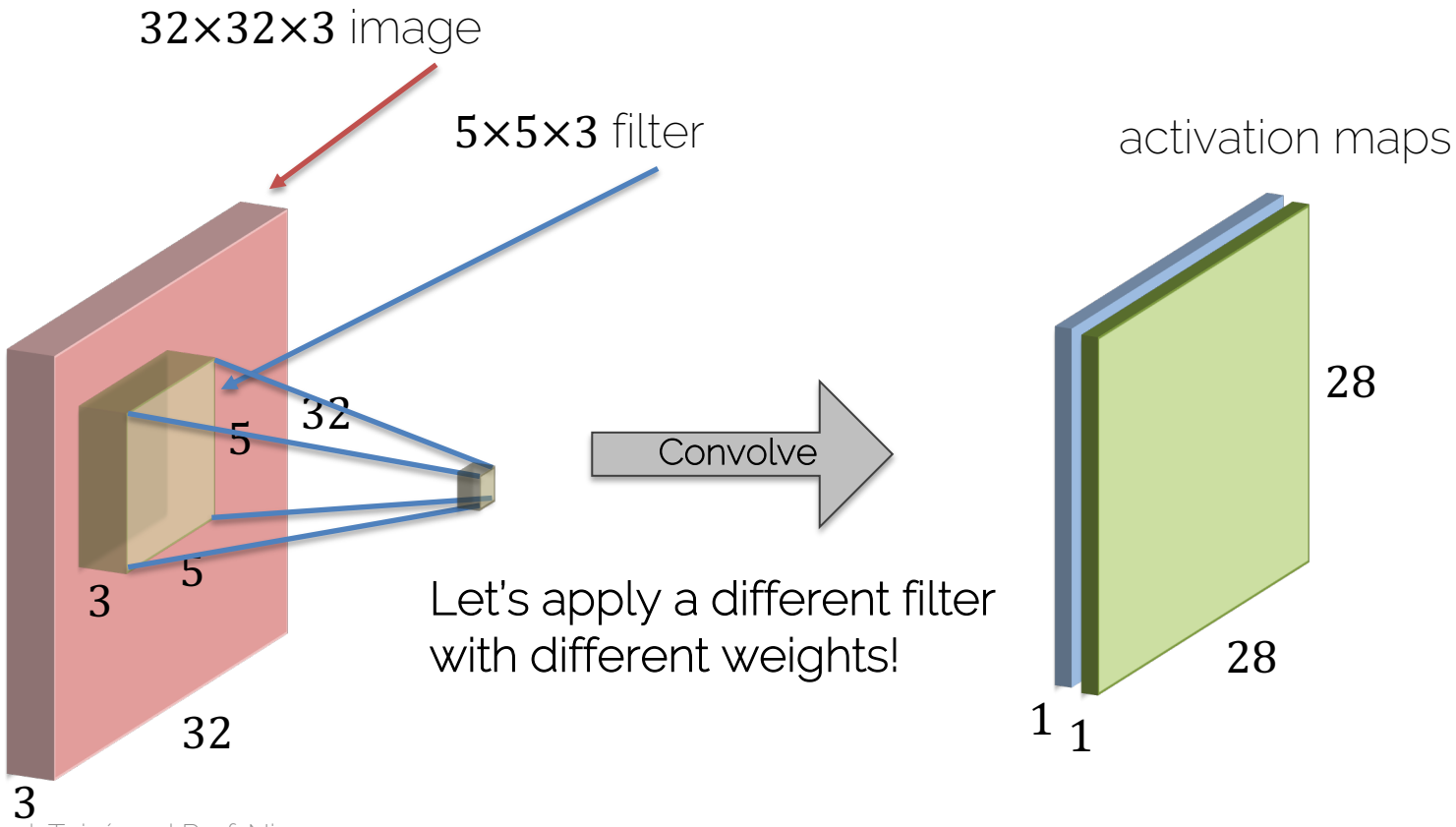
5×5×3 5×5×3 1

Convolutions on RGB Images

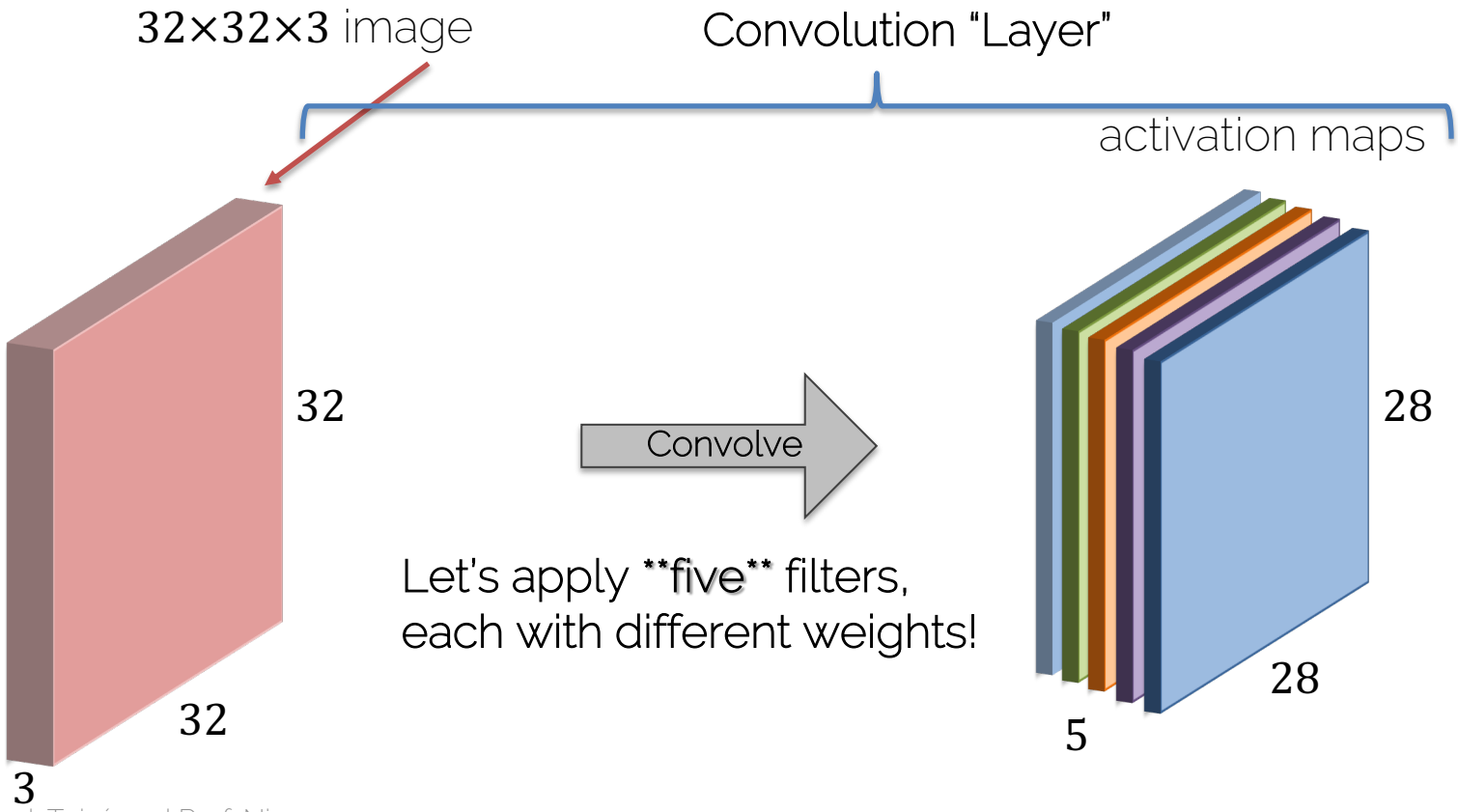


Convolution Layer

Convolution Layer



Convolution Layer

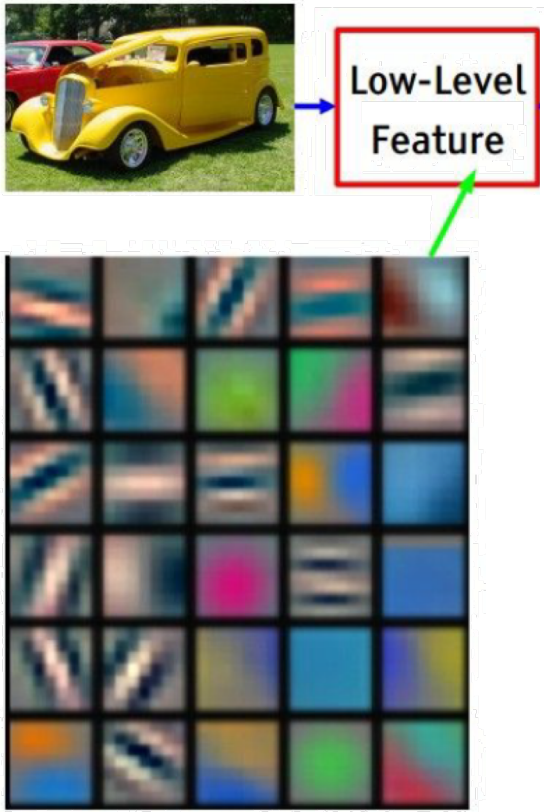


Convolution Layer

- 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

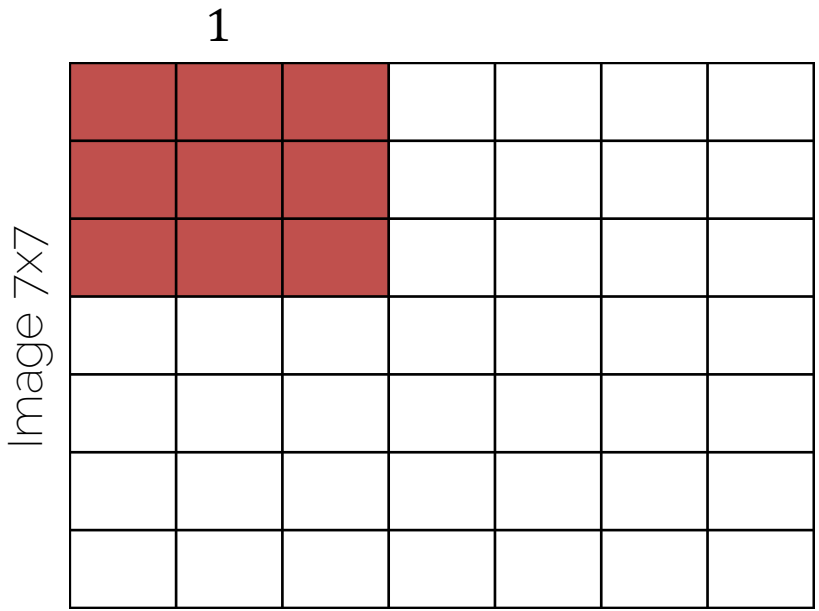


- Each filter captures different image characteristics: horizontal edges, vertical edges, circles, squares….

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

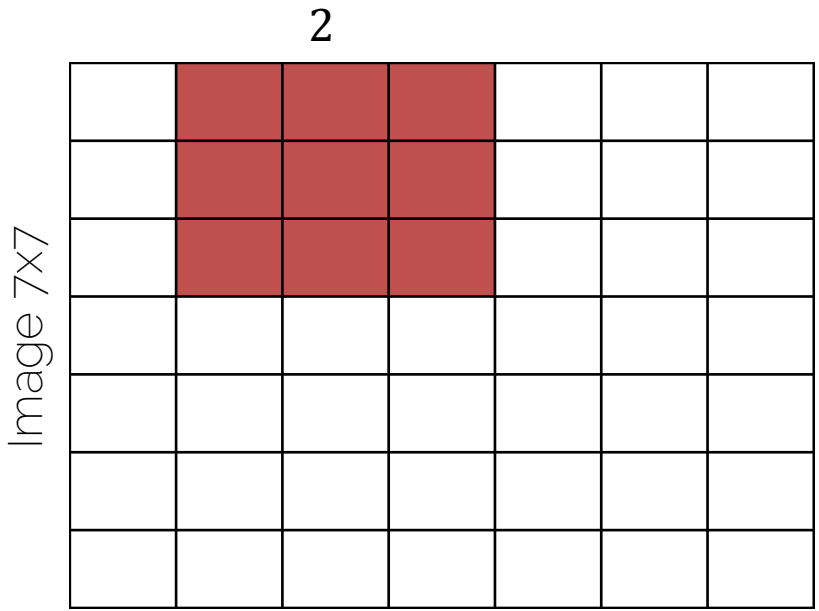
Dimensions of a convolution layer

Convolution Layers: Dimensions



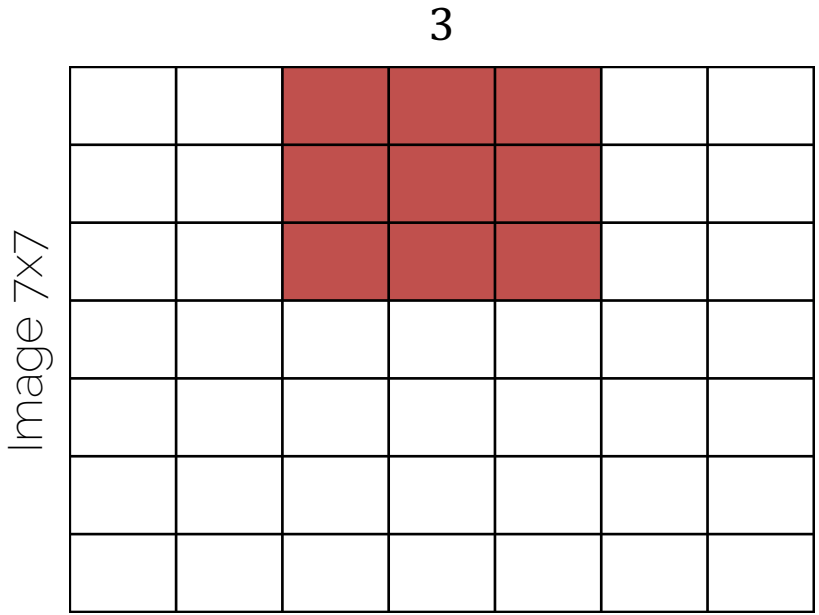
Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Dimensions



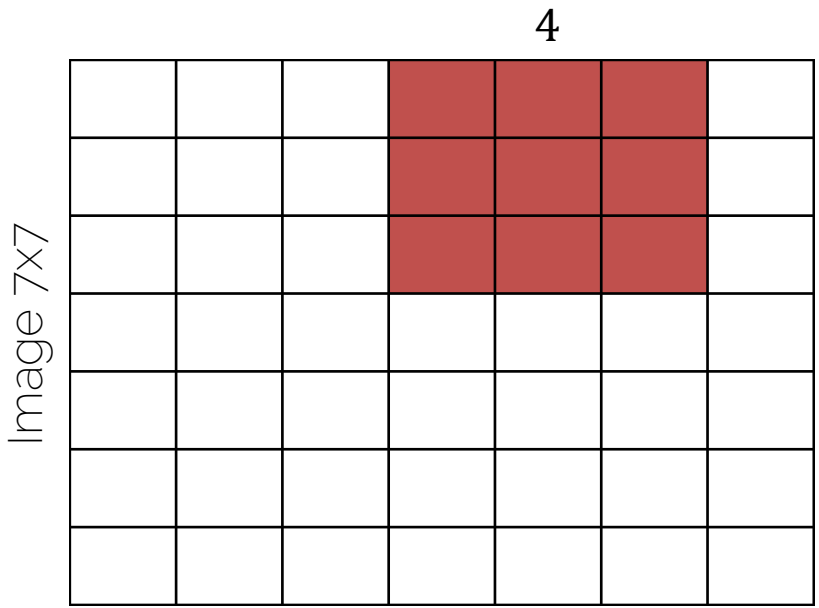
Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Dimensions



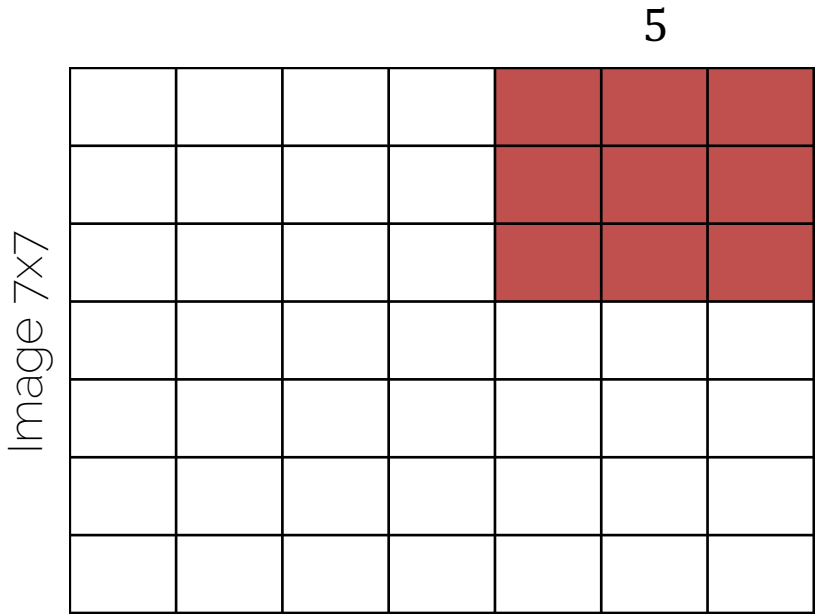
Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Dimensions



Input: 7×7
Filter: 3×3
Output: 5×5

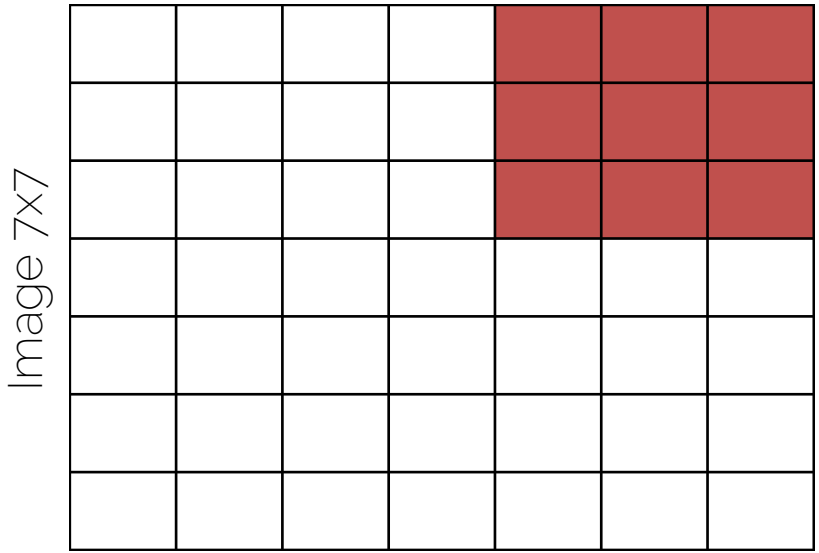
Convolution Layers: Dimensions



Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Stride

With a **stride** of **1**

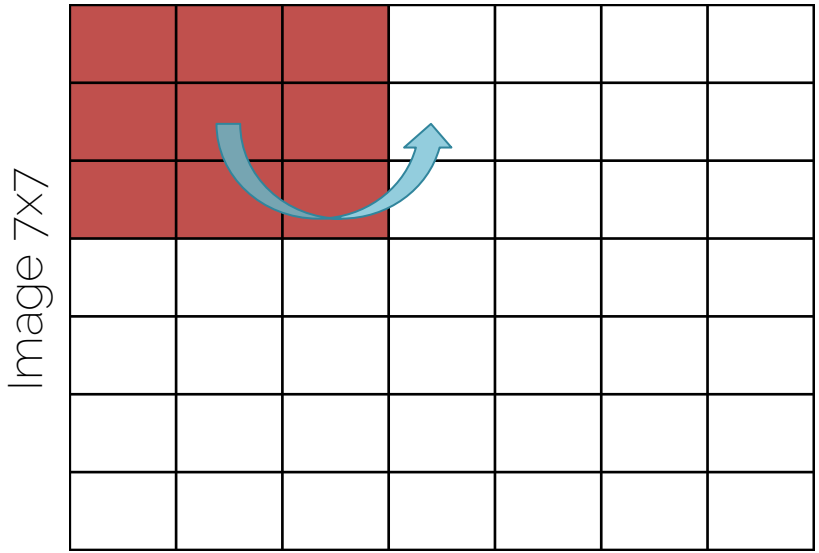


Input: **7×7**
Filter: **3×3**
Stride: **1**
Output: **5×5**

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

Convolution Layers: Stride

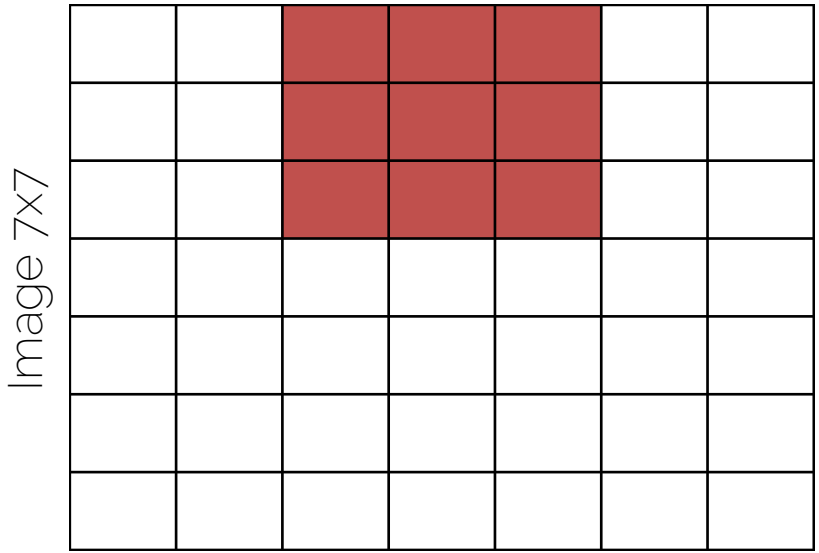
With a **stride** of **2**



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

Convolution Layers: Stride

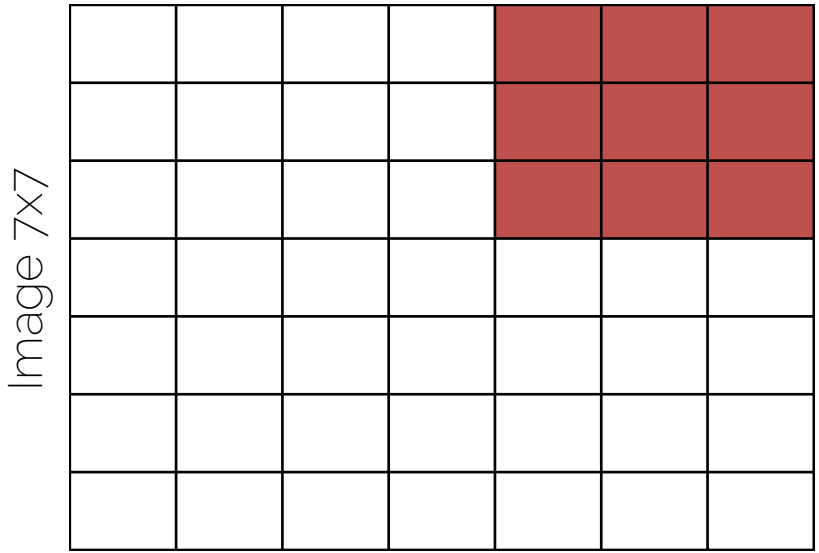
With a **stride** of **2**



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

Convolution Layers: Stride

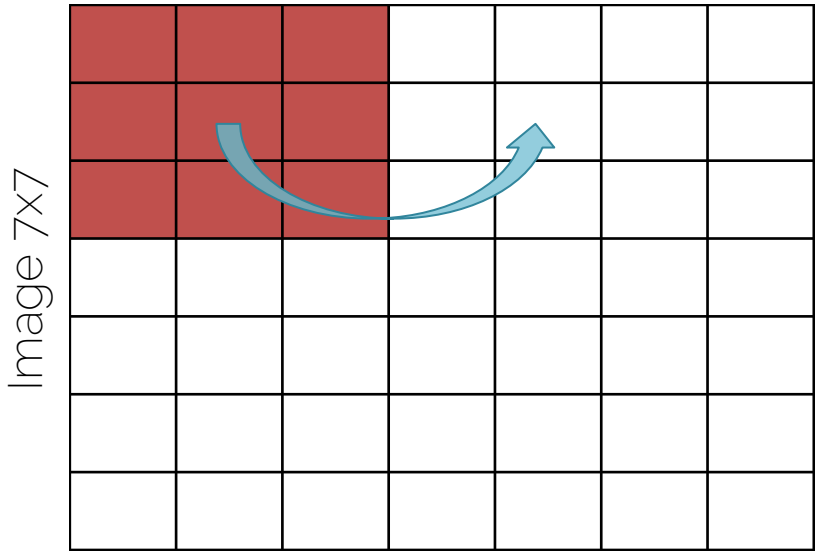
With a **stride** of **2**



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

Convolution Layers: Stride

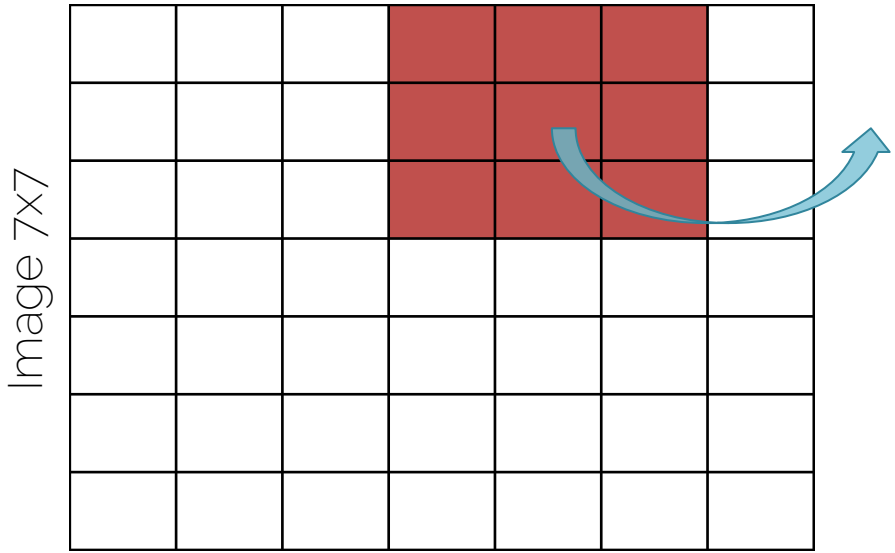
With a **stride** of **3**



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

Convolution Layers: Stride

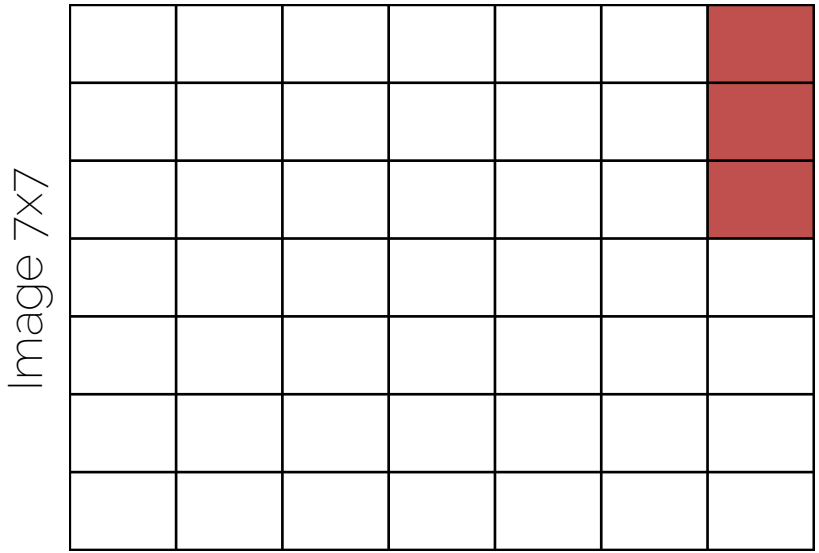
With a **stride** of **3**



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

Convolution Layers: Stride

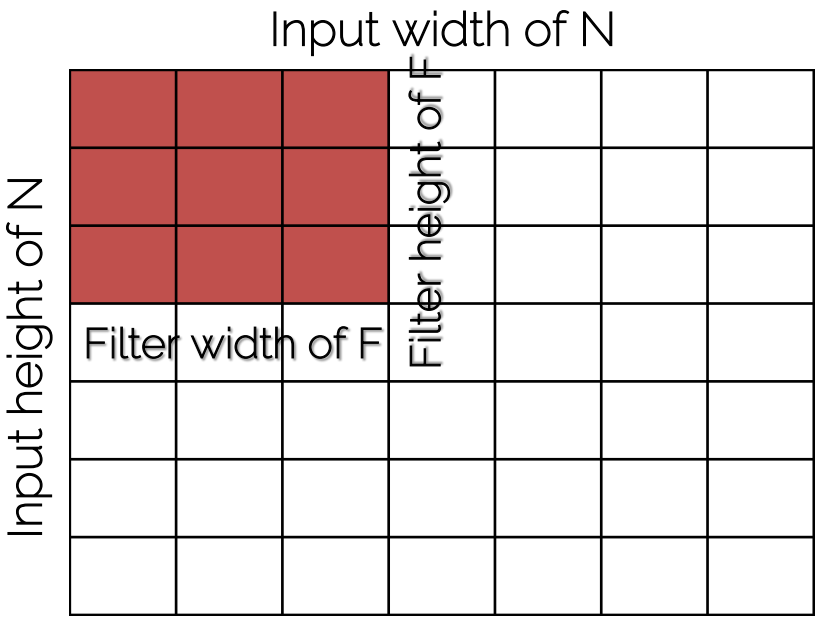
With a **stride** of **3**



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

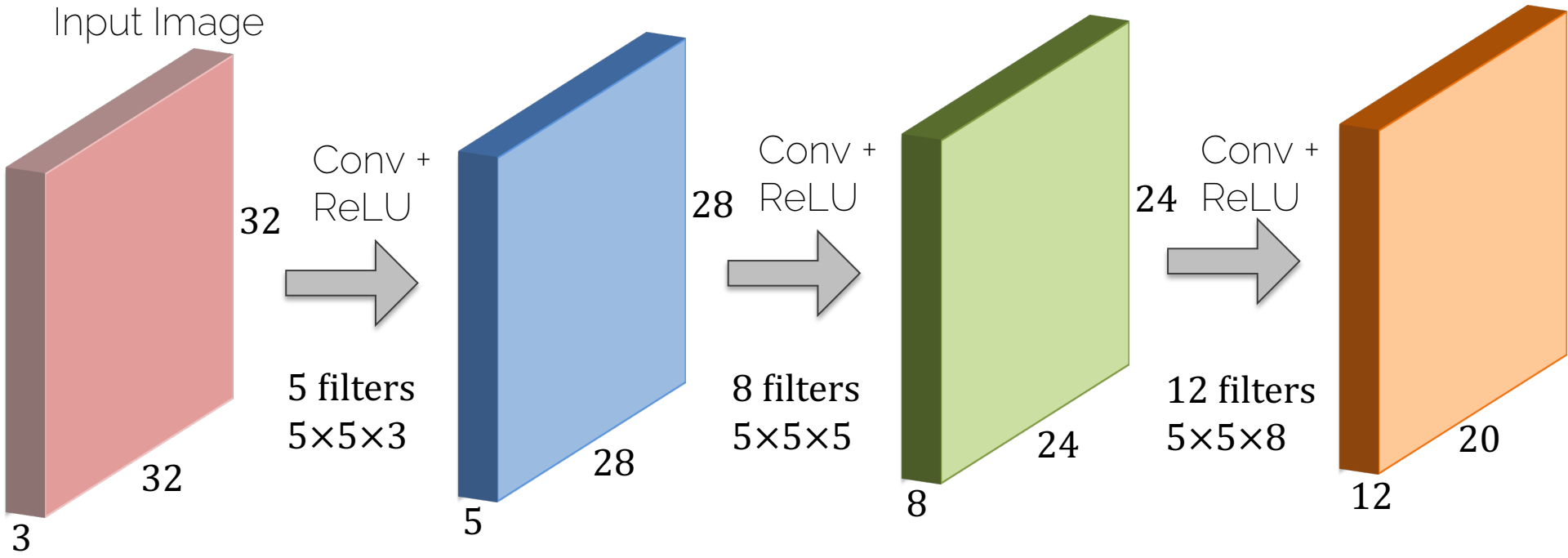
$N = 7, F = 3, S = 1$: $\frac{7-3}{1}+1 = 5$

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

$N = 7, F = 3, S = 3$: $\frac{7-3}{3}+1 = 2.3333$

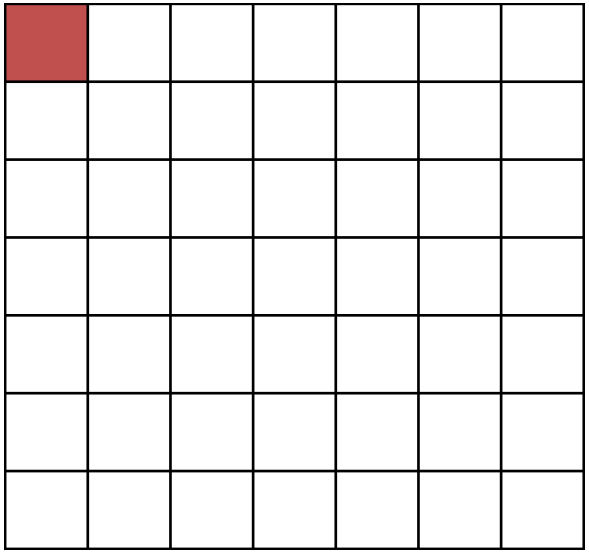
Fractions are illegal

Convolution Layers: Dimensions



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

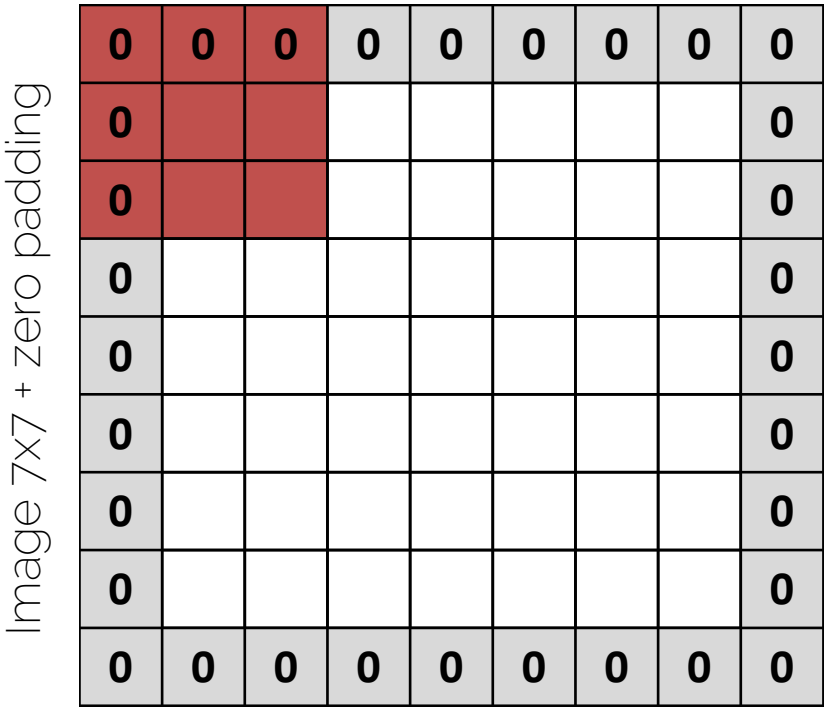
Convolution Layers: Padding



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

Image 7x7 + zero padding

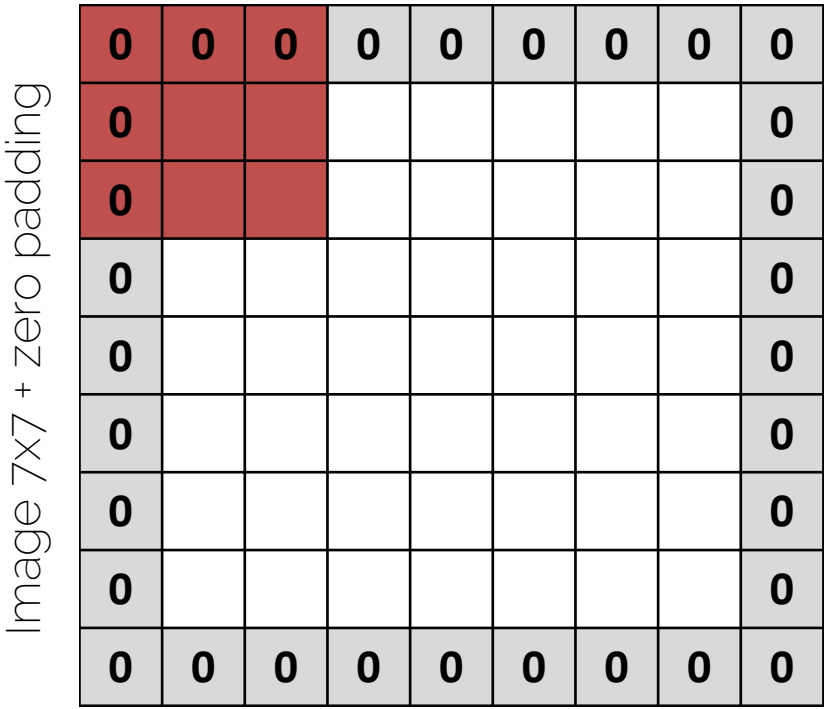
0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

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

Most common is 'zero' padding

Output Size: $(\frac{N+2\cdot P-F}{S}+1)\times(\frac{N+2\cdot P-F}{S}+1)$

Convolution Layers: Padding



Types of convolutions:

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

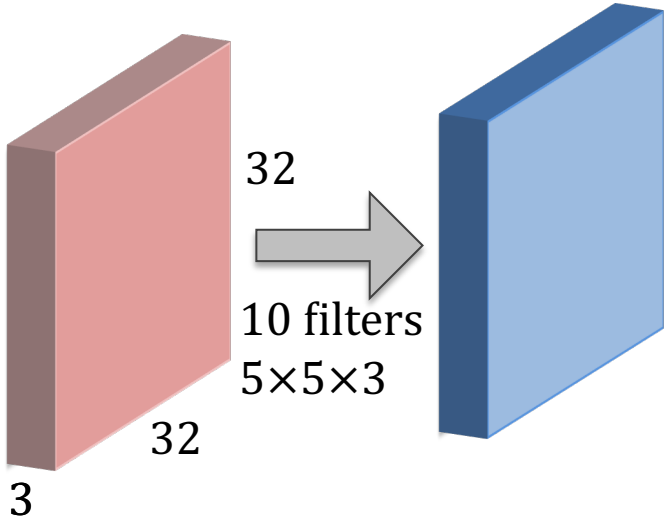
Set padding to $P = \frac{F-1}{2}$

Convolution Layers: Dimensions

Example

Input image: $32\times32\times3$
10 filters 5×5
Stride 1
Pad 2

Depth of 3 is implicitly given



Output size is:

$$\frac{32+2\cdot2-5}{1}+1=32$$

I.e., $32\times32\times10$

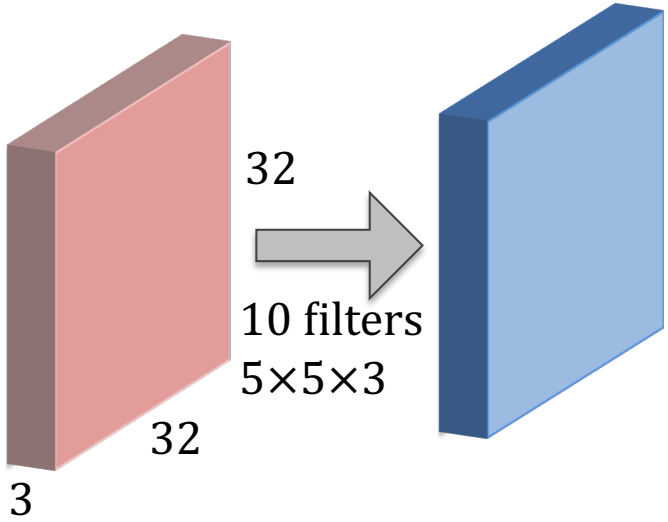
Remember

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

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., $32 \times 32 \times 10$

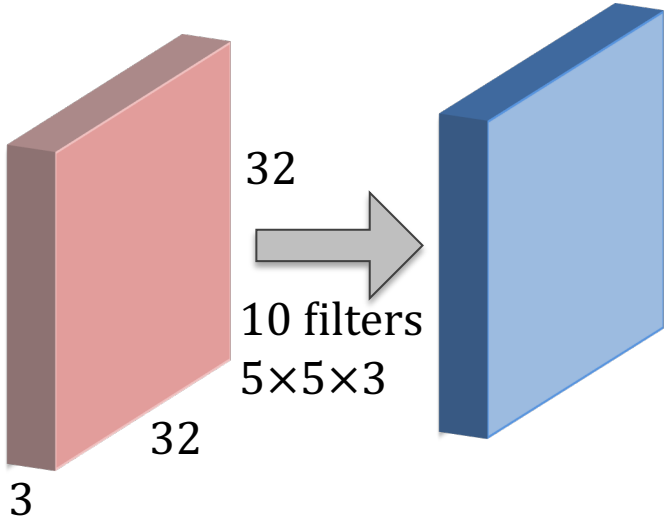
Remember

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

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

Convolution Layers: Dimensions

- Input is a volume of size $W_{in} \times H_{in} \times D_{in}$
- Four hyperparameters
  - Number of filters K
  - Spatial filter extent F
  - Stride S
  - Zero padding P
- Output volume is of size $W_{out} \times H_{out} \times D_{out}$
  - $W_{out} = \frac{W_{in} - F + 2 \cdot P}{S} + 1$
  - $H_{out} = \frac{H_{in} - F + 2 \cdot P}{S} + 1$
  - $D_{out} = K$
- There are $F \cdot F \cdot D_{in}$ weights per filter; i.e., a total of $(F \cdot F \cdot D_{in}) \cdot K$ weights and K biases
- In the output volume, the D-th depth slice of size $(W_{out} \times H_{out})$ is the result of the convolution of the D-th over the input volume with a stride of S, and offset by its bias

Common settings:

$K = '$ powers of $2', \text{e.g.}, 32, 64, 128, 512$

$F = 3,\ S = 1,\ P = 1$

$F = 5,\ S = 1,\ P = 2$

$F = 5,\ S = 2,\ P = (whatever\ fits)$

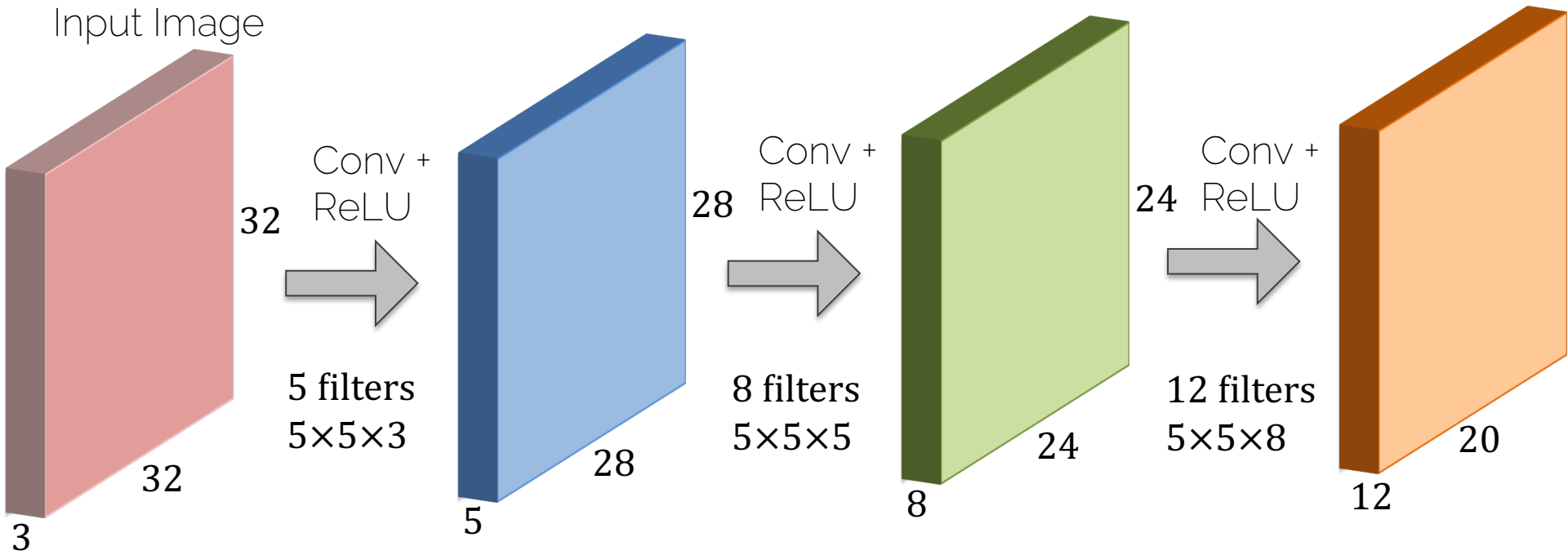
$F = 1,\ S = 1,\ P = 0$

Slide by Li/Karpathy/Johnson

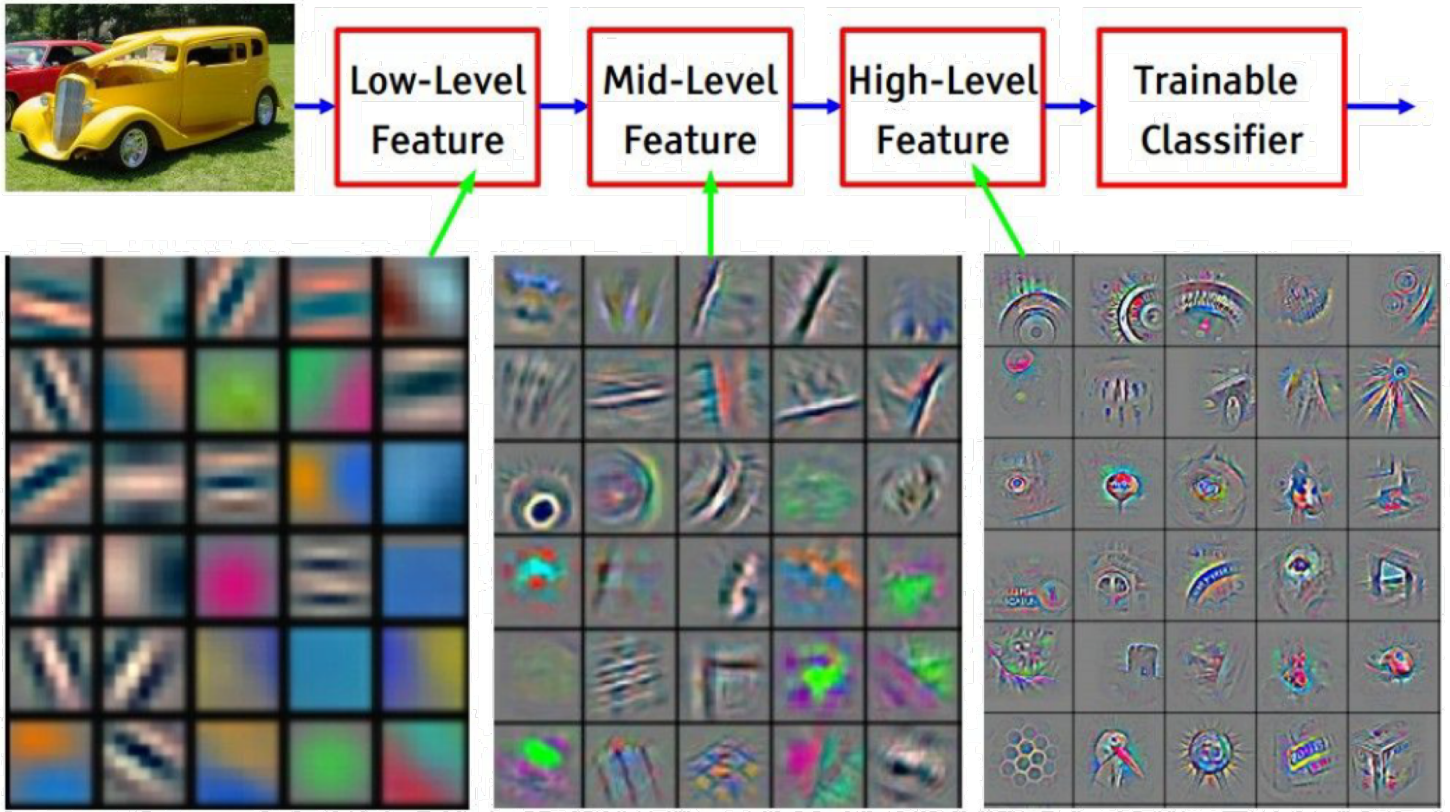
Convolutional Neural Network (CNN)

CNN Prototype

ConvNet is concatenation of Conv Layers and activations

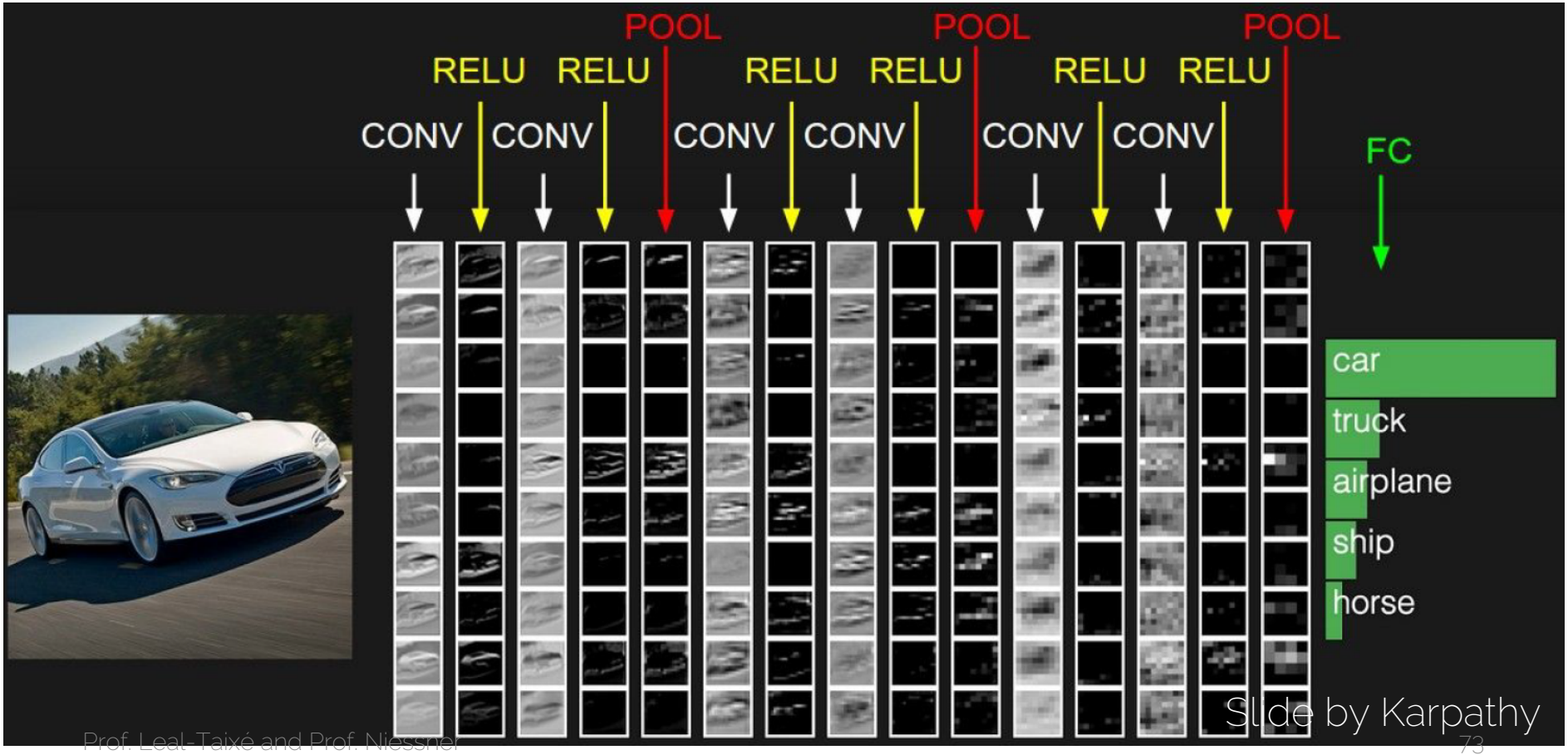


CNN learned filters



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

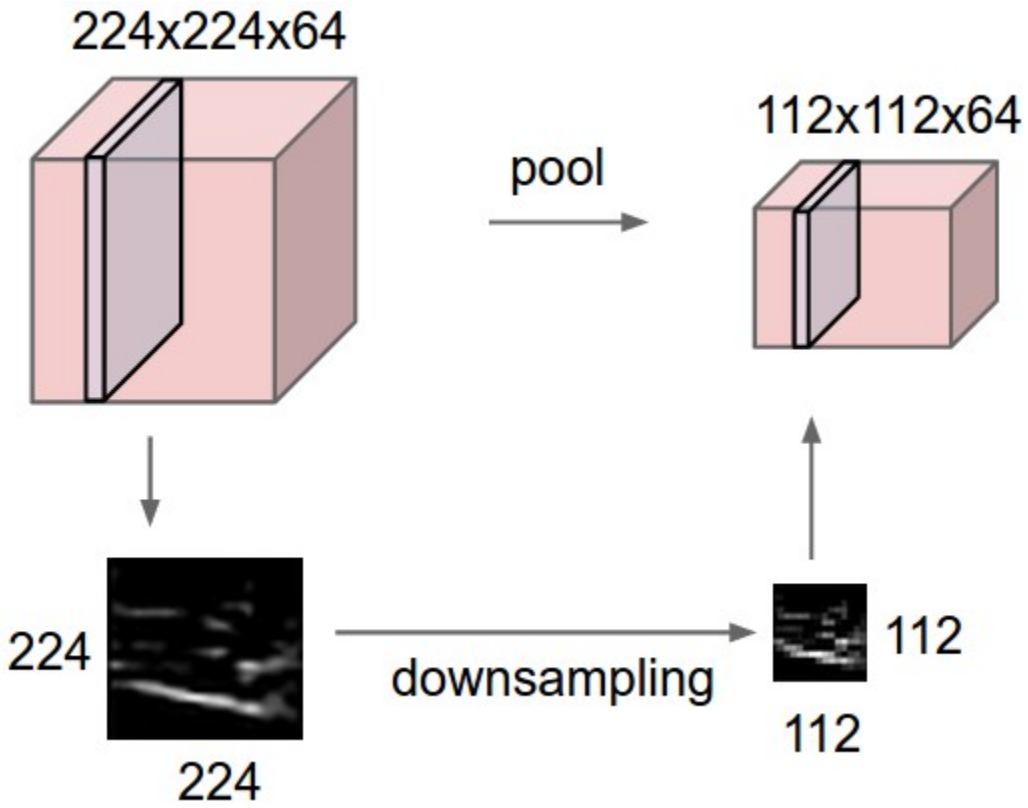
CNN Prototype



Slide by Karpathy

Pooling

Pooling Layer



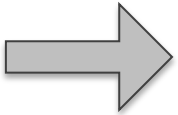
Slide by Li/Karpathy/Johnson

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 $W_{in} \times H_{in} \times D_{in}$
- Four hyperparameters
  - Spatial filter extent F
  - Stride S

} Filter count and padding make no sense here

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

Pooling Layer

- Input is a volume of size $W_{in} \times H_{in} \times D_{in}$
- Four hyperparameters
  - Spatial filter extent F
  - Stride S
- Output volume is of size $W_{out} \times H_{out} \times D_{out}$
  - $W_{out} = \frac{W_{in} - F}{S} + 1$
  - $H_{out} = \frac{H_{in} - F}{S} + 1$
  - $D_{out} = D_{in}$
- Does not contain parameters; e.g., its fixed function

Common settings:

$F = 2, S = 2$

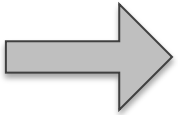
$F = 3, S = 2$

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

Max pool with
2×2 filters and stride 2

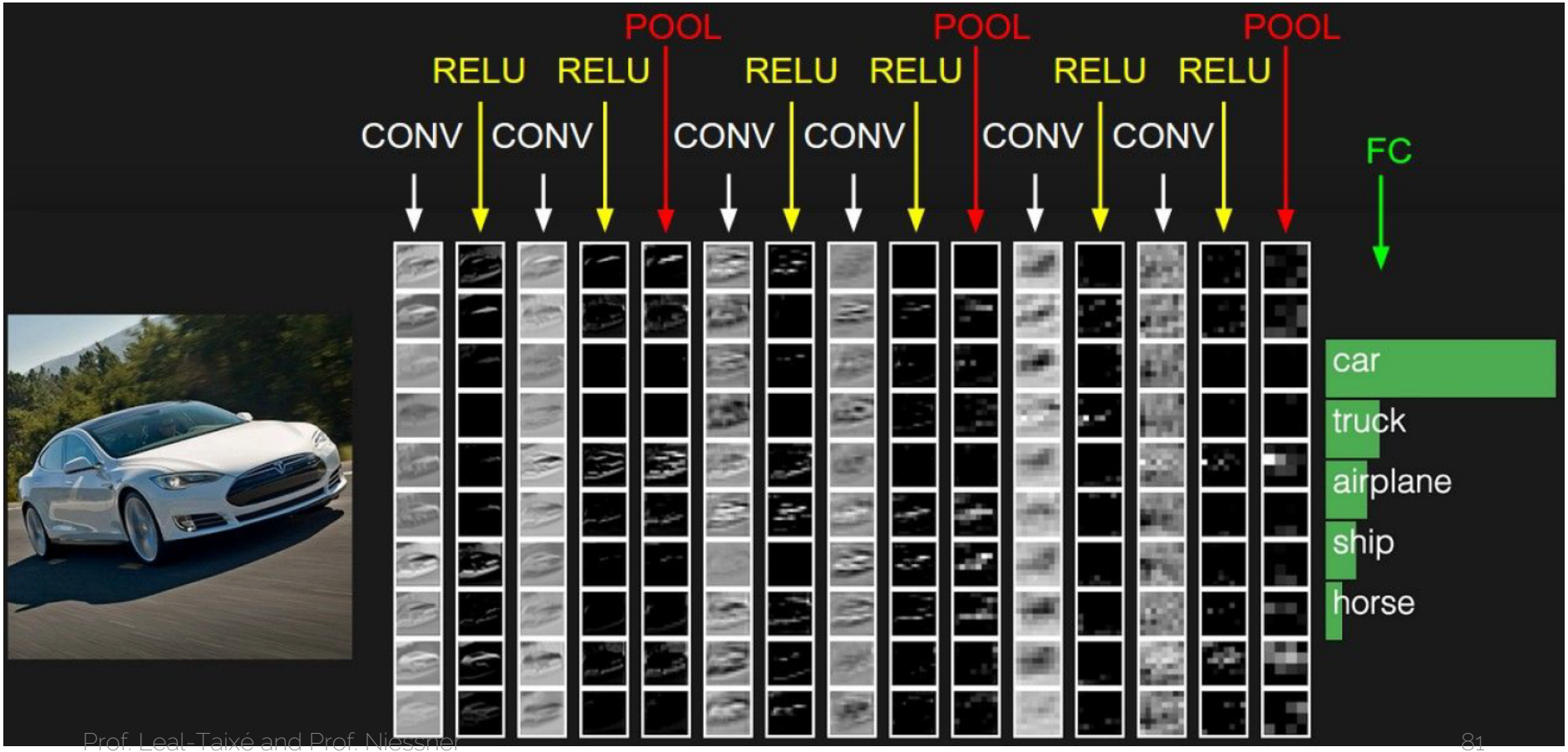


'Pooled' output

2.5	6
1.75	3

- Typically used deeper in the network

Convolutional Neural Network



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

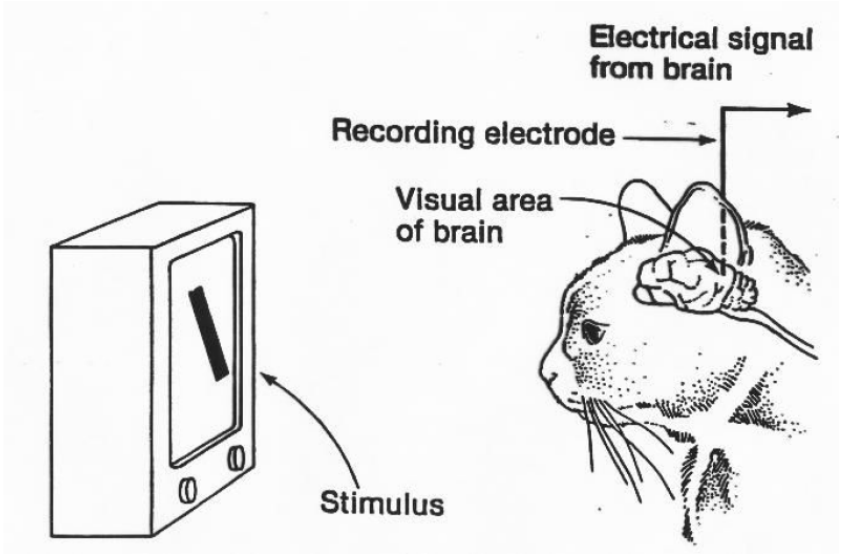
Convolutions vs Fully-Connected

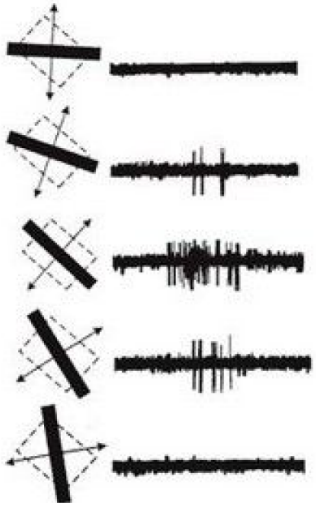
- In contrast to fully-connected layers, we want to restrict the degrees of freedom
  - FC is somewhat brute force
  - Convolutions are structured

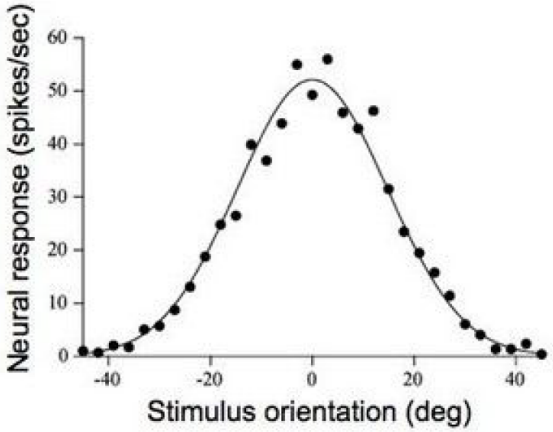
- Sliding window to with the same filter parameters to extract image features
  - Concept of weight sharing
  - Extract same features independent of location

Convolutional Neural Network

- Turns out that CNNs are similar to the visual cortex:



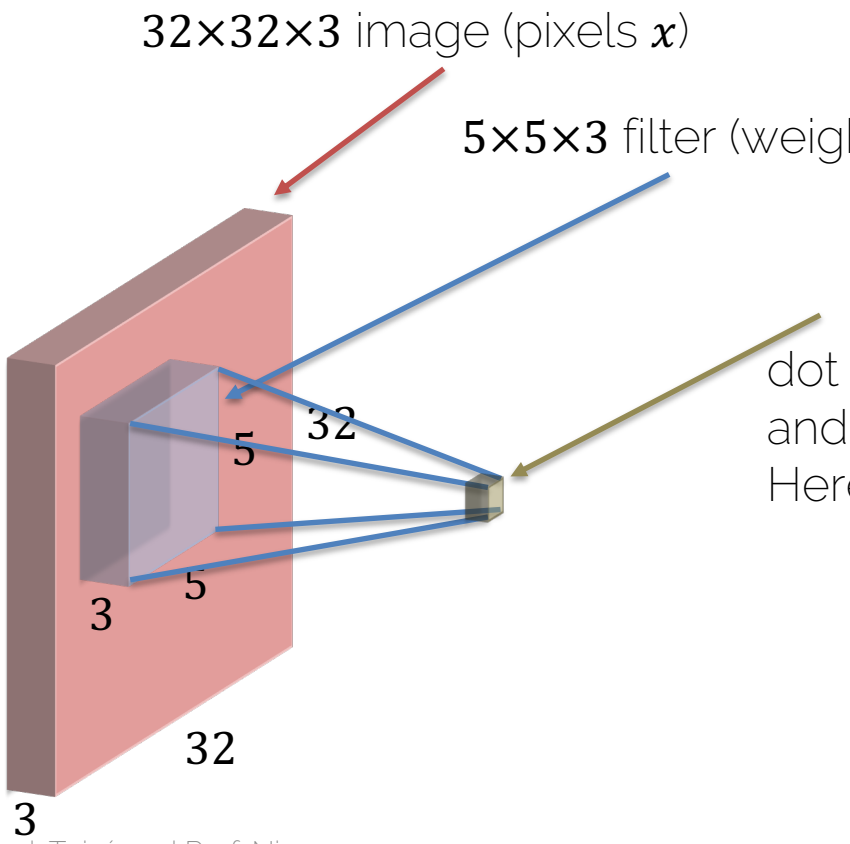




[Hubel & Wiesel, 59, 62, 68, …]

Backprop through CNN layers

Backprop through CNN Layers

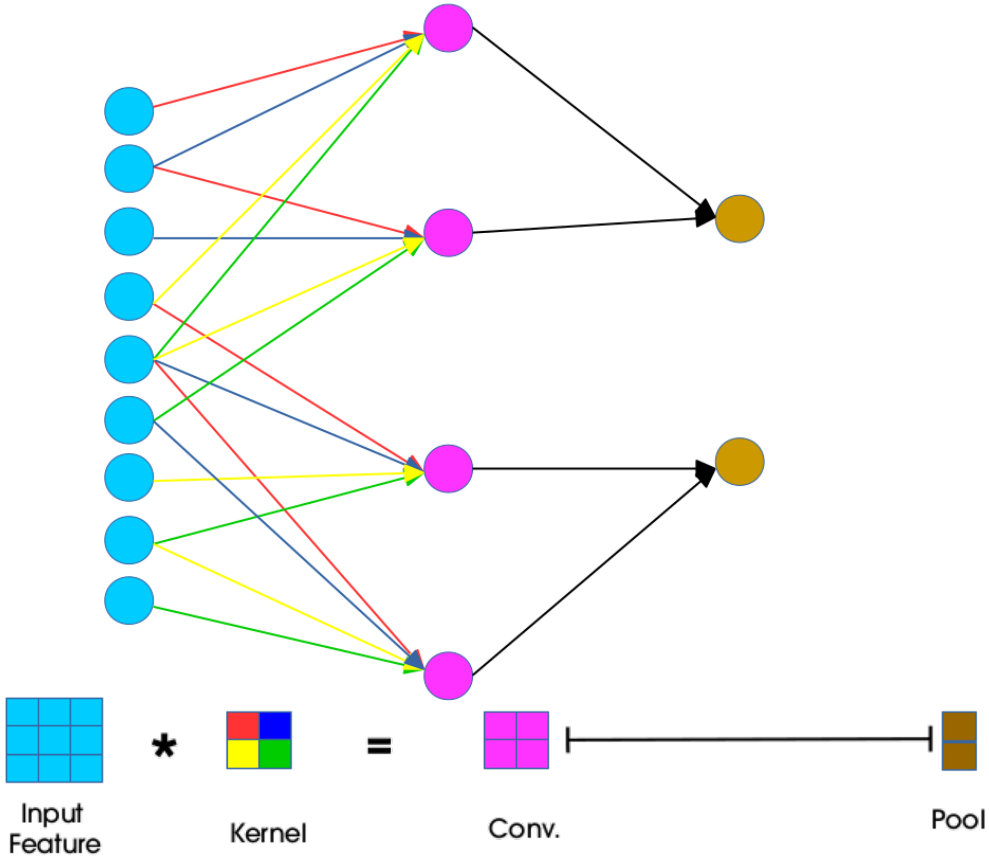


1 number:
dot product between filter weights w
and $x_i - th$ chunk of the image
Here: $5 \cdot 5 \cdot 3 = 75$-dim dot product + bias

$$z_i = w^T x_i + b$$

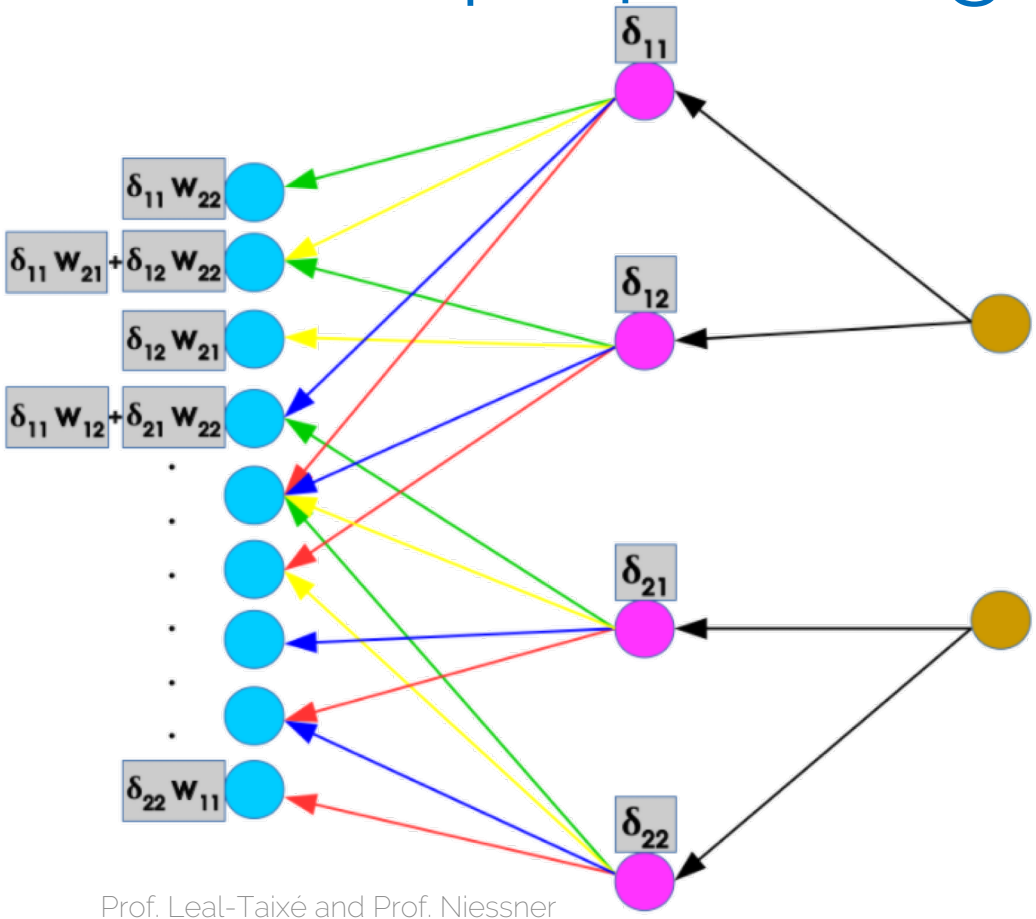
5×5×3 5×5×3 1

Backprop through CNN Layers



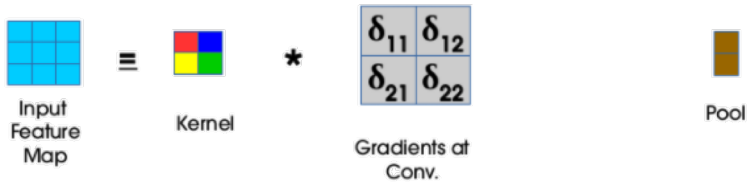
http://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Backprop through CNN Layers



gradient

$\delta_{11}, \delta_{12}, \delta_{21}, \delta_{22}$



http://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Backprop through CNN Layers

$$C = \begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

Input: 16-dim vector
Output: 4-dim vector (will be re-shaped as 2 x 2 eventually)

Backward pass is simply multiplying with C^T

Task for at home: think it through on a piece of paper ☺

[Dumoulin et al. 16]

Administrative Things

- Next Monday: Deep Learning research projects at TUM
- Monday June 25th: second CNN lecture (more about architectures, VGG, Inception)