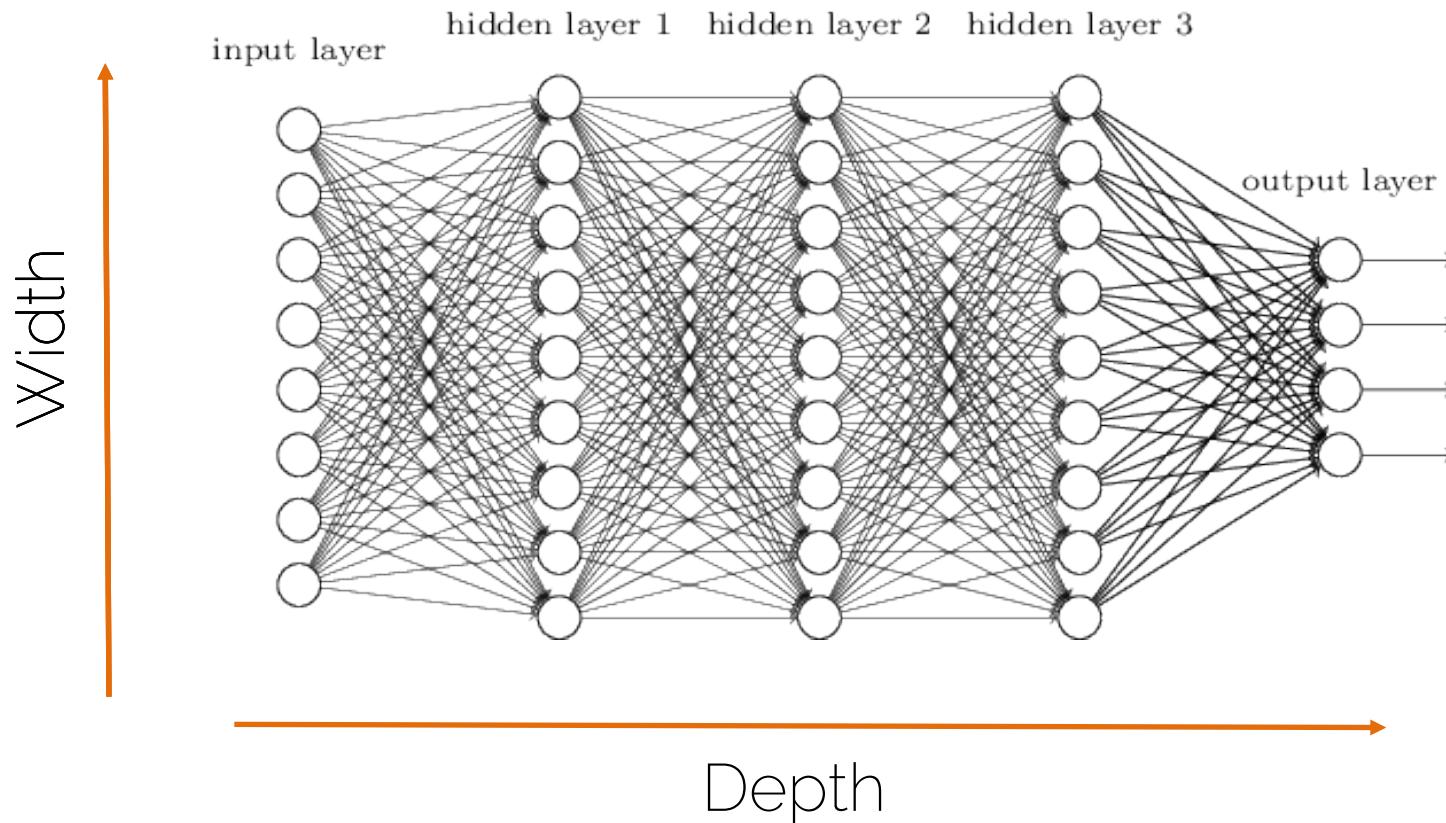


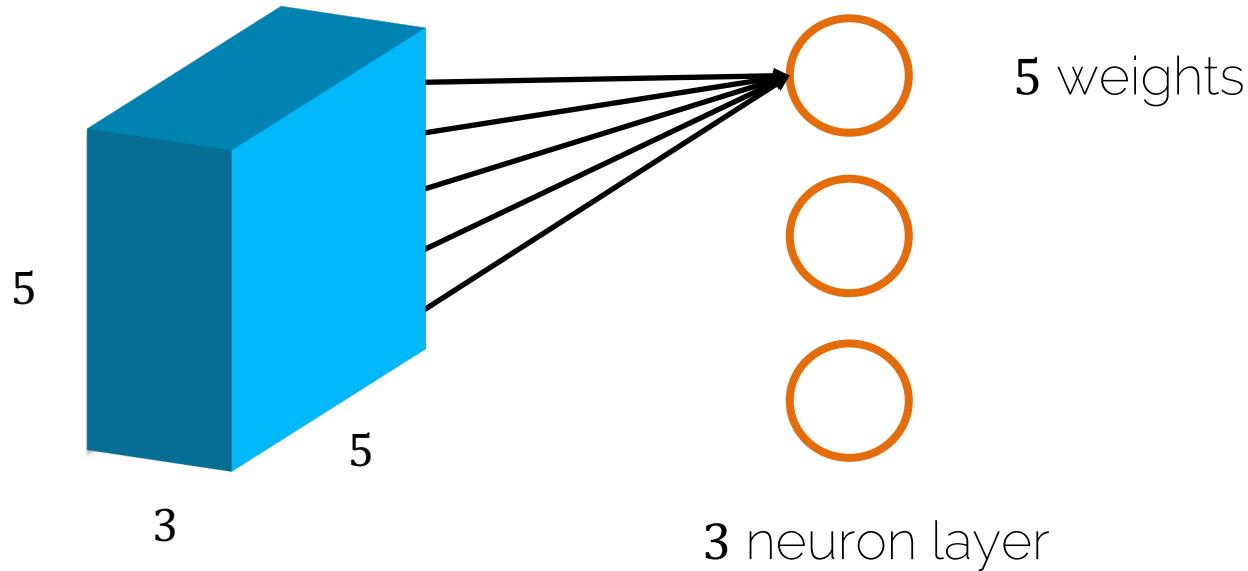
Lecture 9 - Convolutional Neural Networks

Fully Connected Neural Network



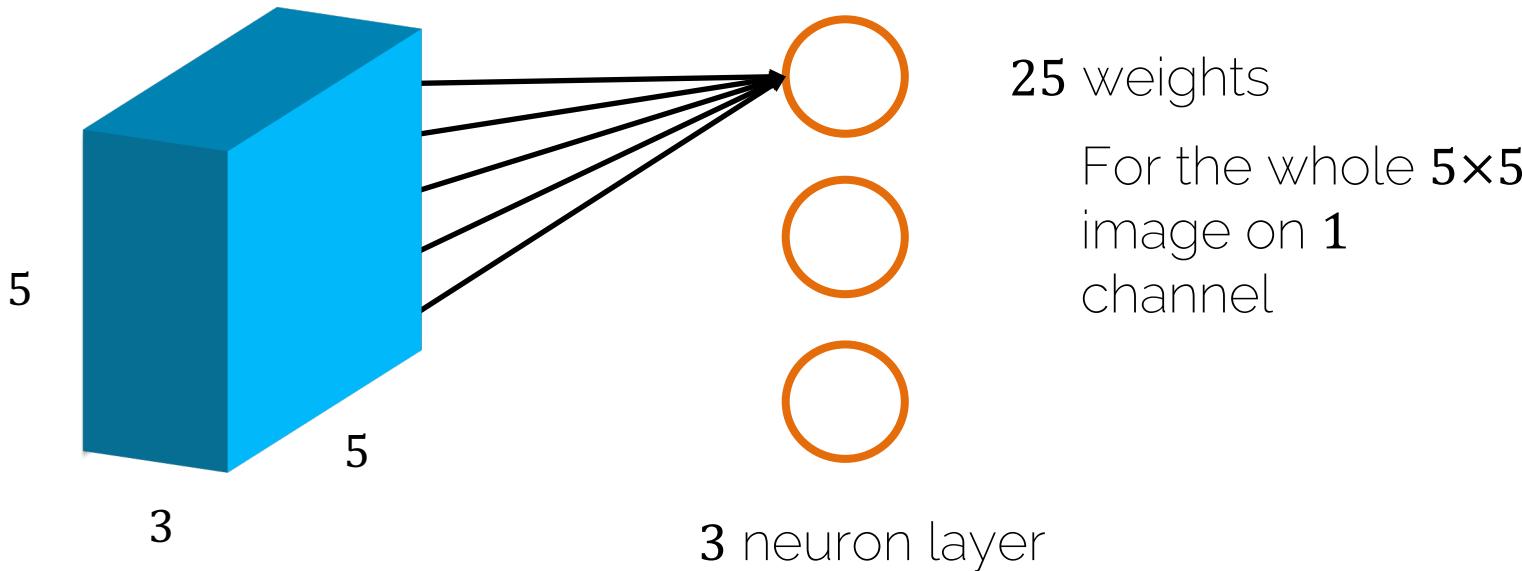
Problems using FC Layers on Images

- How to process a tiny image with FC layers



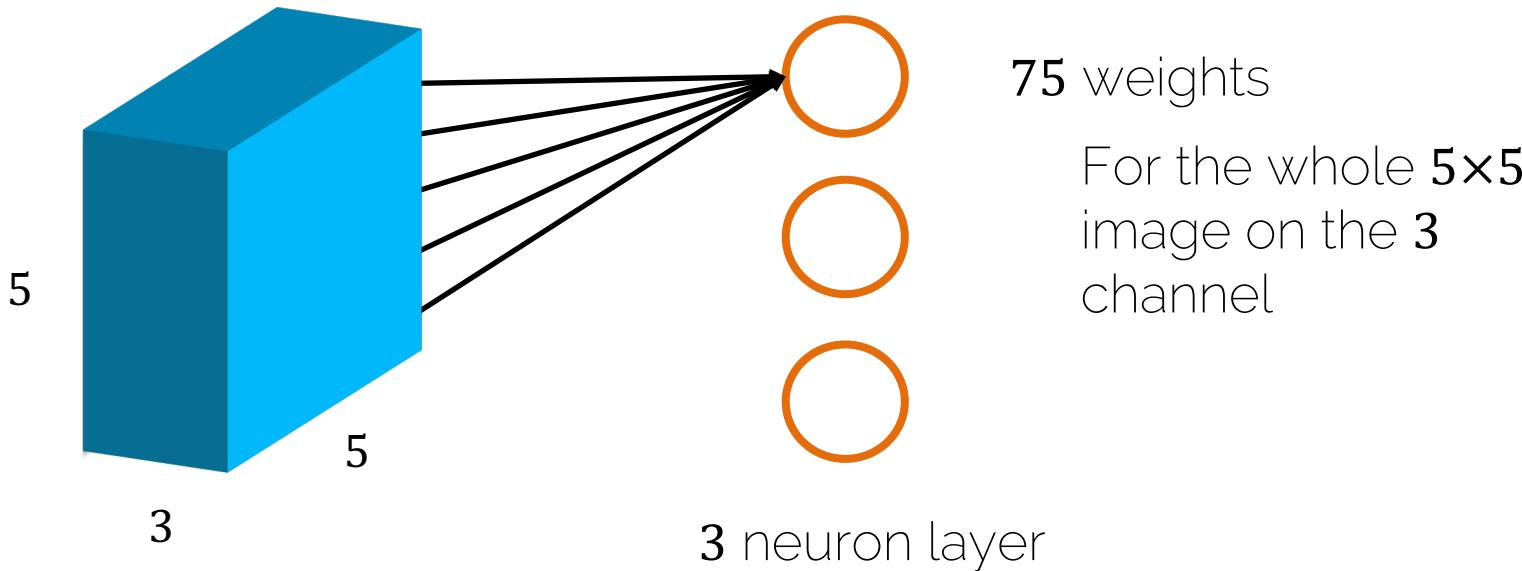
Problems using FC Layers on Images

- How to process a tiny image with FC layers



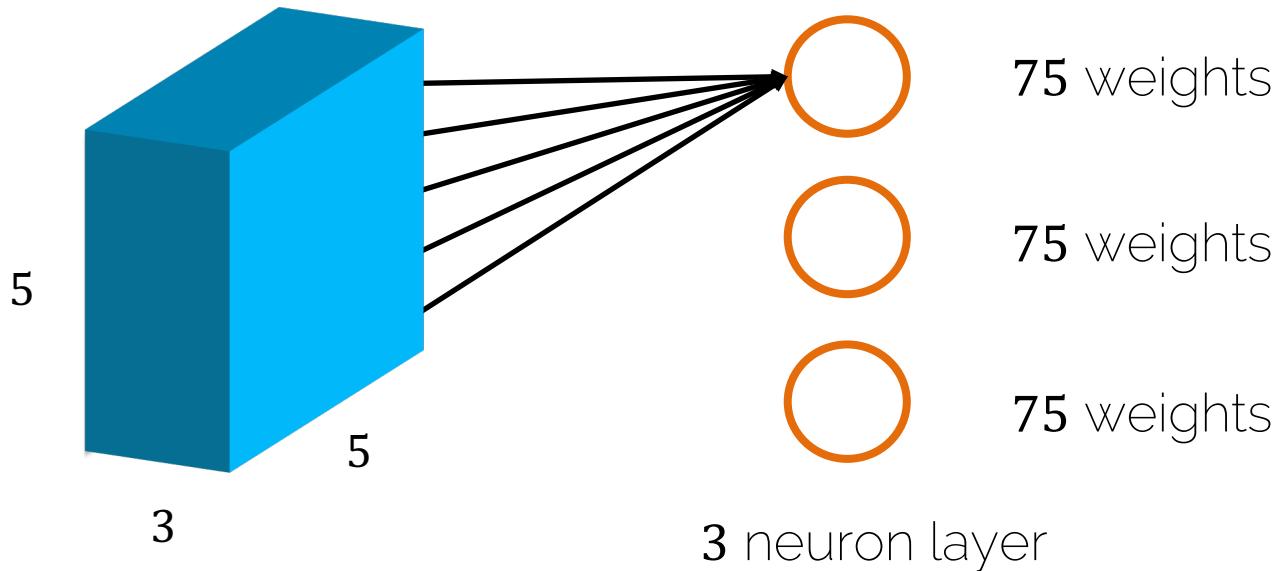
Problems using FC Layers on Images

- How to process a tiny image with FC layers



Problems using FC Layers on Images

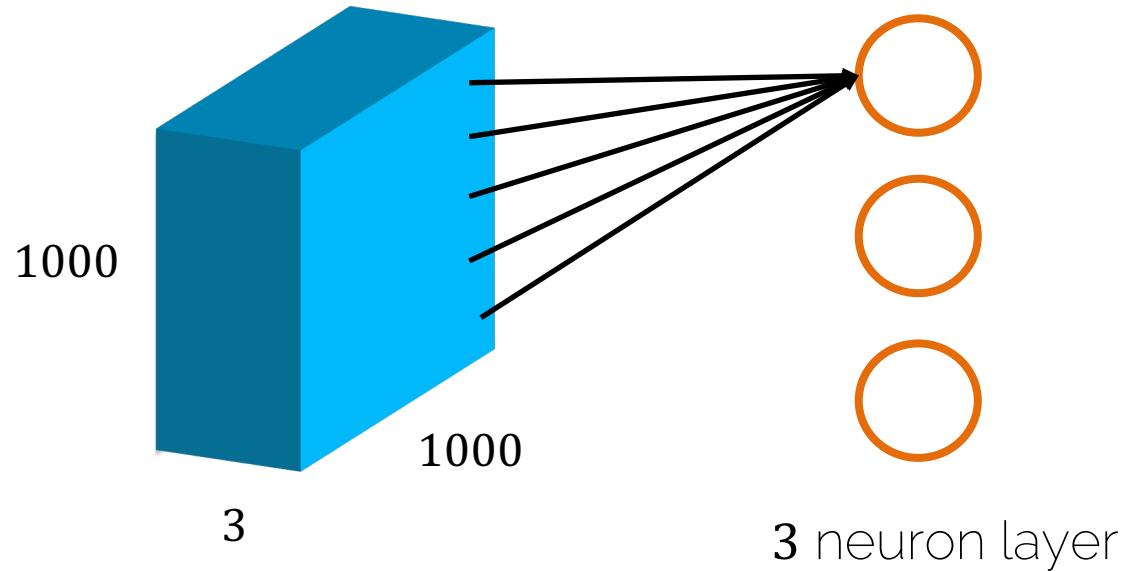
- How to process a tiny image with FC layers



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

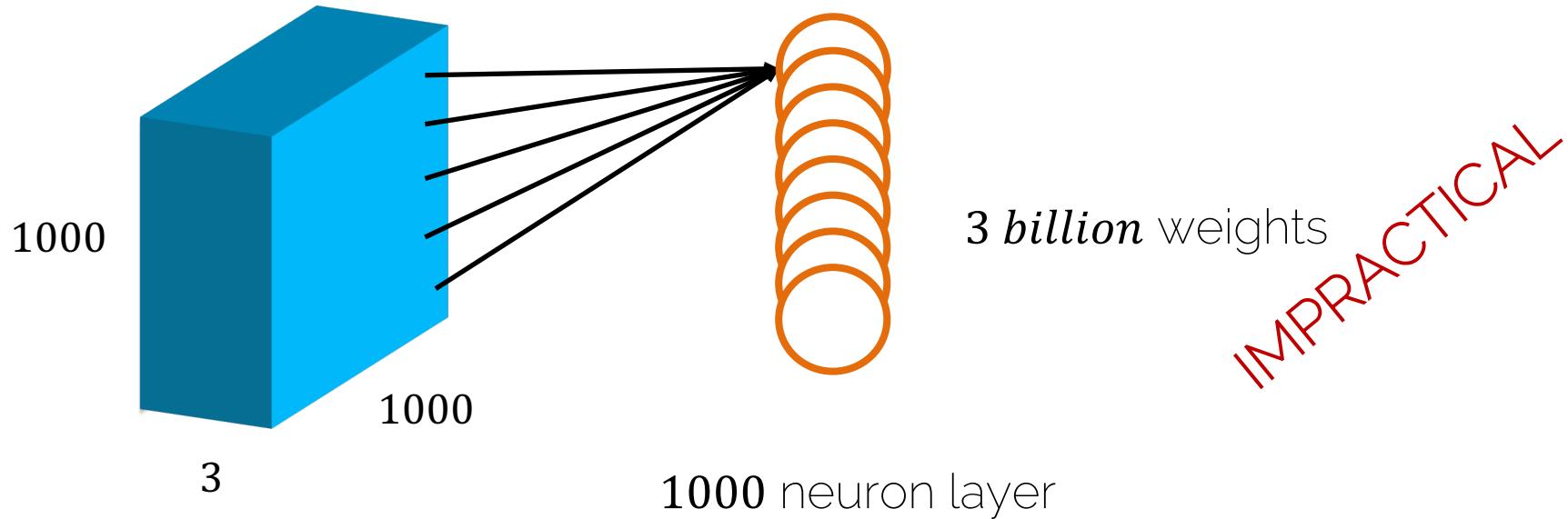
Problems using FC Layers on Images

- 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

Classification



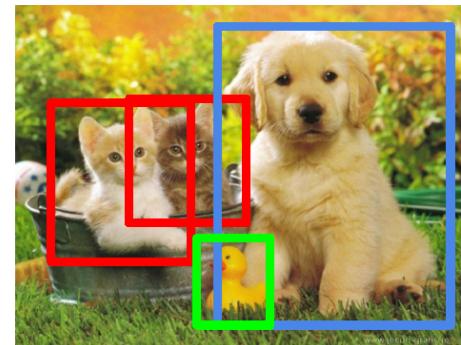
CAT

Classification + Localization



Single object

Object Detection



CAT, DOG, DUCK

Instance Segmentation



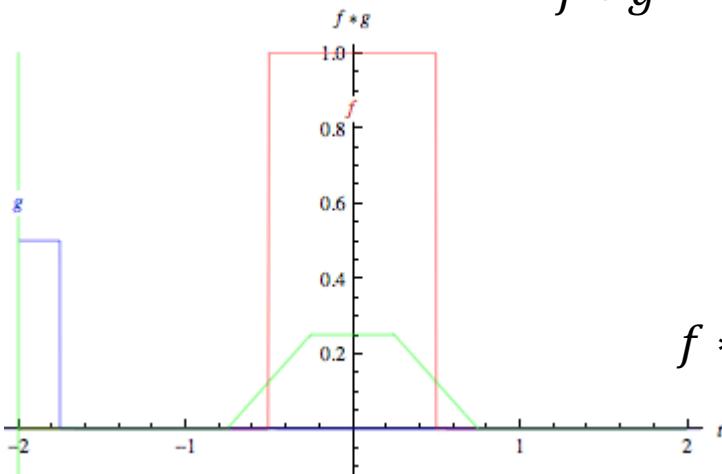
CAT, DOG, DUCK

Multiple objects

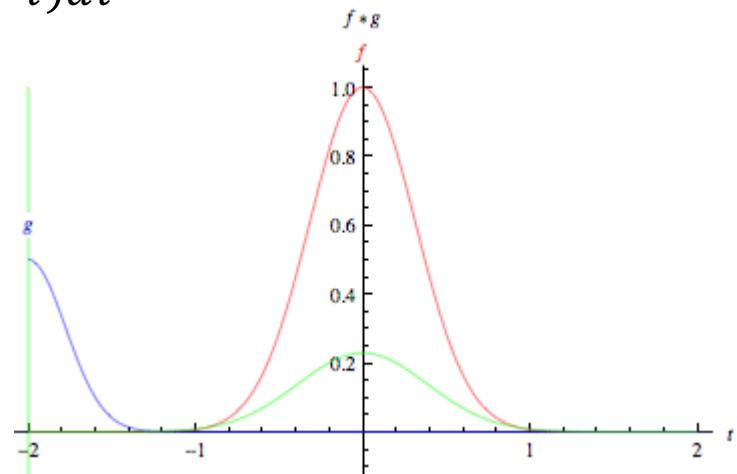
Convolutions

What are Convolutions?

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



Convolution of two box functions



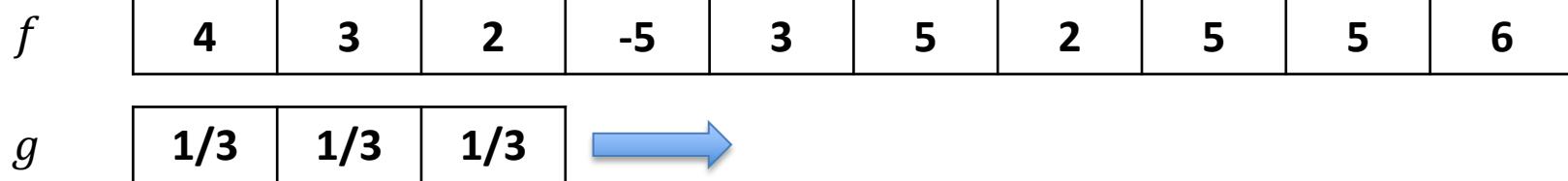
Convolution of two Gaussians

Application of a filter to a function

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

What are Convolutions?

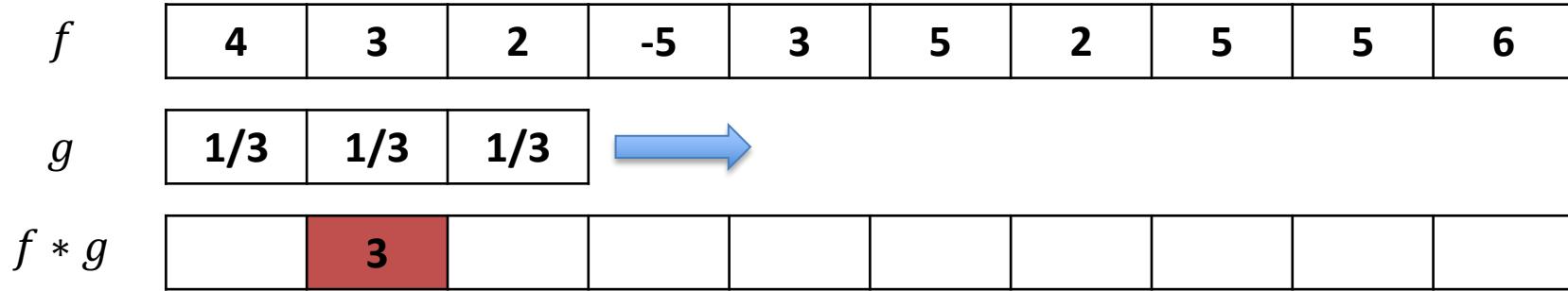
Discrete case: box filter



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

What are Convolutions?

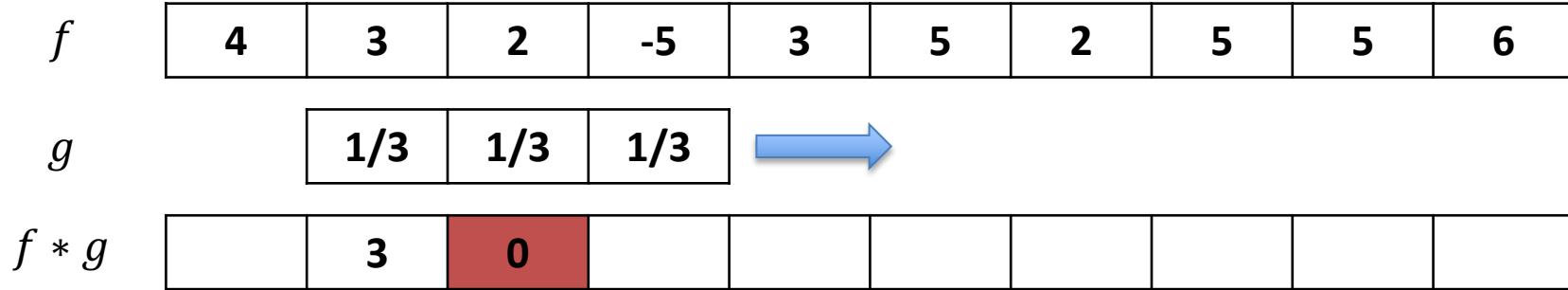
Discrete case: box filter



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

What are Convolutions?

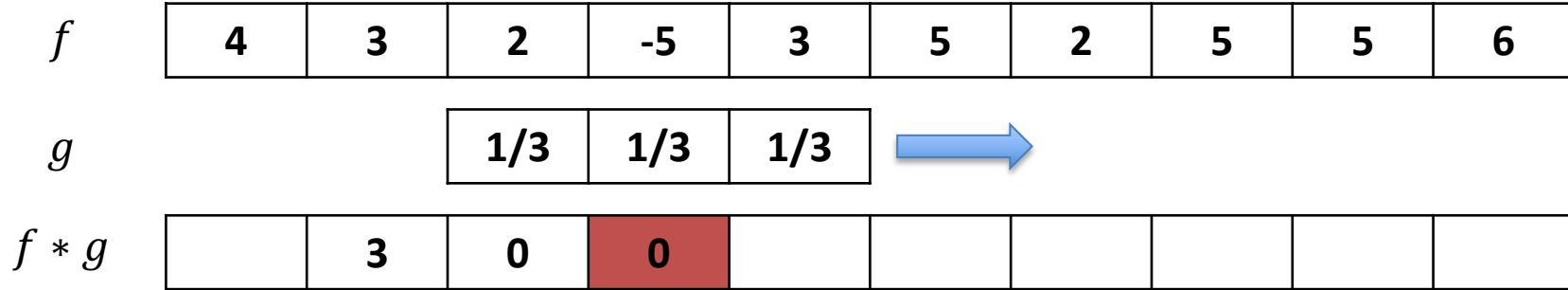
Discrete case: box filter



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

What are Convolutions?

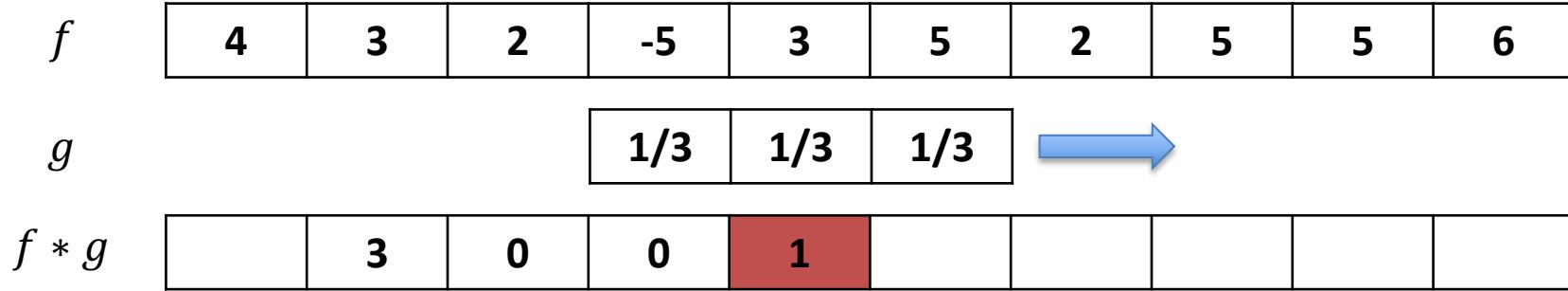
Discrete case: box filter



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

What are Convolutions?

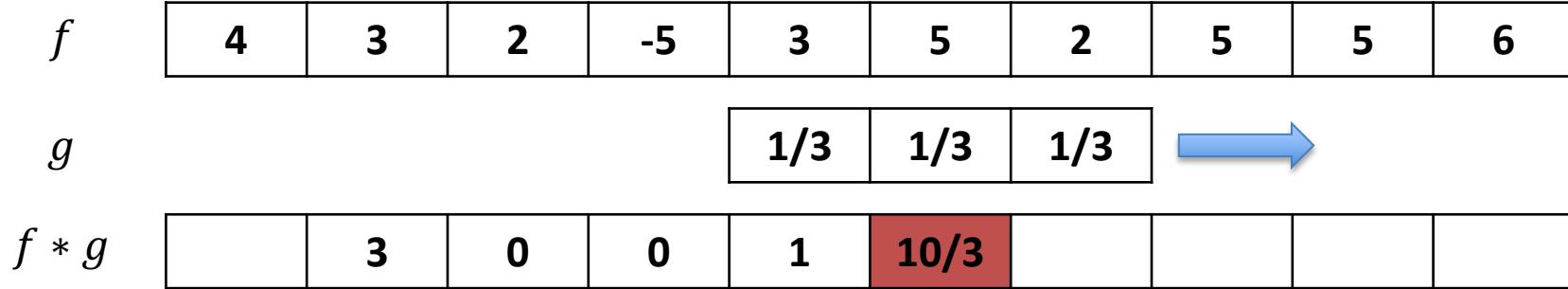
Discrete case: box filter



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

What are Convolutions?

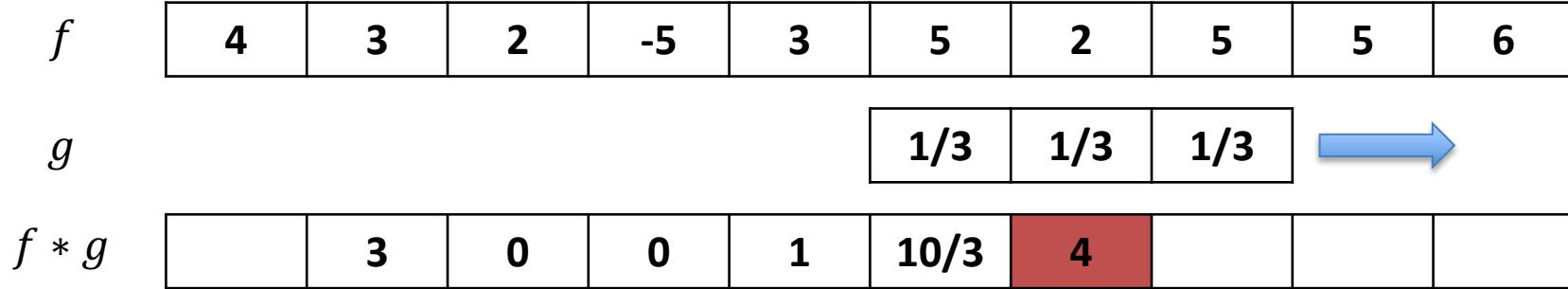
Discrete case: box filter



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

What are Convolutions?

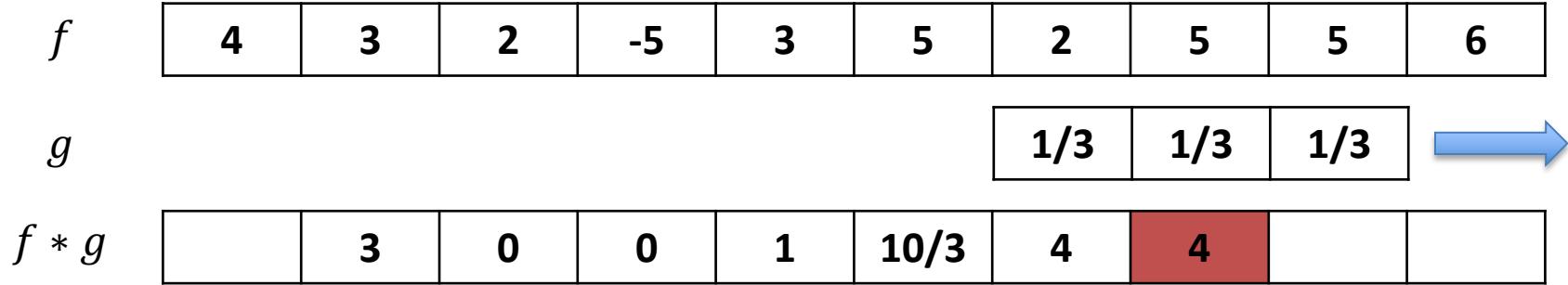
Discrete case: box filter



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

What are Convolutions?

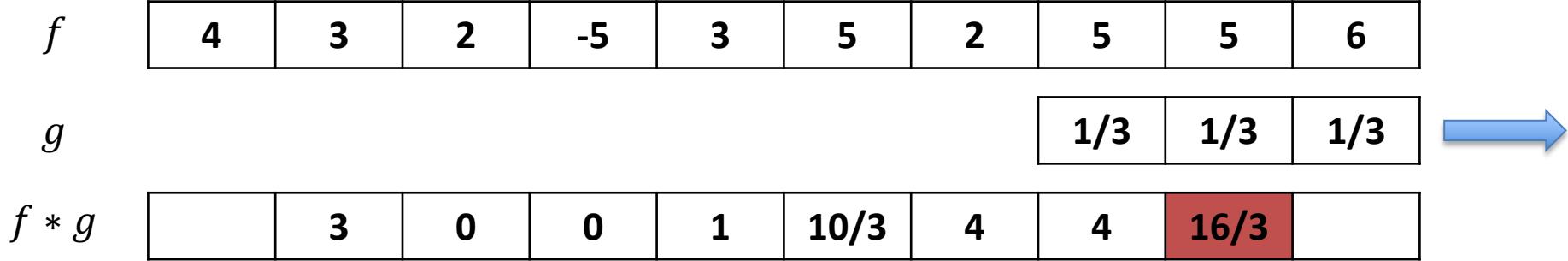
Discrete case: box filter



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

What are Convolutions?

Discrete case: box filter



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

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	??
----	---	---	---	---	------	---	---	------	----

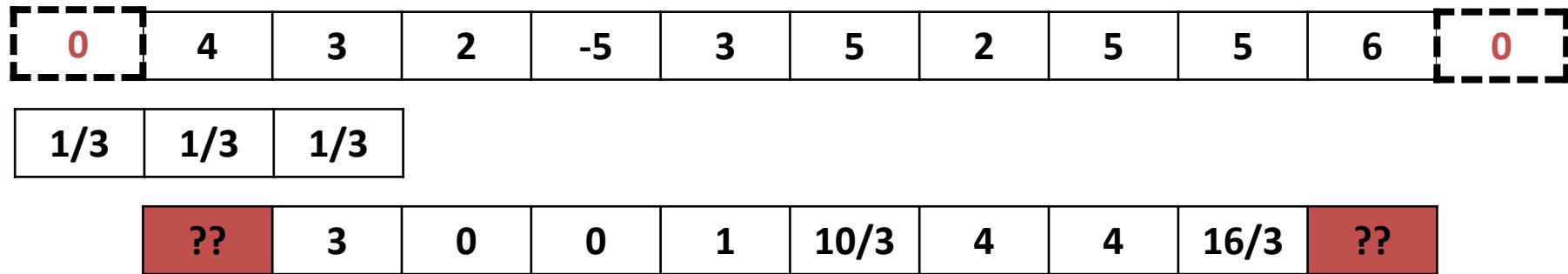
What to do at boundaries?

Option 1: Shrink

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

What are Convolutions?

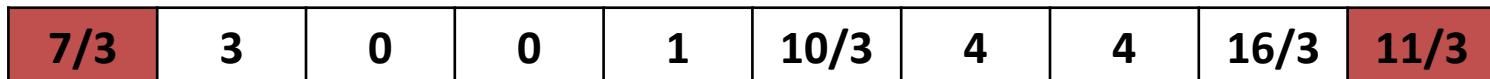
Discrete case: box filter



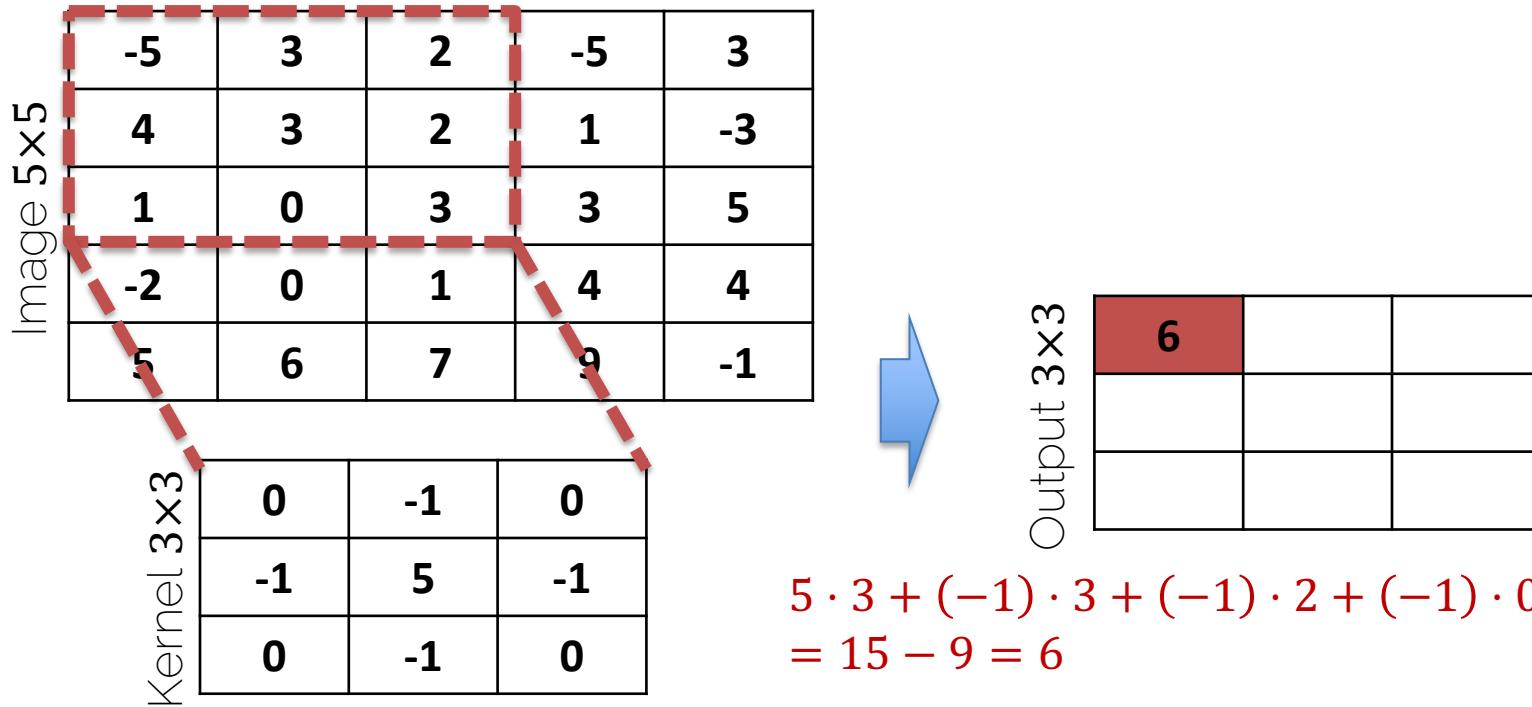
$$0 \cdot \frac{1}{3} + 4 \cdot \frac{1}{3} + 3 \cdot \frac{1}{3} = \frac{7}{3}$$

What to do at boundaries?

Option 2: Pad (often 0's)



Convolutions on Images



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	

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

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

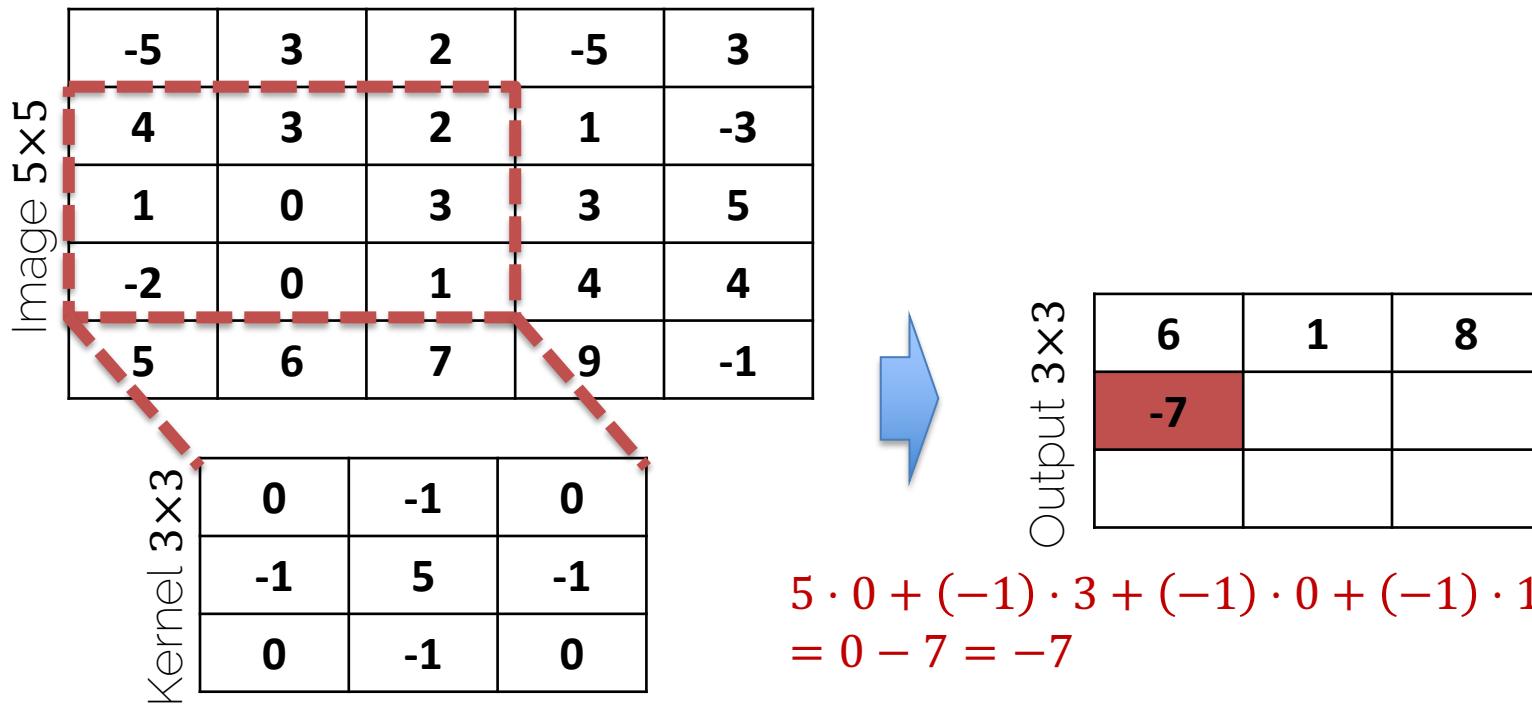


Output 3×3

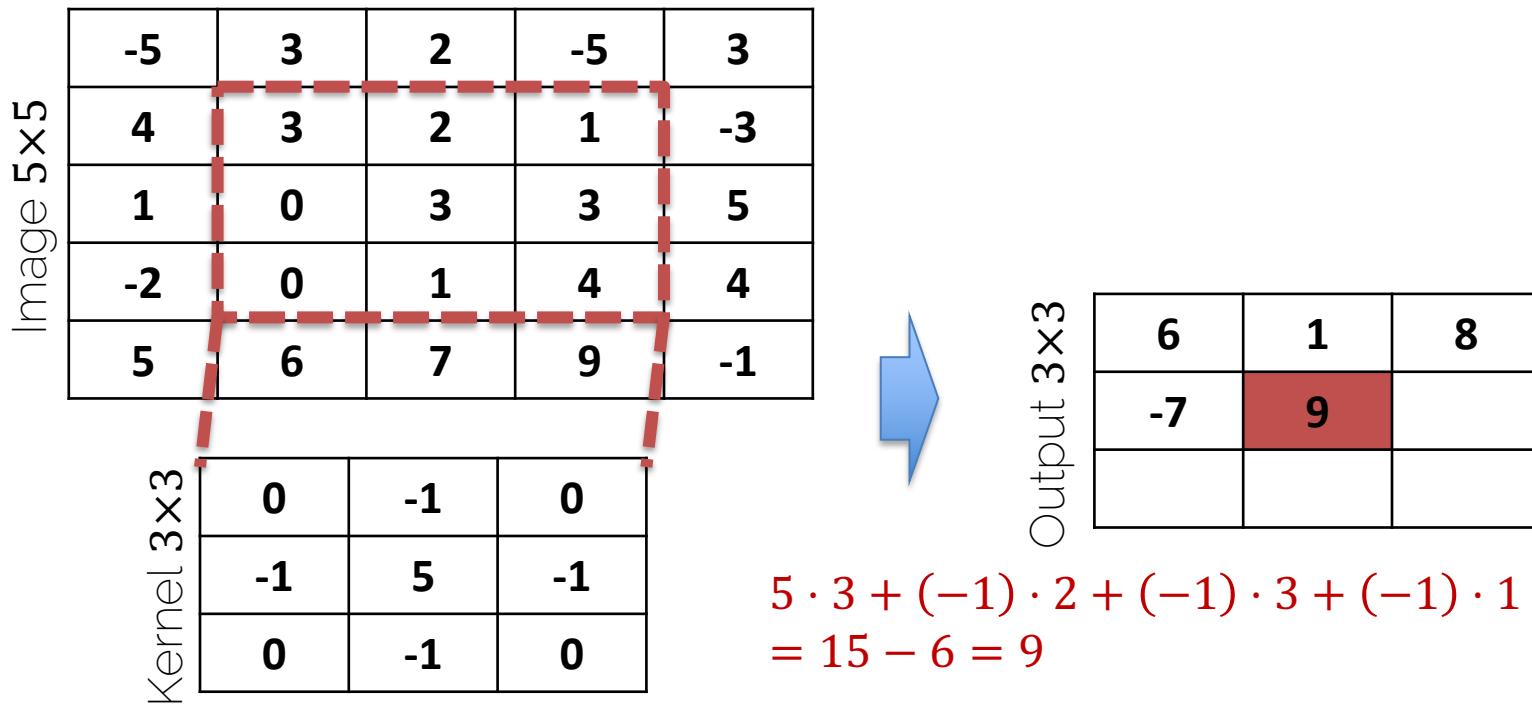
6	1	8

$$\begin{aligned} & 5 \cdot 1 + (-1) \cdot (-5) + (-1) \cdot (-3) + (-1) \cdot 3 \\ & + (-1) \cdot 2 \\ & = 5 + 3 = 8 \end{aligned}$$

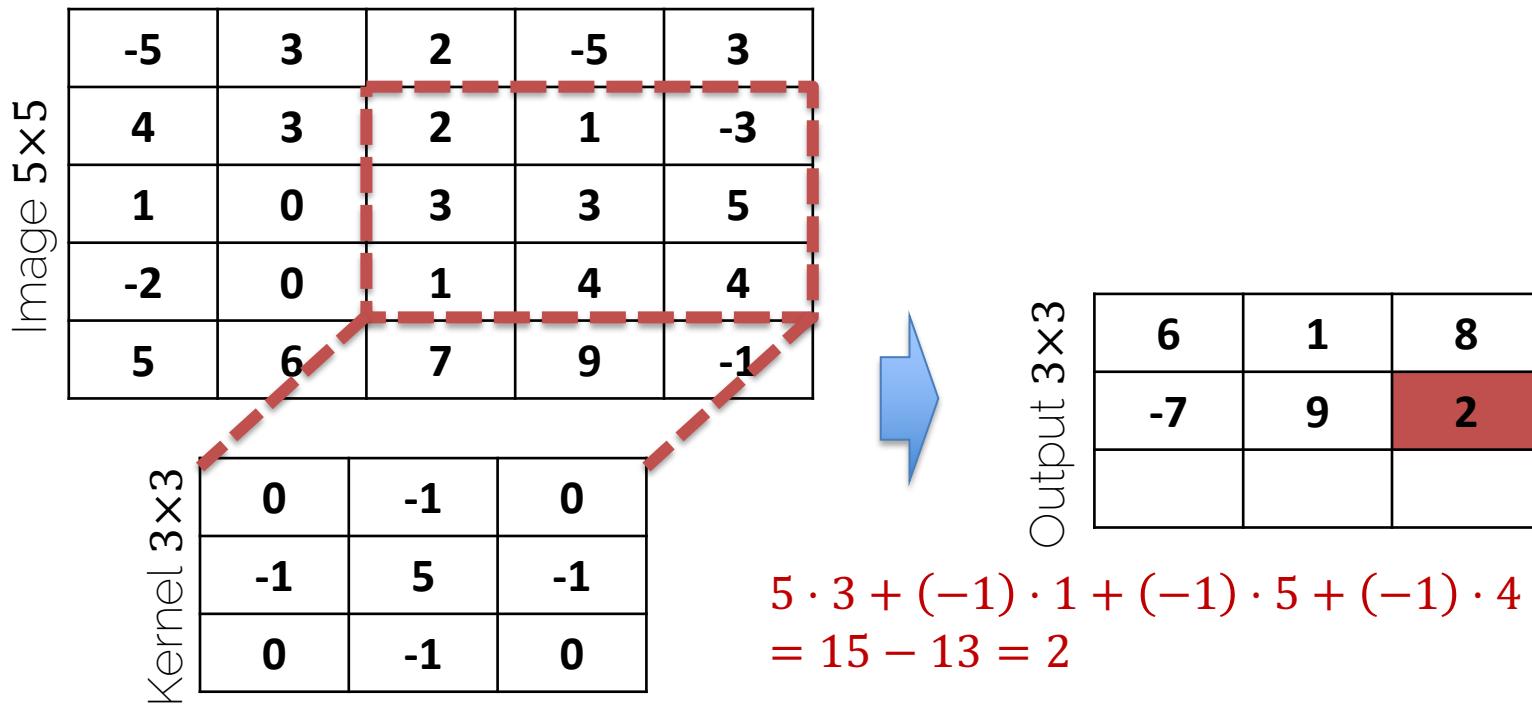
Convolutions on Images



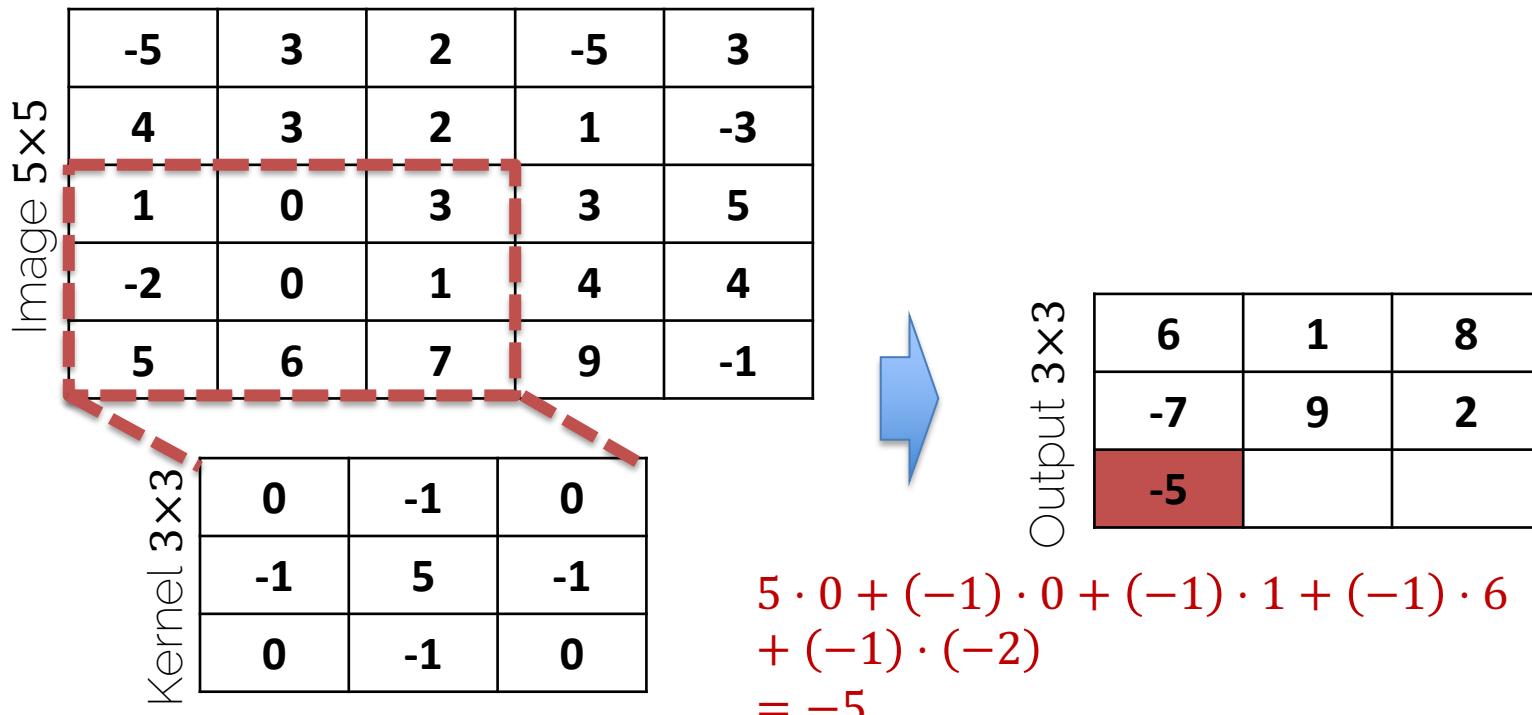
Convolutions on Images



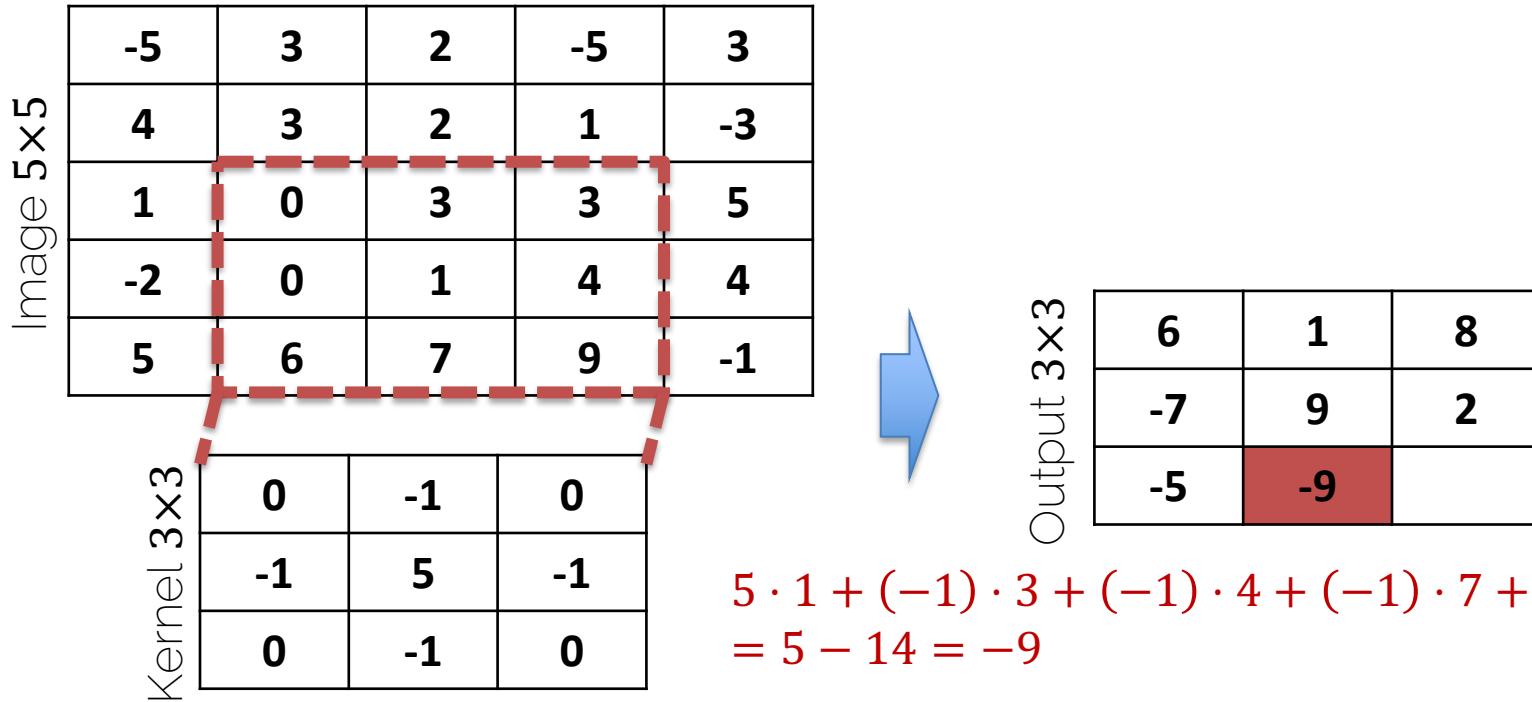
Convolutions on Images



Convolutions on Images



Convolutions on Images



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

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

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

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



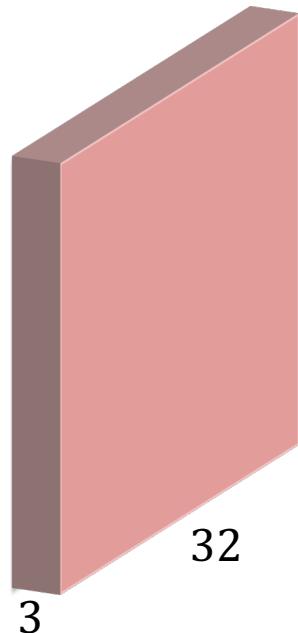
Gaussian blur

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

Convolutions on RGB Images

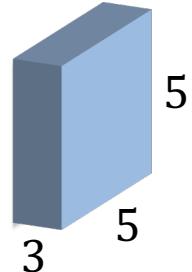
width height depth

image $32 \times 32 \times 3$



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

filter $5 \times 5 \times 3$



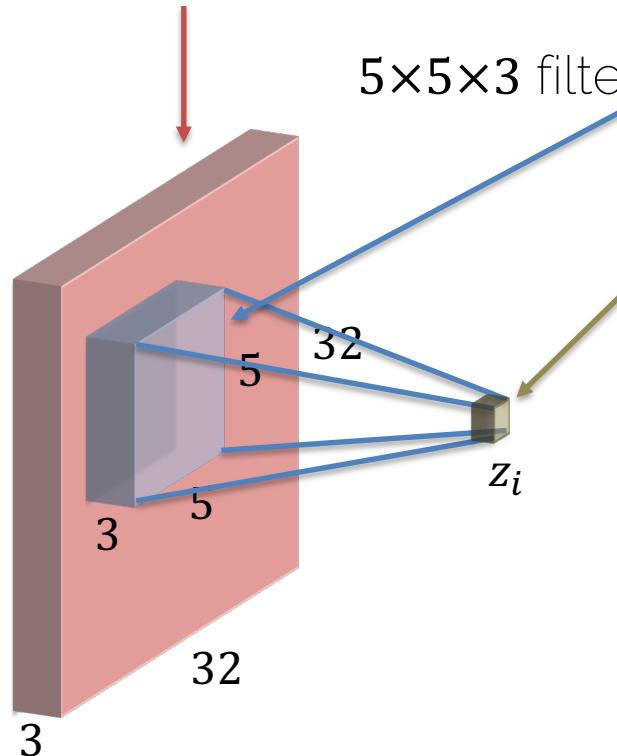
Convolve filter with image
i.e., 'slide' over it and:

- apply filter at each location
- dot products

Images have depth: e.g. RGB \rightarrow 3 channels

Convolutions on RGB Images

32×32×3 image (pixels \mathbf{X})



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

1 number at a time:

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

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

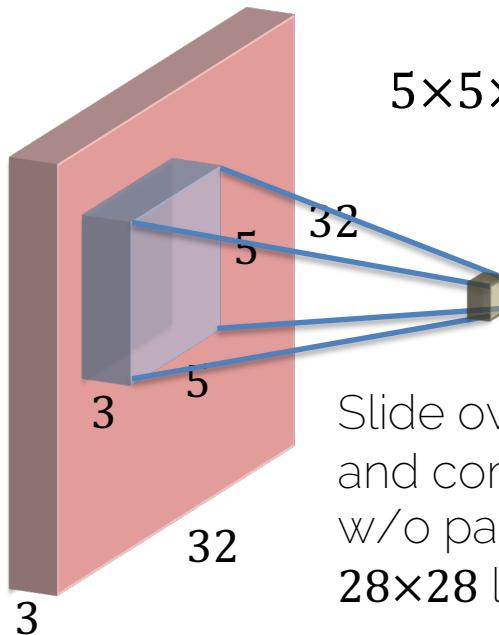
(5×5×3)×1

(5×5×3)×1

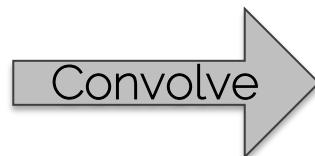
1

Convolutions on RGB Images

32×32×3 image

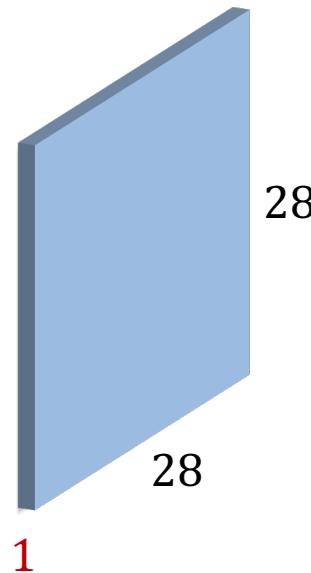


5×5×3 filter



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

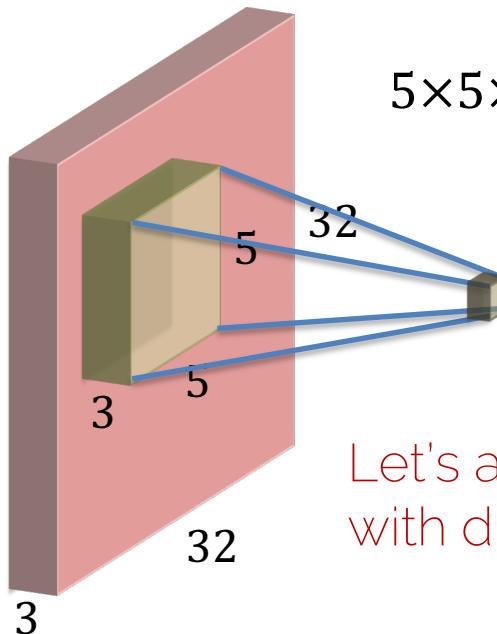
Activation map
(also feature map)



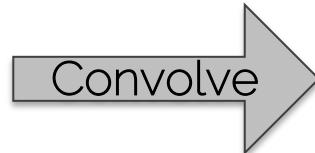
Convolution Layer

Convolution Layer

$32 \times 32 \times 3$ image

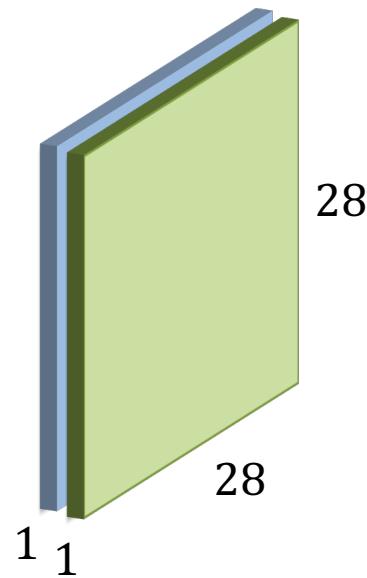


$5 \times 5 \times 3$ filter

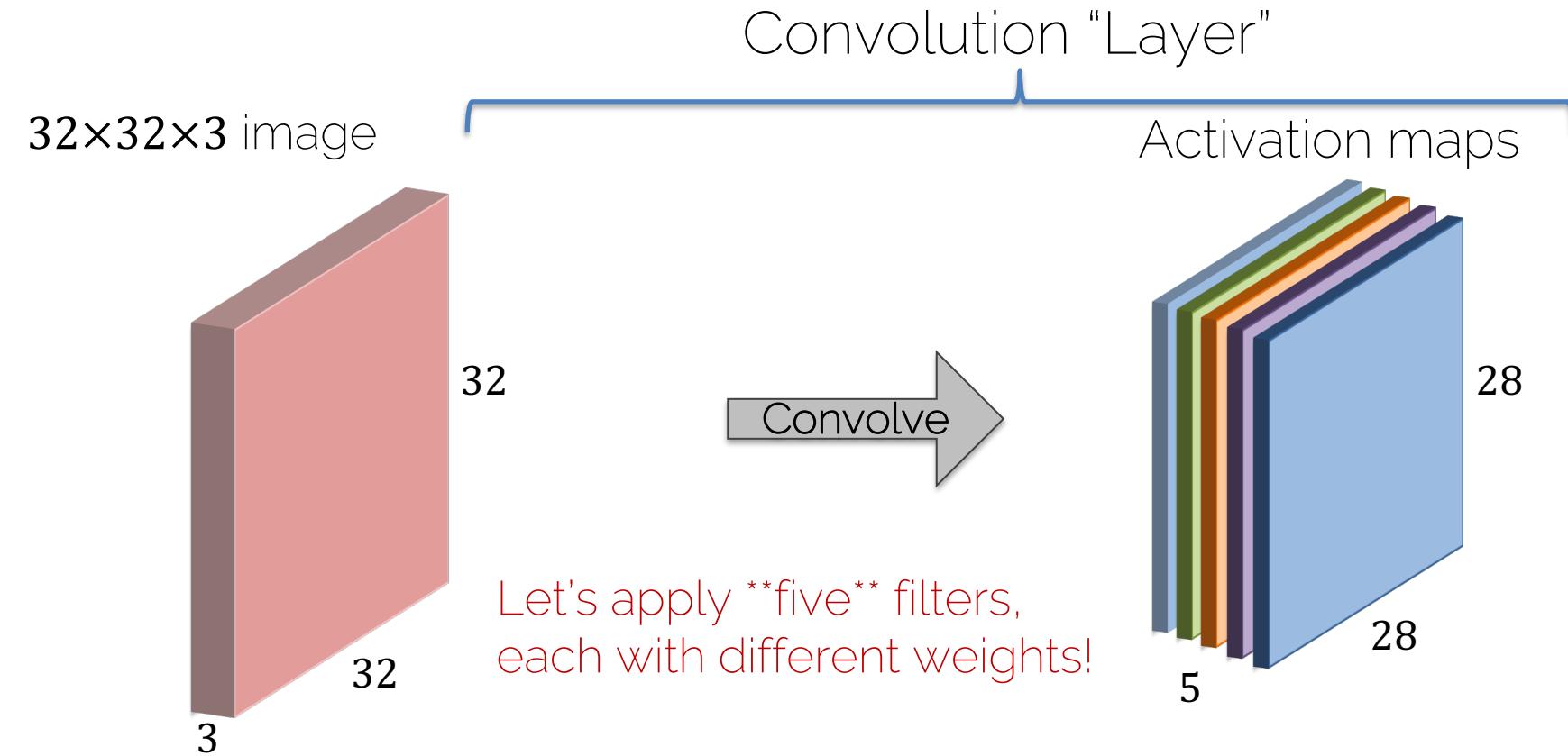


Let's apply a different filter
with different weights!

Activation maps



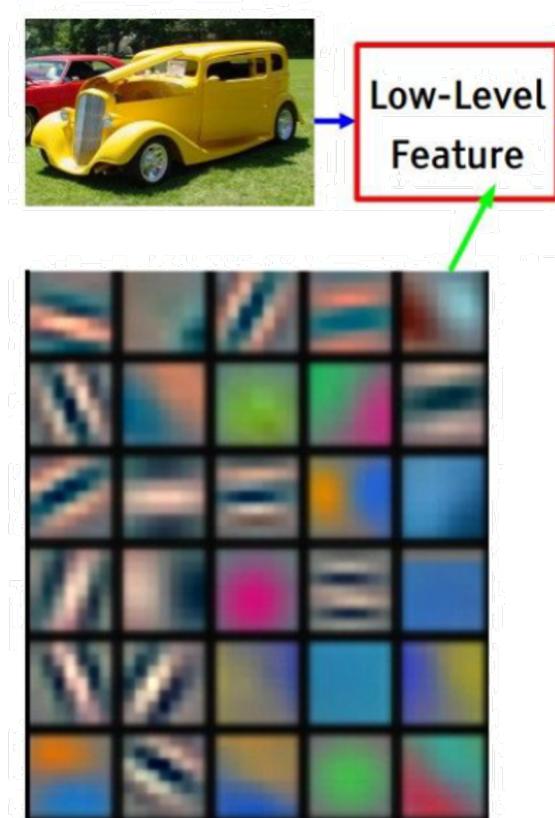
Convolution Layer



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

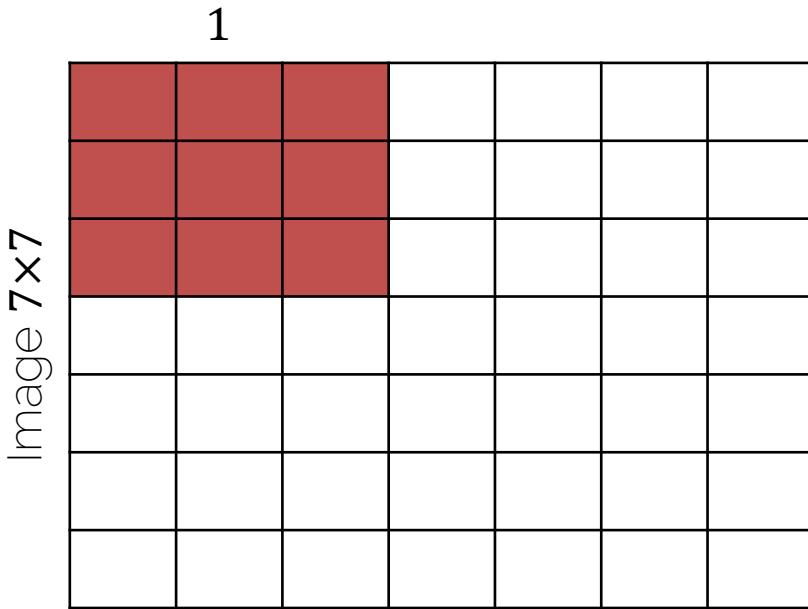
Different Filters



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

Dimensions of a Convolution Layer

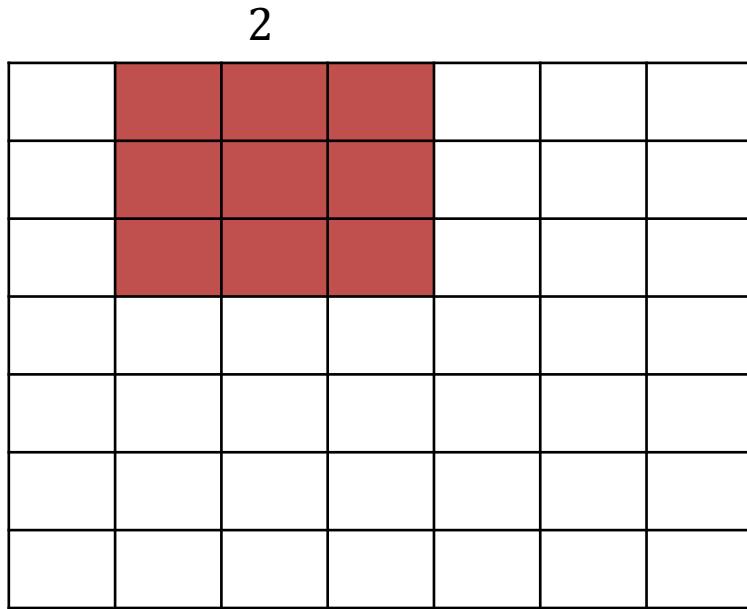
Convolution Layers: Dimensions



Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Dimensions

Image 7×7

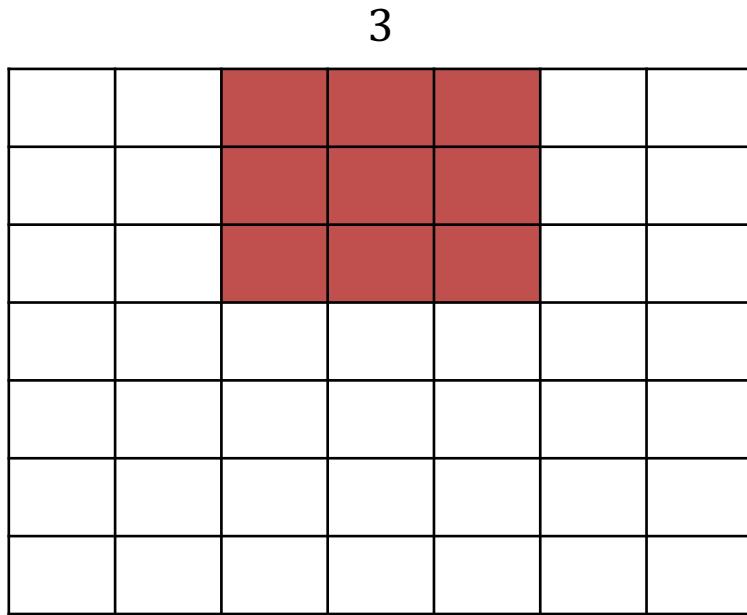


2

Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Dimensions

Image 7×7



Input:

7×7

Filter:

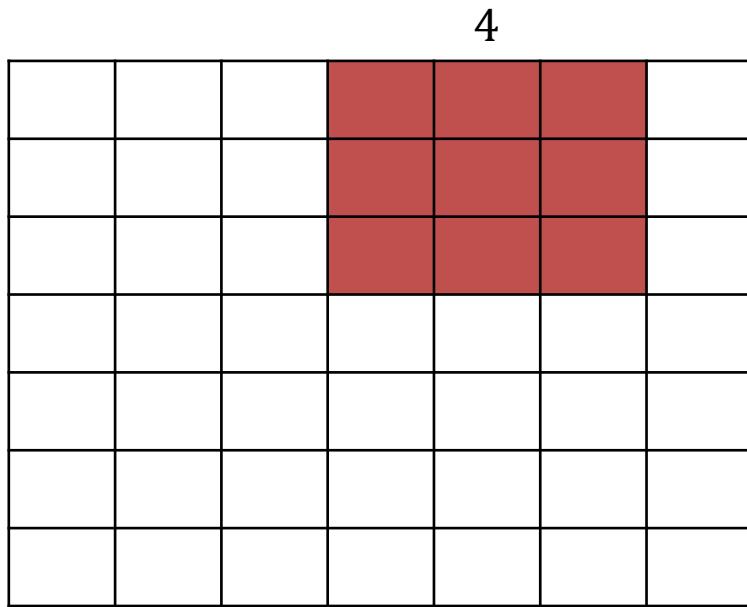
3×3

Output:

5×5

Convolution Layers: Dimensions

Image 7×7



Input:

7×7

Filter:

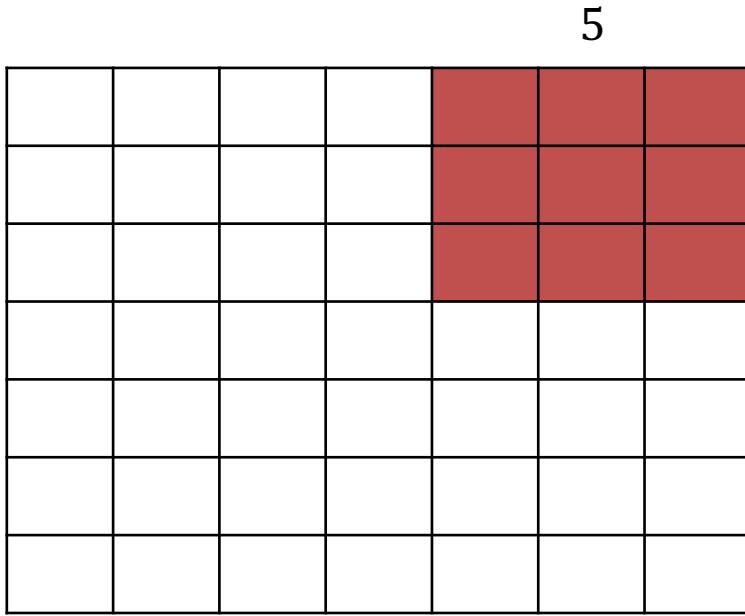
3×3

Output:

5×5

Convolution Layers: Dimensions

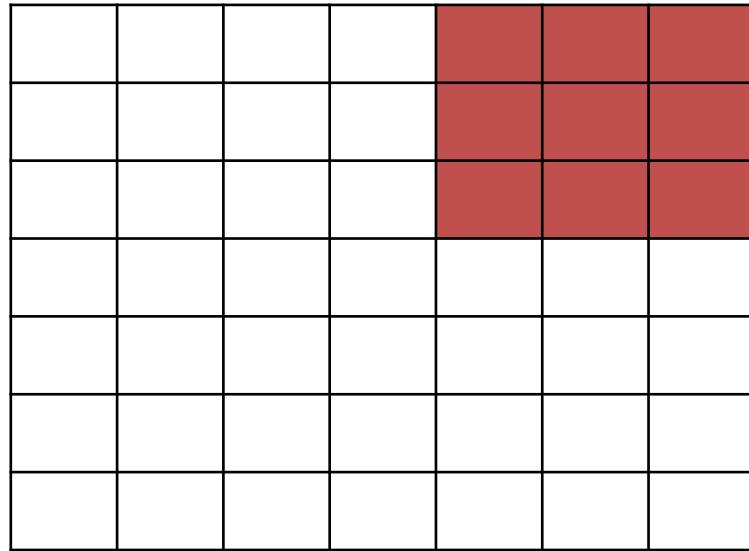
Image 7×7



Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Stride

Image 7×7



With a stride of 1

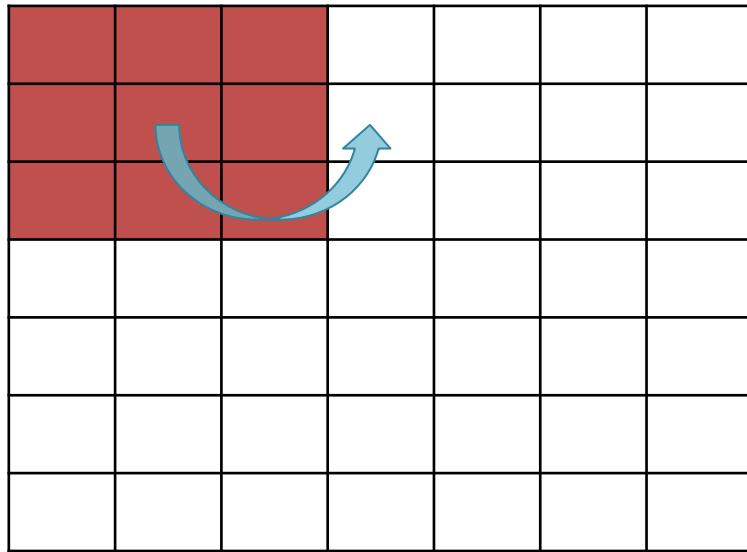
Input:	7×7
Filter:	3×3
Stride:	1
Output:	5×5

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

Convolution Layers: Stride

With a stride of 2

Image 7×7

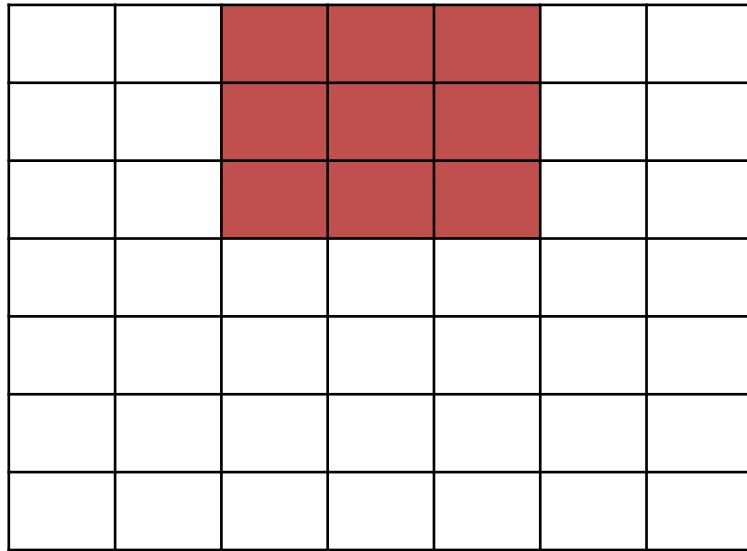


Input:	7×7
Filter:	3×3
Stride:	2
Output:	3×3

Convolution Layers: Stride

With a stride of 2

Image 7×7

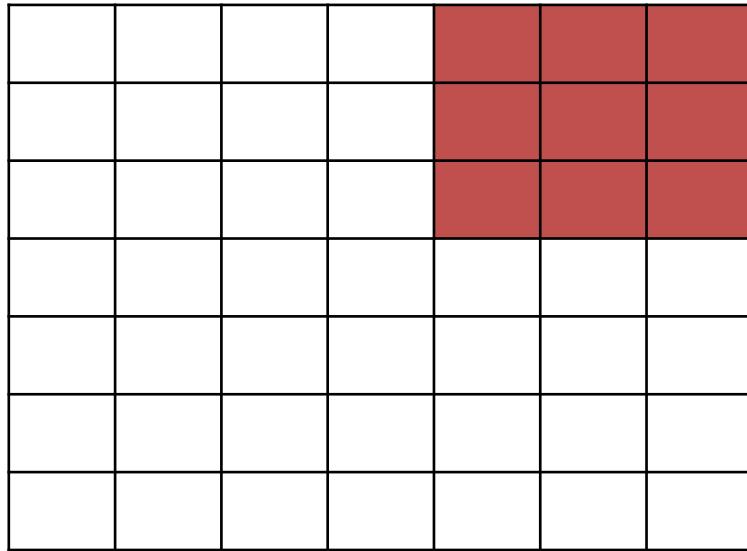


Input:	7×7
Filter:	3×3
Stride:	2
Output:	3×3

Convolution Layers: Stride

With a stride of 2

Image 7×7

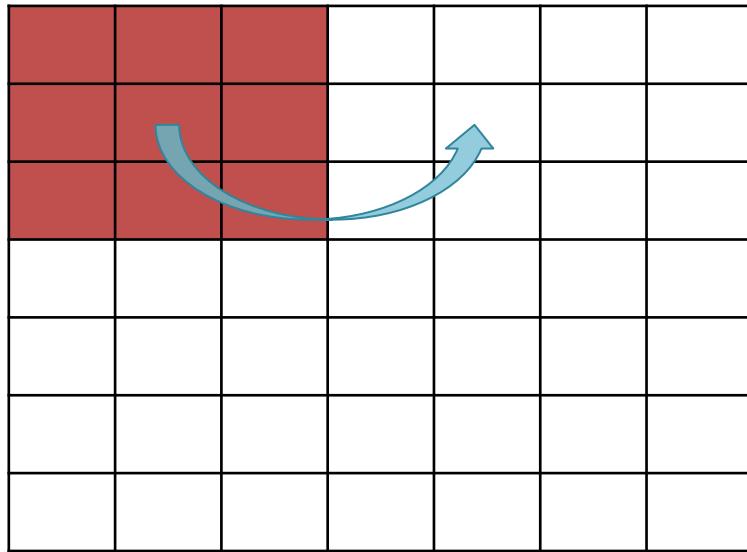


Input:	7×7
Filter:	3×3
Stride:	2
Output:	3×3

Convolution Layers: Stride

With a stride of 3

Image 7×7

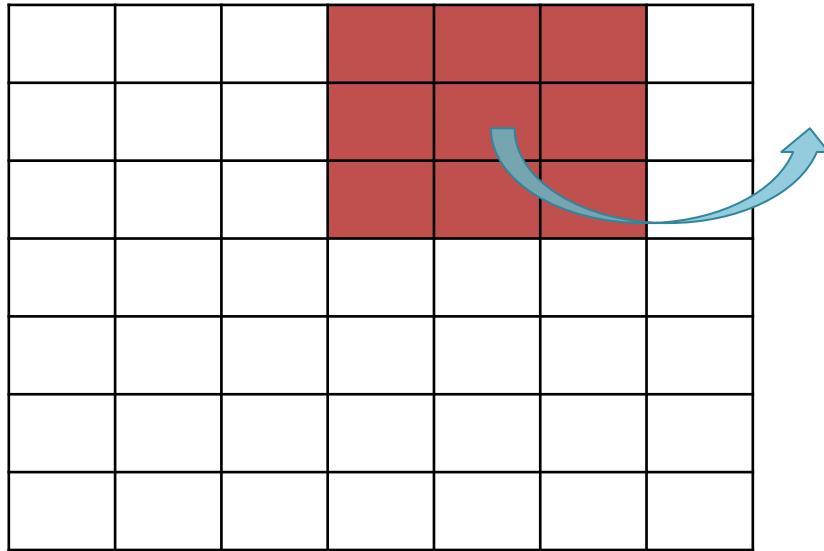


Input:	7×7
Filter:	3×3
Stride:	3
Output:	? × ?

Convolution Layers: Stride

With a stride of 3

Image 7×7



Input:

7×7

Filter:

3×3

Stride:

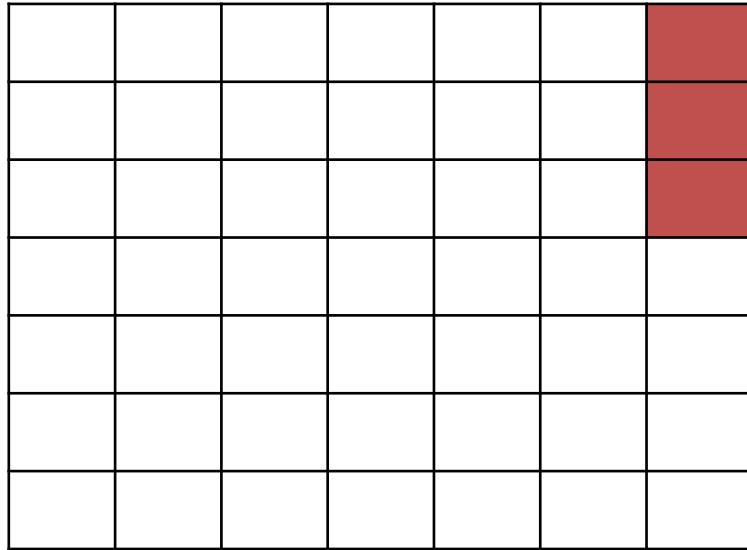
3

Output:

? \times ?

Convolution Layers: Stride

Image 7×7



With a stride of 3

Input:

7×7

Filter:

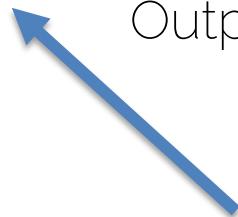
3×3

Stride:

3

Output:

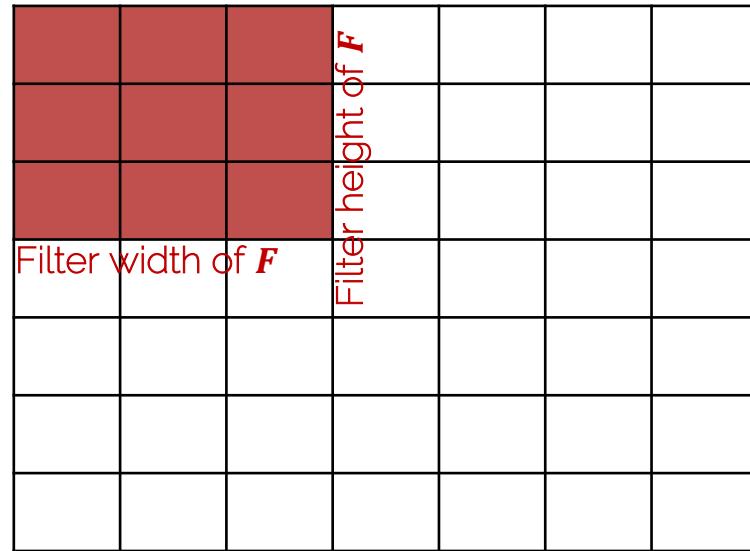
? \times ?



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

Convolution Layers: Dimensions

Input width of N



Input:

$N \times N$

Filter:

$F \times F$

Stride:

S

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

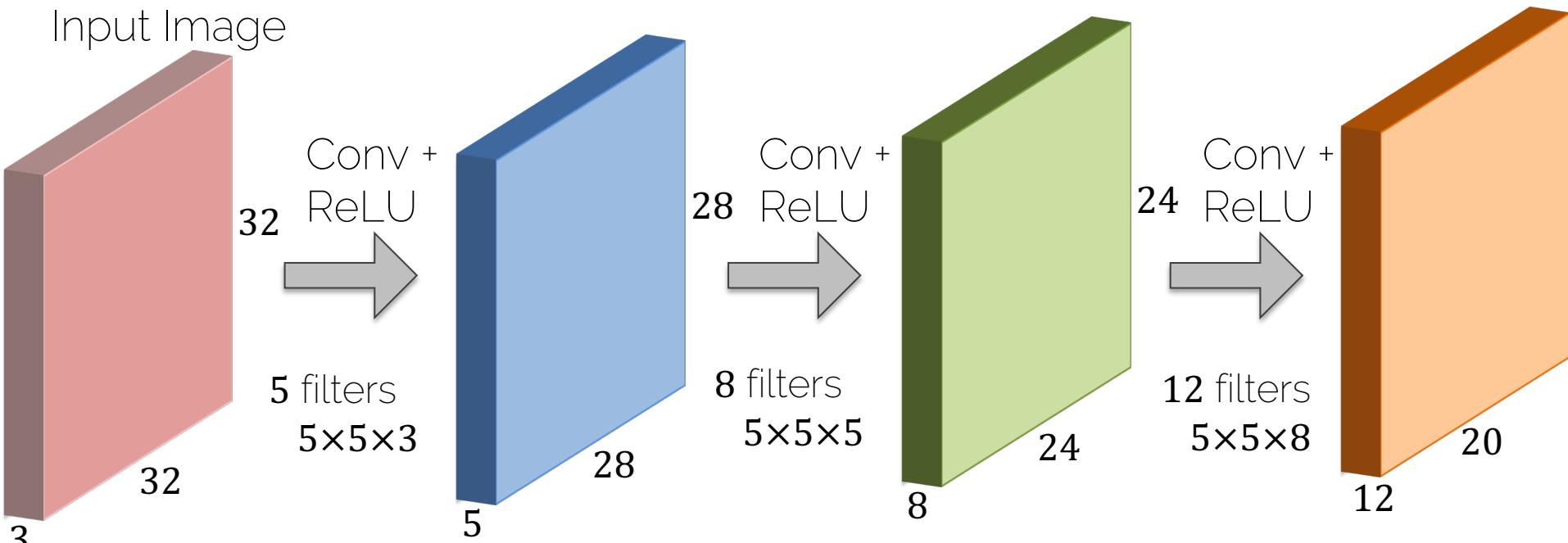
$$N = 7, F = 3, S = 1: \frac{7-3}{1} + 1 = 5$$

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

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

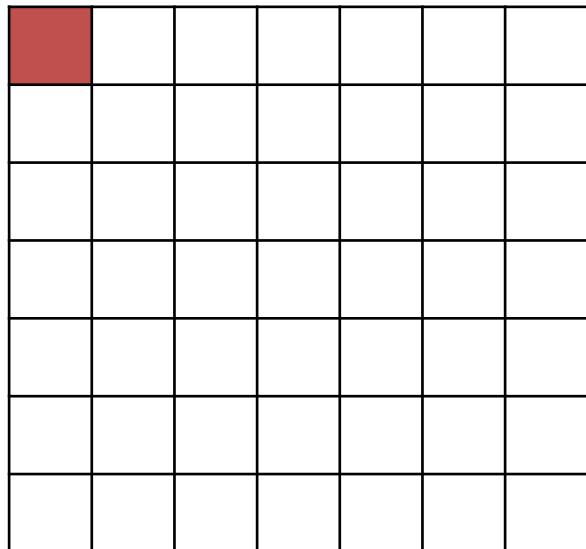
Fractions are illegal

Convolution Layers: Dimensions



Shrinking down so quickly ($32 \rightarrow 28 \rightarrow 24 \rightarrow 20$) 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 7×7 + 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

Why padding?

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

Convolution Layers: Padding

Image 7×7 + 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

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:

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

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

Convolution Layers: Padding

Image 7×7 + 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

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

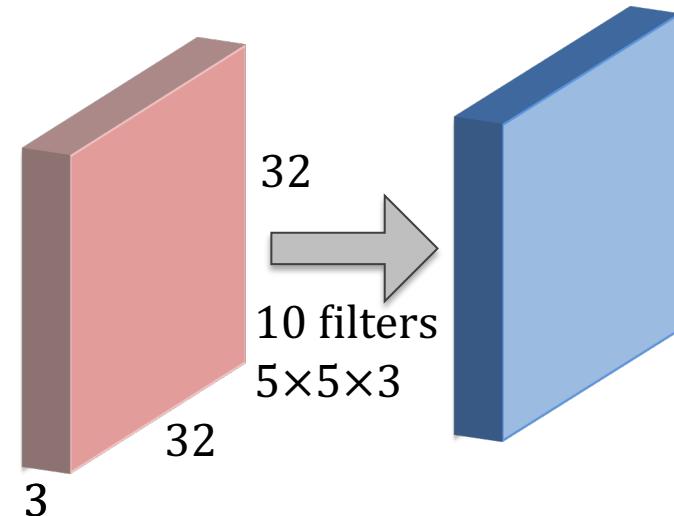
Pad 2

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

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

Convolution Layers: Dimensions

Example

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

10 filters 5×5

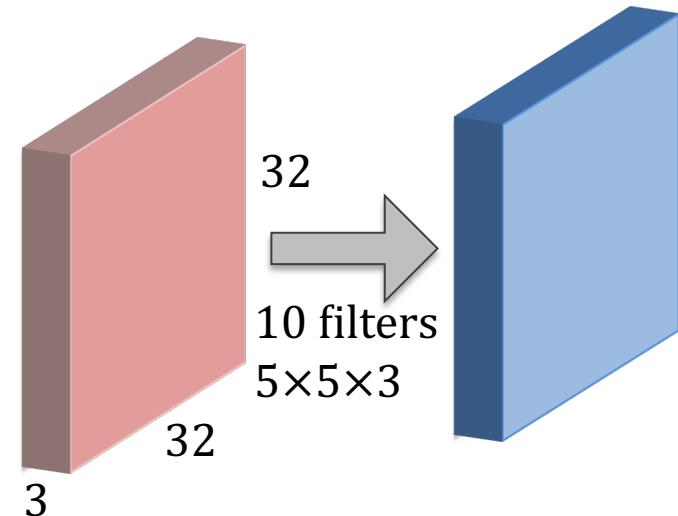
Stride 1

Pad 2

Output size is:

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

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



Remember

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

Convolution Layers: Dimensions

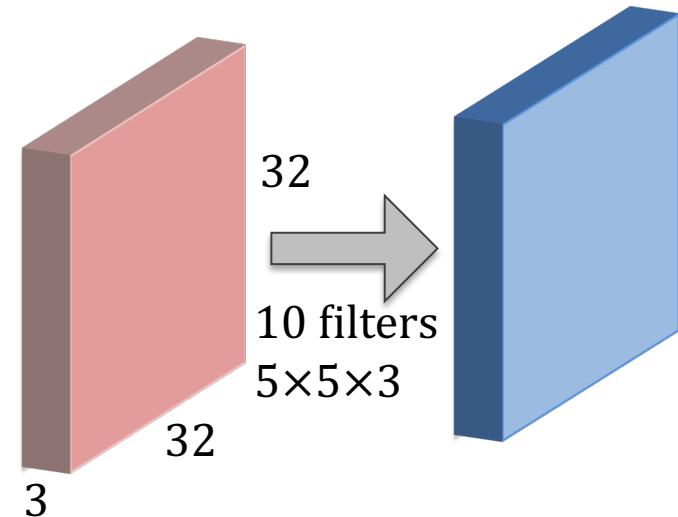
Example

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 **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)**
 - 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: $(3, 4, 5, 5)$

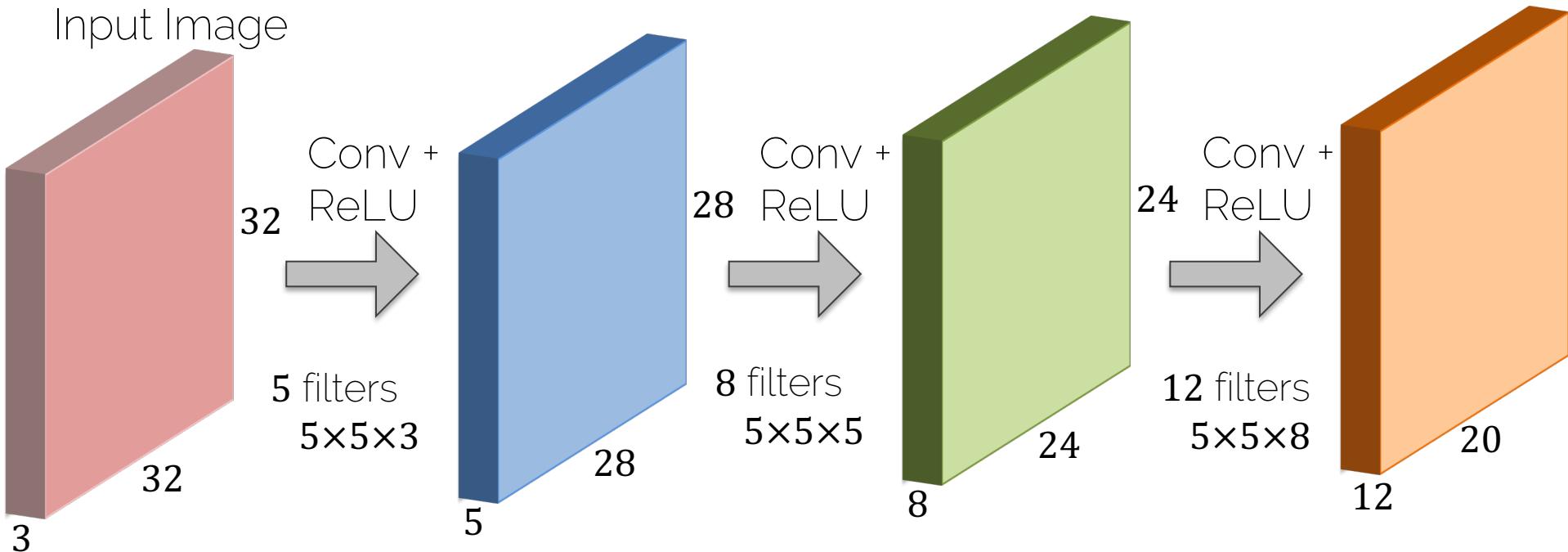
Input channels (RGB = 3) Filter size = 5×5

Output size =
4 filters

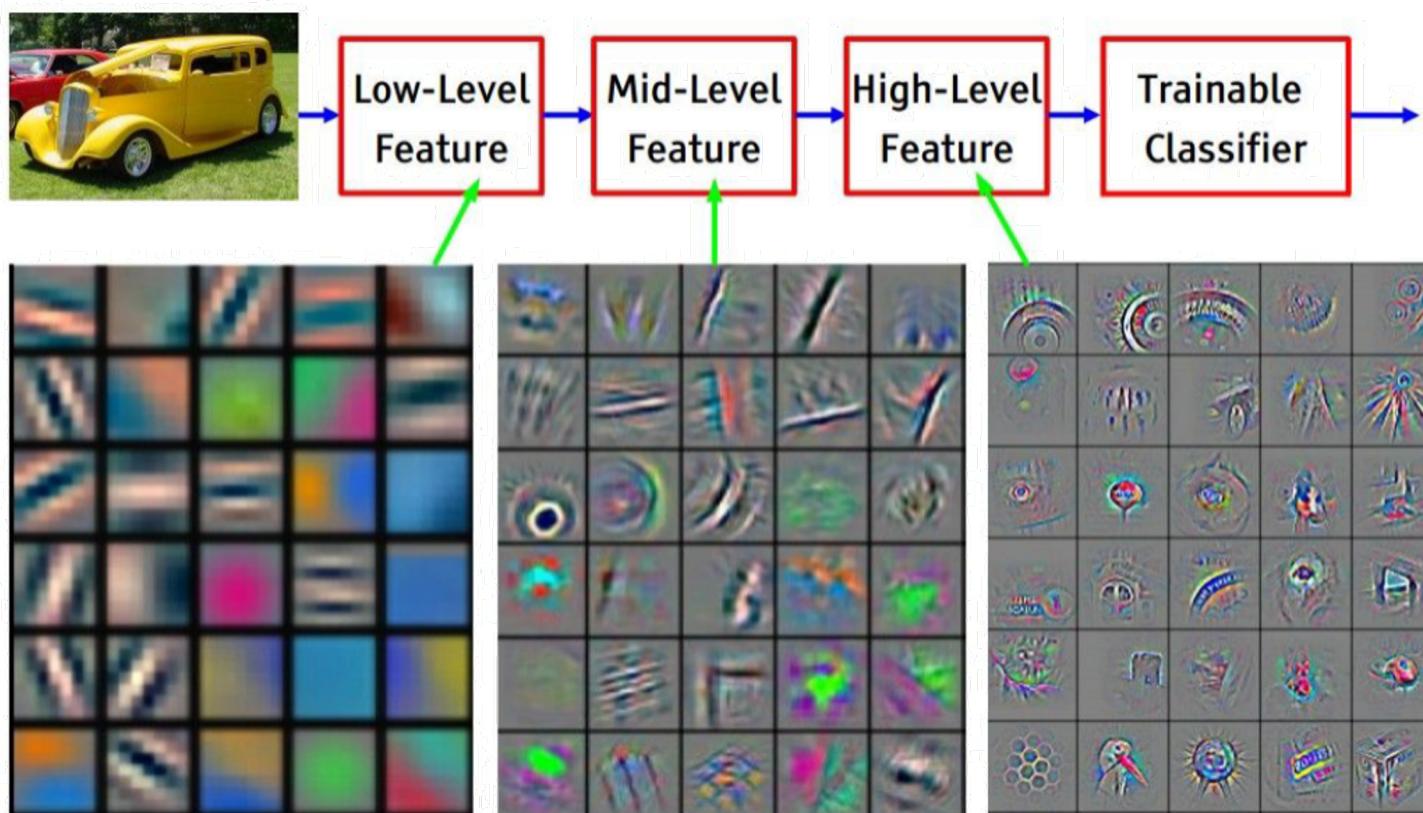
Convolutional Neural Network (CNN)

CNN Prototype

ConvNet is concatenation of Conv Layers and activations



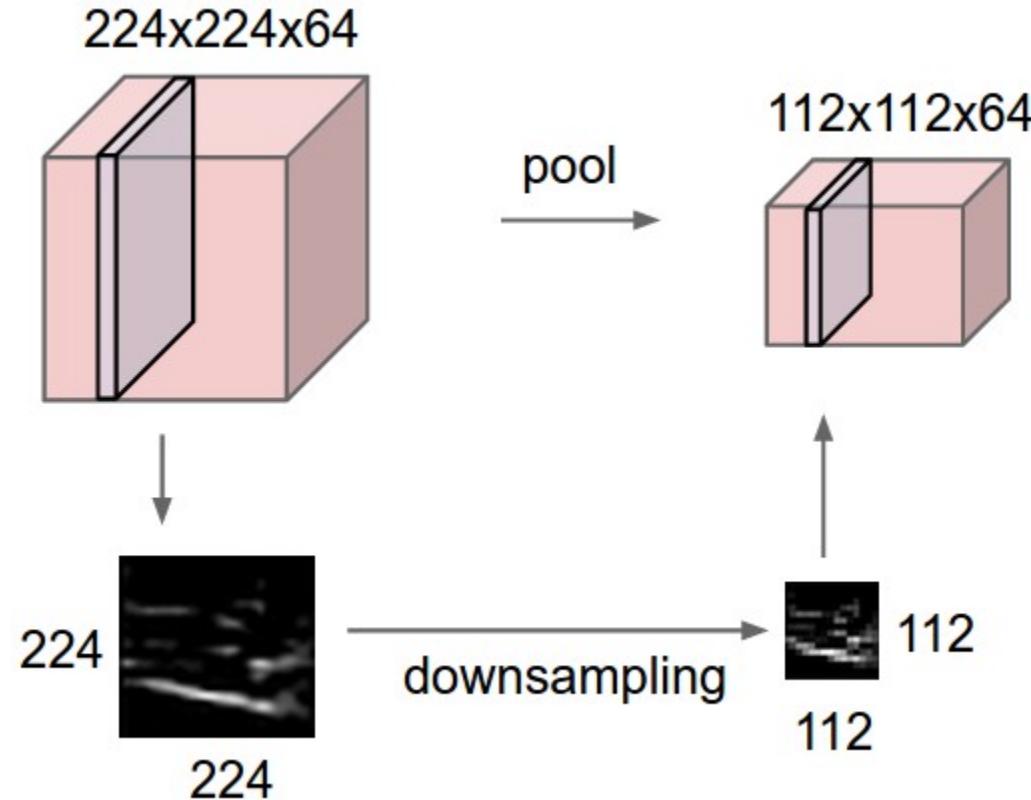
CNN Learned Filters



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

Pooling

Pooling Layer

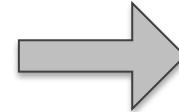


Pooling Layer: Max Pooling

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
- 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. it's fixed function

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

Common settings:

$$F = 2, S = 2$$

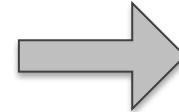
$$F = 3, S = 2$$

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

Average pool with
 2×2 filters and stride 2

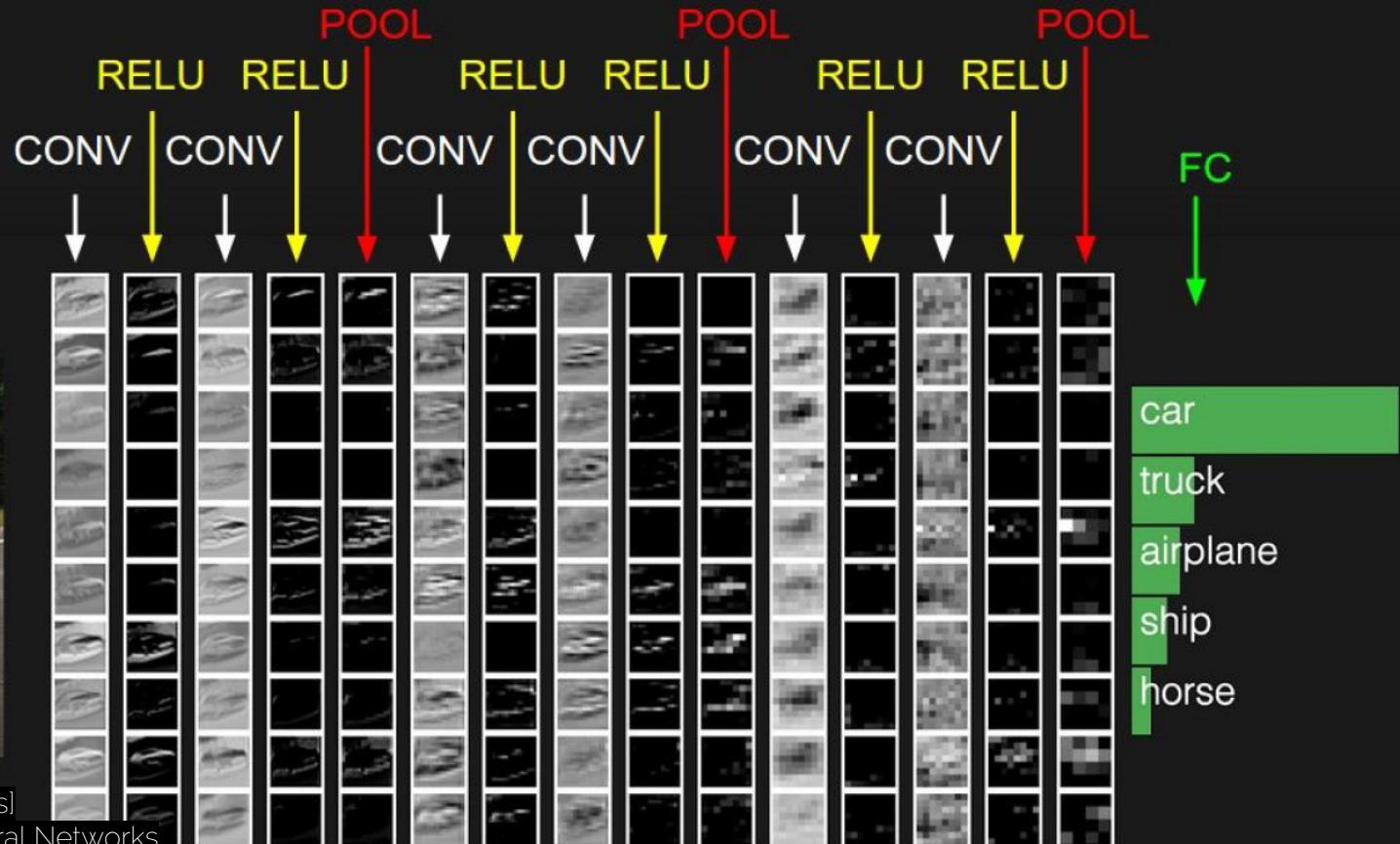


'Pooled' output

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

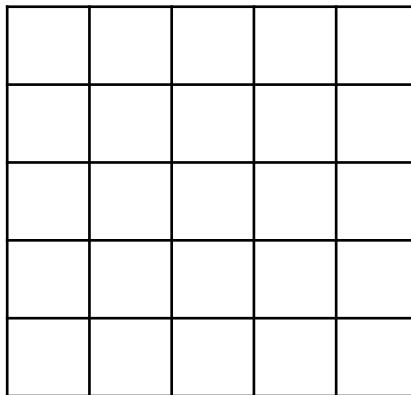
Convolutions vs Fully-Connected

- 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

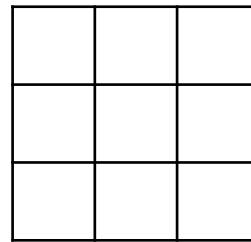
Receptive field

Receptive Field

- Spatial extent of the connectivity of a convolutional filter

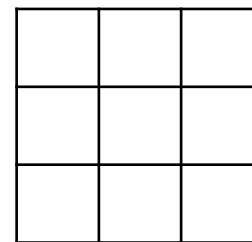


5x5 input



3x3 filter

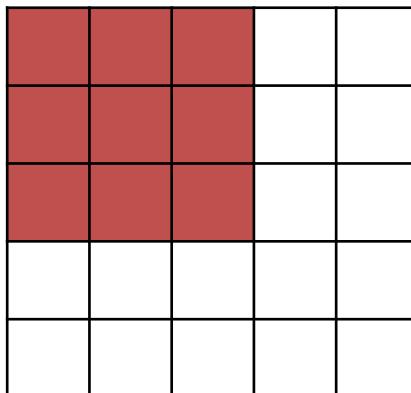
=



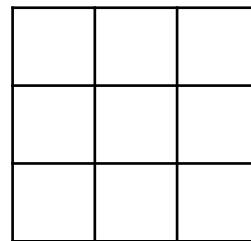
3x3 output

Receptive Field

- Spatial extent of the connectivity of a convolutional filter

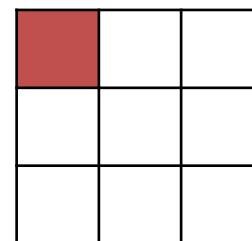


5x5 input



3x3 filter

=

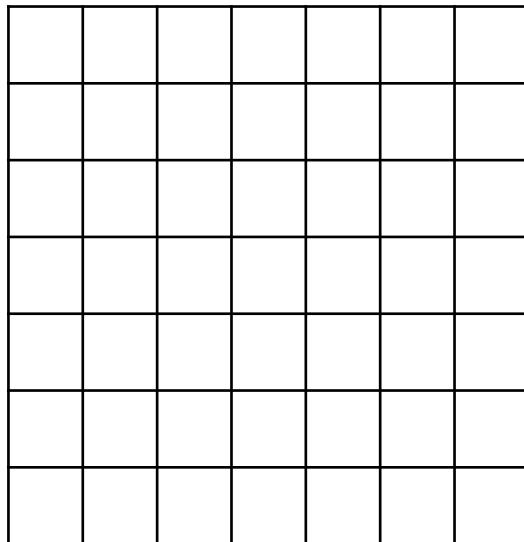


3x3 output

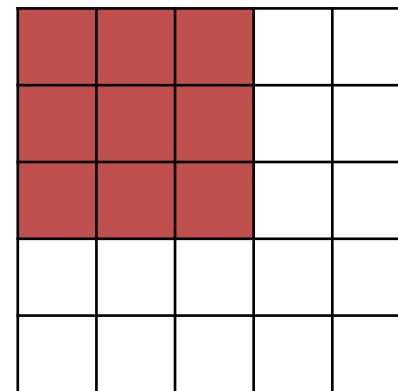
3x3 receptive field - 1 output pixel is connected to 9 input pixels

Receptive Field

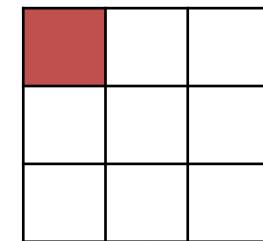
- Spatial extent of the connectivity of a convolutional filter



7x7 input



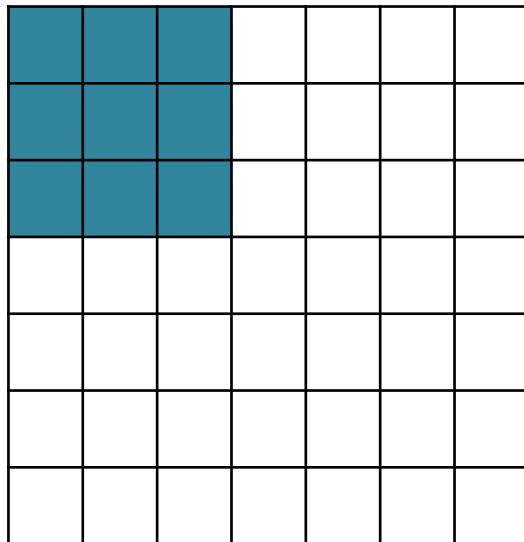
3x3 output



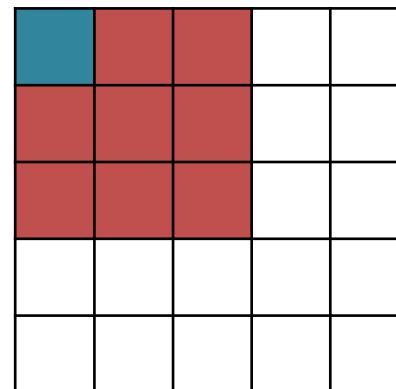
3x3 receptive field = 1 output pixel is connected to 9 input pixels

Receptive Field

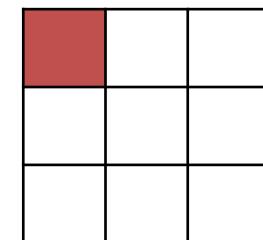
- Spatial extent of the connectivity of a convolutional filter



7x7 input



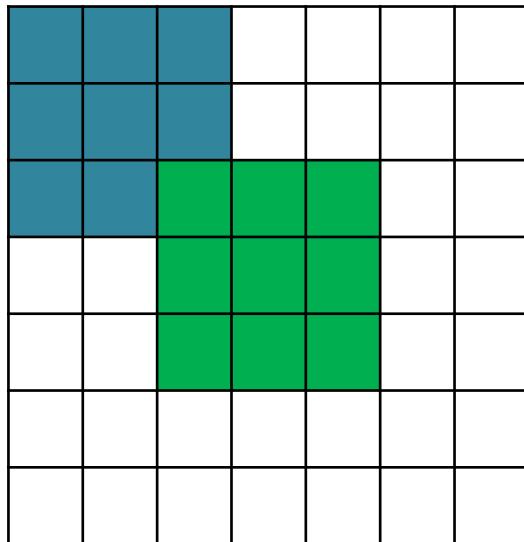
3x3 output



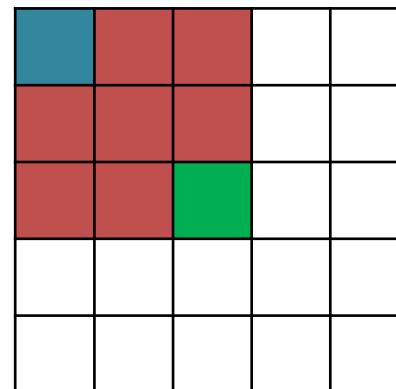
3x3 receptive field = 1 output pixel is connected to 9 input pixels

Receptive Field

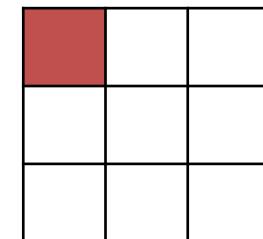
- Spatial extent of the connectivity of a convolutional filter



7x7 input



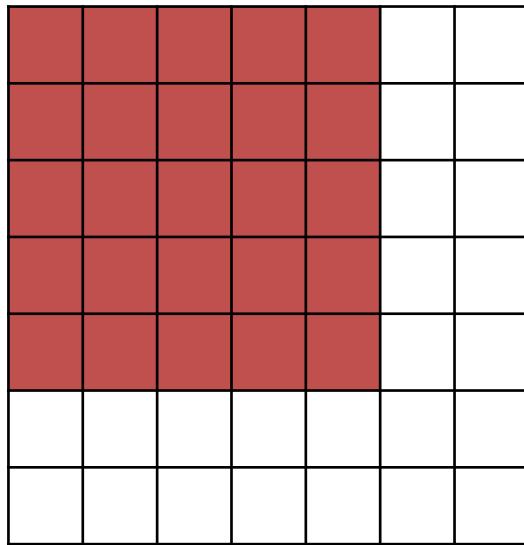
3x3 output



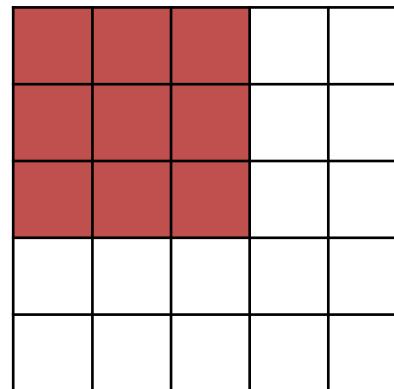
3x3 receptive field = 1 output pixel is connected to 9 input pixels

Receptive Field

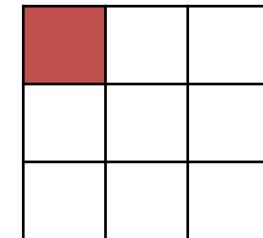
- Spatial extent of the connectivity of a convolutional filter



7x7 input



3x3 output



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

See you next time!

References

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