Lecture 9 – Convolutional Neural Networks

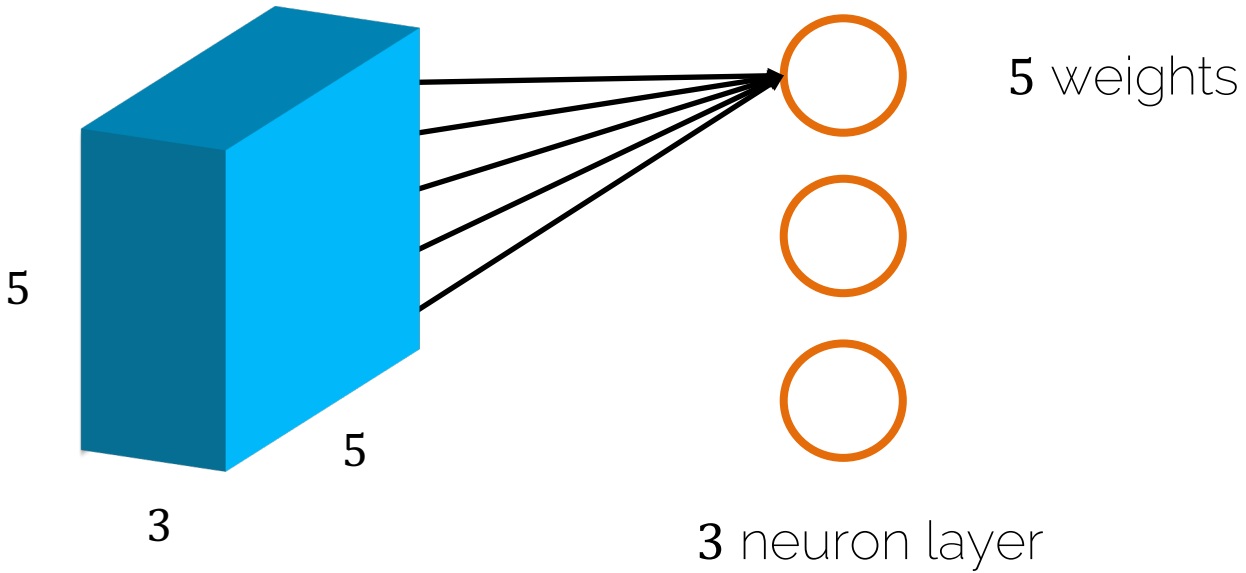
Manual Feature Extraction

domain knowledge → define features → detecting features to extract

* When building pipeline, you need to be invariant to diff variations in images e.g occlusions, deformation, scale, intra-class variations, viewpoint, illuminations.

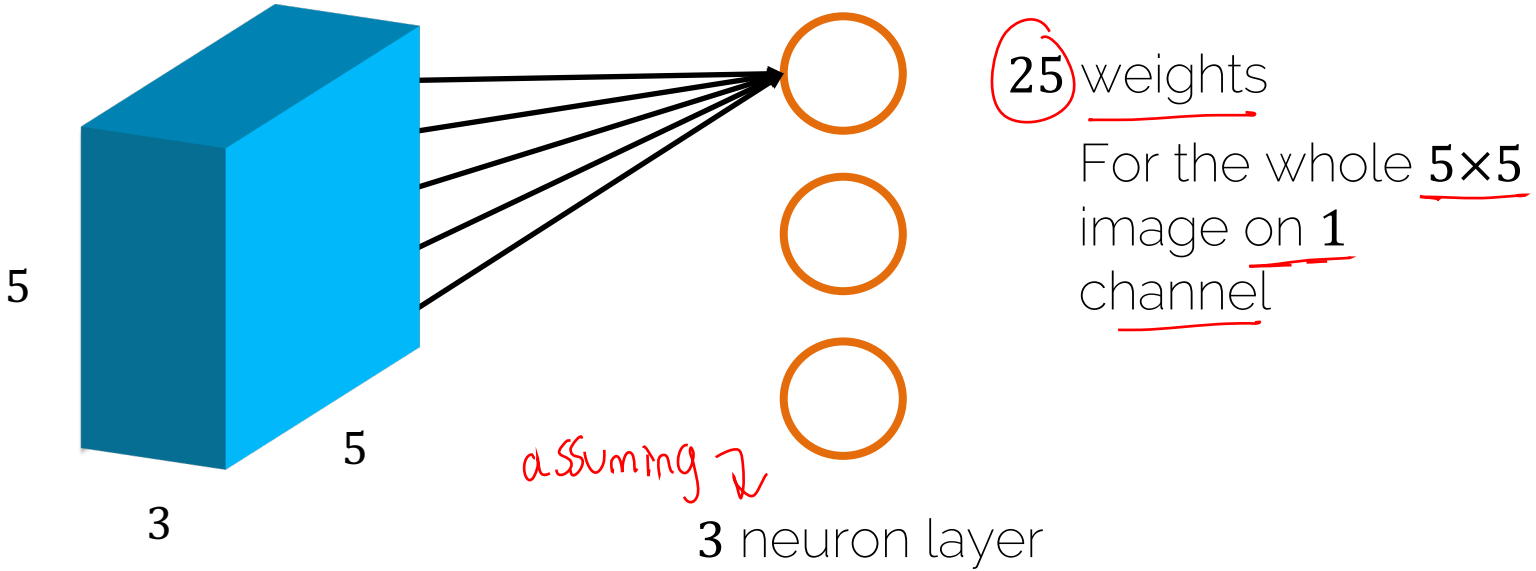
Problems using FC Layers on Images

- How to process a tiny image with FC layers



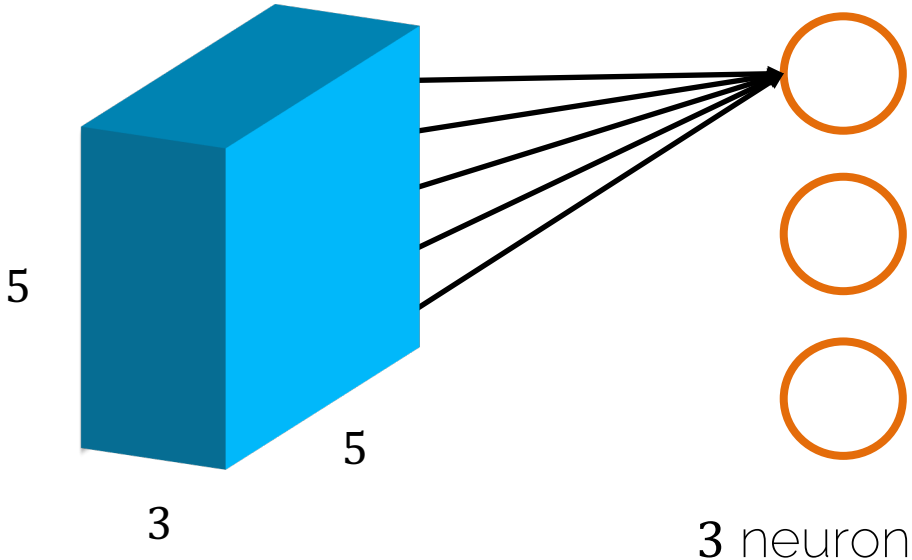
Problems using FC Layers on Images

- How to process a tiny image with FC layers



Problems using FC Layers on Images

- How to process a tiny image with FC layers



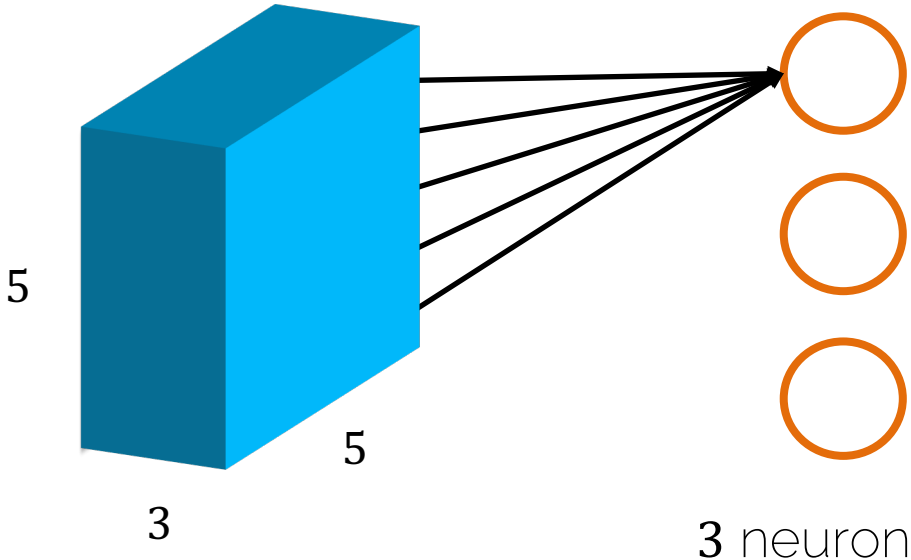
75 weights

For the whole 5×5 image on the 3 channel

only for one neuron

Problems using FC Layers on Images

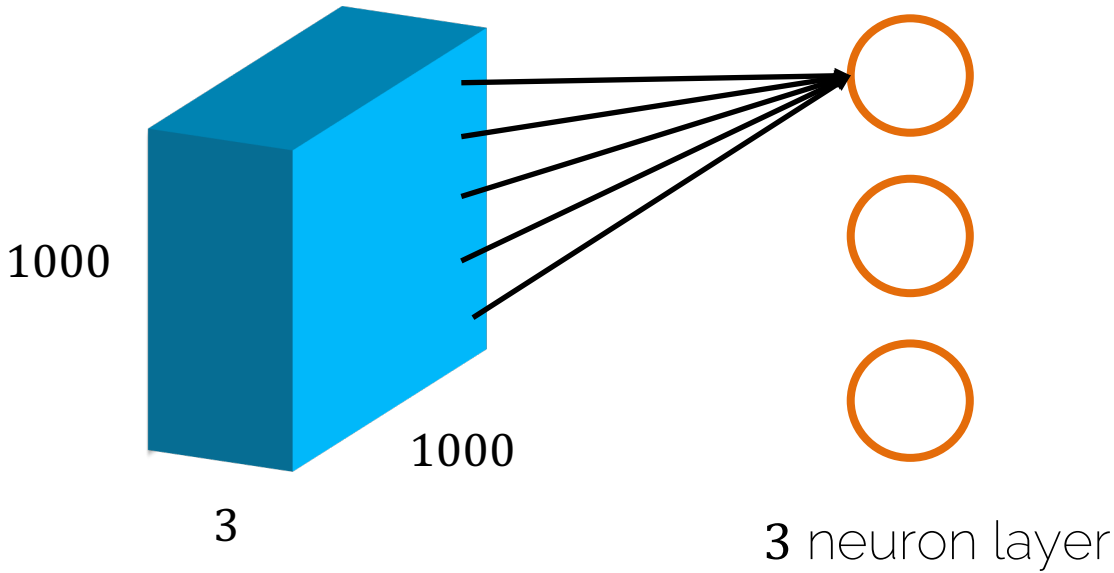
- How to process a tiny image with FC layers



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

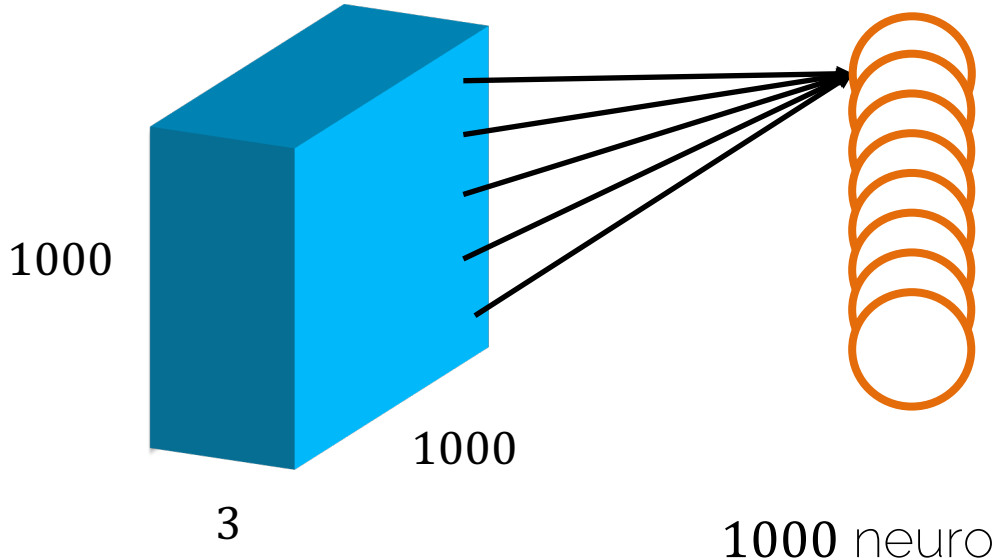
Problems using FC Layers on Images

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



Problems using FC Layers on Images

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



$3\ billion$ weights

IMPRACTICAL

1000 neuron layer

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!

⇒ Spatial structure of image collapse . i.e no spatial info is presered

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

learnable during training

* The filter used on a single 2D plane contains a weight that is shared across all filters, used across the same plane.

Advantage: maintaining same feature detector - used in one part of input data - across all other areas.

Using CNNs in Computer Vision

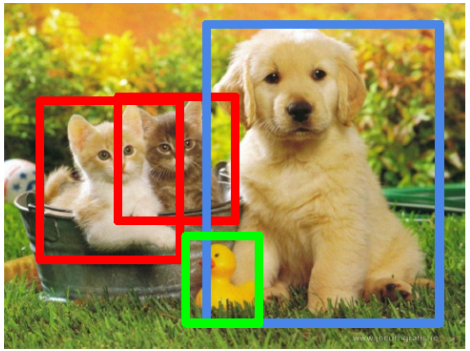
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation

CAT, DOG, DUCK

Single object

Multiple objects

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

Idea → connecting patches of input to neurons in hidden layer.

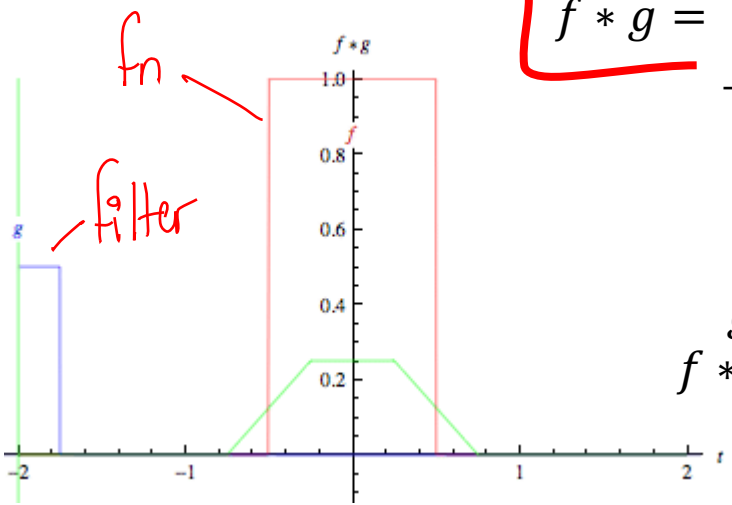
Instead of connecting these patches uniformly to hidden layers, we weight pixels.

Convolutions

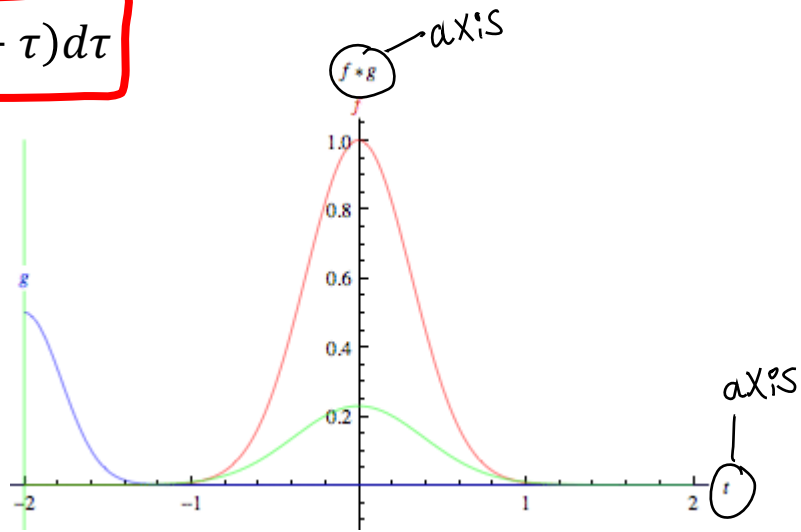
↙ patch operation done element-wise

What are Convolutions?

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



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



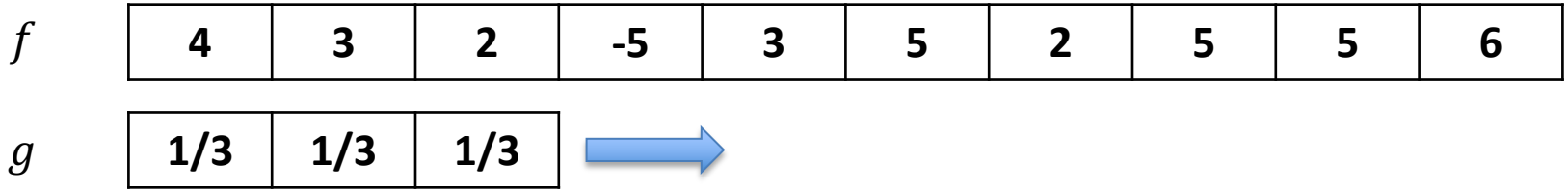
Convolution of two box functions

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

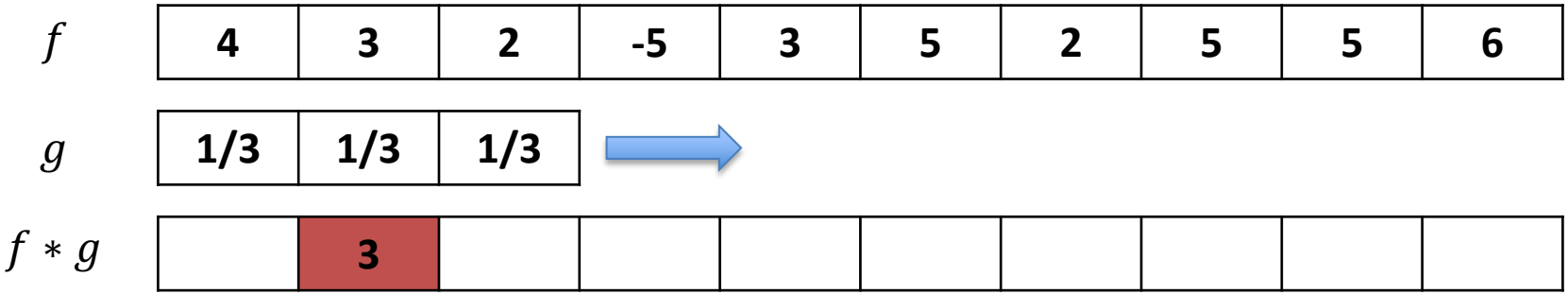


'Slide' filter kernel from left to right; at each position, compute a single value in the output data

think of it as a patch

What are Convolutions?

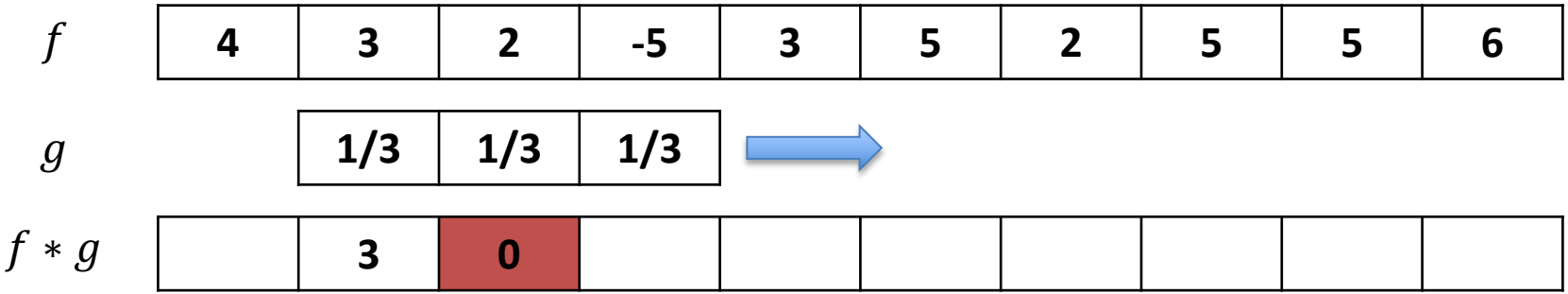
Discrete case: box filter



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

What are Convolutions?

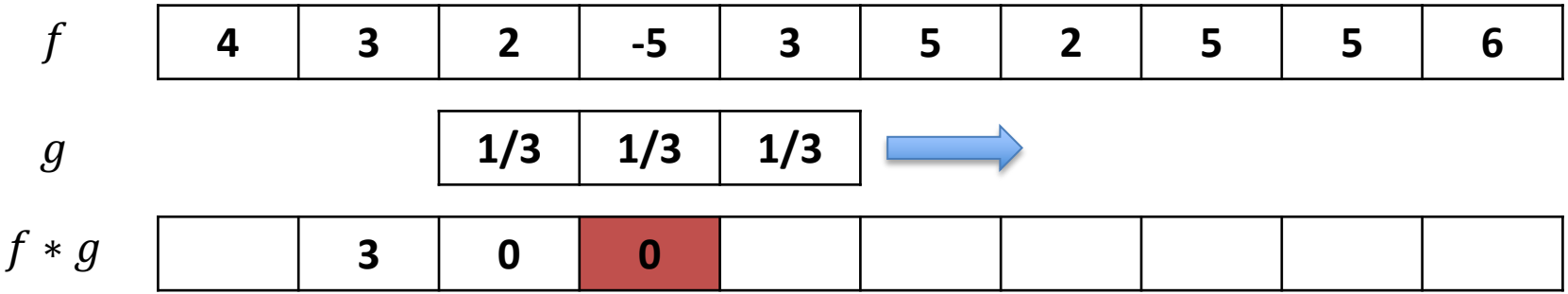
Discrete case: box filter



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

What are Convolutions?

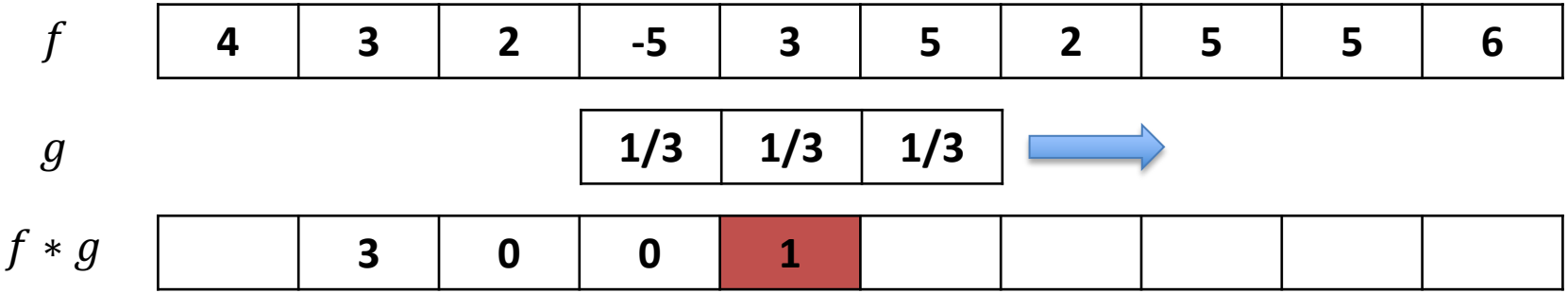
Discrete case: box filter



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

What are Convolutions?

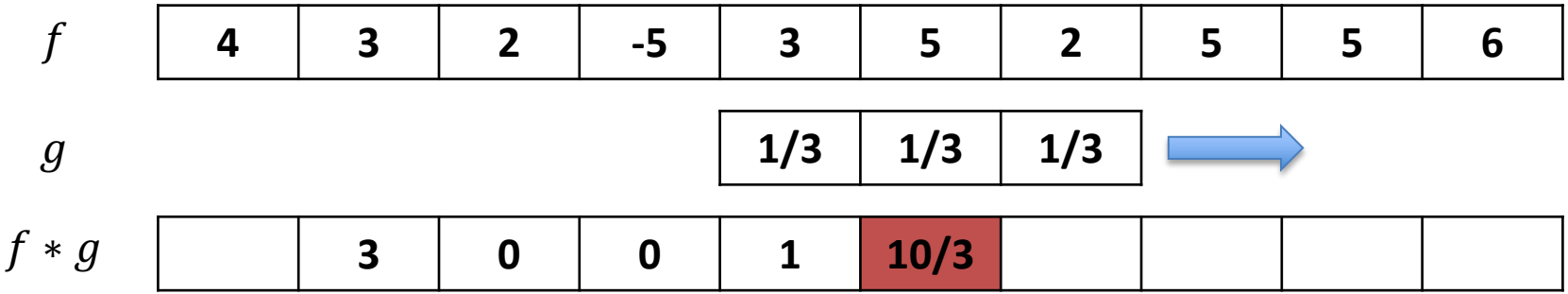
Discrete case: box filter



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

What are Convolutions?

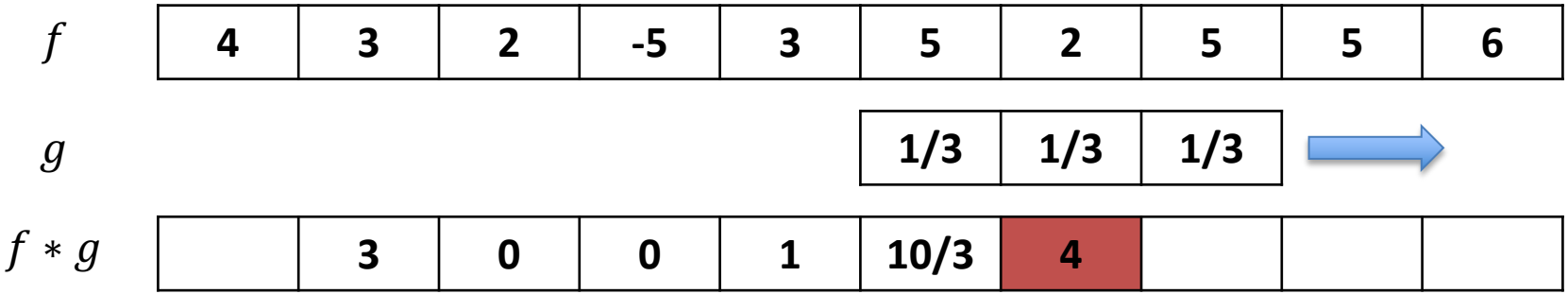
Discrete case: box filter



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

What are Convolutions?

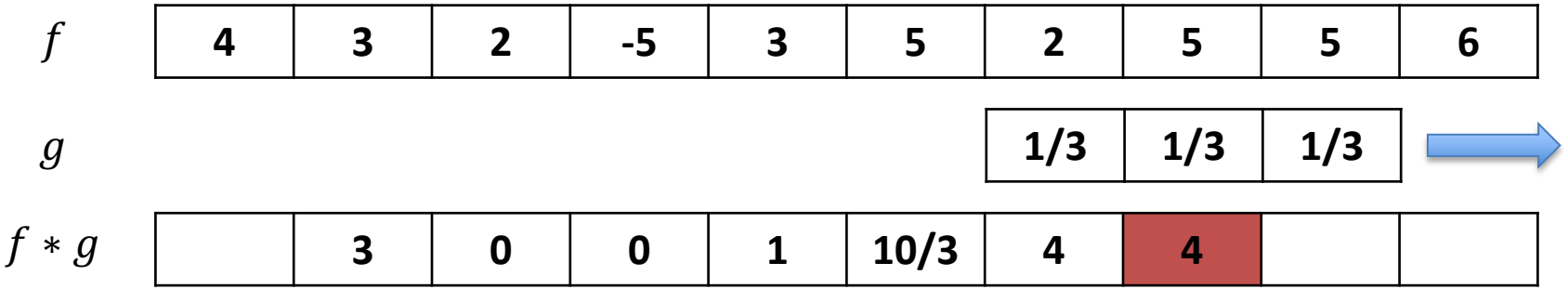
Discrete case: box filter



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

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

	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?

Discrete case: box filter

size: 10

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

no padding → output size reduced

size: 8

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

0 padding is done on input

Discrete case: box filter

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

1/3	1/3	1/3

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

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

What to do at boundaries?

Option 2: Pad (often 0's) preserving input size

Size: 10 →

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

Convolutions on Images

Image 5×5

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

Kernel 3×3

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

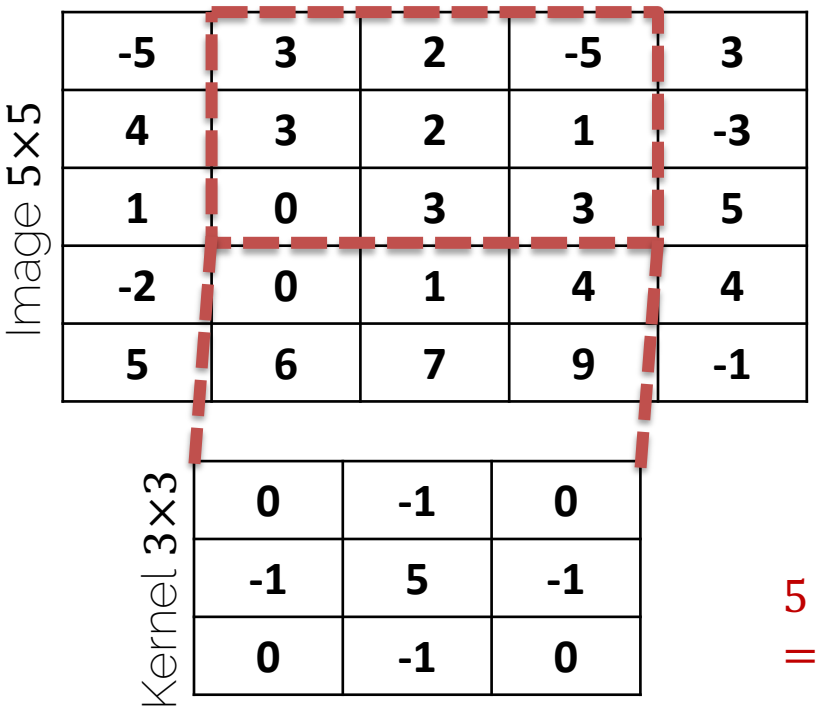
* how does it work?

Output 3×3

6		

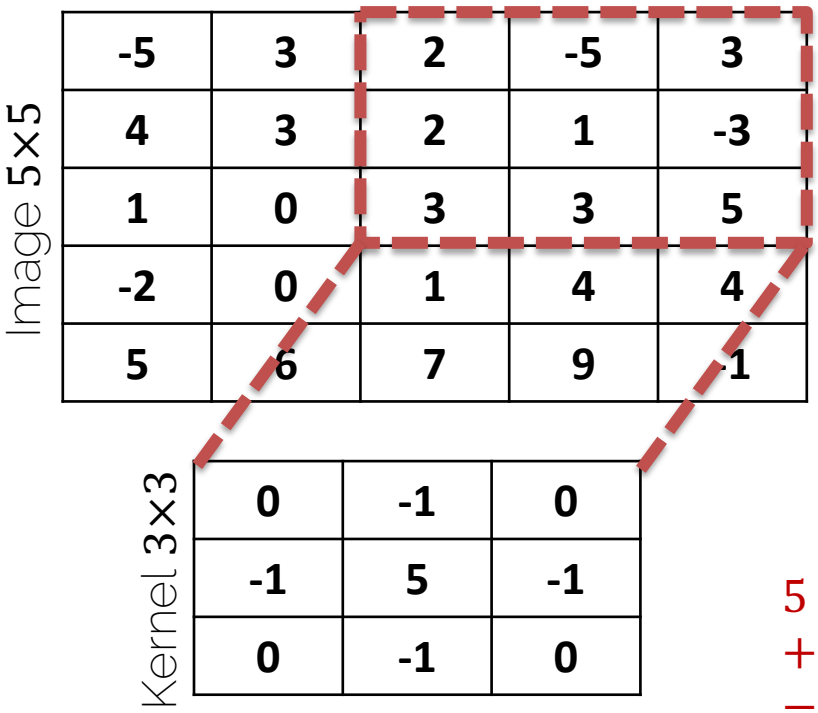
$$5 \cdot 3 + (-1) \cdot 3 + (-1) \cdot 2 + (-1) \cdot 0 + (-1) \cdot 4$$
$$= 15 - 9 = 6$$

Convolutions on Images



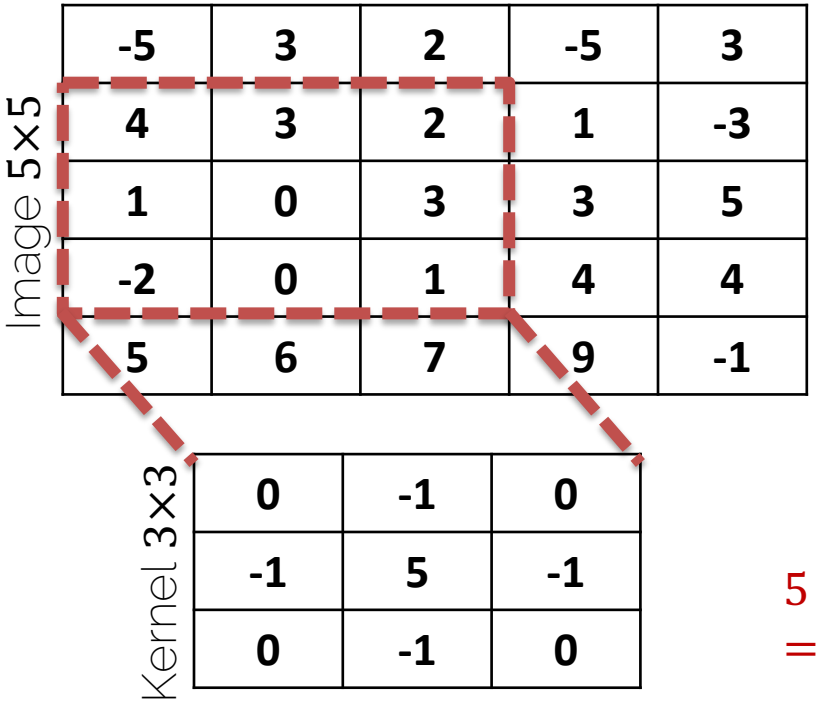
$$5 \cdot 2 + (-1) \cdot 2 + (-1) \cdot 1 + (-1) \cdot 3 + (-1) \cdot 3 = 10 - 9 = 1$$

Convolutions on Images



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

Convolutions on Images



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

Convolutions on Images

Image 5×5

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

Kernel 3×3

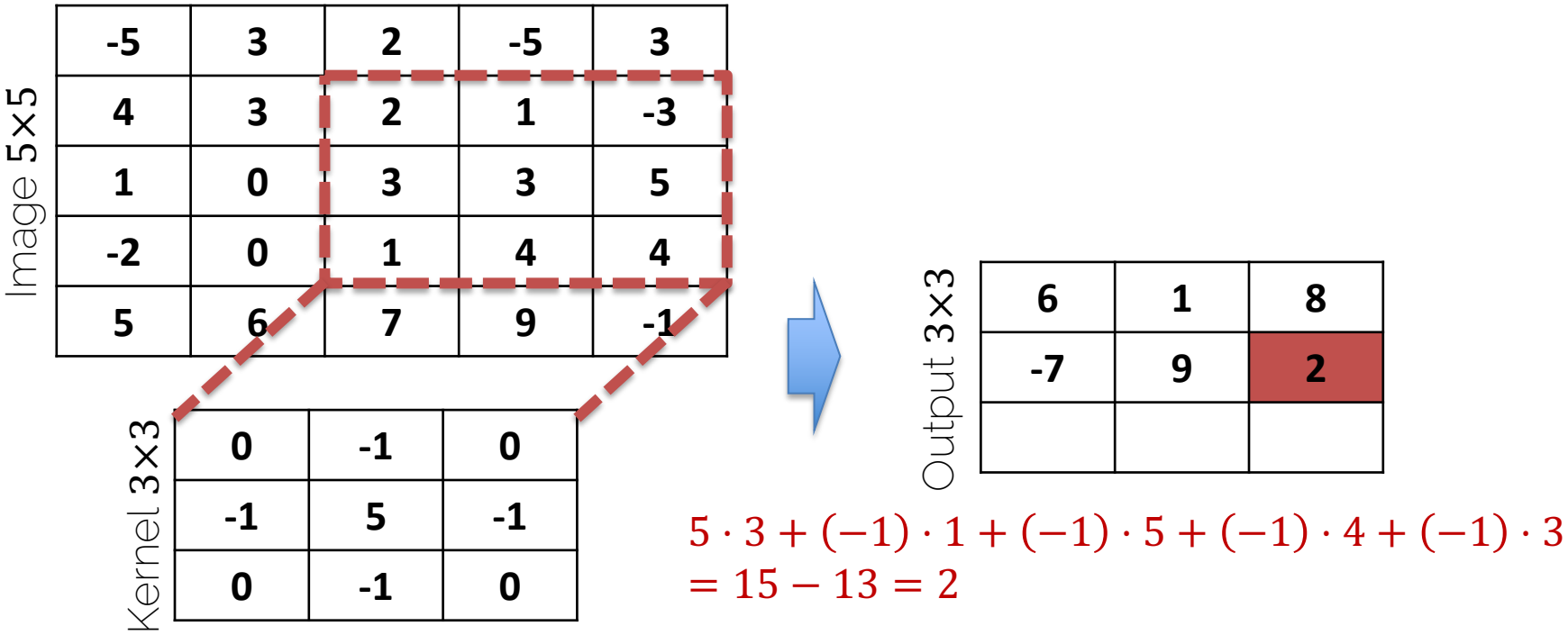
0	-1	0
-1	5	-1
0	-1	0

Output 3×3

6	1	8
-7	9	

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

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		

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

Convolutions on Images

Image 5×5

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

Kernel 3×3

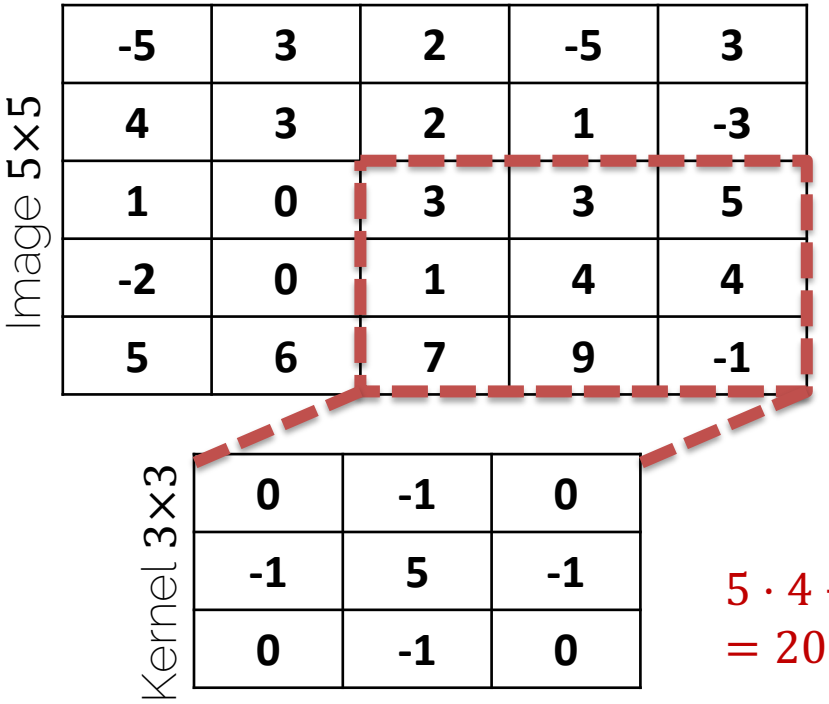
0	-1	0
-1	5	-1
0	-1	0

Output 3×3

6	1	8
-7	9	2
-5	-9	

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

Convolutions on Images



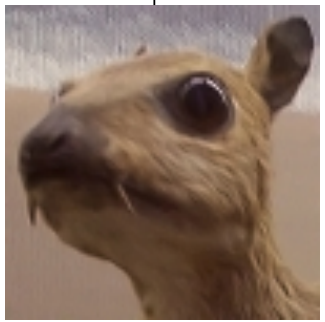
$$5 \cdot 4 + (-1) \cdot 3 + (-1) \cdot 4 + (-1) \cdot 9 + (-1) \cdot 1 = 20 - 17 = 3$$

idea: is to learn those kernels

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!

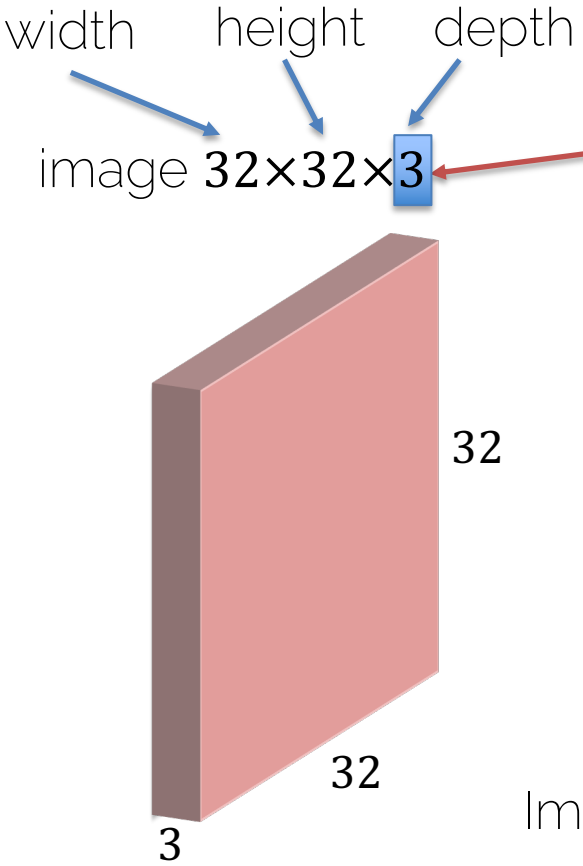
Sharpen

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

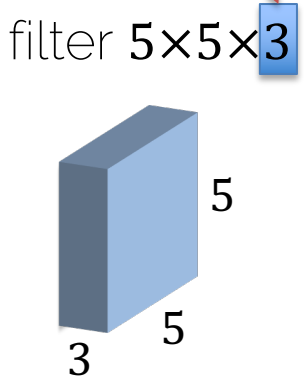
Gaussian blur

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

Convolutions on RGB Images



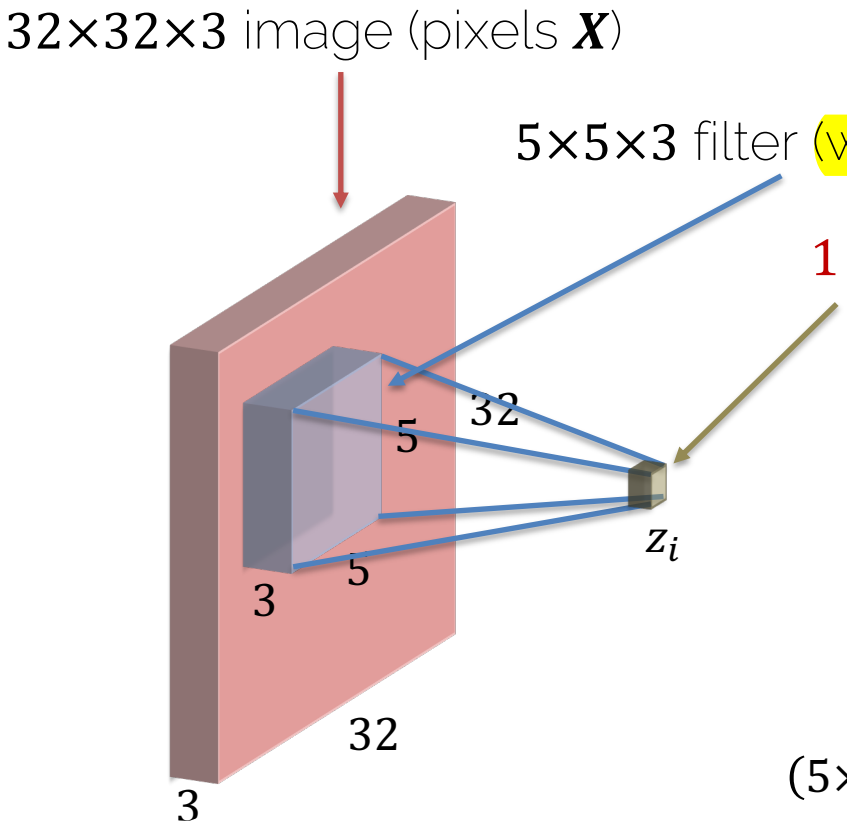
Depth dimension *must* match;
i.e., filter extends the full depth of the input



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

* Feature Mapping is the idea that each convolutional filter can be trained to search for different features in an image.



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

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

$(5\times5\times3)\times1$ $(5\times5\times3)\times1$ 1

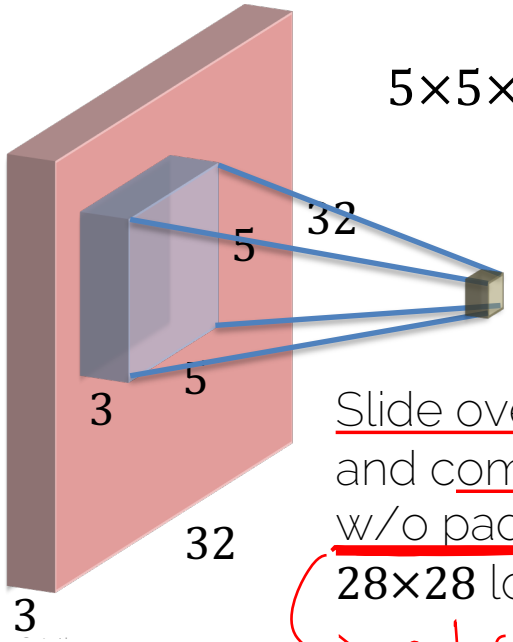
"one single value/pixel"

just a representation of conv operation

fog

Activation map
(also feature map)

32×32×3 image



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

→ reduce output size

whichever feature extracted will be activated in the depth of FM

1 ← # filters applied

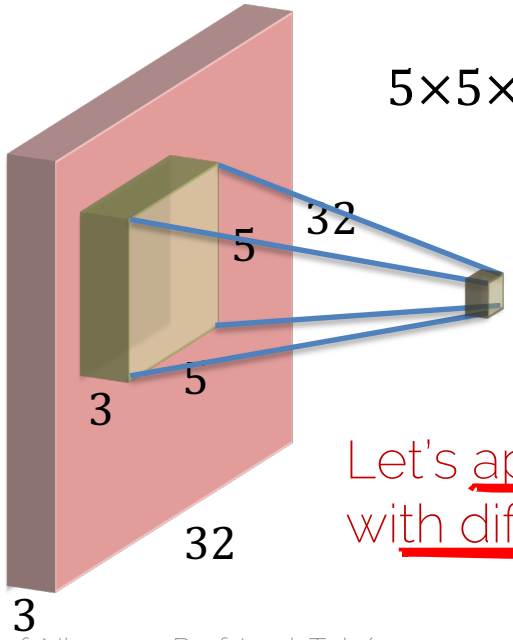
Remember

* think of filter as a set of weights, used to extract local features
* multiple filters are used to extract diff features
* spatially sharing parameters of each filter

Convolution Layer

Convolution Layer

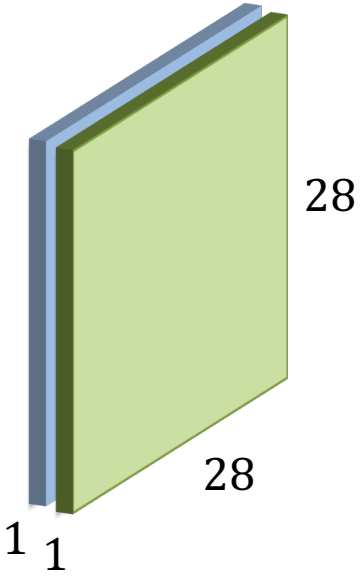
32×32×3 image



5×5×3 filter

Convolve

Activation maps

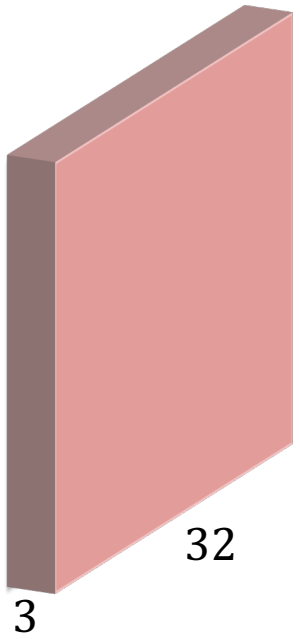


Let's apply a different filter with different weights!

↳ Constitutes applying multiple filters

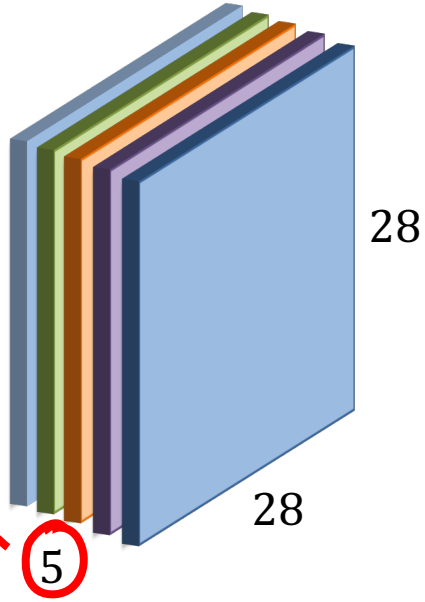
Convolution "Layer"

32×32×3 image



Convolve

Activation maps



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

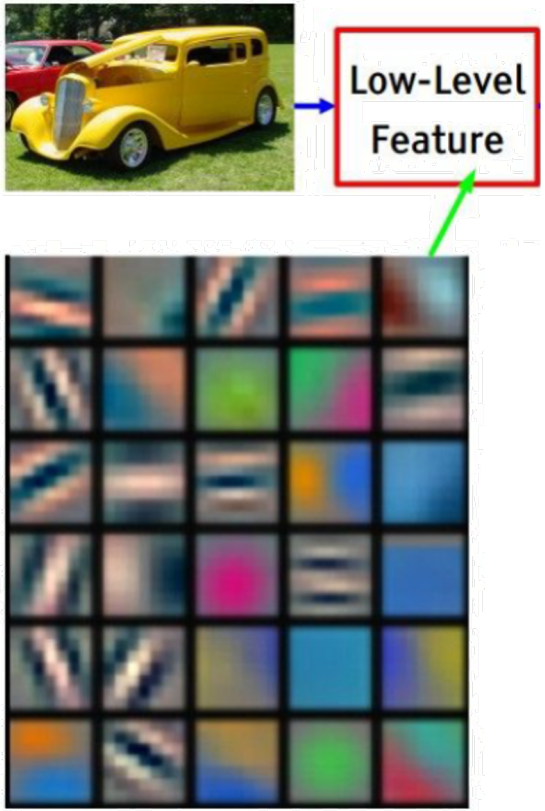
- A basic layer is defined by
  - Filter width and height (depth is implicitly given)
  - Number of different filter banks (#weight sets)

channels of input

* Each filter captures a different image characteristic

* In exam, try to be explicit as much as possible.
  Use H & W instead of size

Different Filters



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

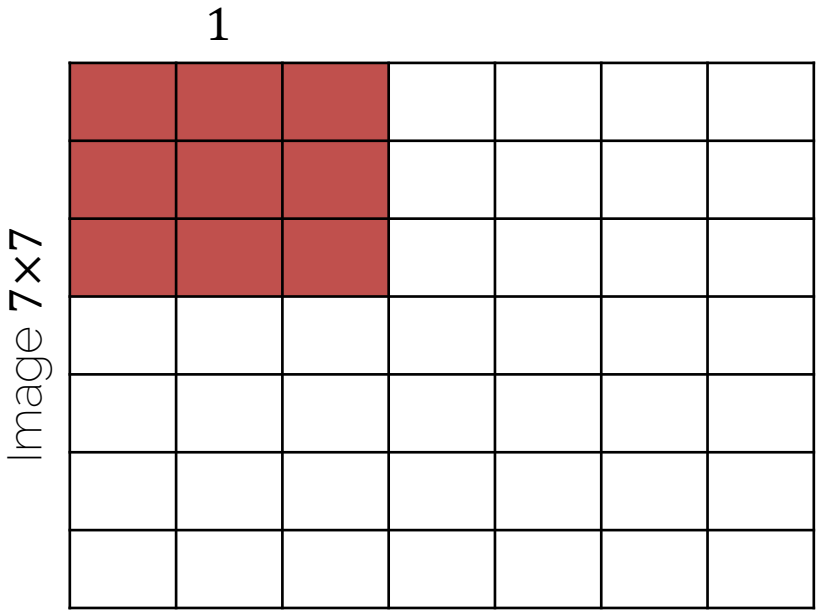
* Extraction is done from prev layer, obviously.

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

* Human detect # features to be extracted by setting # filters to be applied

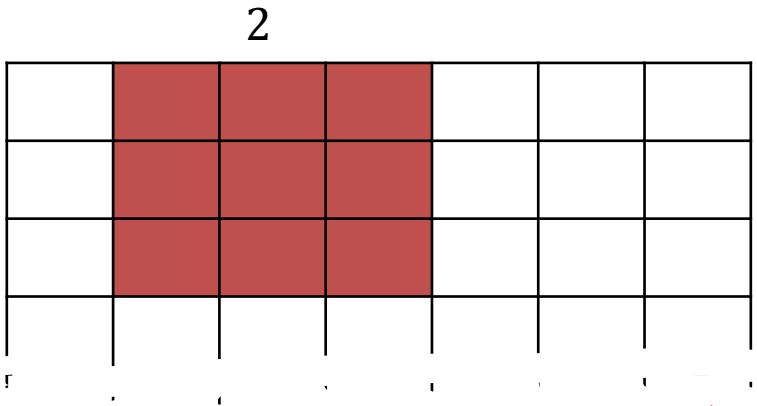
Dimensions of a Convolution Layer

Convolution Layers: Dimensions



Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Dimensions

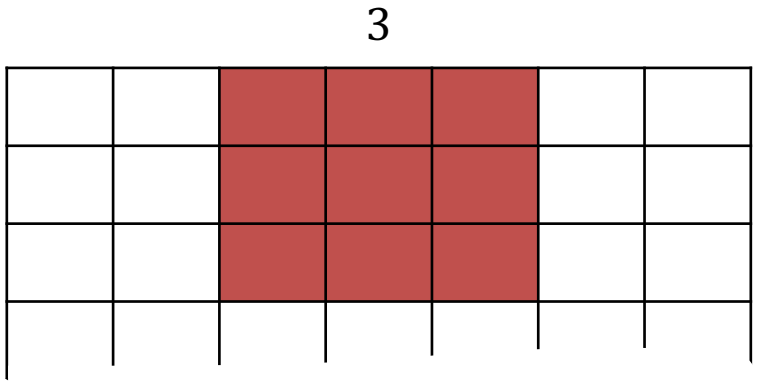


Input:	7×7
Filter:	3×3
Output:	5×5

* Multiple filters are commonly called channels.

* Each of these channels will end up being trained to detect certain key features.

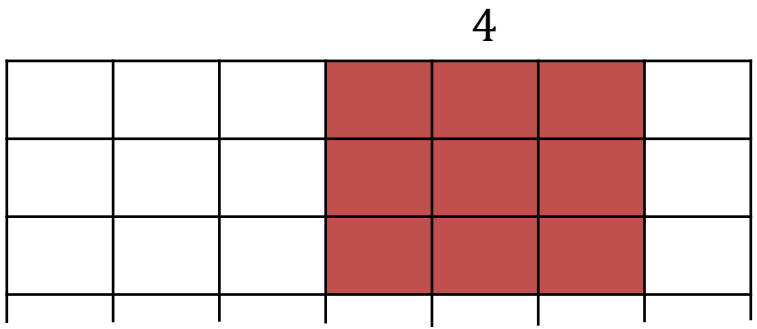
Convolution Layers: Dimensions



Input:	7×7
Filter:	3×3
Output:	5×5

* The output of conv layer, for a gray-scale image, will therefore actually have 3 dim — 2D for each of the channels, then another dim for the number of different channels.

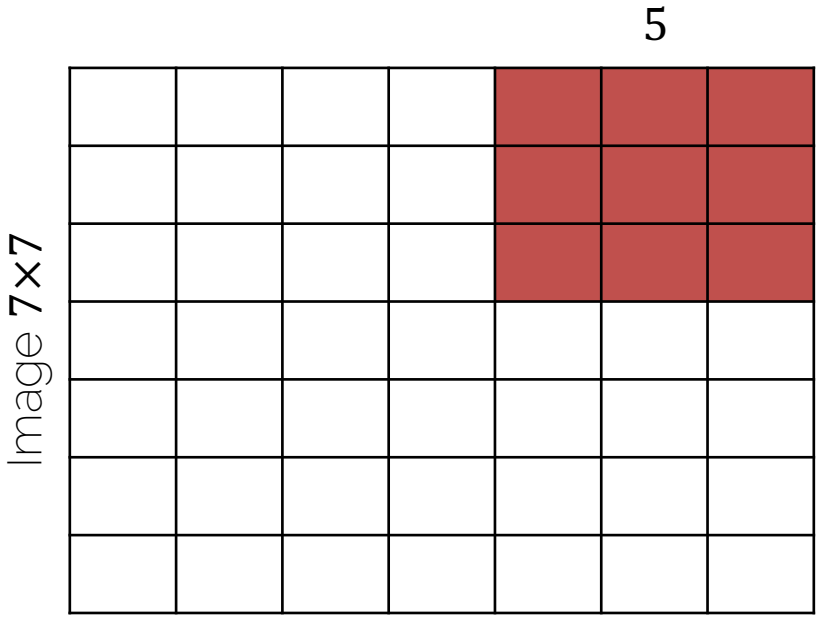
Convolution Layers: Dimensions



Input: 7×7
Filter: 3×3
Output: 5×5

* If the input is itself multi-channelled, as in the case of a color RGB image, the output will actually be 4D.

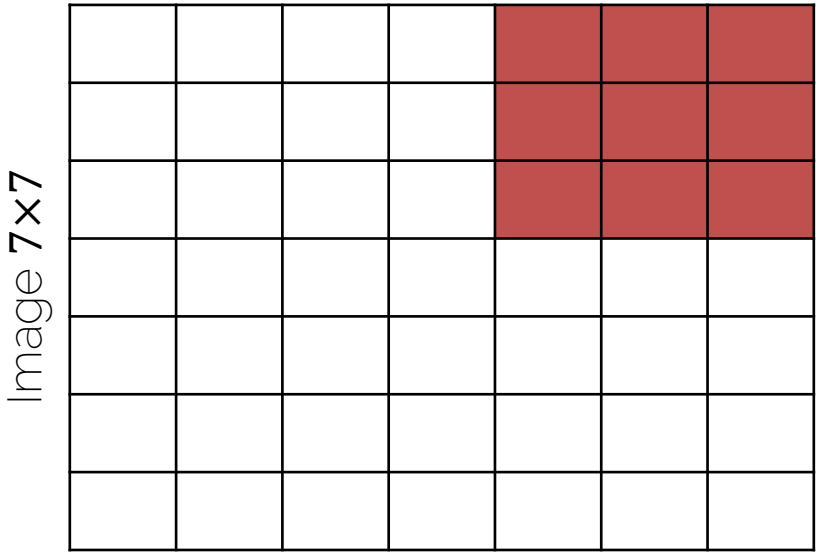
Convolution Layers: Dimensions



Input: 7×7
Filter: 3×3
Output: 5×5

Convolution Layers: Stride

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



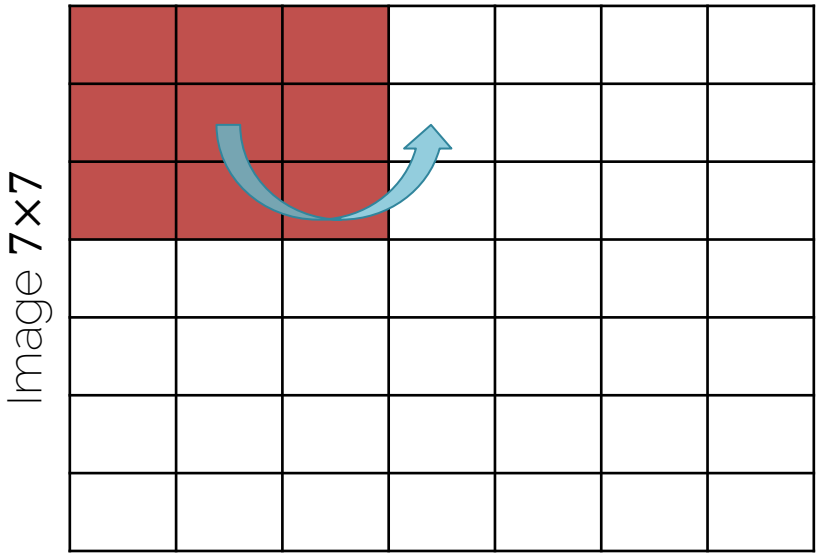
Input: 7×7
Filter: 3×3
Stride: 1
Output: 5×5

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

Controls how the filters convolves around the input volume.

Convolution Layers: Stride

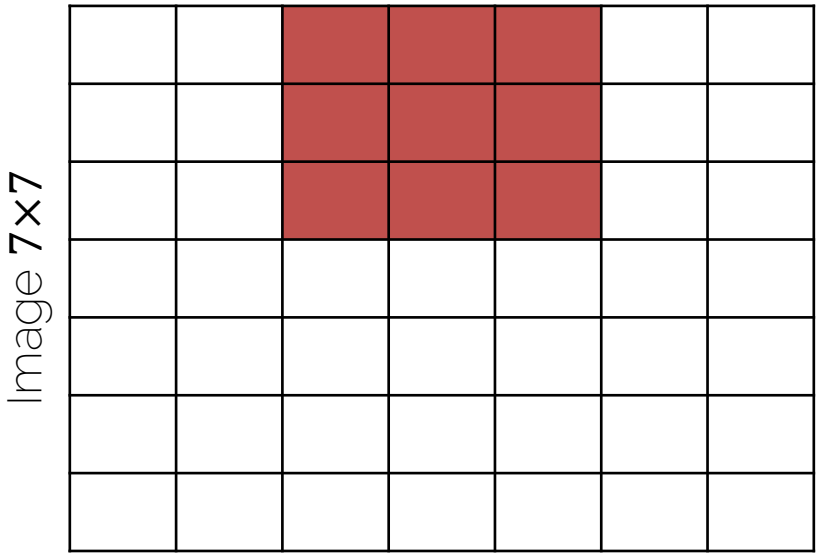
With a **stride** of **2**



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

Convolution Layers: Stride

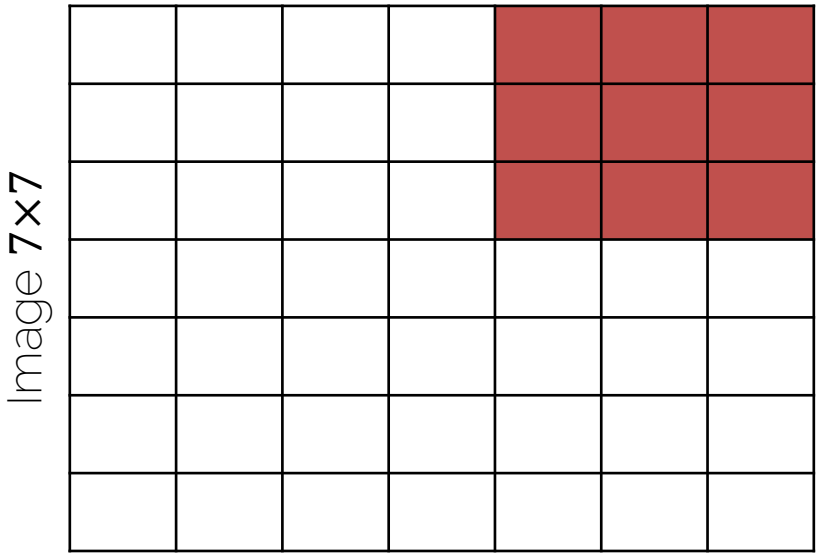
With a **stride** of **2**



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

Convolution Layers: Stride

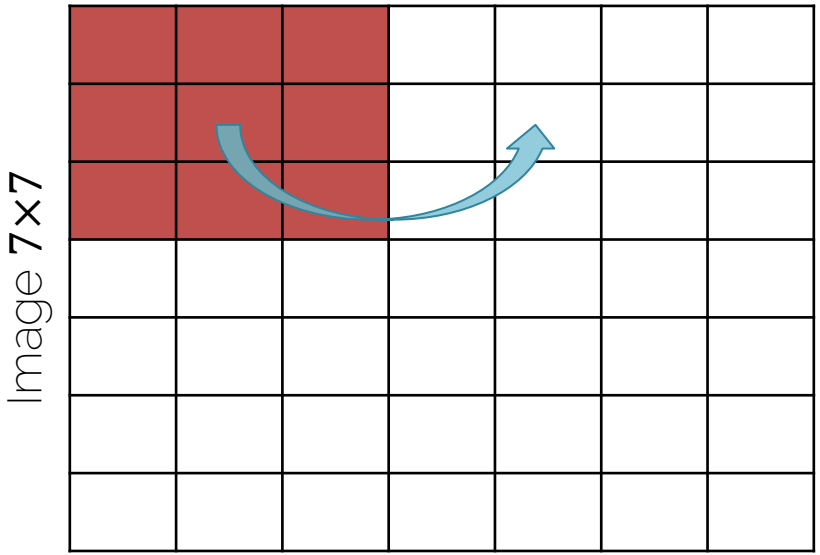
With a **stride** of **2**



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

Convolution Layers: Stride

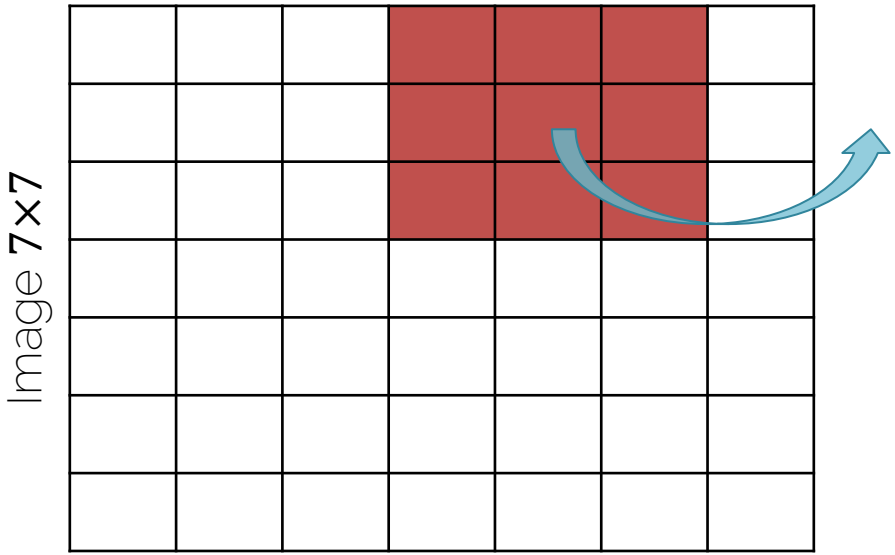
With a **stride** of **3**



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

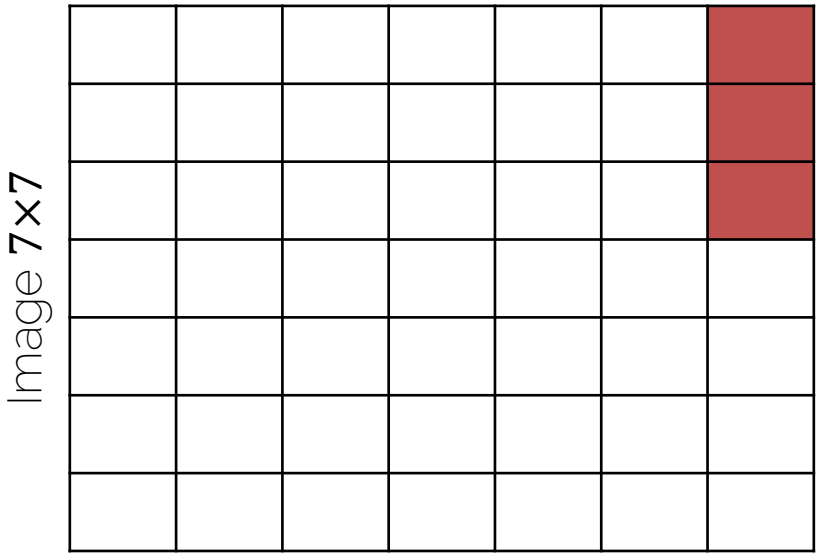
Convolution Layers: Stride

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



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

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



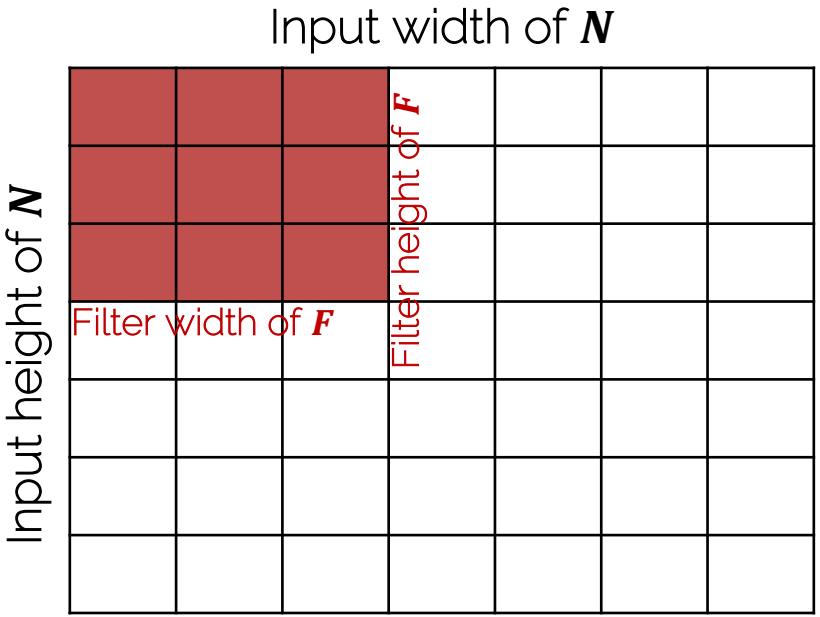
Input:	7×7
Filter:	3×3
Stride:	3
Output:	? × ?

NB

Does not really fit (remainder left)

→ Illegal stride for input & filter size!

Dimensions



Input: $N \times N$

Filter: $F \times F$

Stride: S

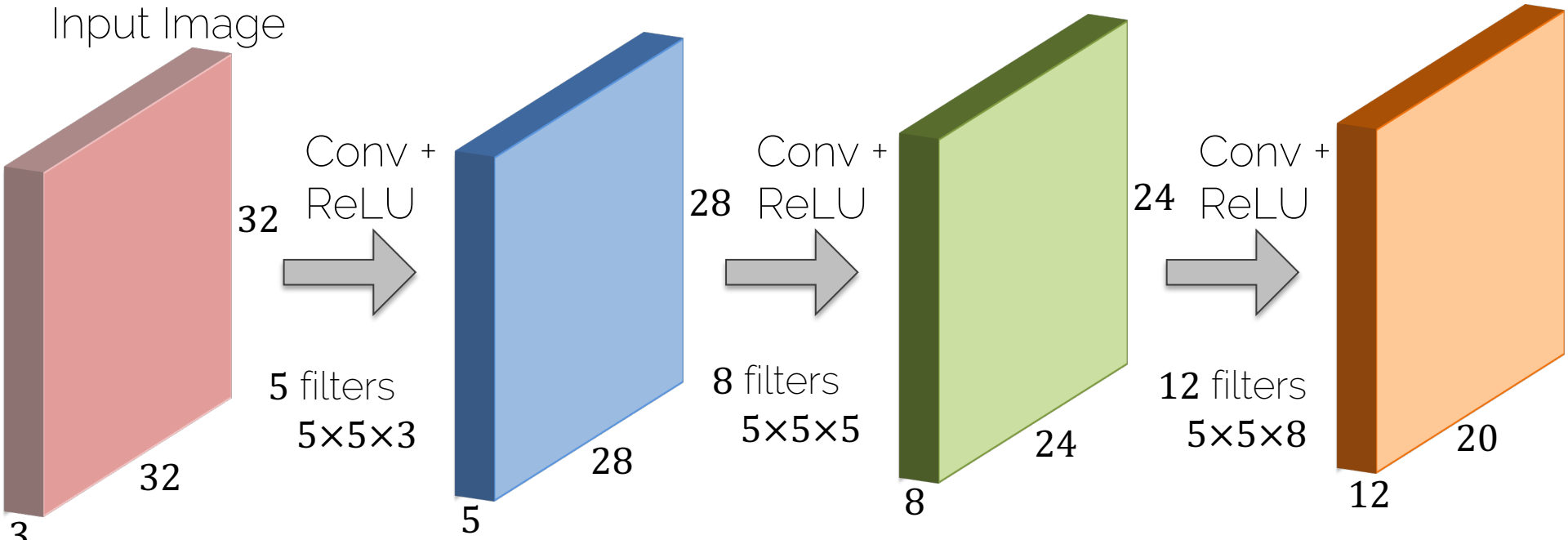
Output: $\left(\frac{N-F}{S}+1\right) \times \left(\frac{N-F}{S}+1\right)$

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

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

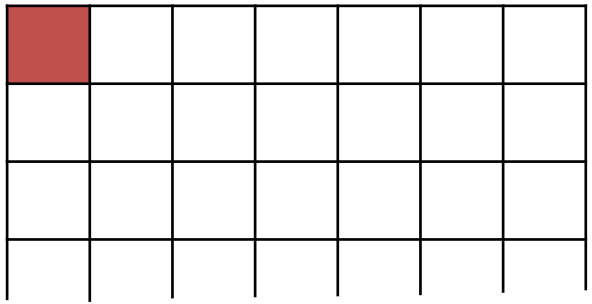
$N = 7, F = 3, S = 3$: $\frac{7-3}{3}+1 = 2.\bar{3}$

Fractions are illegal



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

Convolution Layers: Padding



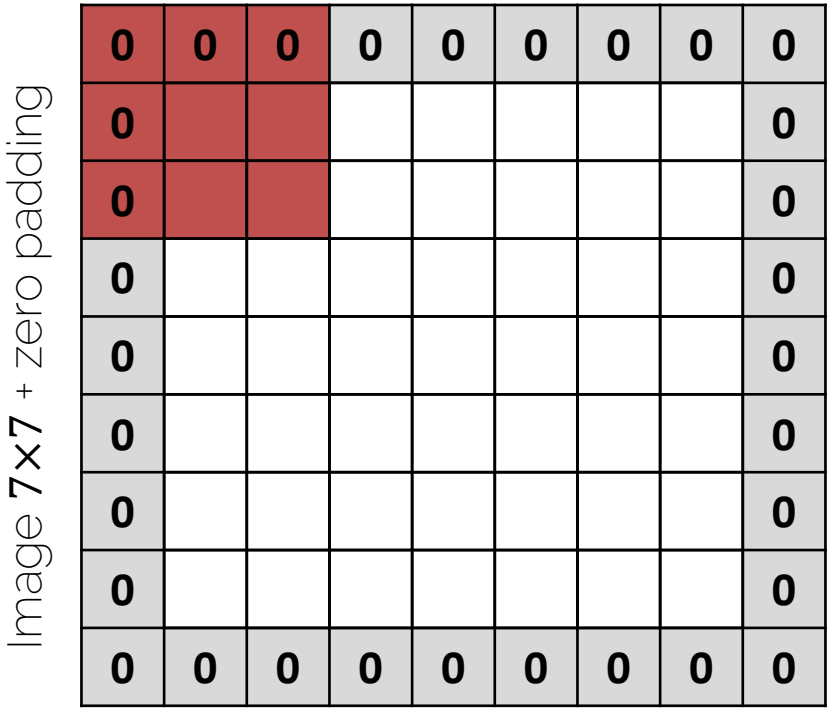
Why padding?

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

NB Adding padding to an image by a CNN allows for a more accurate analysis of images.

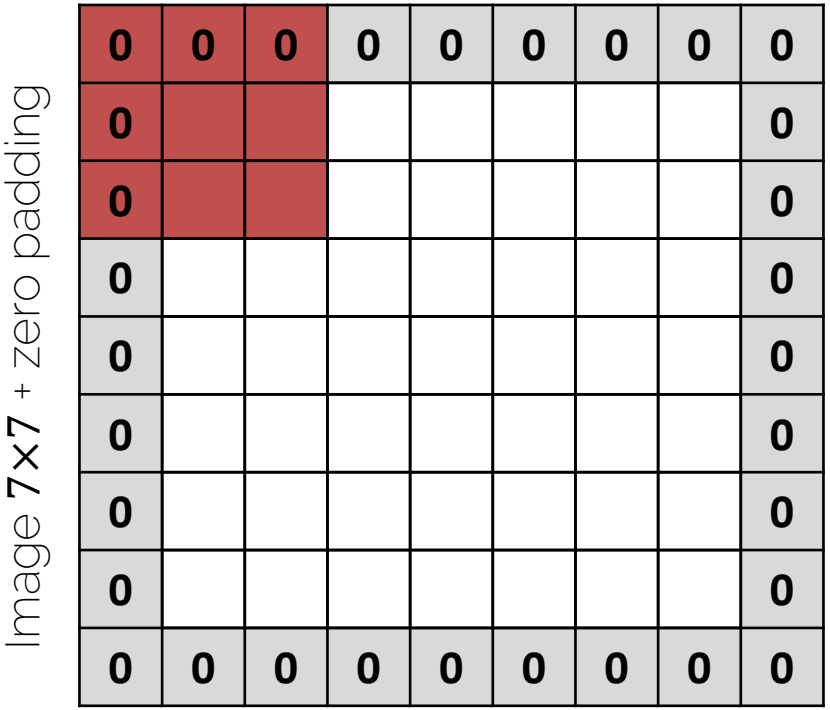
* In order to assist the Kernel with processing the image, padding is added to the frame of the image to allow for more space for the Kernel to cover the image.

Convolution Layers: Padding



Why padding?

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



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:

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

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

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 $P = \frac{F-1}{2}$

Example

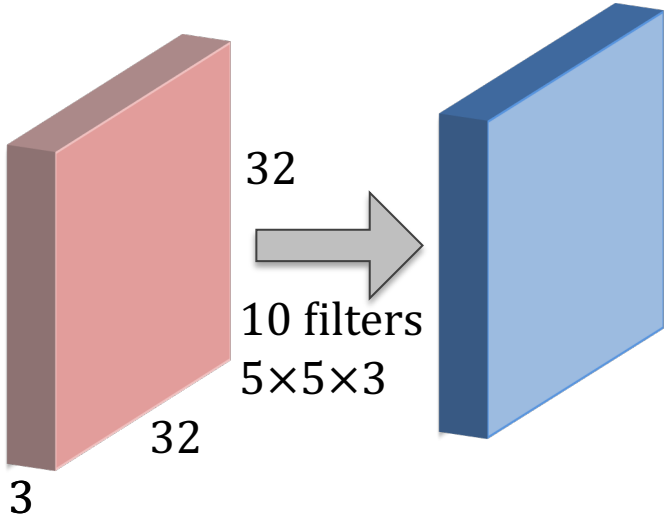
Input image: $32 \times 32 \times 3$

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

Dim Color-Coded

Example

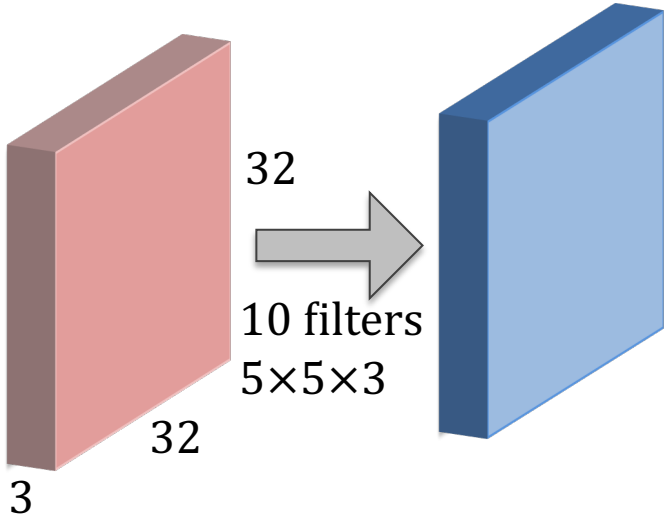
Input image: 32×32×3

10 filters 5×5

Stride 1

Pad 2

Spatial Size



Output size is:

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

i.e. 32×32×10

Remember

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

parameters - weights

Example

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

└ does not depend on input size, stride or padding

└ only depends on depth of input, # filters applied & filter spatial size (H, W)

* 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

↘ Usually forgotten !

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?

↓

directly linked with filter & filter size, not with input size

However depth of input is important

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?

NB order is important!

A1: (3, 4, 5, 5)

Input channels (RGB = **3**)

Output size = depth
4 filters

Filter size = **5×5**

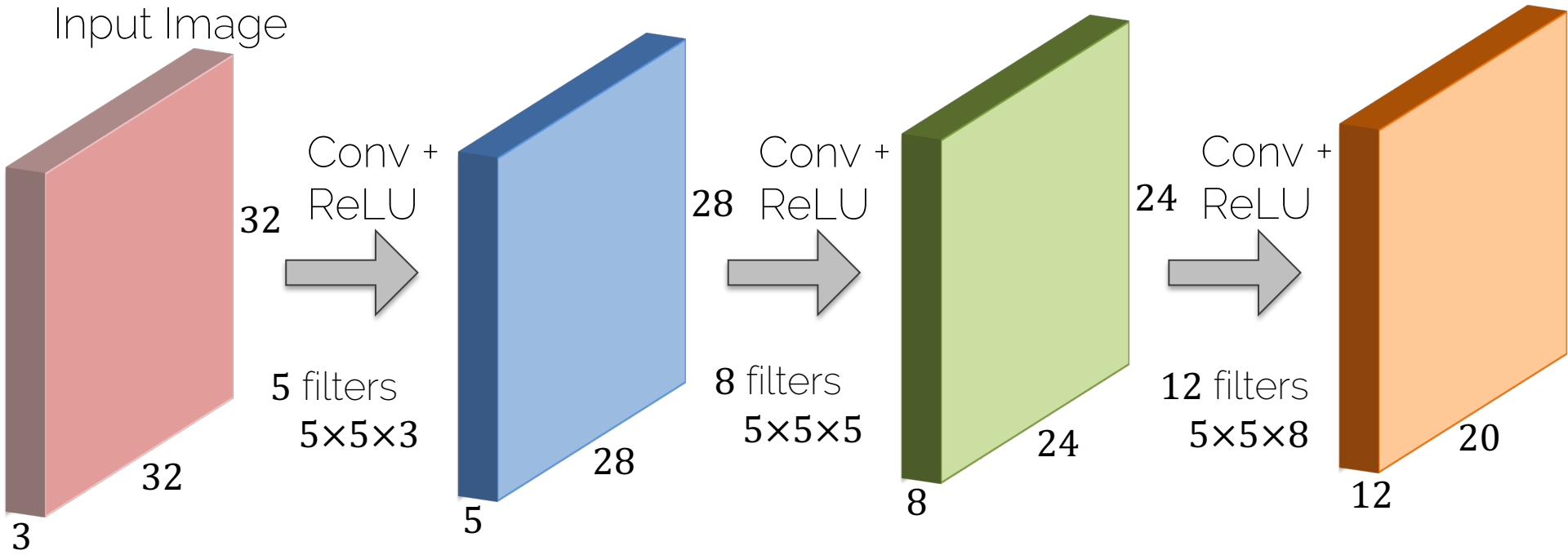
Convolutional Neural Network (CNN)

One adv

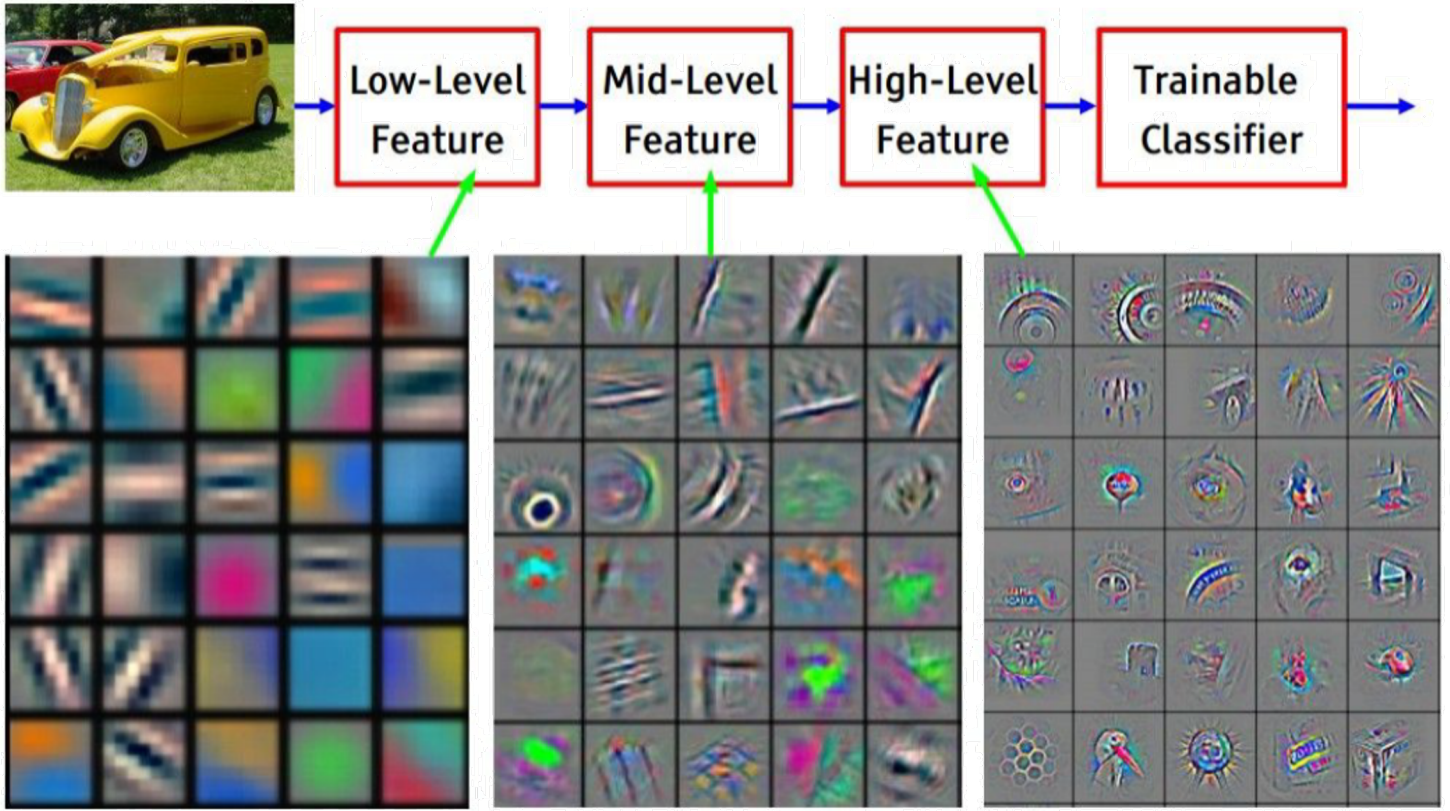
ability of feature map to reflect any affine transformations applied on input image.

CNN Prototype

ConvNet is concatenation of Conv Layers and activations



CNN Learned Filters



Still Something missing!

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

Pooling layers are used to reduce the dimensions of feature maps.

Thus it reduces the # parameters to learn & the amount of computation performed in the network.

Pooling

DownSampling

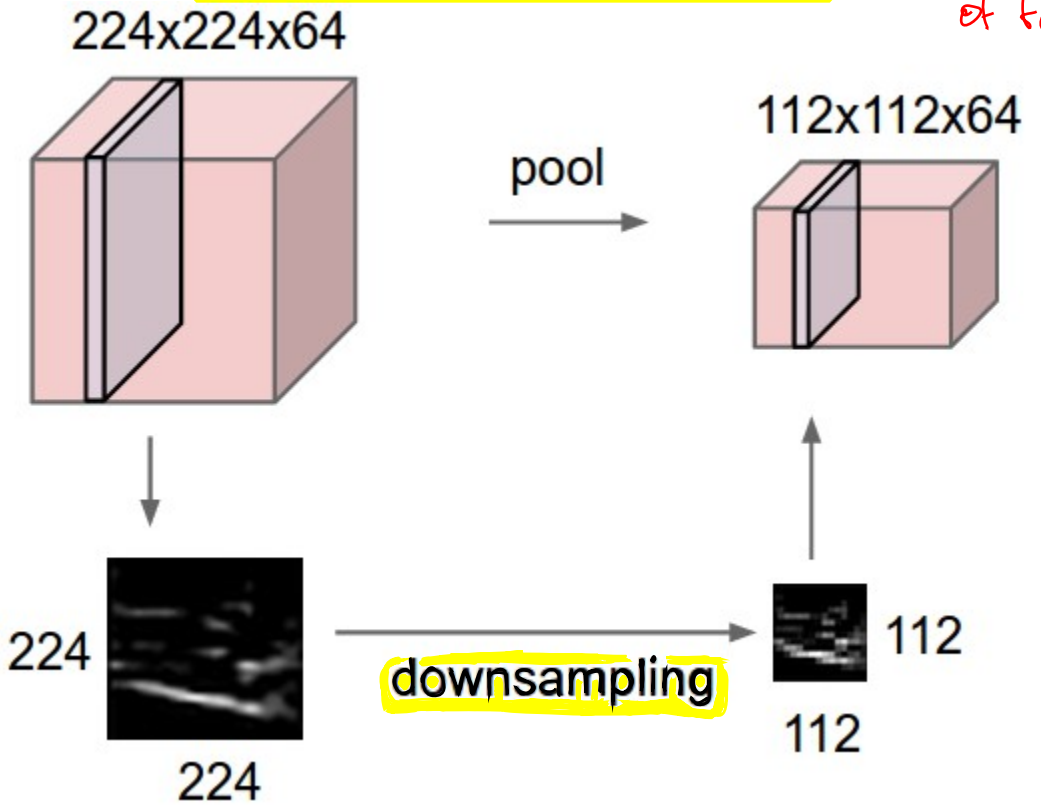
Mainly, used for
- dim reduction, thus computation
- Spatial invariance feature extraction
  - pos
  - rot

Pooling Layer

idea

reducing input size

deals with multiple scales of features with image

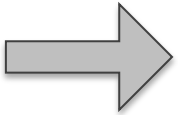


Max Pooling

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

NB – Pooling acts as a generalizer of the lower level data, and so, in away enables the network to move from high resoln data to lower resoln info.

Pooling coupled with conv filters attempts to detect objects with an image.

Important

- Conv Layer = 'Feature Extraction'
  - Computes a feature in a given region

- Pooling Layer = 'Feature Selection'
  - Picks the strongest activation in a region

* Pooling makes the detection of certain features somewhat invariant to scale & orientation changes. It generalizes over lower level, more complex info.

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

* Assuming fewer convolutional filters, they will learn to activate when they "see" the object in the image (i.e return a large output).
  - via ReLU
  - is contained in a small region in the image.

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

However they will activate more or less strongly depending what orientation the object has.

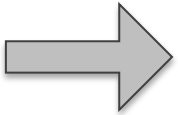
* Pooling comes to play; as it looks over the output of these filters & gives a high output as long as any one of these filters has a high activation.

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

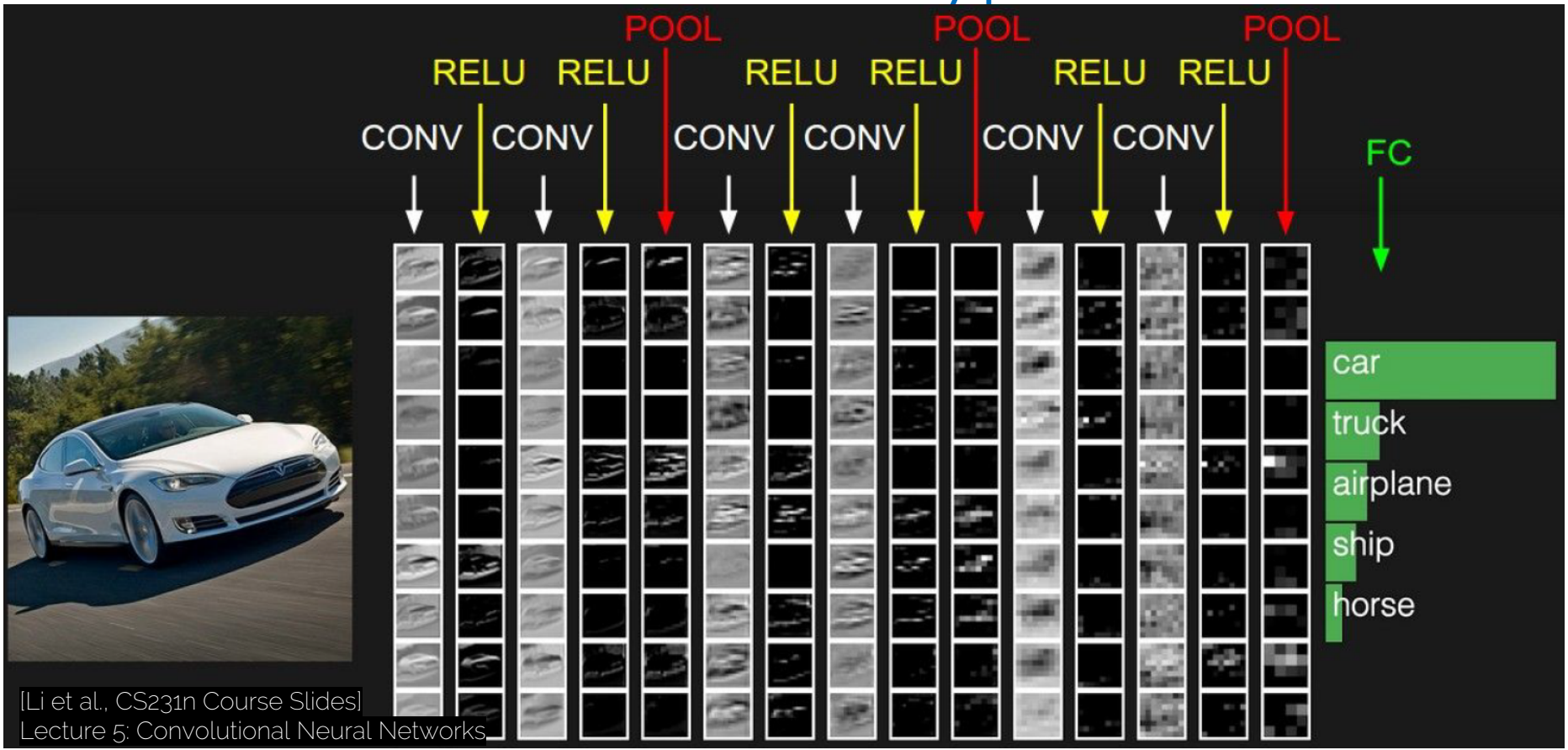


'Pooled' output

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

→ think of it as mapping

Convolutions vs Fully-Connected

- In contrast to fully-connected layers, we want to restrict the degrees of freedom → allowing learning higher rep without exploding # parameters
  - FC is somewhat brute force
  - Convolutions are structured for images

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

[Spatial invariant]

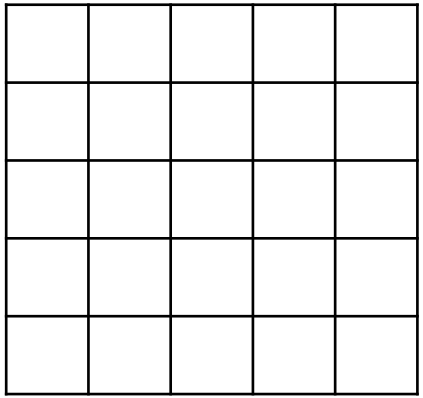
* A powerful tool to distill info from image.

Receptive field

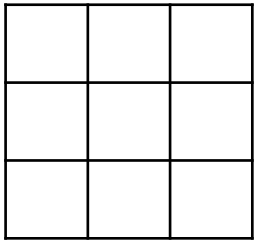
The region of the input space that a particular feature is looking at (i.e affected by)

* Each neuron of hidden layer is only seeing a patch of original input image

- Spatial extent of the connectivity of a convolutional filter

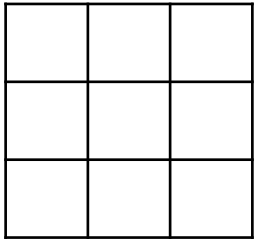


5x5 input



3x3 filter

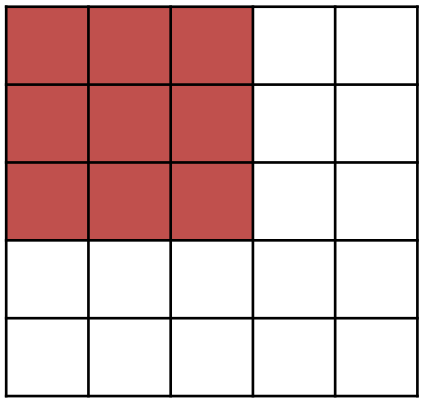
=



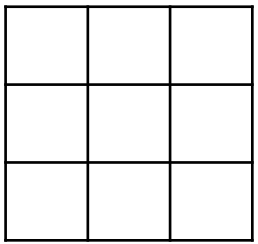
3x3 output

→ how many input pixels are used to compute a particular output pixel

we say a 1 pixel in output has a RF of 3x3 pixels in input

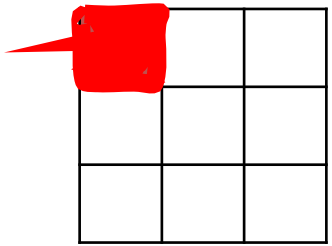


5x5 input



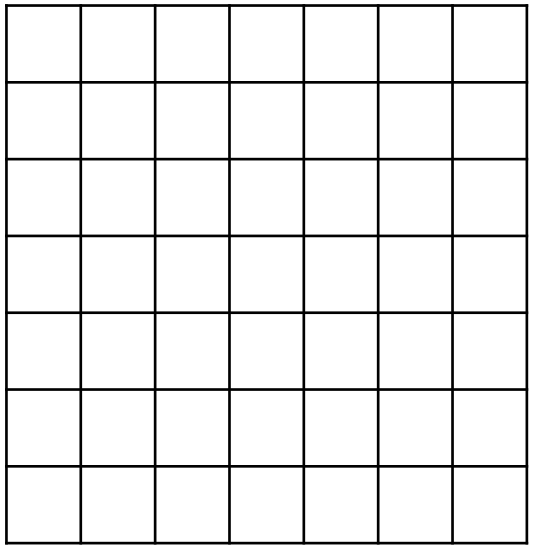
3x3 filter

=

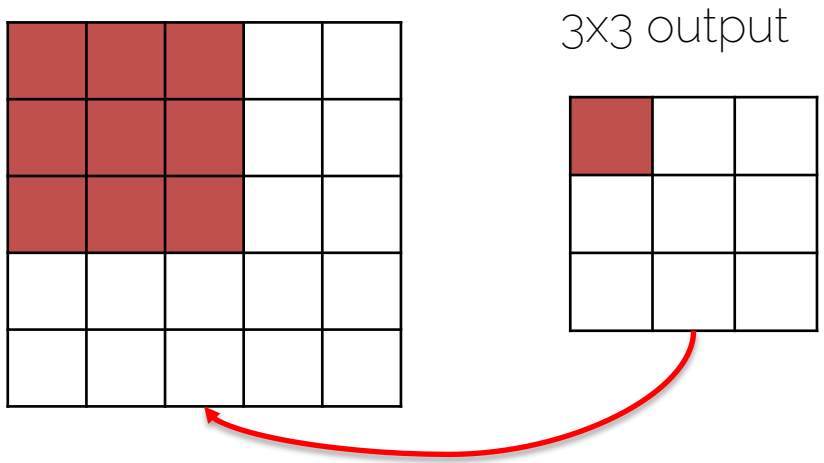


3x3 output

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

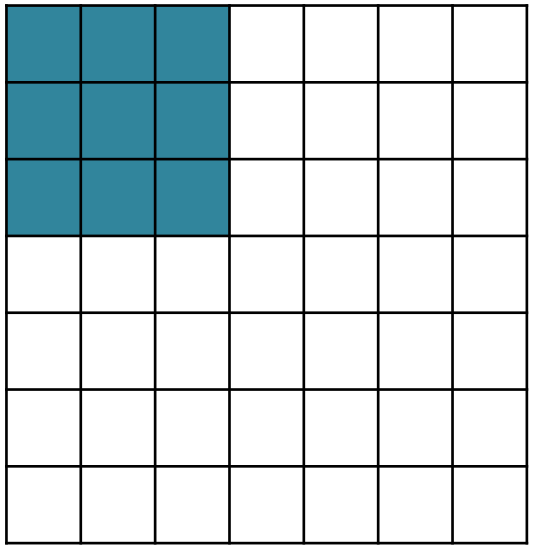


7x7 input

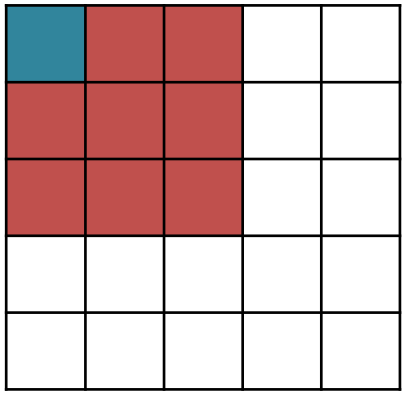


3x3 receptive field = 1 output pixel is connected to 9 input pixels

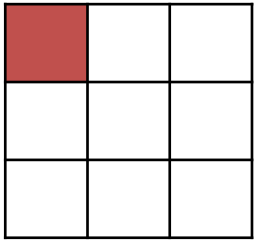
going one step back



7x7 input



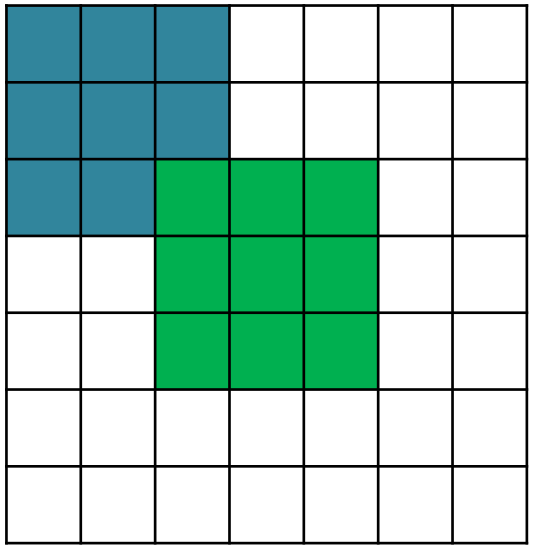
3x3 output



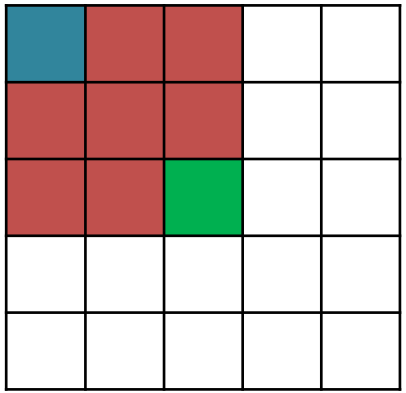
3x3 receptive field = 1 output pixel is connected to 9 input pixels

RF of 1 pixel in the 3x3 will be a 5x5 RF of the 7x7.

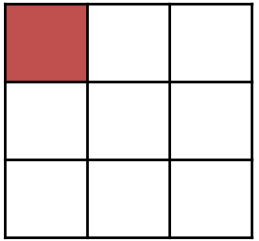
i.e 25 values are used to compute 1 value



7x7 input

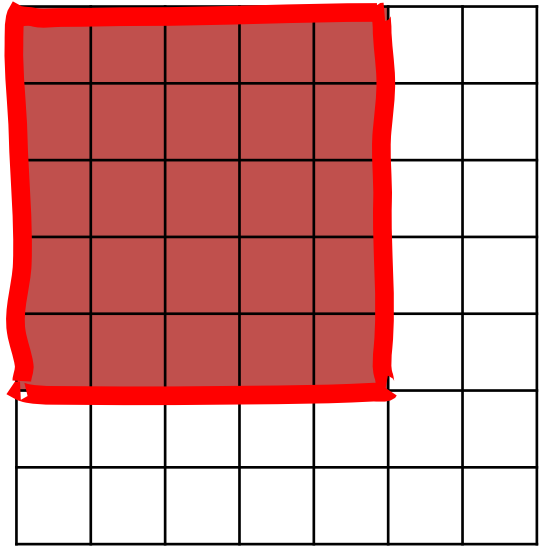


3x3 output

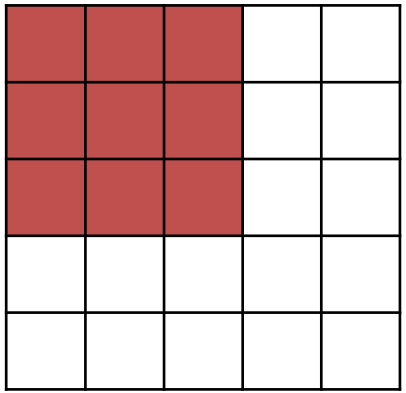


3x3 receptive field = 1 output pixel is connected to 9 input pixels

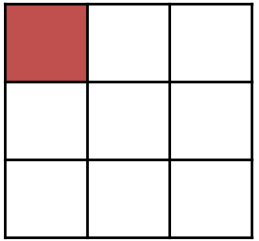
Q: how many pixels in the original 7x7 input were used to actually create this 1 pixel in the 3x3 output?



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/