University of Lancashire

Lecture 13

Convolutional Neural Network

Course overview

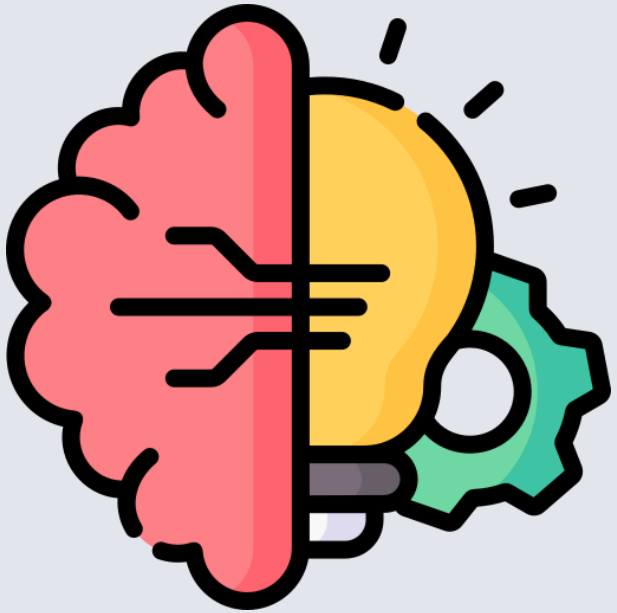
- **Learning Objectives**
  - Introduce and familiarise you with the ConvNet
  - Learning how CNN works considering strides
  - How to construct a complete CNN as a end-to-end model

- **Important Directions**

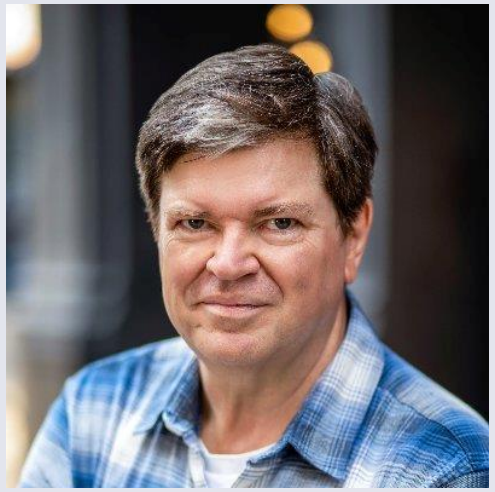


Today's Contents

- Convolutional Neural Network
- Building blocks of CNN
- Convolutional Operation
- Constructing CNN



Brief about CNN



Yann LeCun, Professor of Computer Science
The Courant Institute of Mathematical Sciences,
New York University
Room 1220, 715 Broadway, New York, NY 10003, USA.
(212)998-3283

Email: yann@cs.nyu.edu

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

Brief about CNN

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

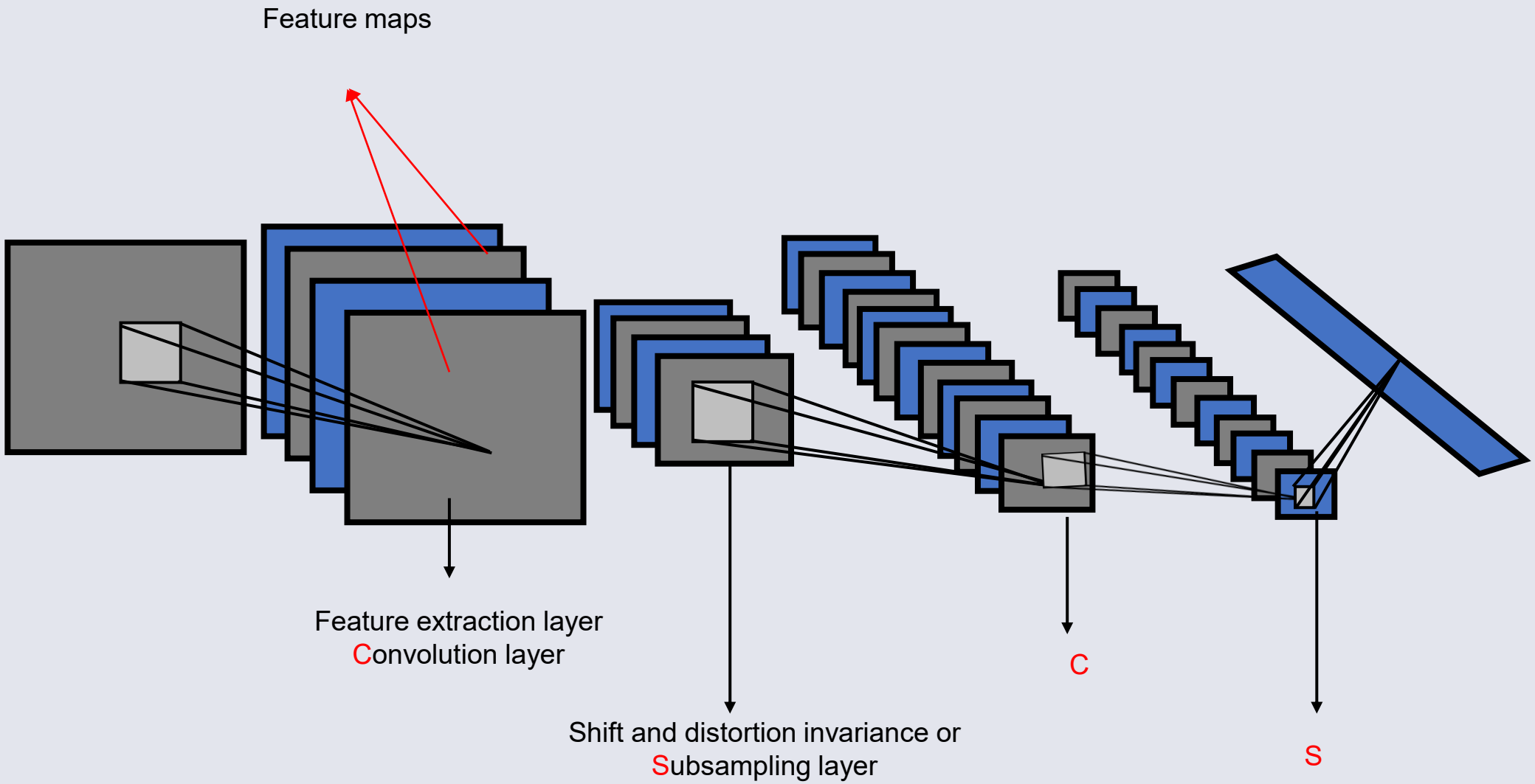
CNN is a feed-forward network that can extract topological properties from an image.

Like almost every other neural networks they are trained with a version of the back-propagation algorithm.

Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing.

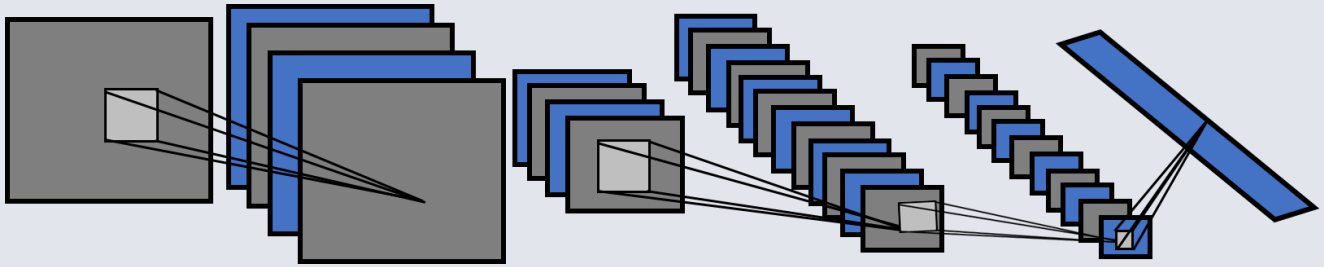
They can recognize patterns with extreme variability (such as handwritten characters).

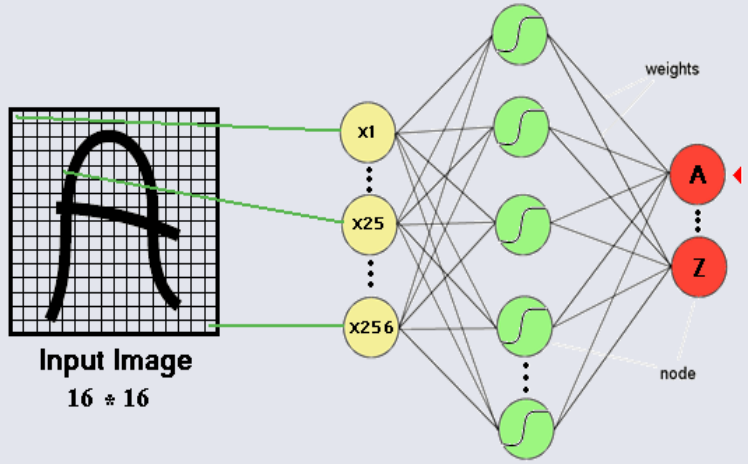
CNN Building Blocks

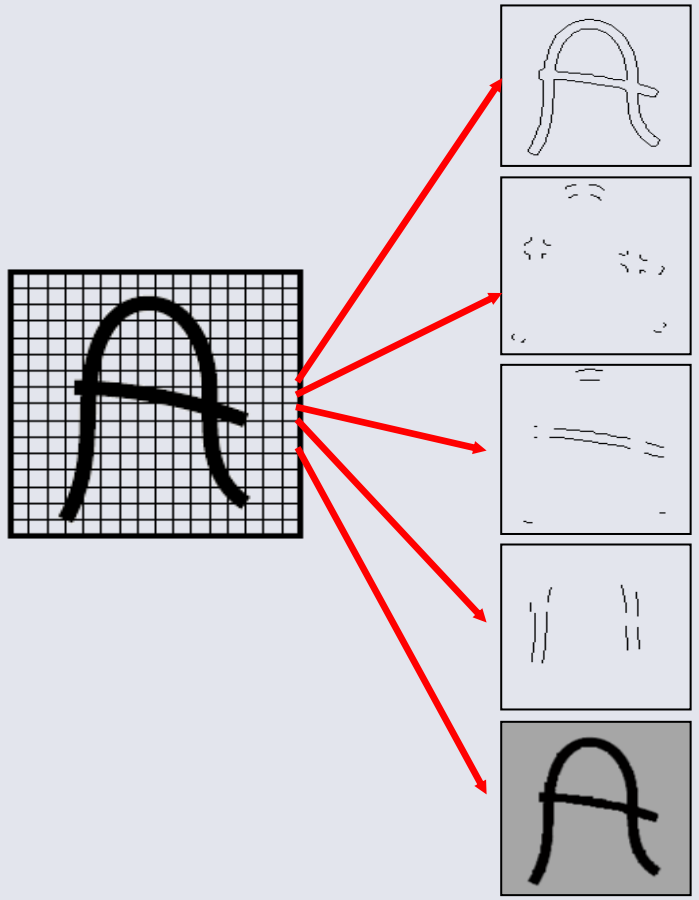


Convolutional Layer

Detect the same feature at different positions in the input image.



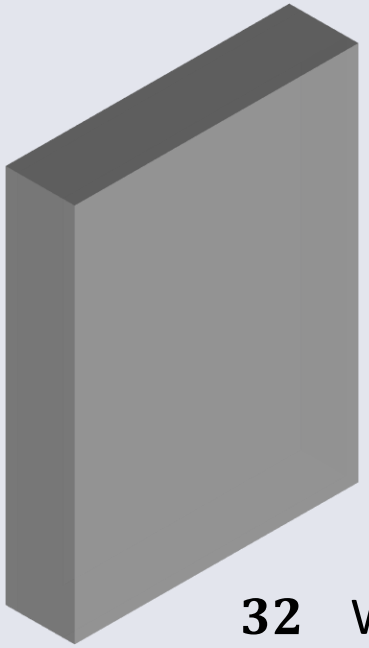




Features

Convolutional Layer

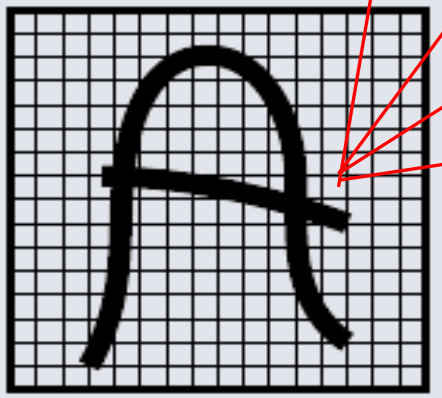
$32 \times 32 \times 3$ Image



32 Height

32 Width

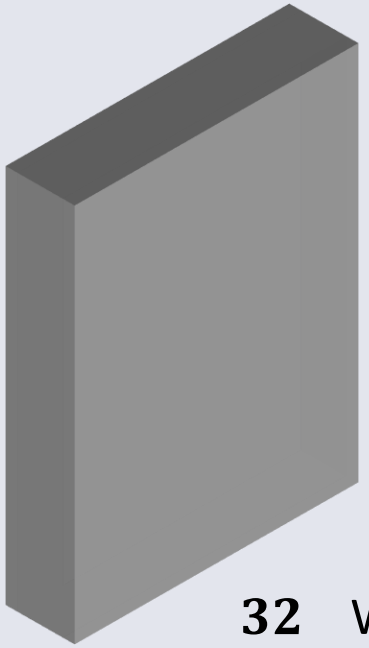
3 Depth



22	562	72	67	56	25	79	95	78	41
489	23	52	54	717	47	89	101	123	74
56	45	41	89	42	758	65	45	214	45
456	414	252	414	64	42	471	32	584	74
89	74	452	75	45	737	95	452	75	452
56	45	85	54	87	120	12	85	54	85
894	74	788	34	012	37	35	788	34	788
101	45	65	78	89	32	91	51	46	56

Convolutional Layer

$32 \times 32 \times 3$ Image



32 Height

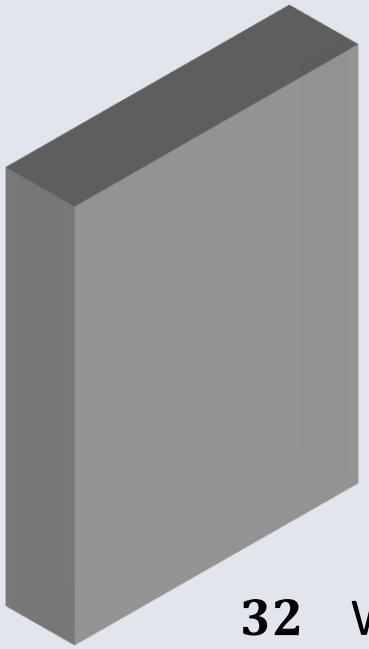
32 Width

3 Depth

Convolutional Layer

Filters always extend the full depth of the input volume

$32 \times 32 \times 3$ Image

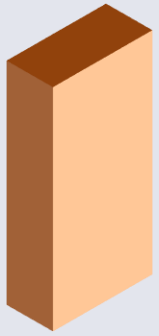


32 Height

32 Width

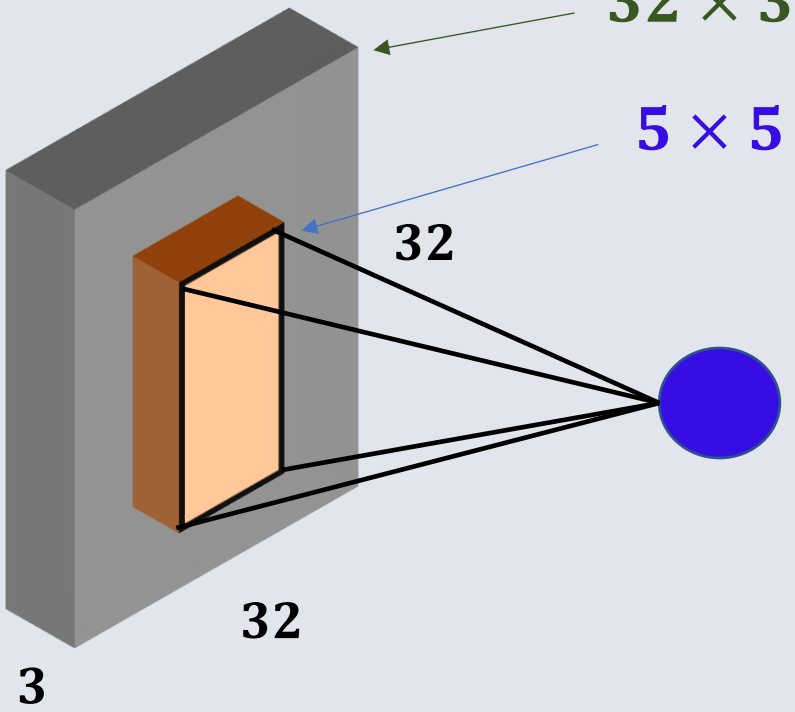
3 Depth

$5 \times 5 \times 3$ Filter



Convolve the filter with the image
i.e. “Slide over the image spatially, computing dot products”

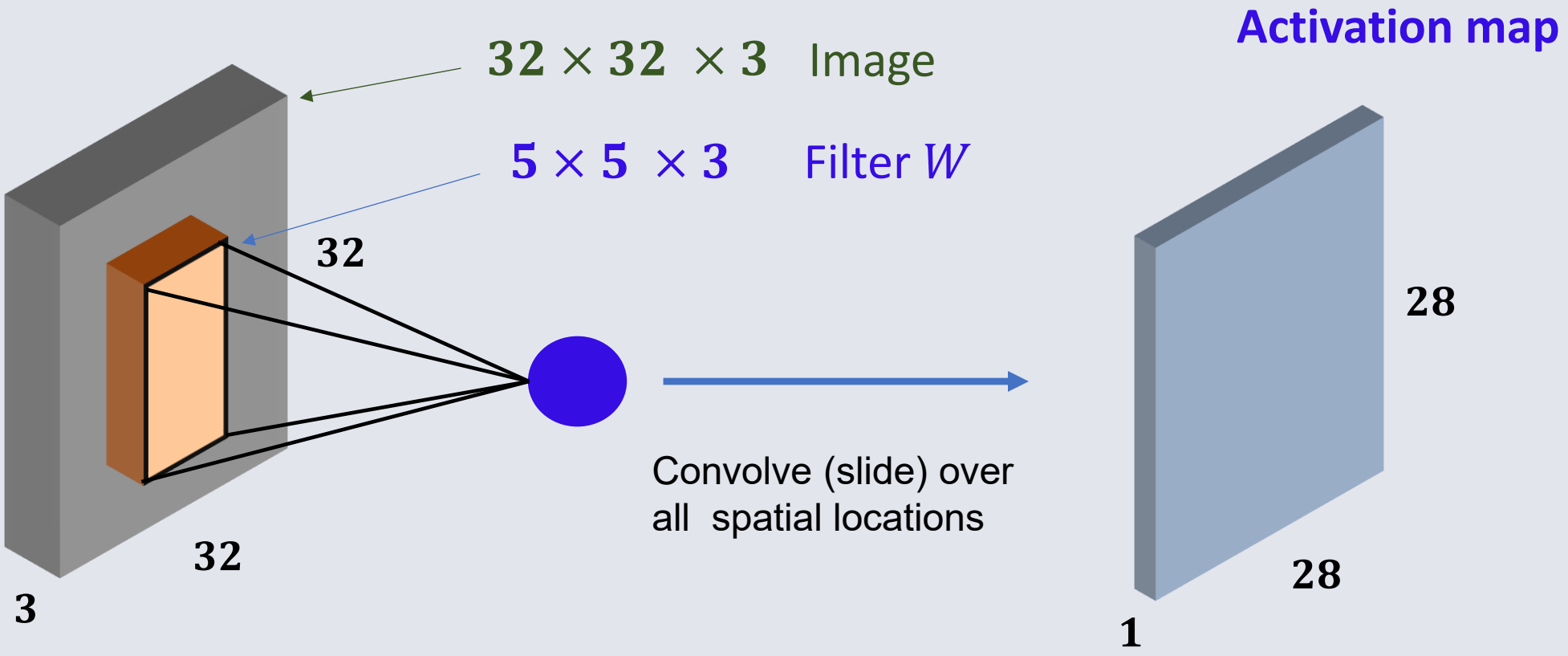
Convolutional Layer



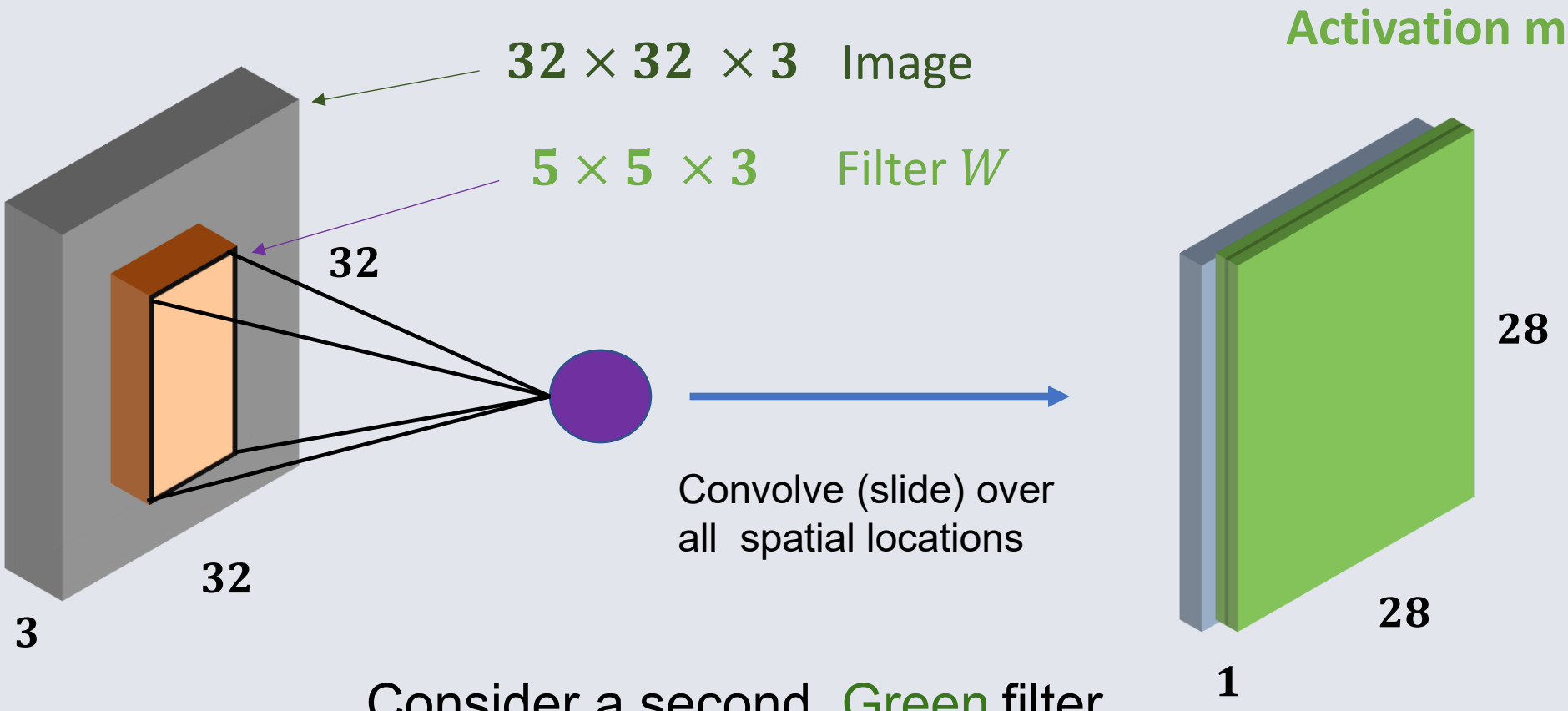
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. $5 \times 5 \times 3 = 75$-dimensional dot product + bias)

$$w^T x + b$$

Convolutional Layer



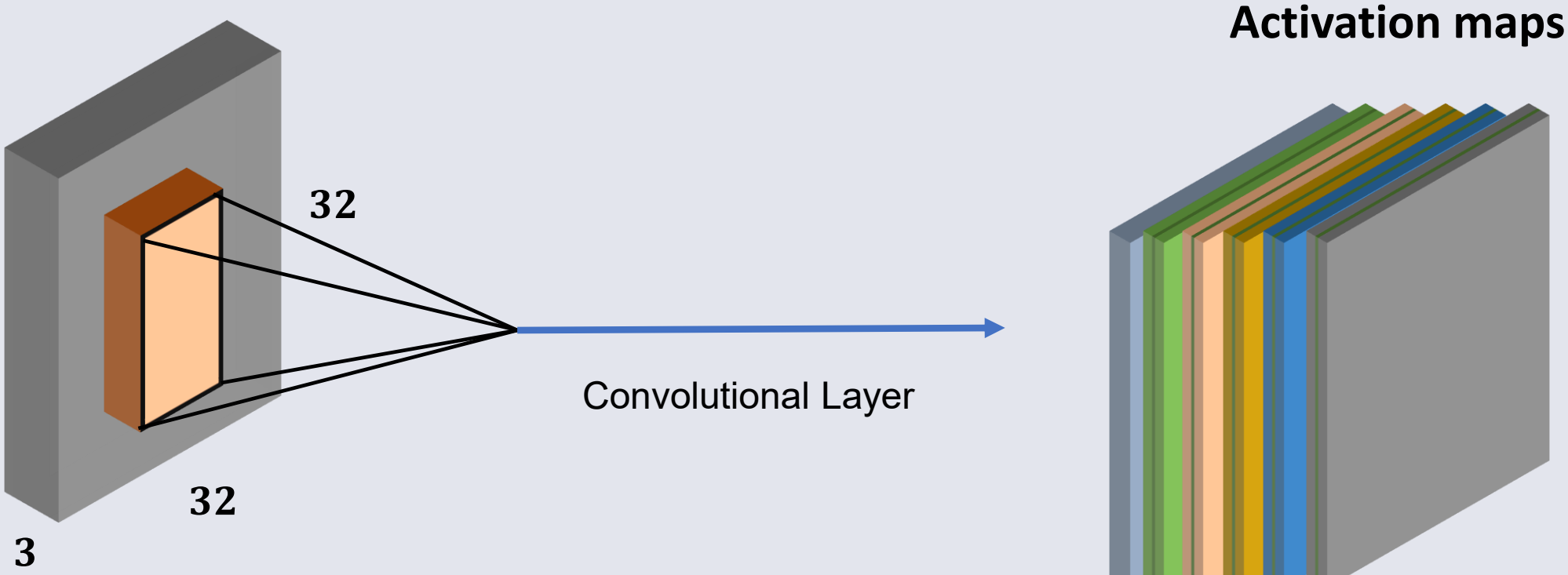
Convolutional Layer



Consider a second, Green filter

Convolutional Layer

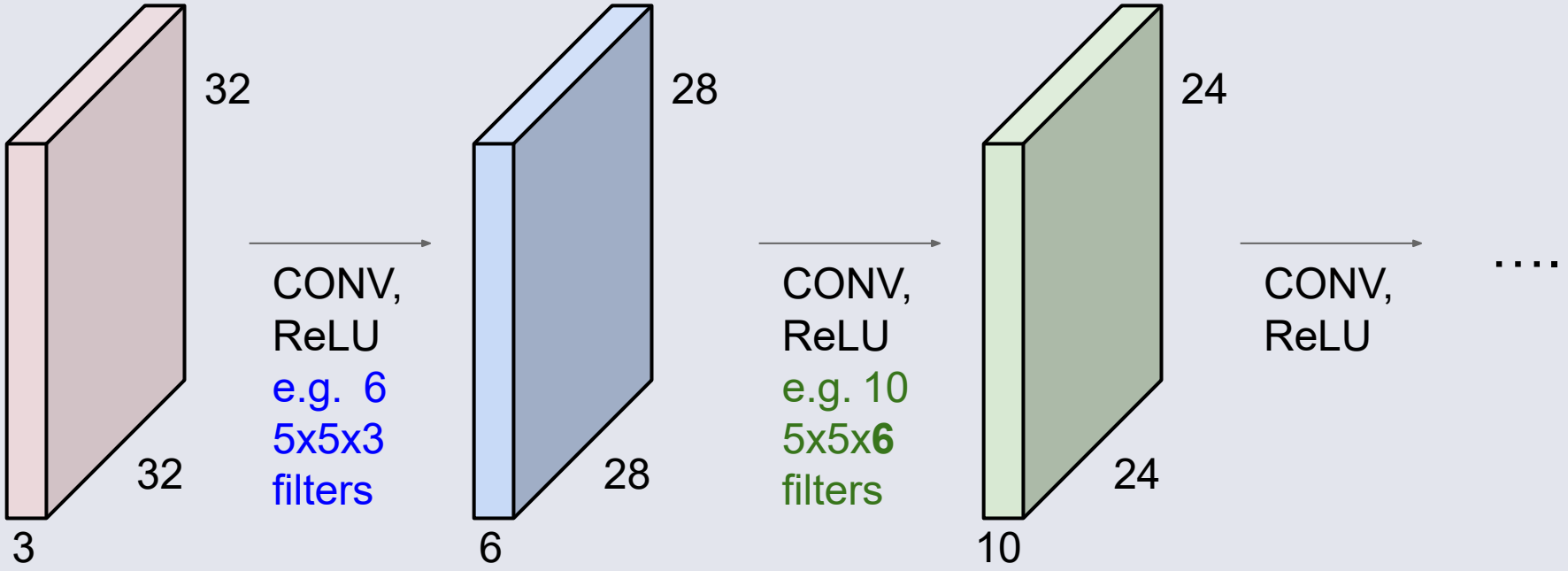
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps



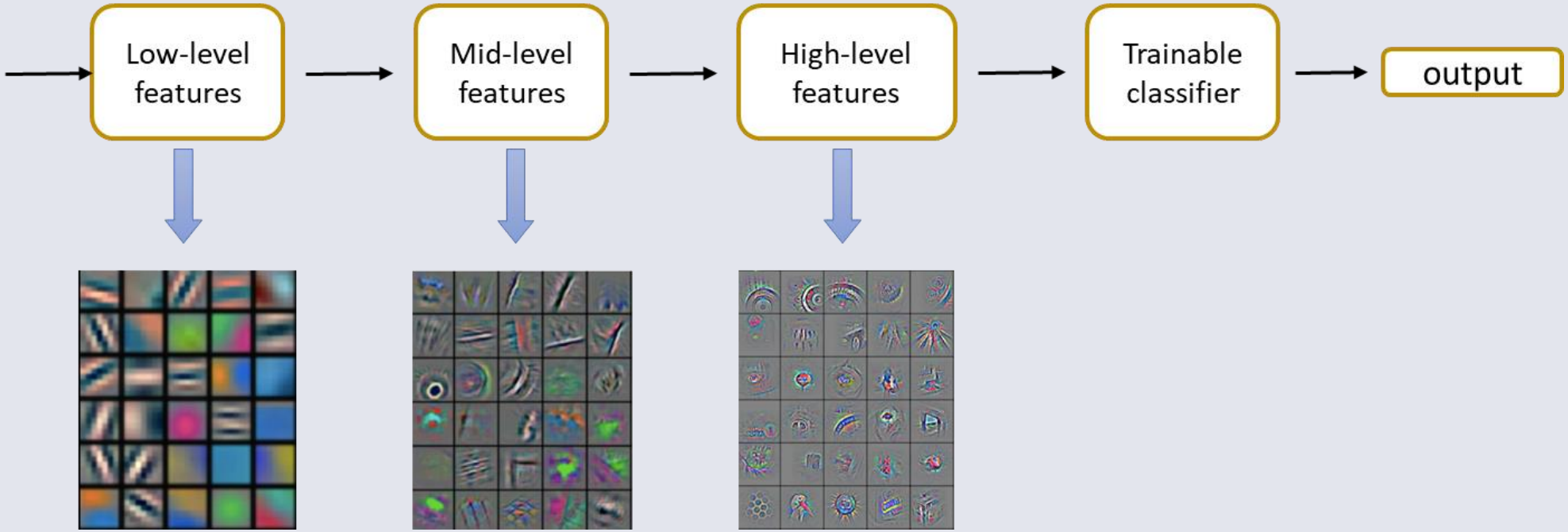
We stack these up to get 'new image ' of size of $28 \times 28 \times 6$

Convolutional Layer

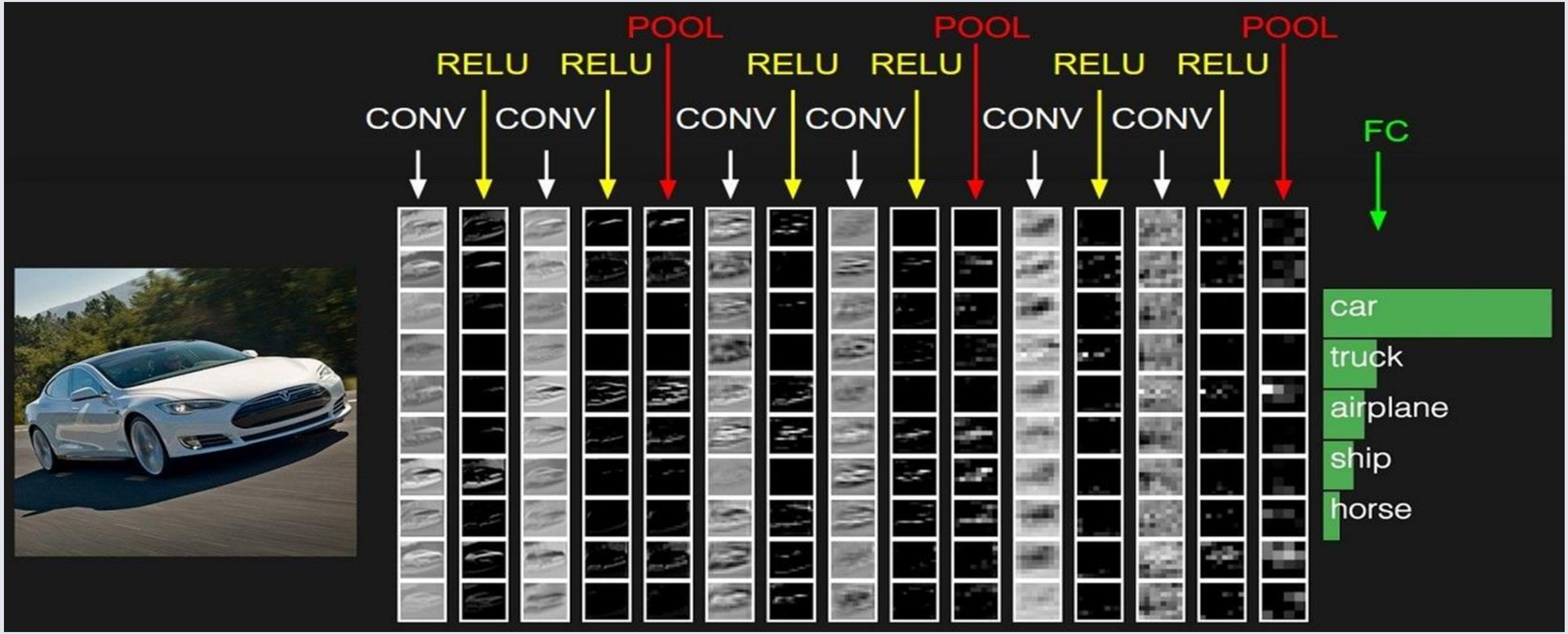
CNN/ConvNet is a sequence of Convolution Layers, interspersed with activation functions



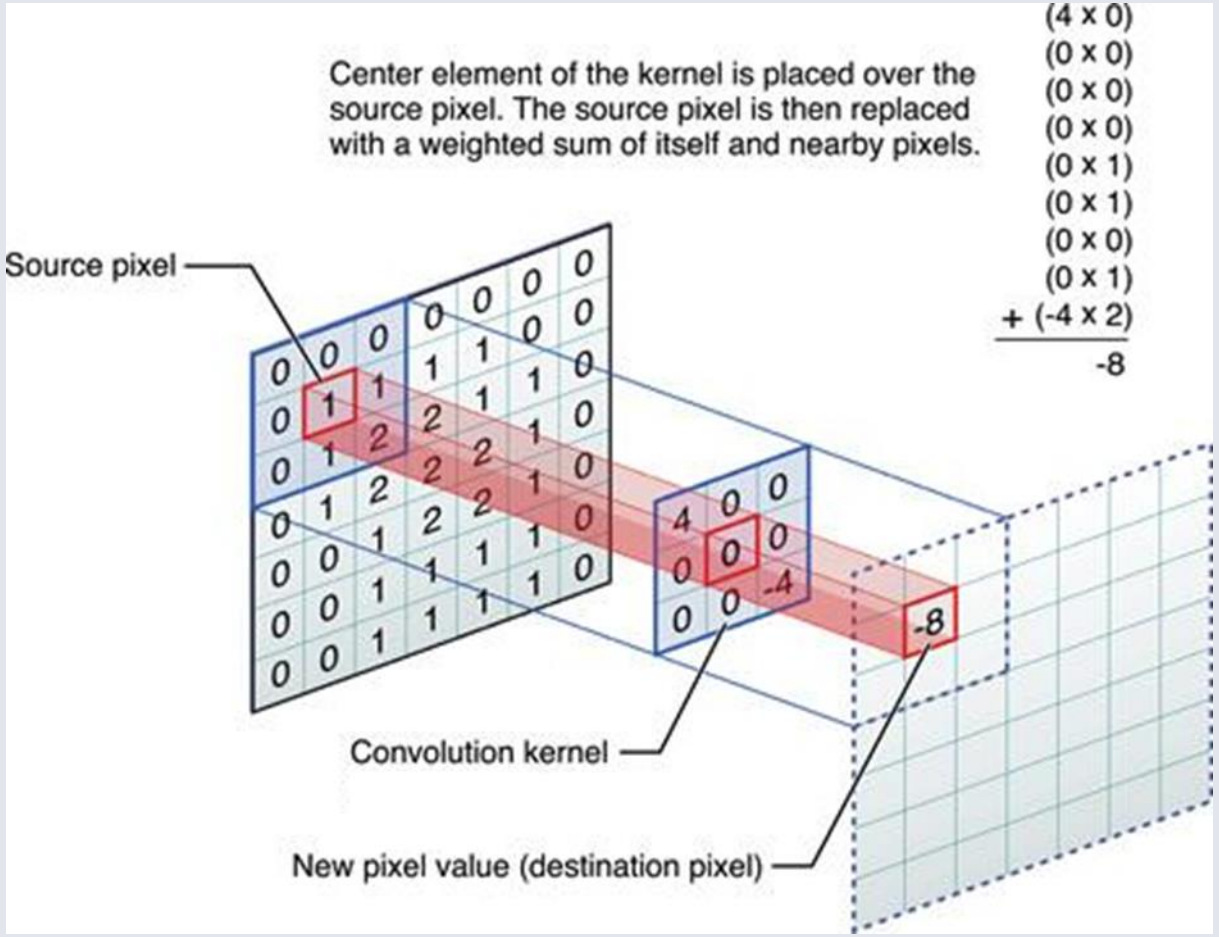
Convolutional Layer



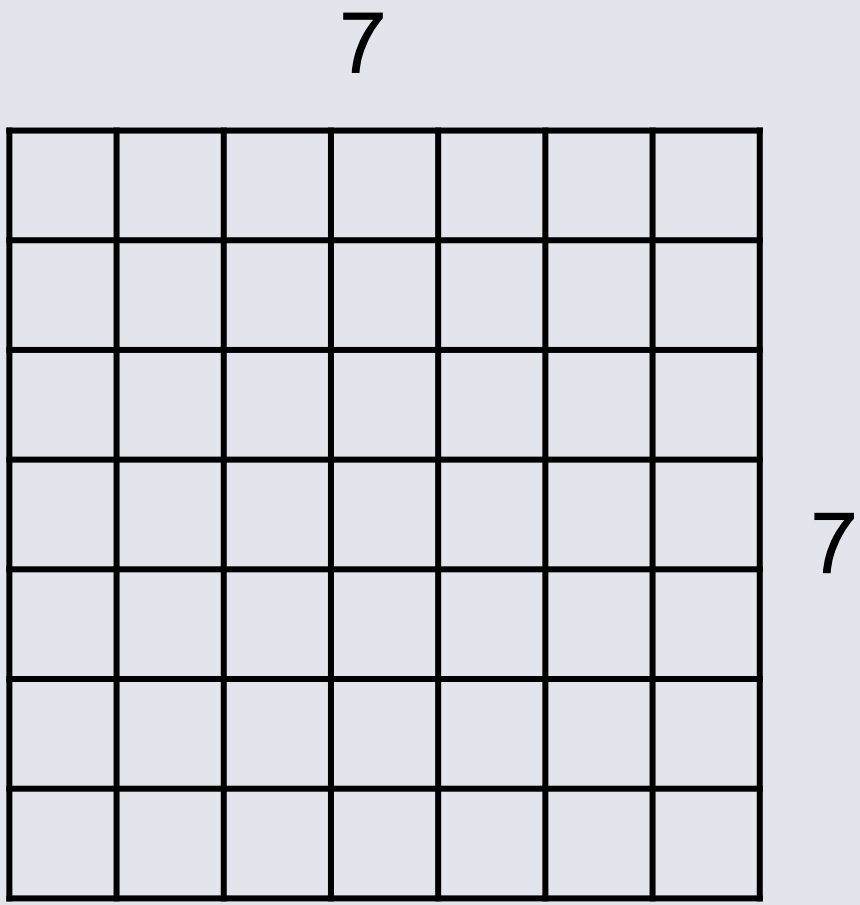
Convolutional Layer



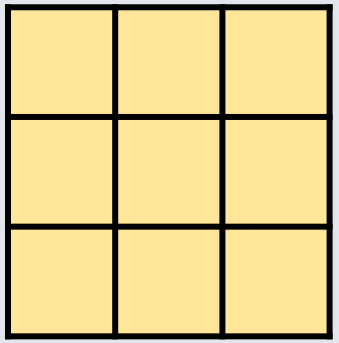
Closer Look at the Spatial Dimensions



Closer Look at the Spatial Dimensions

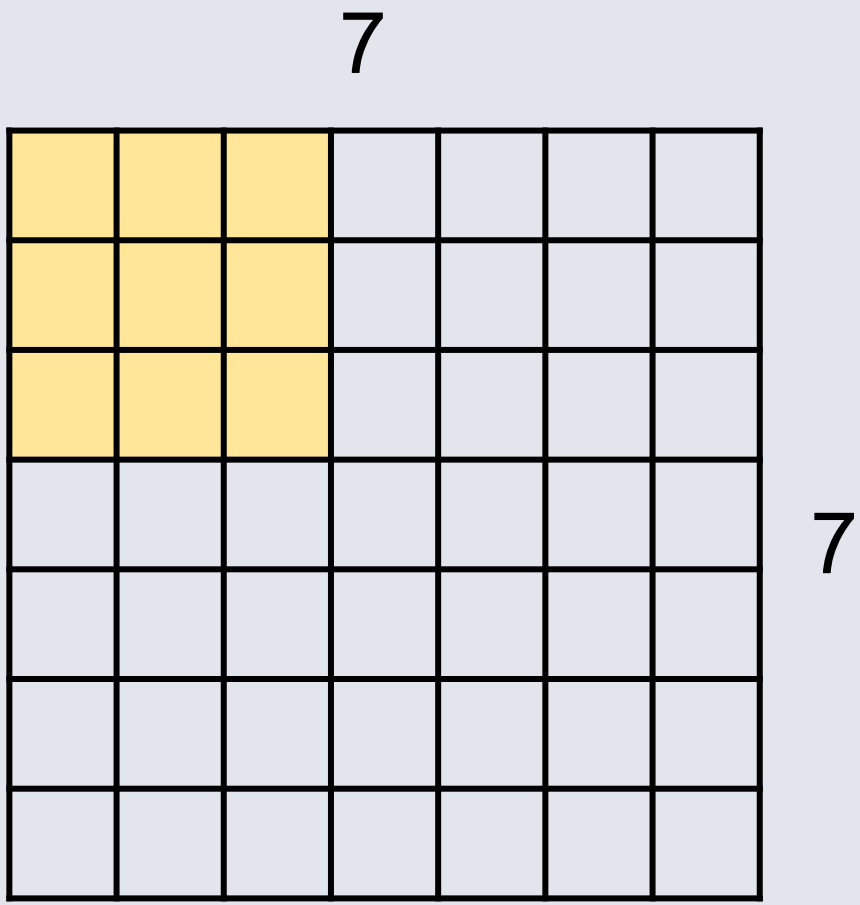


7×7 input (Spatially)

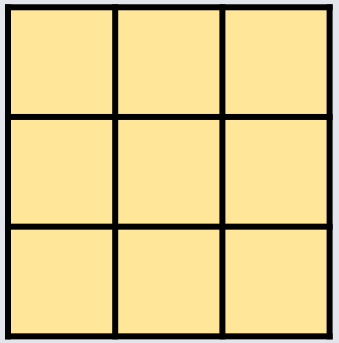


Assume 3×3 filter

Closer Look at the Spatial Dimensions

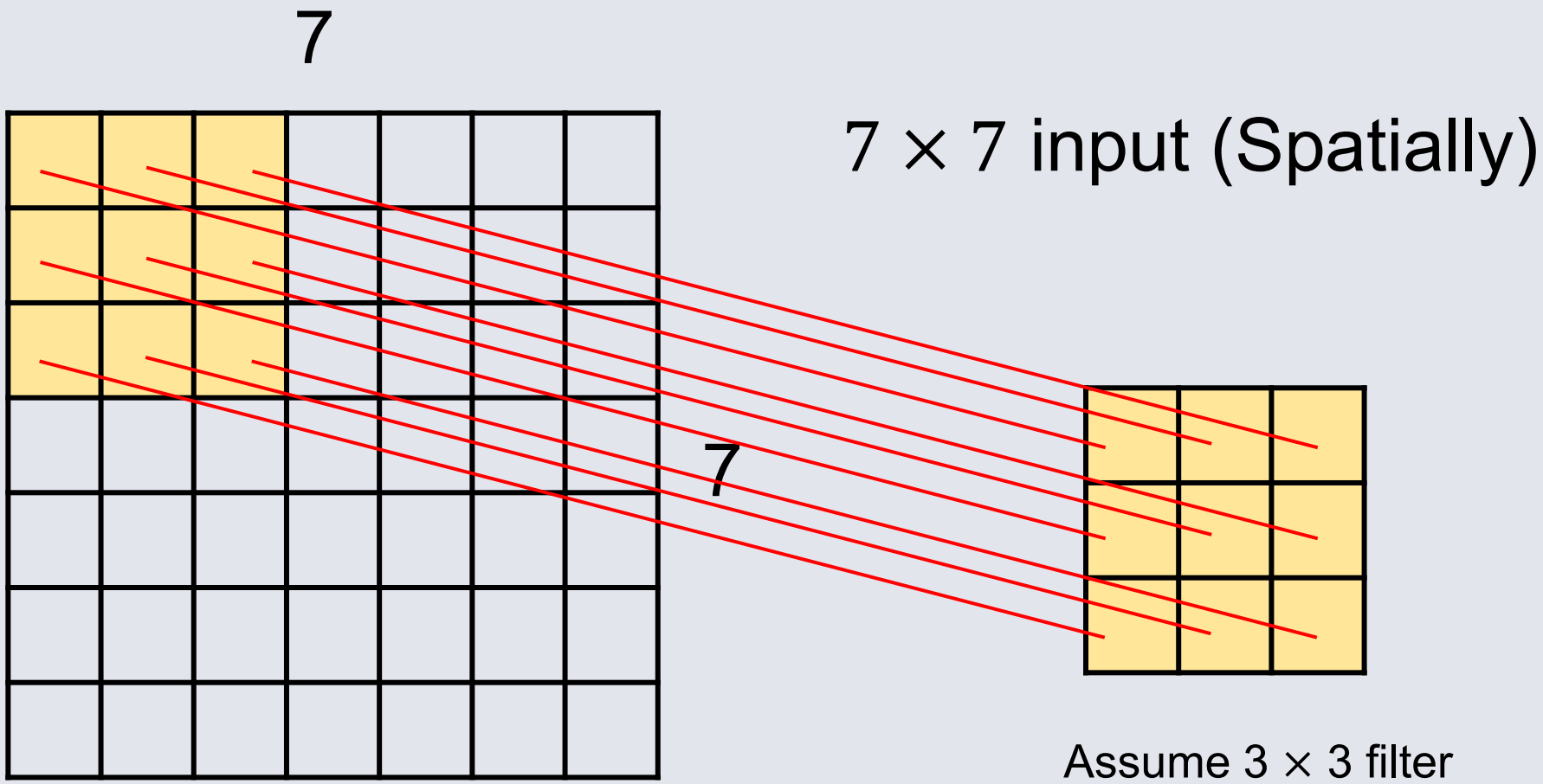


7×7 input (Spatially)

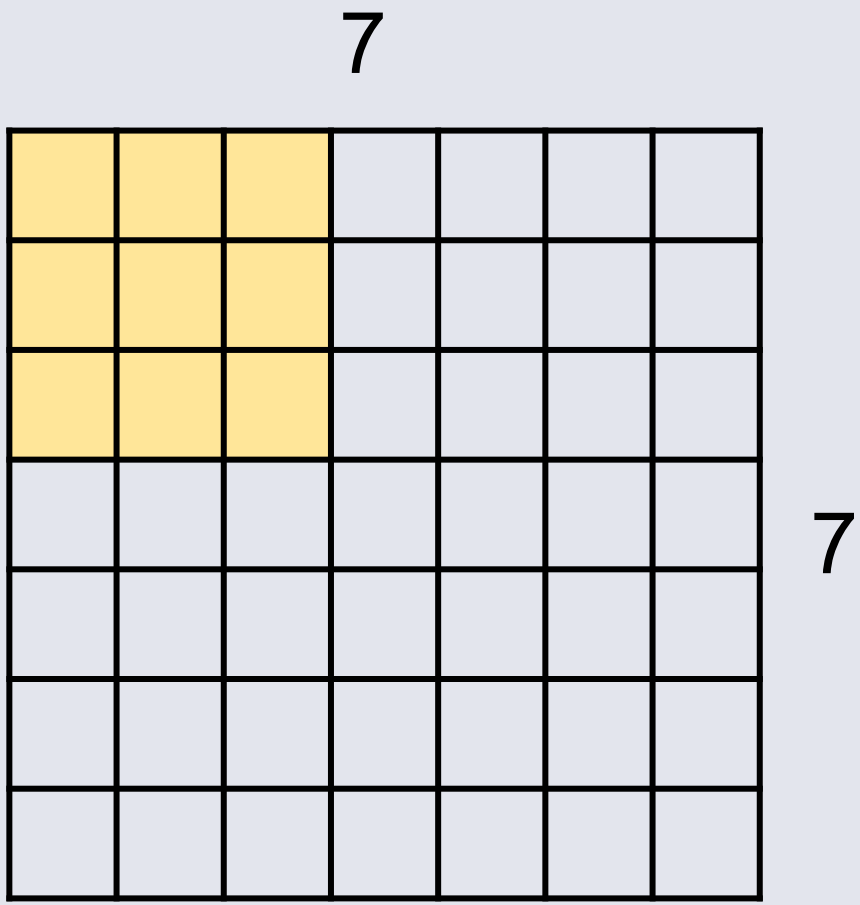


Assume 3×3 filter

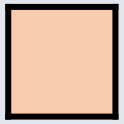
Closer Look at the Spatial Dimensions



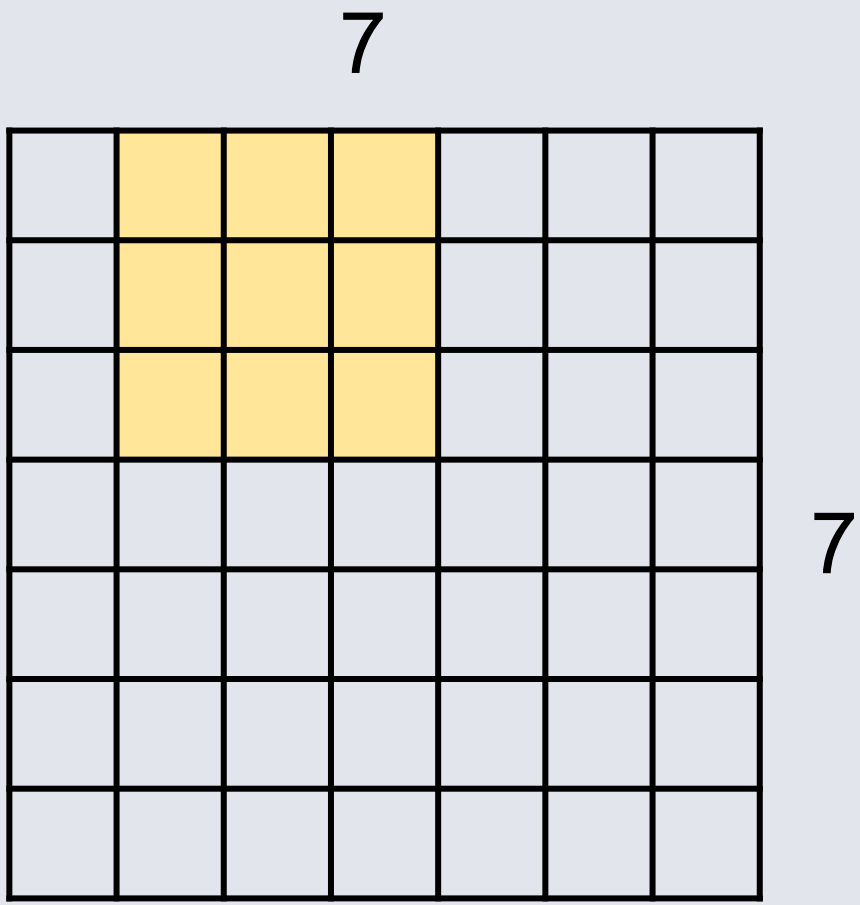
Closer Look at the Spatial Dimensions



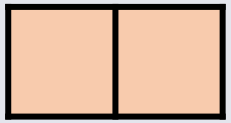
7×7 input (Spatially)



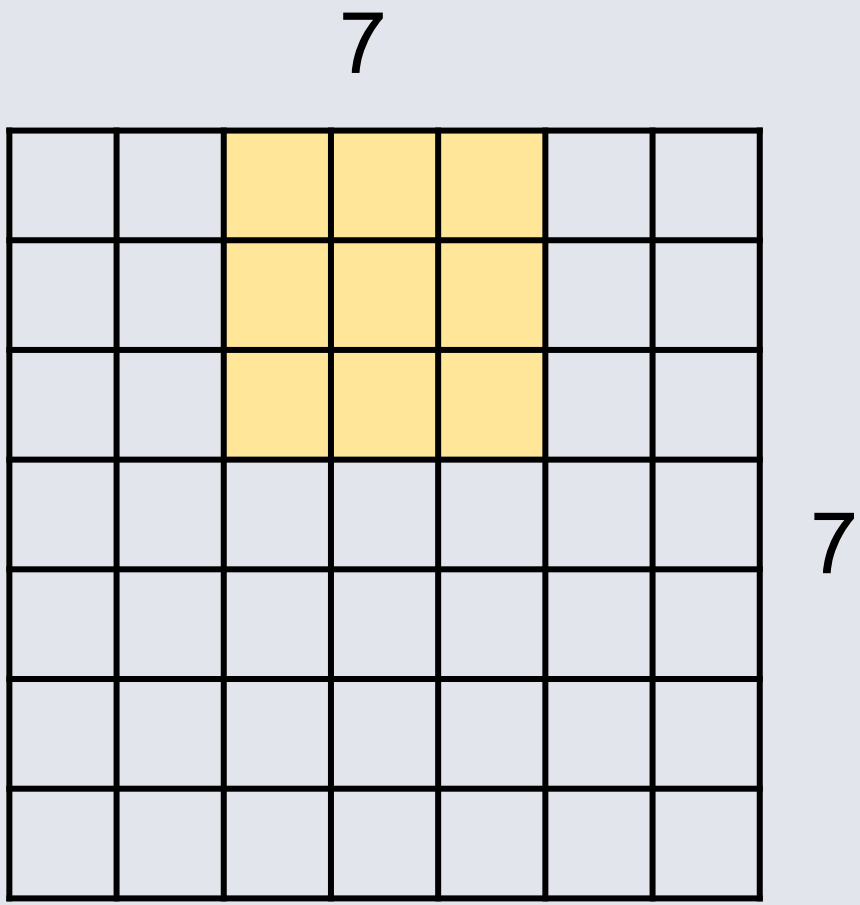
Closer Look at the Spatial Dimensions



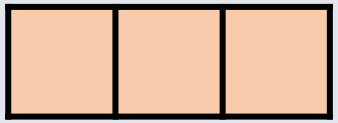
7×7 input (Spatially)



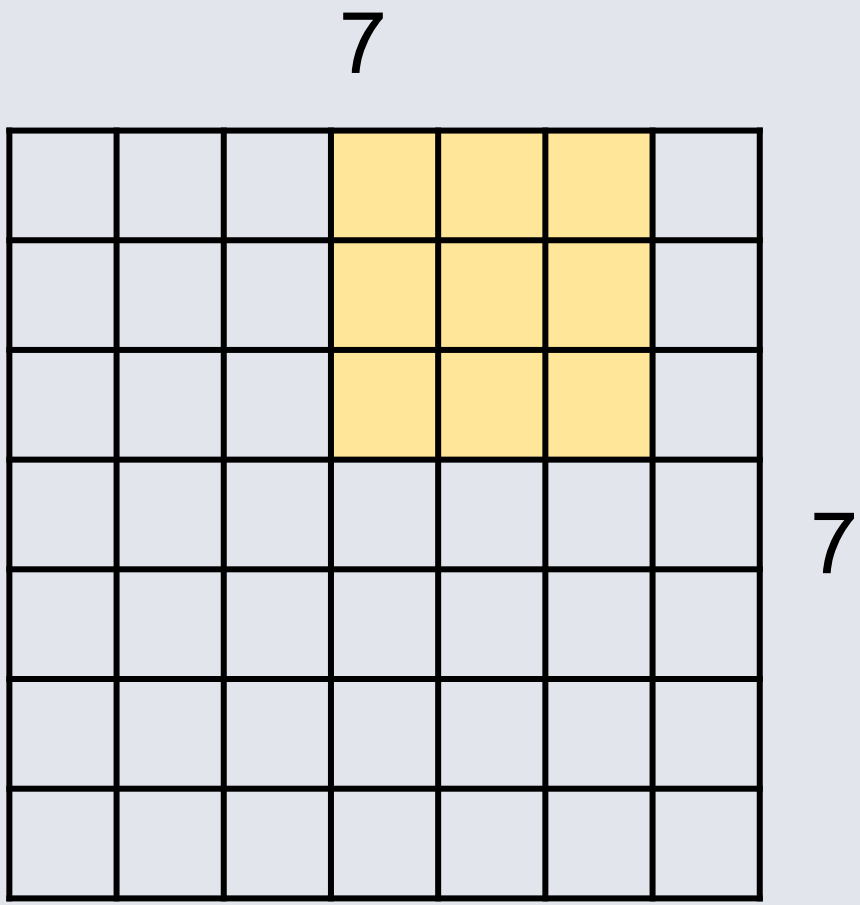
Closer Look at the Spatial Dimensions



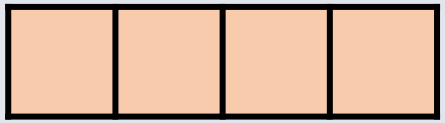
7×7 input (Spatially)



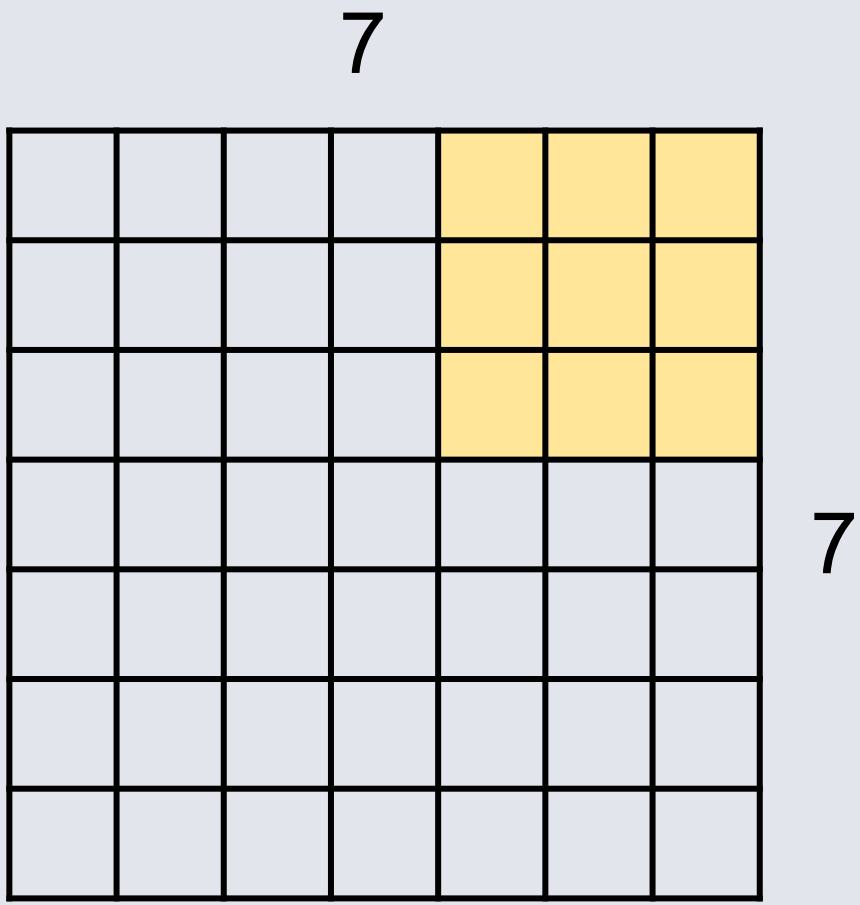
Closer Look at the Spatial Dimensions



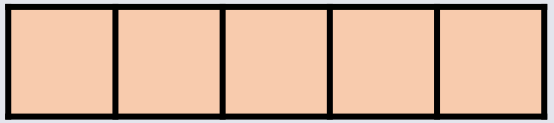
7×7 input (Spatially)



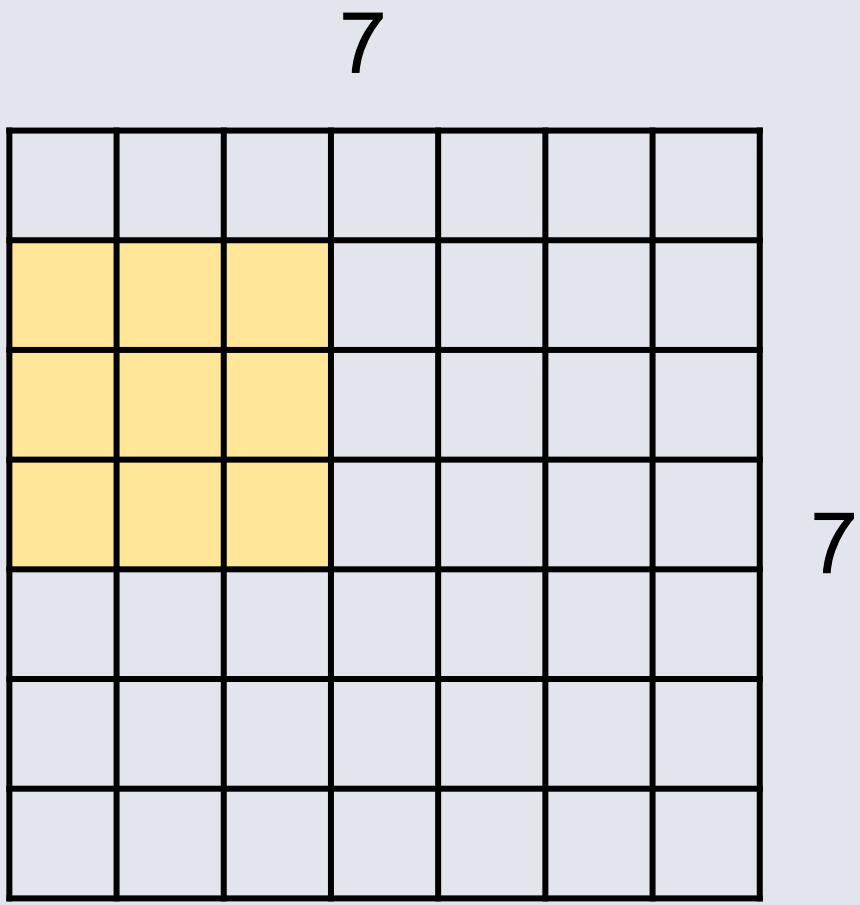
Closer Look at the Spatial Dimensions



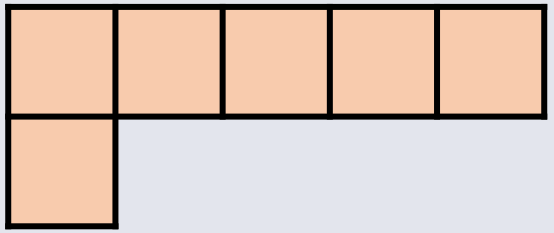
7×7 input (Spatially)



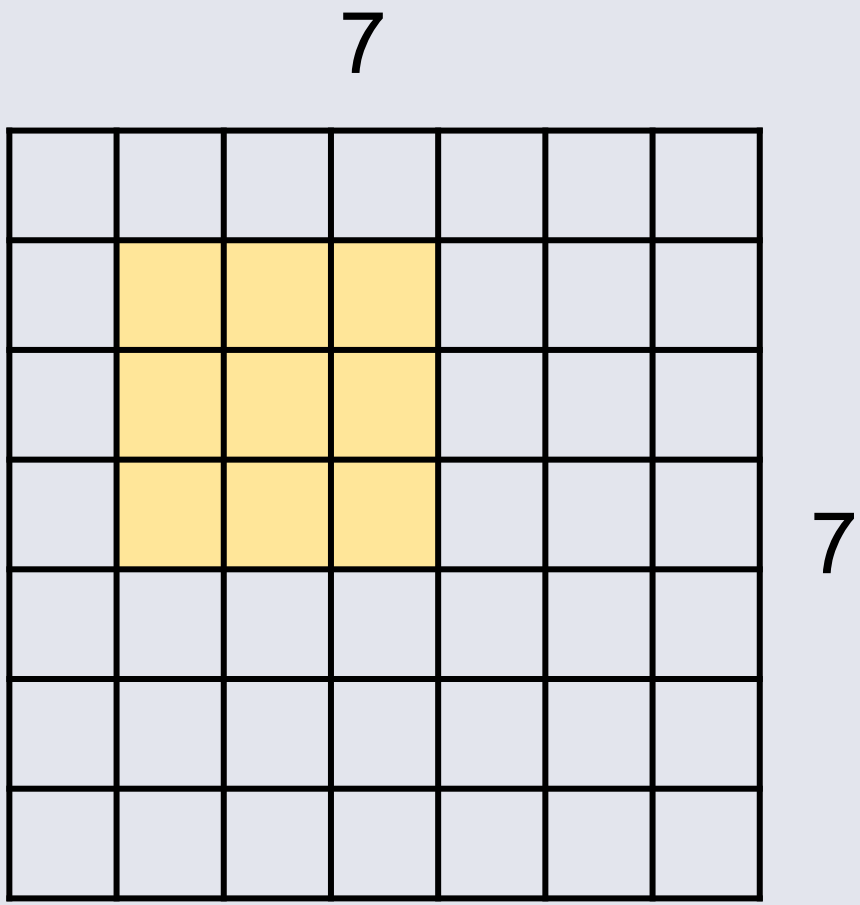
Closer Look at the Spatial Dimensions



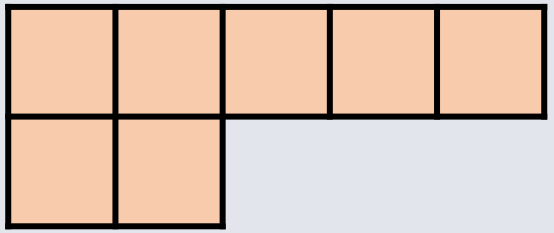
7×7 input (Spatially)



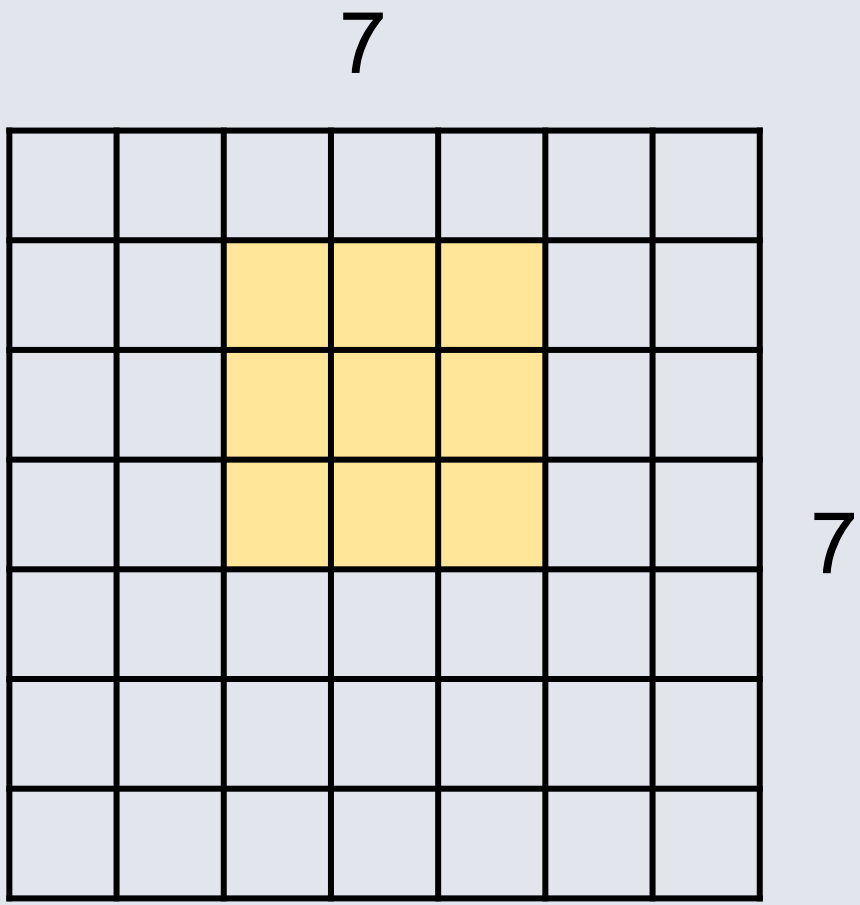
Closer Look at the Spatial Dimensions



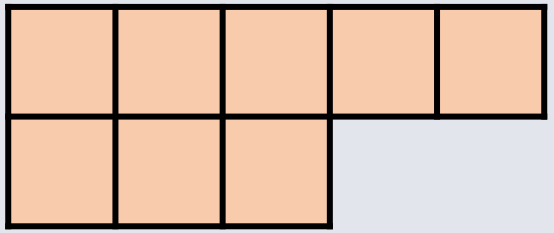
7×7 input (Spatially)



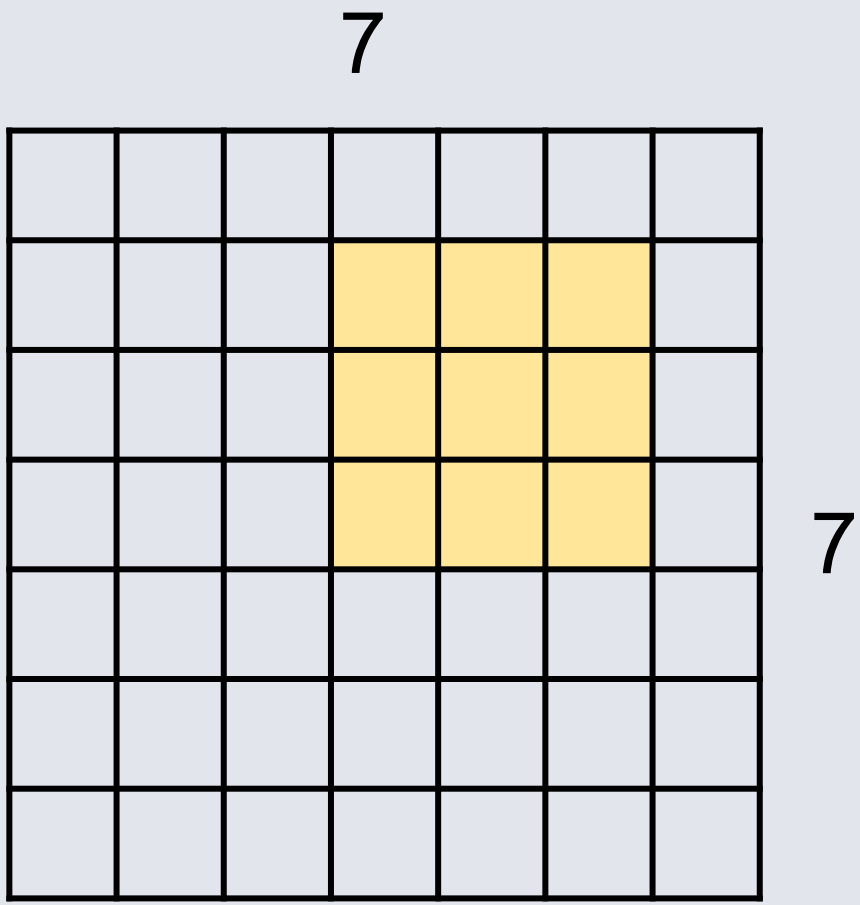
Closer Look at the Spatial Dimensions



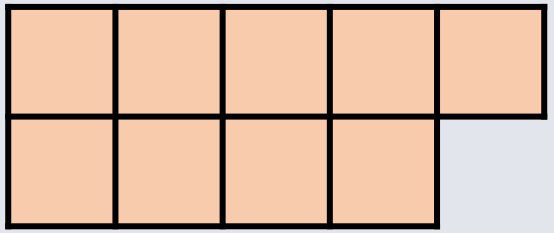
7×7 input (Spatially)



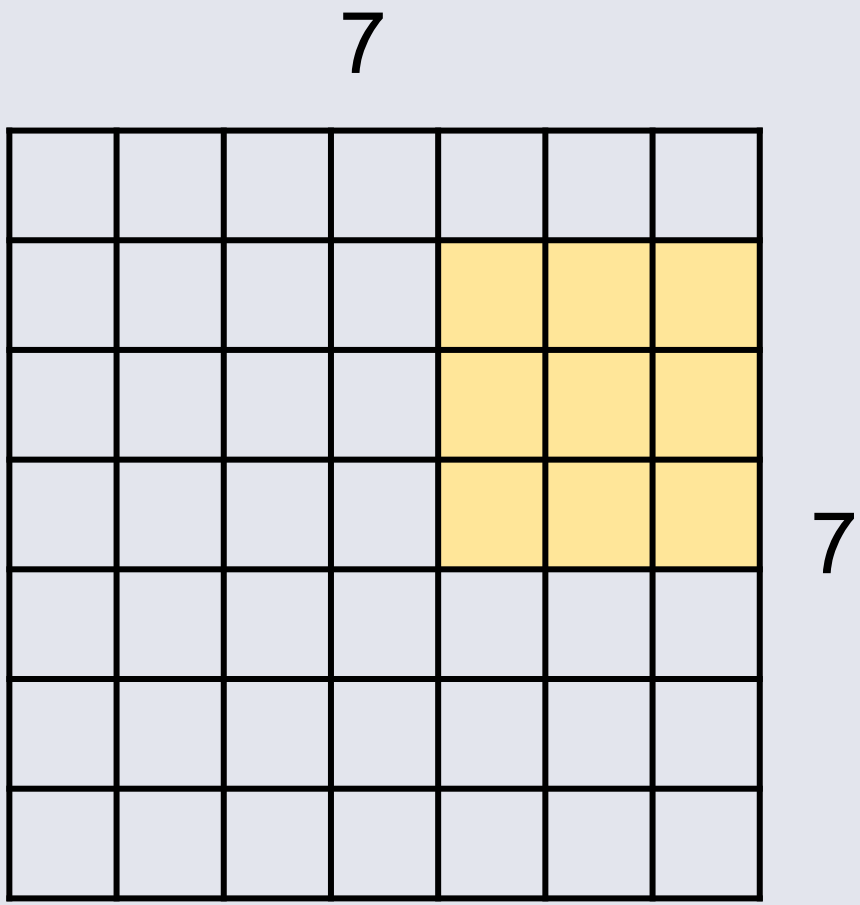
Closer Look at the Spatial Dimensions



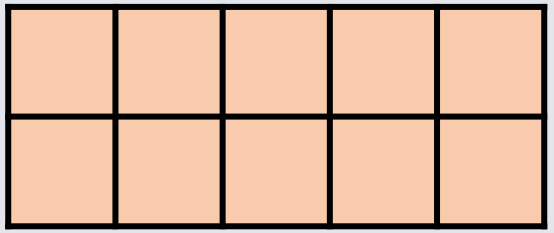
7×7 input (Spatially)



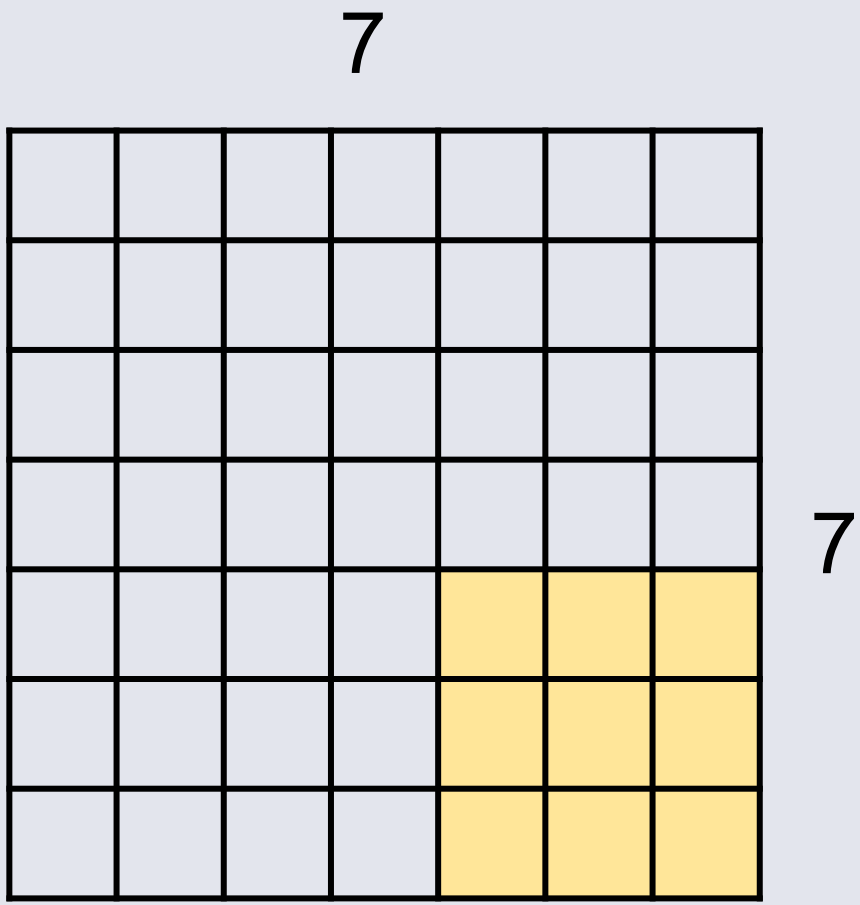
Closer Look at the Spatial Dimensions



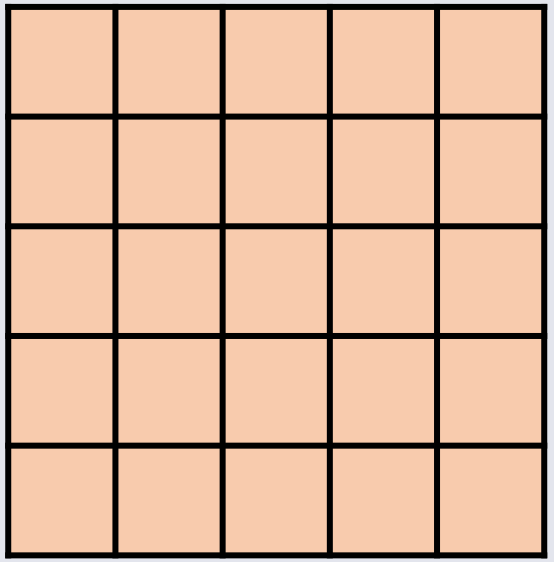
7×7 input (Spatially)



Closer Look at the Spatial Dimensions



7×7 input (Spatially)



=> 5x5 output

Building Blocks of CNN

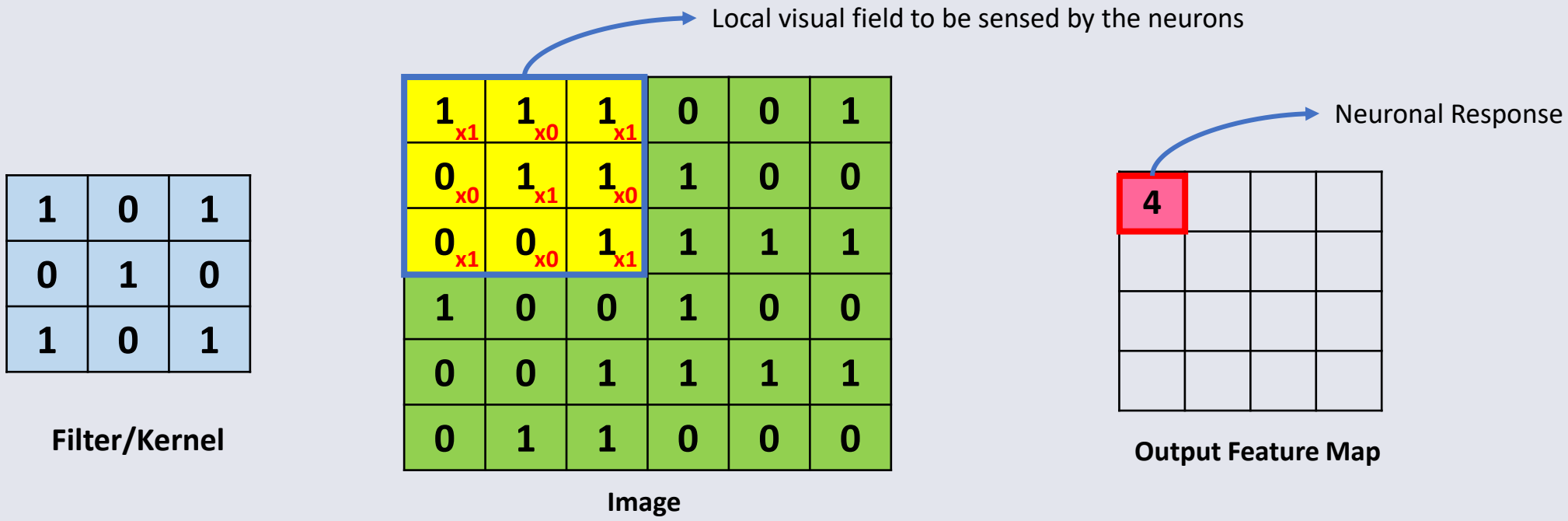
1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0	0	1
0	1	1	1	0	0
0	0	1	1	1	1
1	0	0	1	0	0
0	0	1	1	1	1
0	1	1	0	0	0

Image

Building Blocks of CNN



Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1_{x1}	1_{x0}	0_{x1}	0	1
0	1_{x0}	1_{x1}	1_{x0}	0	0
0	0_{x1}	1_{x0}	1_{x1}	1	1
1	0	0	1	0	0
0	0	1	1	1	1
0	1	1	0	0	0

Image

4	3		

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1_{x1}	0_{x0}	0_{x1}	1
0	1	1_{x0}	1_{x1}	0_{x0}	0
0	0	1_{x1}	1_{x0}	1_{x1}	1
1	0	0	1	0	0
0	0	1	1	1	1
0	1	1	0	0	0

Image

4	3	4	

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0_{x1}	0_{x0}	1_{x1}
0	1	1	1_{x0}	0_{x1}	0_{x0}
0	0	1	1_{x1}	1_{x0}	1_{x1}
1	0	0	1	0	0
0	0	1	1	1	1
0	1	1	0	0	0

Image

4	3	4	3

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0	0	1
0 x1	1 x0	1 x1	1	0	0
0 x0	0 x1	1 x0	1	1	1
1 x1	0 x0	0 x1	1	0	0
0	0	1	1	1	1
0	1	1	0	0	0

Image

4	3	4	3
2			

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0	0	1
0	1_{x1}	1_{x0}	1_{x1}	0	0
0	0_{x0}	1_{x1}	1_{x0}	1	1
1	0_{x1}	0_{x0}	1_{x1}	0	0
0	0	1	1	1	1
0	1	1	0	0	0

Image

4	3	4	3
2	4		

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0	0	1
0	1	1	1_{x1}	0_{x0}	0_{x1}
0	0	1	1_{x0}	1_{x1}	1_{x0}
1	0	0	1_{x1}	0_{x0}	0_{x1}
0	0	1	1	1	1
0	1	1	0	0	0

Image

4	3	4	3
2	4	2	3

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0	0	1
0	1	1	1	0	0
0_{x1}	0_{x0}	1_{x1}	1	1	1
1_{x0}	0_{x1}	0_{x0}	1	0	0
0_{x1}	0_{x0}	1_{x1}	1	1	1
0	1	1	0	0	0

Image

4	3	4	3
2	4	2	3
2			

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0	0	1
0	1	1	1	0	0
0	0_{x1}	1_{x0}	1_{x1}	1	1
1	0_{x0}	0_{x1}	1_{x0}	0	0
0	0_{x1}	1_{x0}	1_{x1}	1	1
0	1	1	0	0	0

Image

4	3	4	3
2	4	2	3
2	2		

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0	0	1
0	1	1	1	0	0
0	0	1_{x1}	1_{x0}	1_{x1}	1
1	0	0_{x0}	1_{x1}	0_{x0}	0
0	0	1_{x1}	1_{x0}	1_{x1}	1
0	1	1	0	0	0

Image

4	3	4	3
2	4	2	3
2	2	5	

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0	0	1
0	1	1	1	0	0
0	0	1	1 x1	1 x0	1 x1
1	0	0	1 x0	0 x1	0 x0
0	0	1	1 x1	1 x0	1 x1
0	1	1	0	0	0

Image

4	3	4	3
2	4	2	3
2	2	5	4

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0	0	1
0	1	1	1	0	0
0	0	1	1	1	1
1_{x1}	0_{x0}	0_{x1}	1	0	0
0_{x0}	0_{x1}	1_{x0}	1	1	1
0_{x1}	1_{x0}	1_{x1}	0	0	0

Image

4	3	4	3
2	4	2	3
2	2	5	4
2			

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0	0	1
0	1	1	1	0	0
0	0	1	1	1	1
1	0 x1	0 x0	1 x1	0	0
0	0 x0	1 x1	1 x0	1	1
0	1 x1	1 x0	0 x1	0	0

Image

4	3	4	3
2	4	2	3
2	2	5	4
2	3		

Output Feature Map

Building Blocks of CNN

1	0	1
0	1	0
1	0	1

Filter/Kernel

1	1	1	0	0	1
0	1	1	1	0	0
0	0	1	1	1	1
1	0	0 x1	1 x0	0 x1	0
0	0	1 x0	1 x1	1 x0	1
0	1	1 x1	0 x0	0 x1	0

Image

4	3	4	3
2	4	2	3
2	2	5	4
2	3	2	

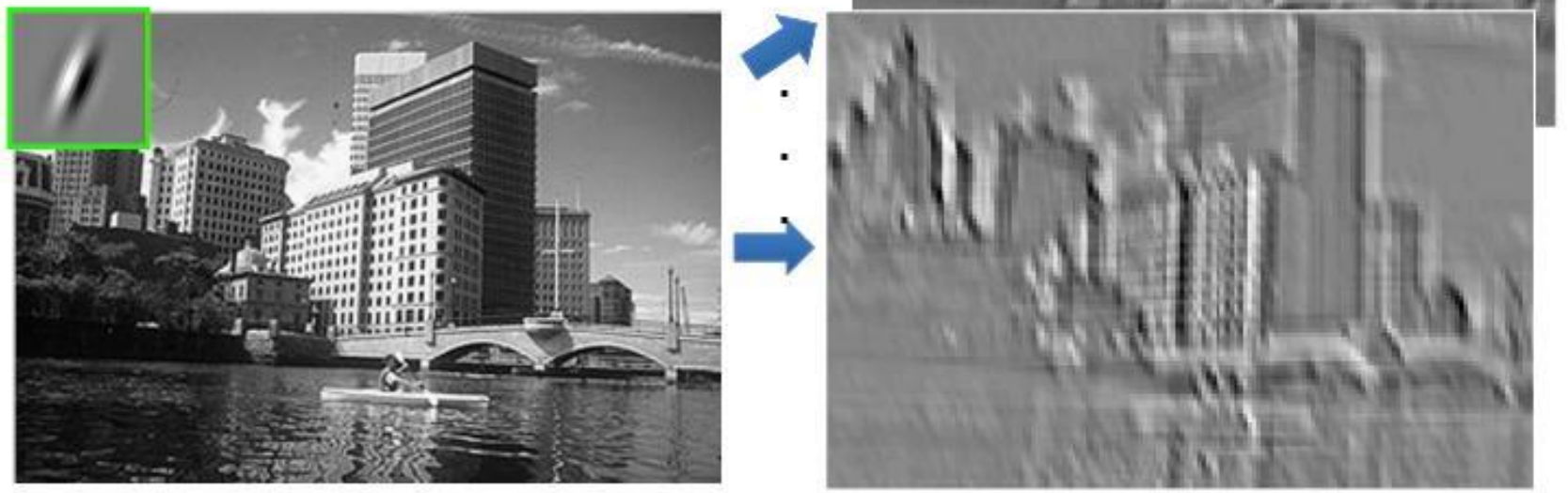
Output Feature Map

Building Blocks of CNN

The output feature map depends on the weights of this filter.
These filter weights are learned from the input training data

Building Blocks of CNN

- Dependencies are local
- Translation invariance
- Few parameters (filter weights)
- Stride can be greater than 1 (faster, less memory)



Input

Feature Map

Pooling Layer

A **subsampling layer**, commonly referred to as a **pooling layer**, reduces the spatial size of feature maps, which helps achieve translation invariance and reduces computational complexity.

Types of Pooling (Subsampling) Layers

Max Pooling: Takes the maximum value from a given window. Helps preserve the most important features.

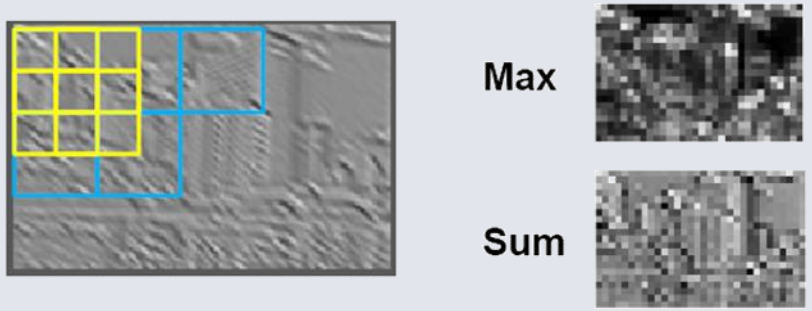
Average Pooling: Computes the average value in the window, resulting in a smoother representation.

Global Pooling: Reduces an entire feature map to a single value, often used before fully connected layers.

Building Blocks of CNN

Pooling Layer

- In order to reduce variance, pooling layers compute the max or average value of a particular feature over a region of the image
- This will ensure that the same result will be obtained, even when image features have small translations
- This is an important operation for object classification and detection



Building Blocks of CNN

Pooling Layer

4	3	4	3
2	4	2	3
2	2	5	4
2	3	2	2

Feature Map

Feature Pooling

Pooling Scheme: Max Pooling
Stride: 2
Filter Size: 2x2

Building Blocks of CNN

Pooling Layer

4	3	4	3
2	4	2	3
2	2	5	4
2	3	2	2

Feature Map

4	

Feature Pooling

Pooling Scheme: Max Pooling
Stride: 2
Filter Size: 2x2

Building Blocks of CNN

Pooling Layer

4	3	4	3
2	4	2	3
2	2	5	4
2	3	2	2

Feature Map

4	4

Feature Pooling

Pooling Scheme: Max Pooling
Stride: 2
Filter Size: 2x2

Building Blocks of CNN

Pooling Layer

4	3	4	3
2	4	2	3
2	2	5	4
2	3	2	2

Feature Map

4	4
3	

Feature Pooling

Pooling Scheme: Max Pooling
Stride: 2
Filter Size: 2x2

Building Blocks of CNN

Pooling Layer

4	3	4	3
2	4	2	3
2	2	5	4
2	3	2	2

Feature Map

4	4
3	5

Feature Pooling

Pooling Scheme: Max Pooling
Stride: 2
Filter Size: 2x2

Building Blocks of CNN

Pooling Layer

4	3	4	3
2	4	2	3
2	2	5	4
2	3	2	2

Feature Map

4	4
3	5

Feature Pooling

Pooling Scheme: Max Pooling
Stride: 2
Filter Size: 2x2

After Pooling the size of feature map gets reduced by a factor of 2
Before pooling:
Feature Map = 4 x 4
After pooling:
Feature Map = 2 x 2

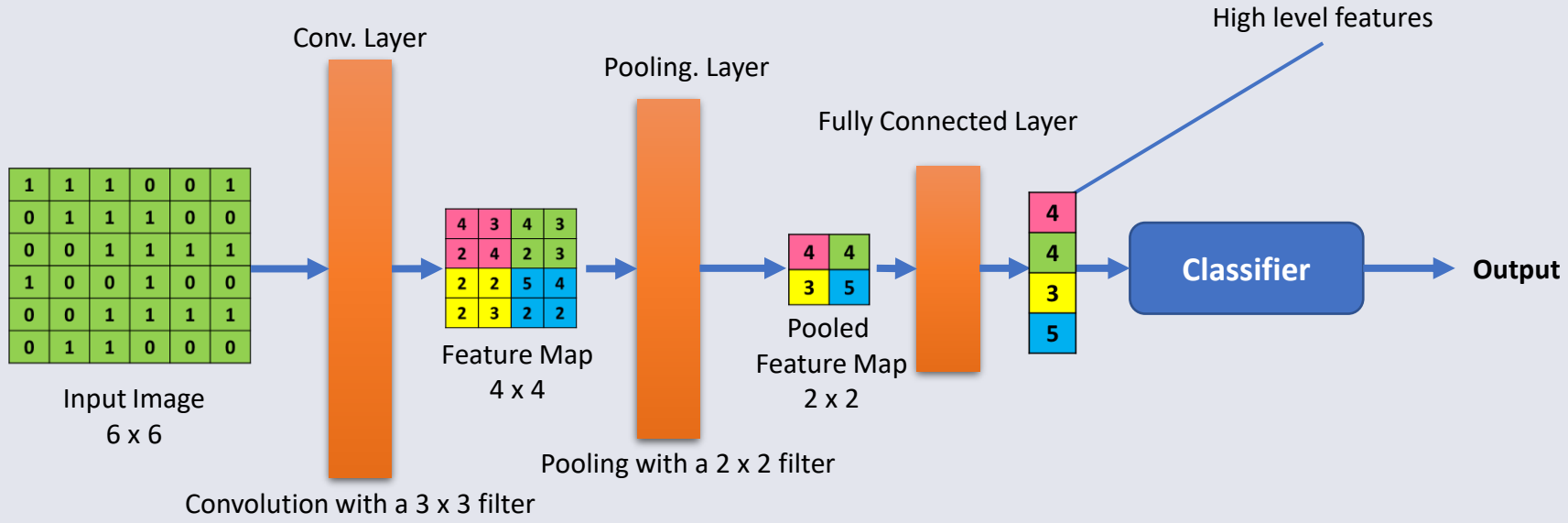
Constructing CNNs

- Construct a CNN by placing a number of convolution and pooling layers in cascade

  Specify input size

  Specify filter size at each layer, etc.

- Add a number of fully connected layers after the conv. and pooling layers
- Add a classifier at the end of the network



Next week

Extension of CNN

Any Question?