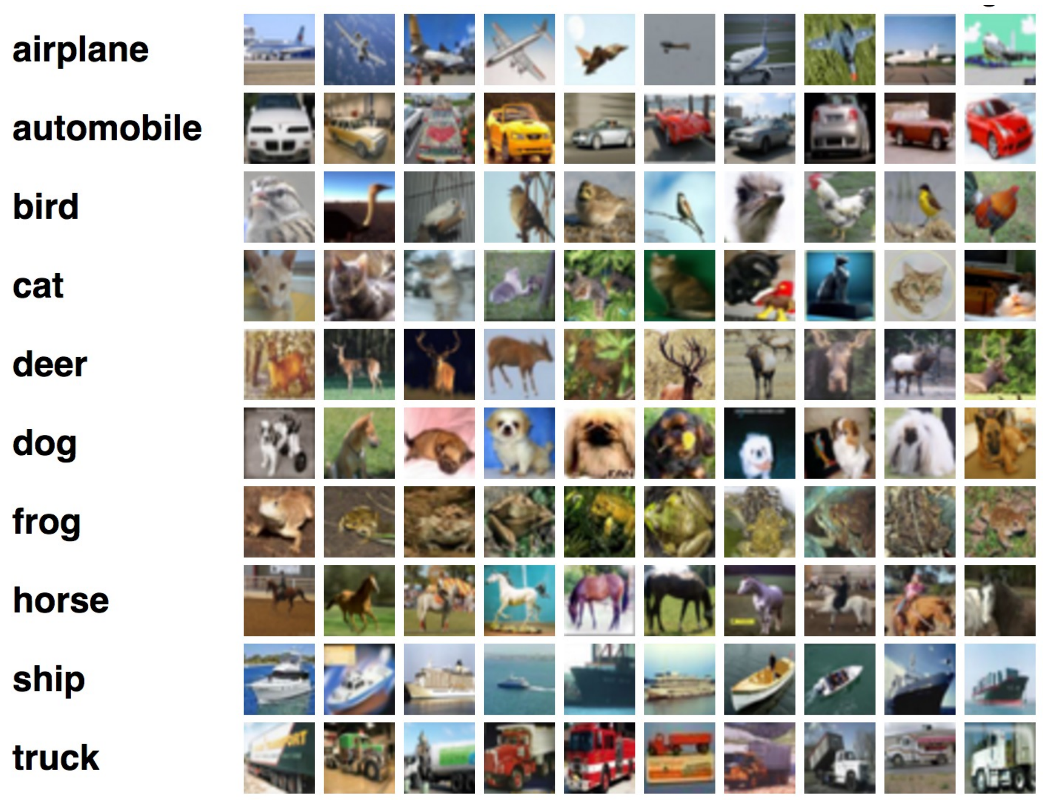
DS-UA 301
Advanced Topics in Data Science
Advanced Techniques in ML and Deep Learning

LECTURE 5

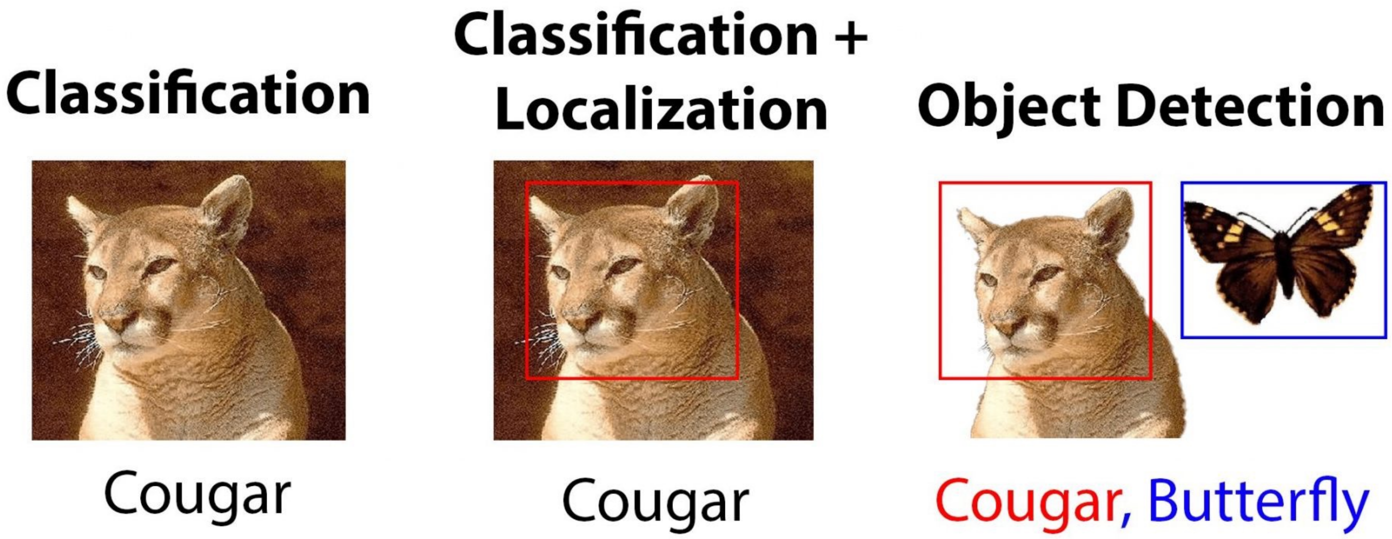
Parijat Dube

Deep Learning for Computer Vision

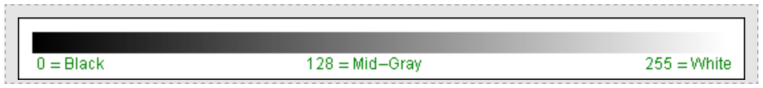
Image classification



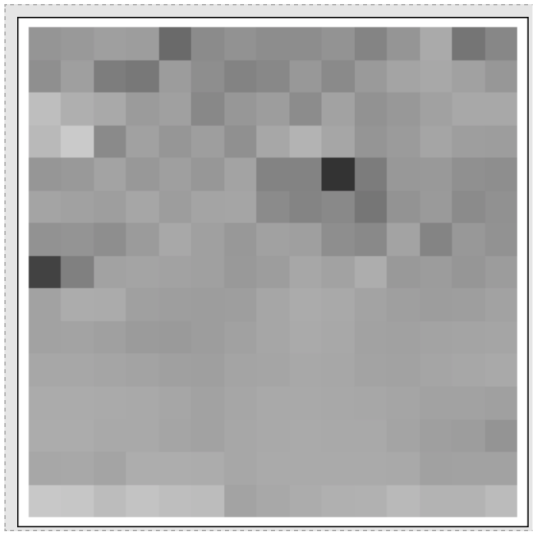
Other vision tasks

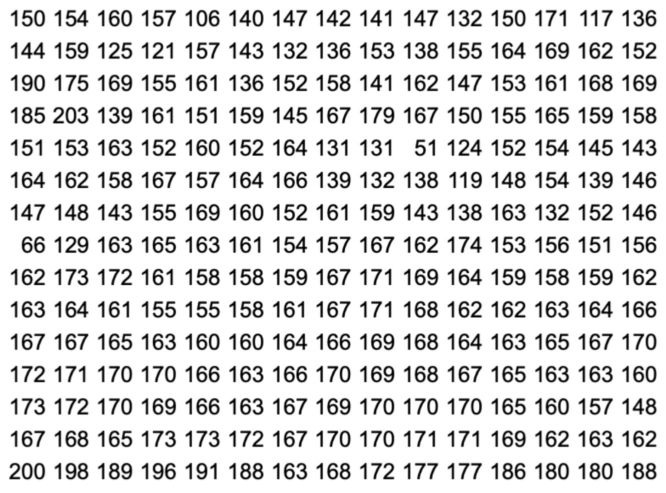


Pixel and their values



The range of intensity values from 0 (black) to 255 (white).

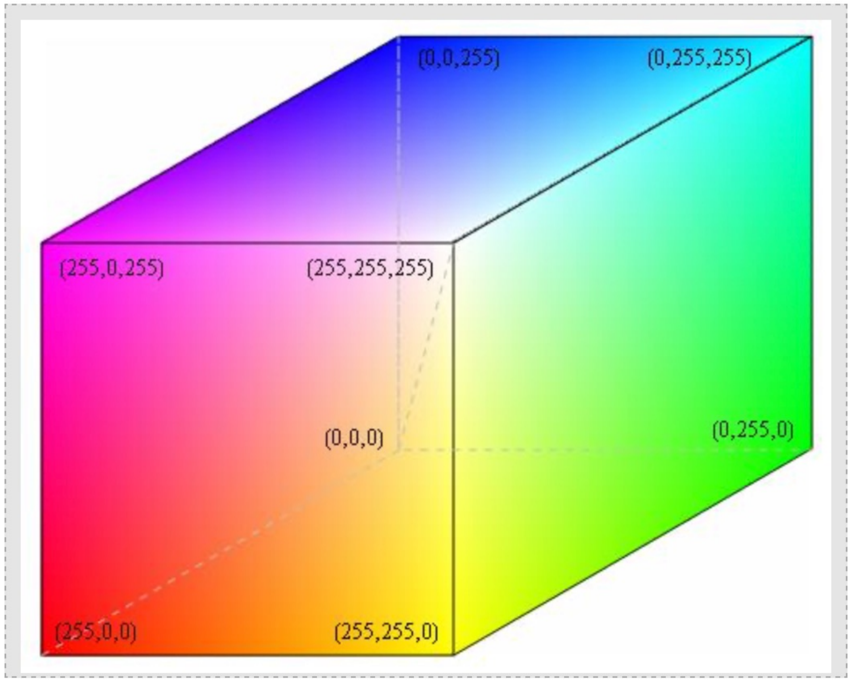




150	154	160	157	106	140	147	142	141	147	132	150	171	117	136
144	159	125	121	157	143	132	136	153	138	155	164	169	162	152
190	175	169	155	161	136	152	158	141	162	147	153	161	168	169
185	203	139	161	151	159	145	167	179	167	150	155	165	159	158
151	153	163	152	160	152	164	131	131	51	124	152	154	145	143
164	162	158	167	157	164	166	139	132	138	119	148	154	139	146
147	148	143	155	169	160	152	161	159	143	138	163	132	152	146
66	129	163	165	163	161	154	157	167	162	174	153	156	151	156
162	173	172	161	158	158	159	167	171	169	164	159	158	159	162
163	164	161	155	155	158	161	167	171	168	162	162	163	164	166
167	167	165	163	160	160	164	166	169	168	164	163	165	167	170
172	171	170	170	166	163	166	170	169	168	167	165	163	163	160
173	172	170	169	166	163	167	169	170	170	170	165	160	157	148
167	168	165	173	173	172	167	170	170	171	171	169	162	163	162
200	198	189	196	191	188	163	168	172	177	177	186	180	180	188

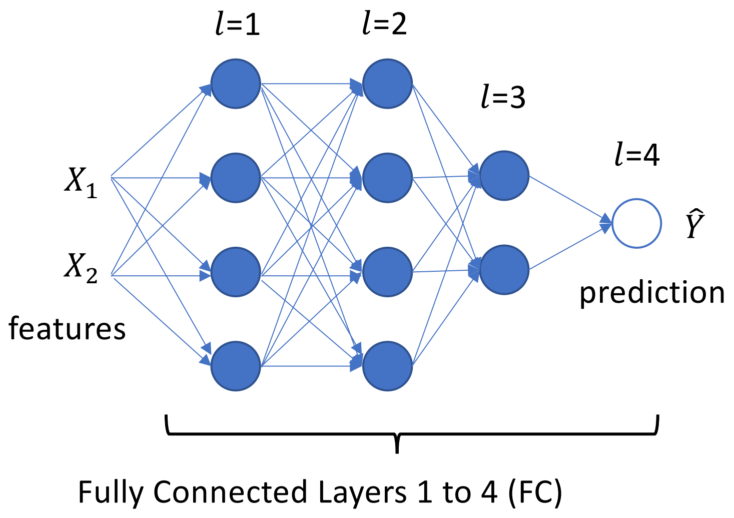
http://whydomath.org/node/wavlets/imagebasics.html

RGB Colorspace



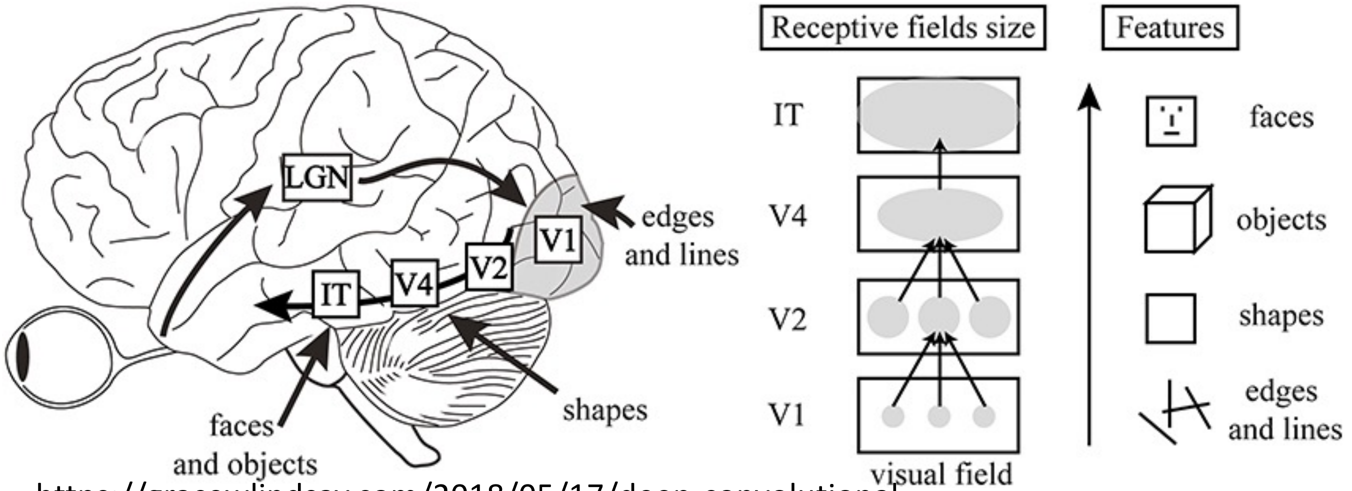
Need for special neural network architectures

- **Fully-Connected Neural Networks** for image recognition/classification:
  1. **Dimension Problem**:
     - Reshaping to 1D: loosing the spatial structure
  2. **Size Problem:**
     - Example features: 1000 × 1000 × 3 [Y × X × RGB]
     - Weights: 3M for each neuron just for first layer
     - 100 layers: about 300M DP weights => **2.4GB model size**
  3. **Overfitting problem:**
     - One weight per pixel leads to **overfitting**

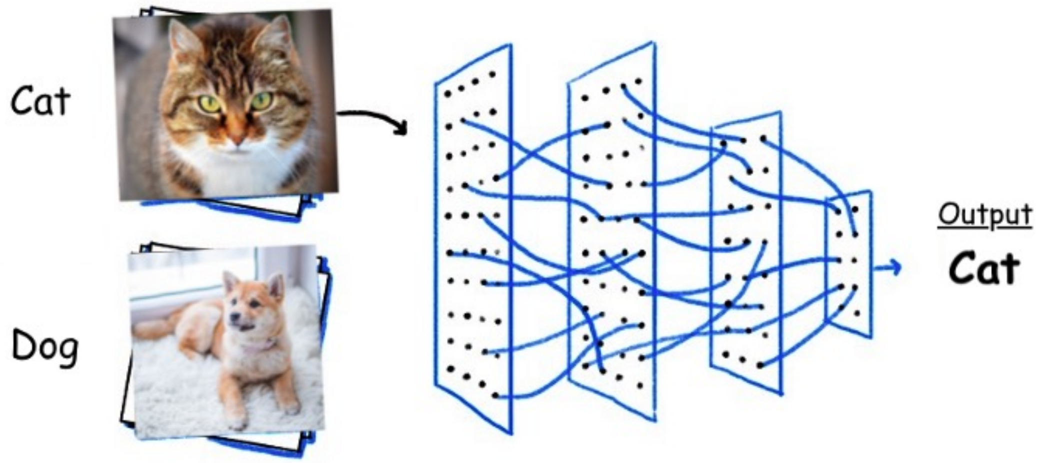


Which Architecture?

- Convolutional neural networks have three main traits that support their use as models of biological vision: (1) they can perform visual tasks at near-human levels, (2) they do this with an architecture that replicates basic features known about the visual system, and (3) they produce activity that is directly relatable to the activity of different areas in the visual system.



https://gracewlindsay.com/2018/05/17/deep-convolutional-neural-networks-as-models-of-the-visual-system-qa/


Cat
Dog
Output
Cat

Convolutional Neural Networks (CNNs)

- Widely used in computer vision
- Specialized structure: inspired by Hubel and Wiesel experiments on visual cortexes of cat and monkey
- Success in ImageNet (ILSVRC) competitions brought them into limelight since 2012
- Layers have *volume*: length, width, depth
- The depth is 3 for color images, 1 for grayscale, and an arbitrary value for hidden layers.
- Three basic operations in CNNs are convolution, pooling, and ReLU.
    - The convolution operation is analogous to the matrix multiplication in a conventional network

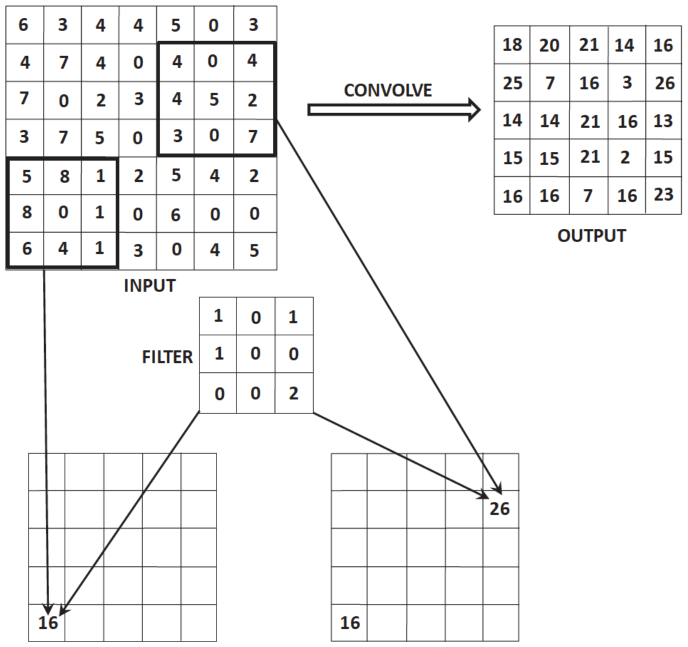
How CNNs work?

- https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/

Example of MNIST classification

- https://python-course.eu/neural_network_mnist.php

Example of Convolution



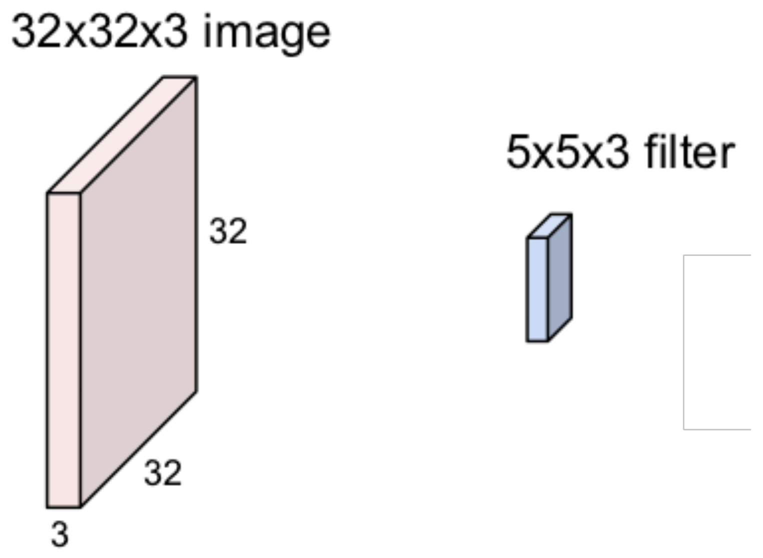
6	3	4	4	5	0	3
4	7	4	0	4	0	4
7	0	2	3	4	5	2
3	7	5	0	3	0	7
5	8	1	2	5	4	2
8	0	1	0	6	0	0
6	4	1	3	0	4	5

1	0	1
1	0	0
0	0	2

18	20	21	14	16
25	7	16	3	26
14	14	21	16	13
15	15	21	2	15
16	16	7	16	23

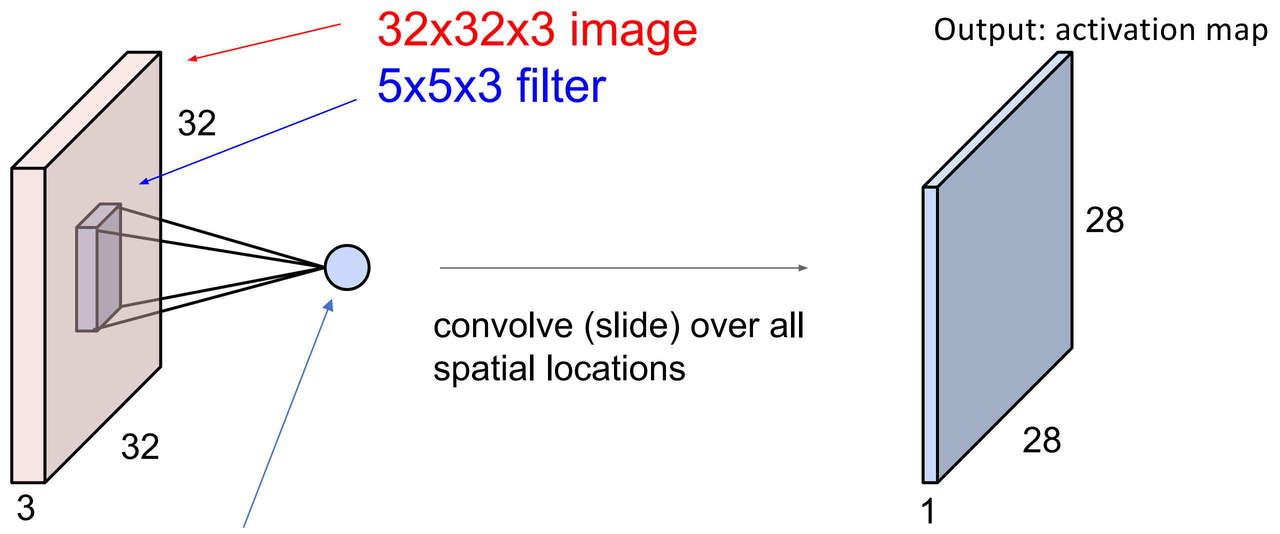
- Add up over multiple activations maps from the depth

Convolution Layer with 3 dimensions



- Preserve the spatial structure of the image
- Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”
- Filters always extend the full depth of the input volume (the 3 channels in this example)

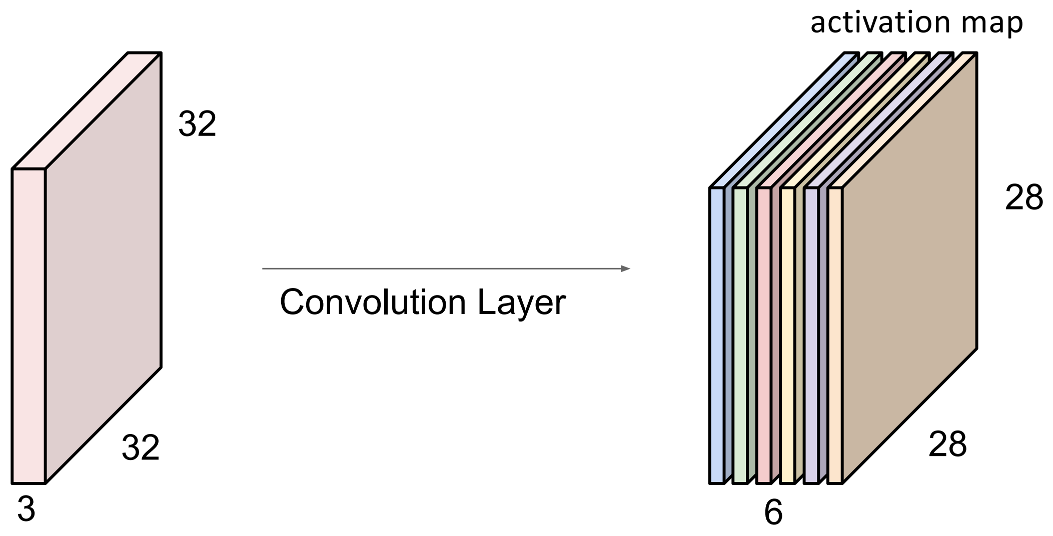
Output image: Activation Map



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

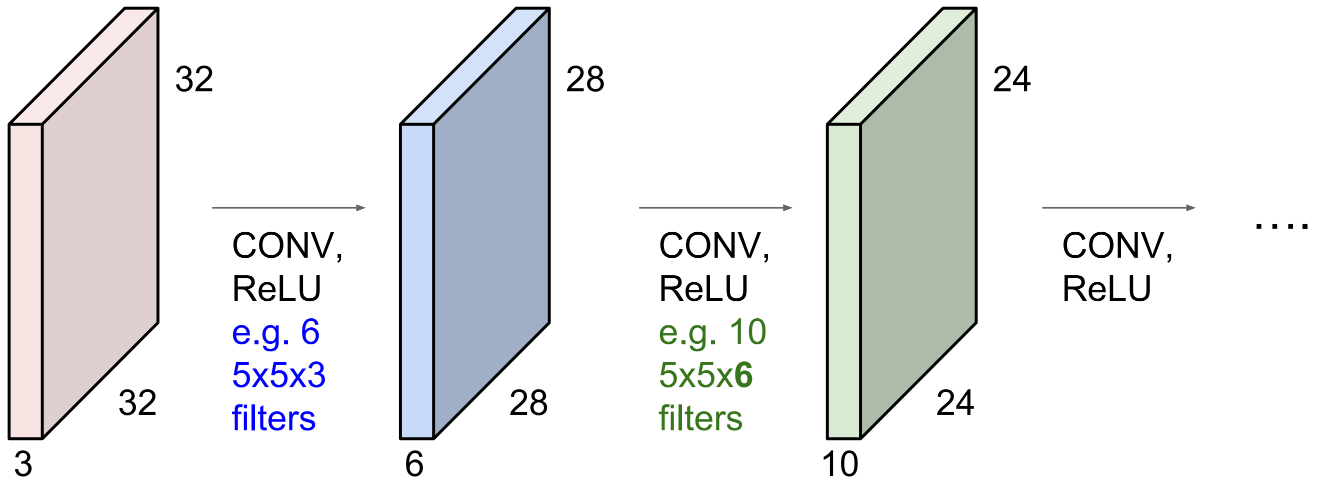
Multiple filters Activation Map

- If we had 6 5x5x3 filters, we'll get **6 separate activation maps**
- We stack these up to get a "new image" of size 28x28x6!



Convolutional Network

- A Convolution Network is a sequence of Convolutional Layers, interspersed with activation functions



Convolution Layer: Size of output

$$\begin{tikzcd} L_q \times B_q \times d_q \arrow[d, "F_q \times F_q \times \boxed{d_q}"] \\ (L_q - F_q + 1) \times (B_q - F_q + 1) \end{tikzcd}$$

with stride S_q

$$\begin{tikzcd} L_q \times B_q \times d_q \arrow[d, "F_q \times F_q \times d_q"] \\ (L_q - F_q)/S_q + 1 \times (B_q - F_q)/S_q + 1 \end{tikzcd}$$

When a stride of S_q is used in the qth layer, the convolution is performed at the locations 1, $S_q + 1$, $2\,S_q + 1$, and so on along both spatial dimensions of the layer.

Convolution Neural Networks Properties

- Sparse connectivity because we are creating a feature from a region in the input volume of the size of the filter (usually much smaller than the input size)
    - Each neuron in the output layer has connectivity to a small subset of neurons in the input layer which are spatially close
- Shared weights because we use the same filter across entire spatial volume
    - Interpret a shape in various parts of the image in the same way
    - Reduces the number of parameters to learn
- A feature in a hidden layer captures some properties of a region of the input image.

Padding

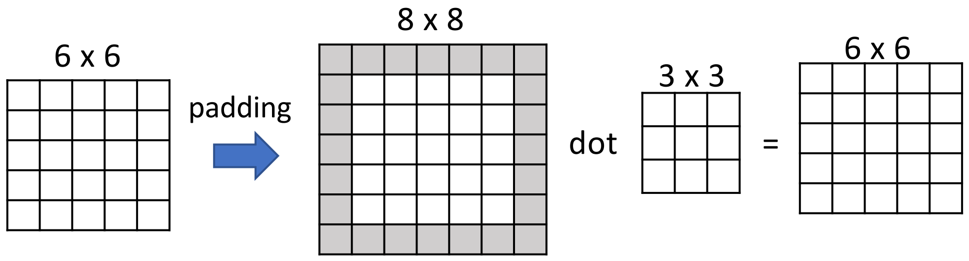
- Convolution
  - Image size 6 × 6 (n = 6)
  - Filter size f = 3 × 3 (f = 3)
  - Output: 4 × 4 (n' = 4)
  - Output size formula:
    - size: **n' × n'**
    - where **n' = n – f + 1**

$$\overset{\text{6 x 6 image}}{\begin{array}{|c|c|c|c|c|c|}\hline 8 & 8 & 8 & 0 & 0 & 0 \\ \hline 8 & 8 & 8 & 0 & 0 & 0 \\ \hline 8 & 8 & 8 & 0 & 0 & 0 \\ \hline 8 & 8 & 8 & 0 & 0 & 0 \\ \hline 8 & 8 & 8 & 0 & 0 & 0 \\ \hline 8 & 8 & 8 & 0 & 0 & 0 \\ \hline \end{array}} \quad \text{dot} \quad \overset{\text{3 x 3 filter}}{\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array}} \quad = \quad \overset{\text{4 x 4 image}}{\begin{array}{|c|c|c|c|}\hline 0 & 24 & 24 & 0 \\ \hline 0 & 24 & 24 & 0 \\ \hline 0 & 24 & 24 & 0 \\ \hline 0 & 24 & 24 & 0 \\ \hline \end{array}}$$

- Problem:
  - Edges are used less by the convolution (less relevant)
  - Output image shrinks

Padding - Definition

- Padding:
  - Extend input image adding a “frame”
- Convolution size:
  - Input image: **n × n**
  - Filter: **f × f**
  - Output image: **n' × n’**
  - **Without padding**:
    - **n' = n – f + 1**
  - **With padding** of **p**:
    - **n' = n + 2p – f + 1**
- More definitions:
  - “Same” convolution: convolution with padding when output image and input image size are the same
  - “Valid” convolution: no padding
- In PyTorch you have to specify the size of the padding (Default is 0)

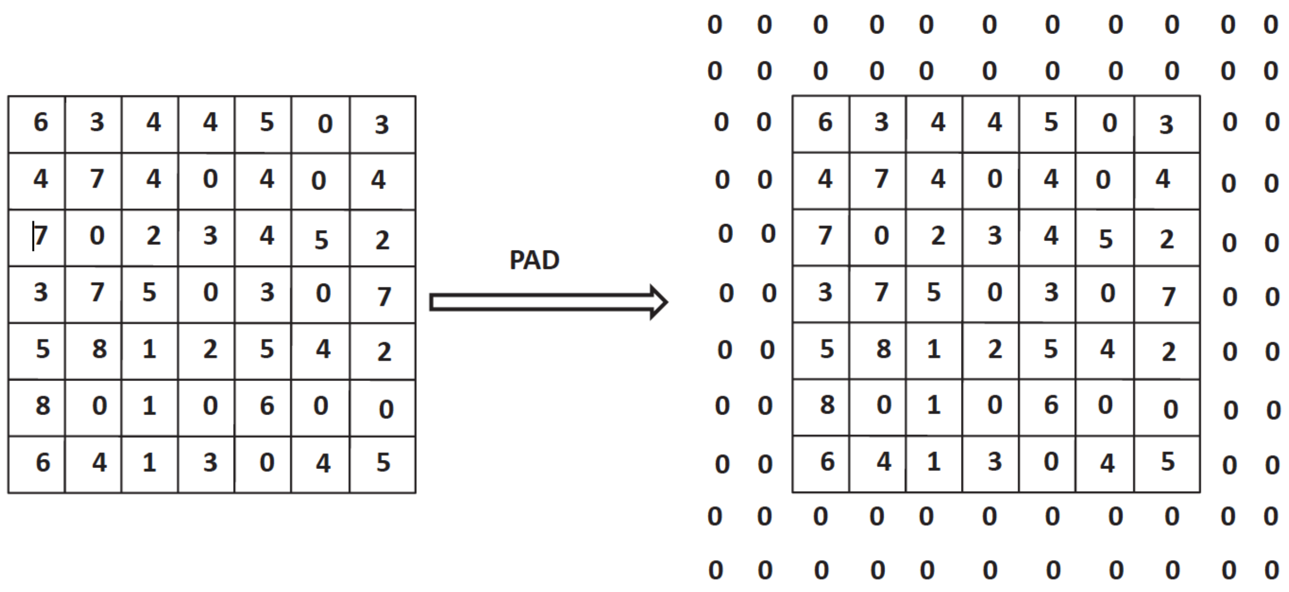


- Example Convolution with padding of 1:
  1. Do padding and obtain a 8 x 8 image
  2. Do convolution and obtain a 6 x 6 image

Padding

- Convolution results in loss of information
  - Size of the output layer is smaller than the size of the input layer
  - Loss of information along the borders of the image (or of the feature map, in the case of hidden layers).
- By adding (Fq − 1)/2 “pixels” all around the borders of the feature map, one can maintain the size of the spatial image by taking stride =1
- Padding can be of any size up to Fq-1
- What happens if you add a padding of size Fq or greater ?

Padding example



6	3	4	4	5	0	3
4	7	4	0	4	0	4
7	0	2	3	4	5	2
3	7	5	0	3	0	7
5	8	1	2	5	4	2
8	0	1	0	6	0	0
6	4	1	3	0	4	5

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	6	3	4	4	5	0	3	0	0
0	0	4	7	4	0	4	0	4	0	0
0	0	7	0	2	3	4	5	2	0	0
0	0	3	7	5	0	3	0	7	0	0
0	0	5	8	1	2	5	4	2	0	0
0	0	8	0	1	0	6	0	0	0	0
0	0	6	4	1	3	0	4	5	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

- A padding of size P increases both the length and width of input by 2P

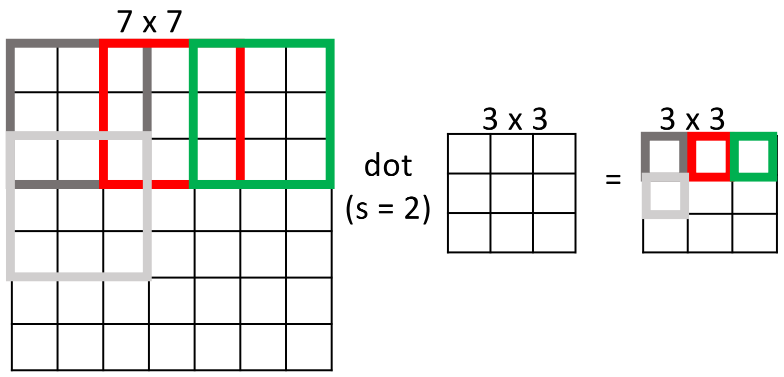
Stride

- Sometimes we want to obtain a **smaller output image** than the standard convolution or we want to have a lighter computation
- We can have the filter **jump a few pixels** at each step (stride)
- There are several implications:
  - The **compute time will diminish**
  - The **output size also will diminish**
  - The **convolution will be less fine-grained** (precise)

Stride - Definition

- Stride:
  - Jump S elements when moving filter
  - If filter falls outside image:
    - ignore computation: no output
- Convolution size:
  - Input image: **n × n**
  - Filter: **f × f**
  - Output image: **n' × n'**
  - Stride size: **s**
  - **With padding** of **p** and stride **s**:

$$n' = \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

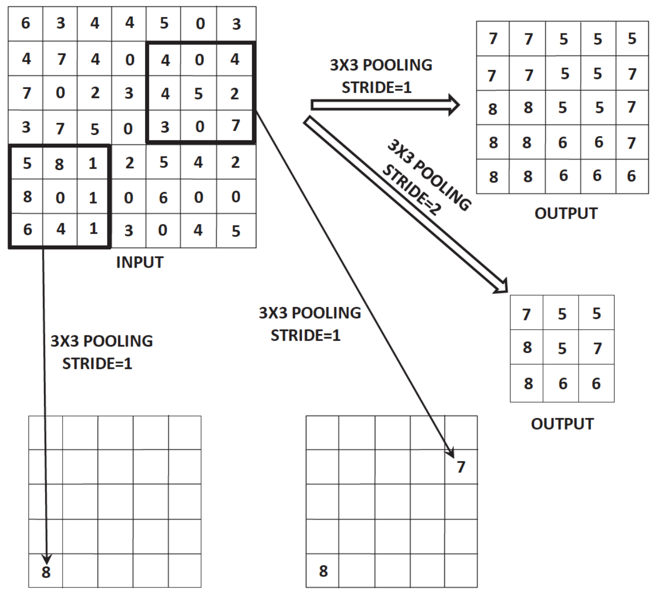


- Example Convolution with stride of 2
  - Input image: 7x7
  - Filter: 3x3
  - Stride: 2
  - Output image **n' × n'** : 3x3

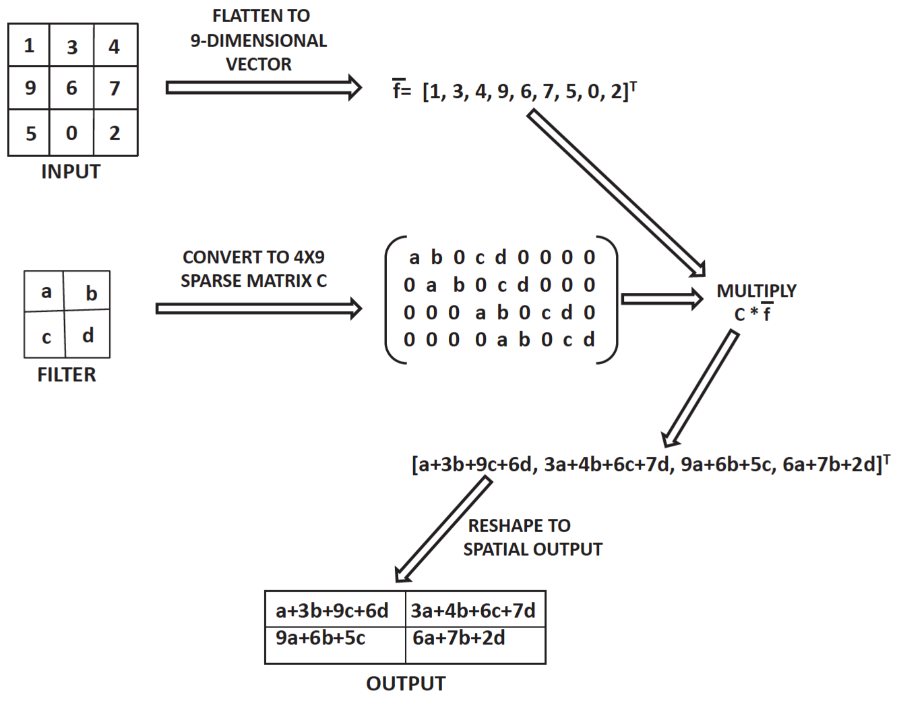
$$n' = \left\lfloor \frac{7+0-3}{2} + 1 \right\rfloor = 3$$

Max Pooling

- Pooling operates on square regions of size P x P in each of the activation maps of the input and returns maximum of the values in those regions
- Pooling preserves the depth; depth of output is same as input
- Pooling is a subsampling technique; Greatly reduces the size of output activation maps and hence the number of parameters to learn in subsequent layers

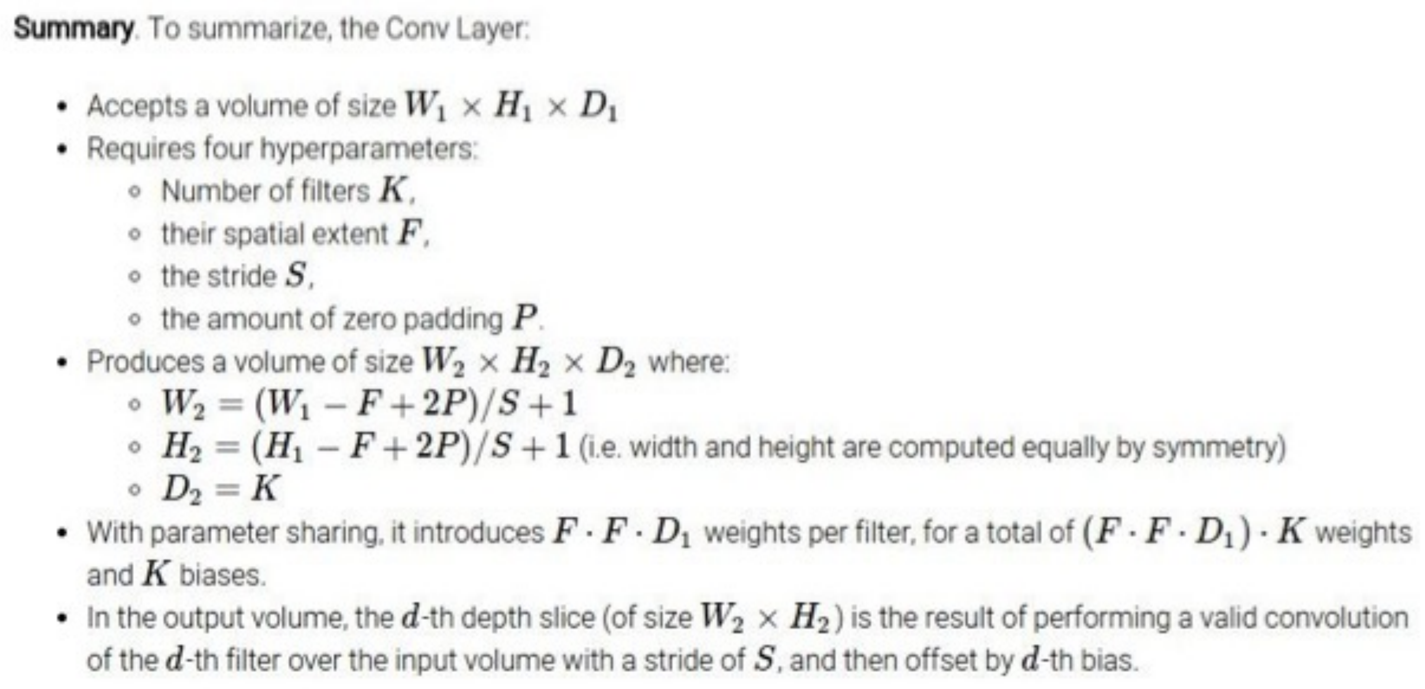


Convolution as a Matrix Operation



1 3 4
9 6 7
5 0 2
INPUT
FLATTEN TO 9-DIMENSIONAL VECTOR
$\overline{f}$= [1, 3, 4, 9, 6, 7, 5, 0, 2]^T
a b
c d
FILTER
CONVERT TO 4X9 SPARSE MATRIX C
a b 0 c d 0 0 0 0
0 a b 0 c d 0 0 0
0 0 0 a b 0 c d 0
0 0 0 0 a b 0 c d
MULTIPLY C * $\overline{f}$
[a+3b+9c+6d, 3a+4b+6c+7d, 9a+6b+5c, 6a+7b+2d]^T
RESHAPE TO SPATIAL OUTPUT
a+3b+9c+6d
3a+4b+6c+7d
9a+6b+5c
6a+7b+2d
OUTPUT

Convolution Summary

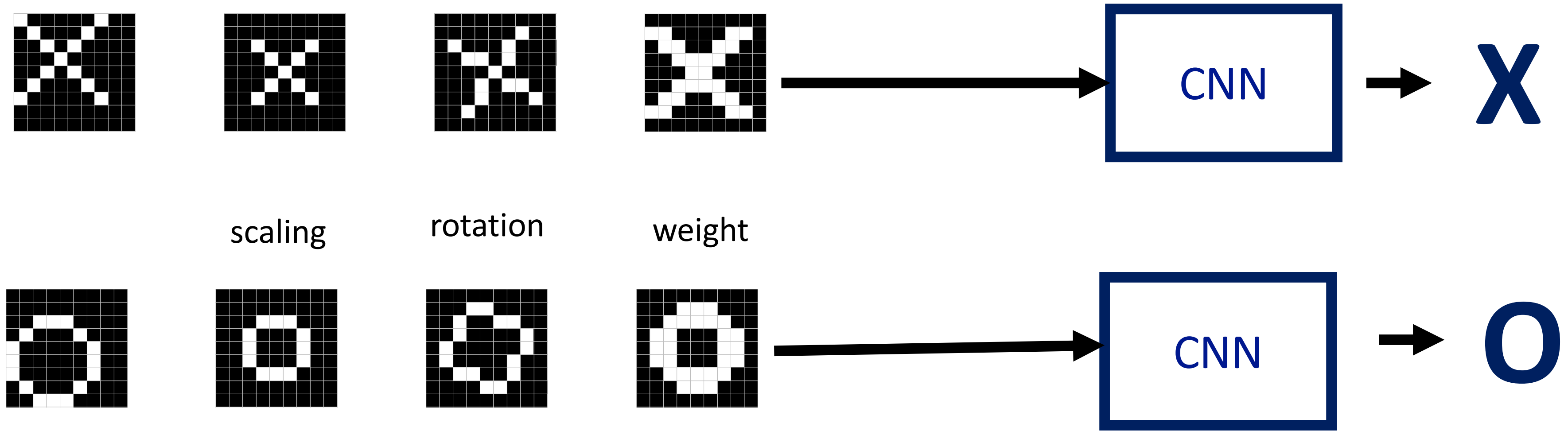


Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

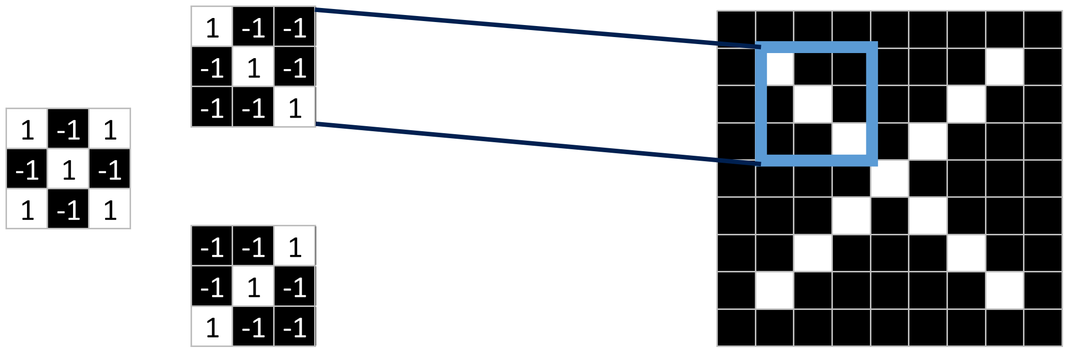
Convolution example – Image Classification

- Says whether a picture is of an X or an O

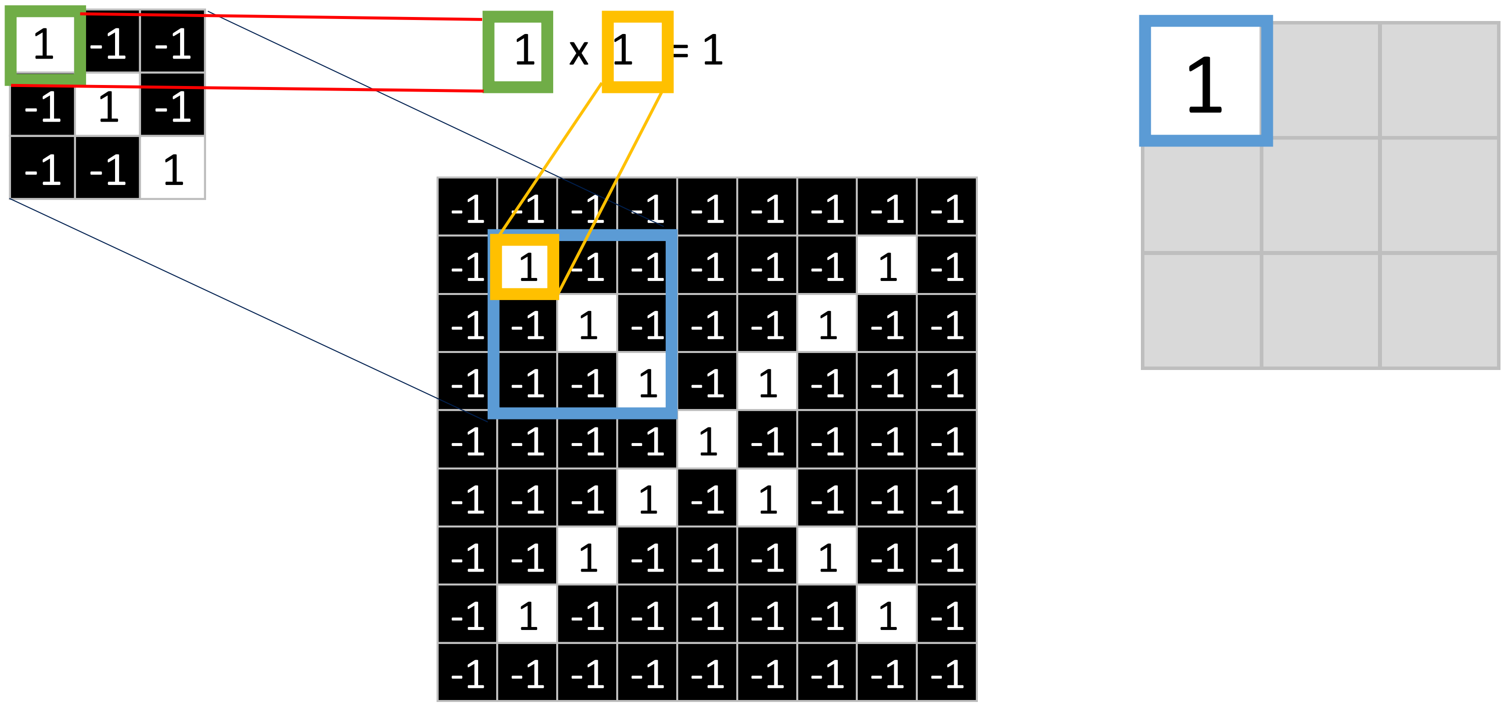


Feature recognition Example

1. Line up the feature and the image patch.
2. Multiply each image pixel by the corresponding feature pixel.
3. Add them up.
4. Divide by the total number of pixels in the feature **[optional: to normalize values]**



Feature recognition

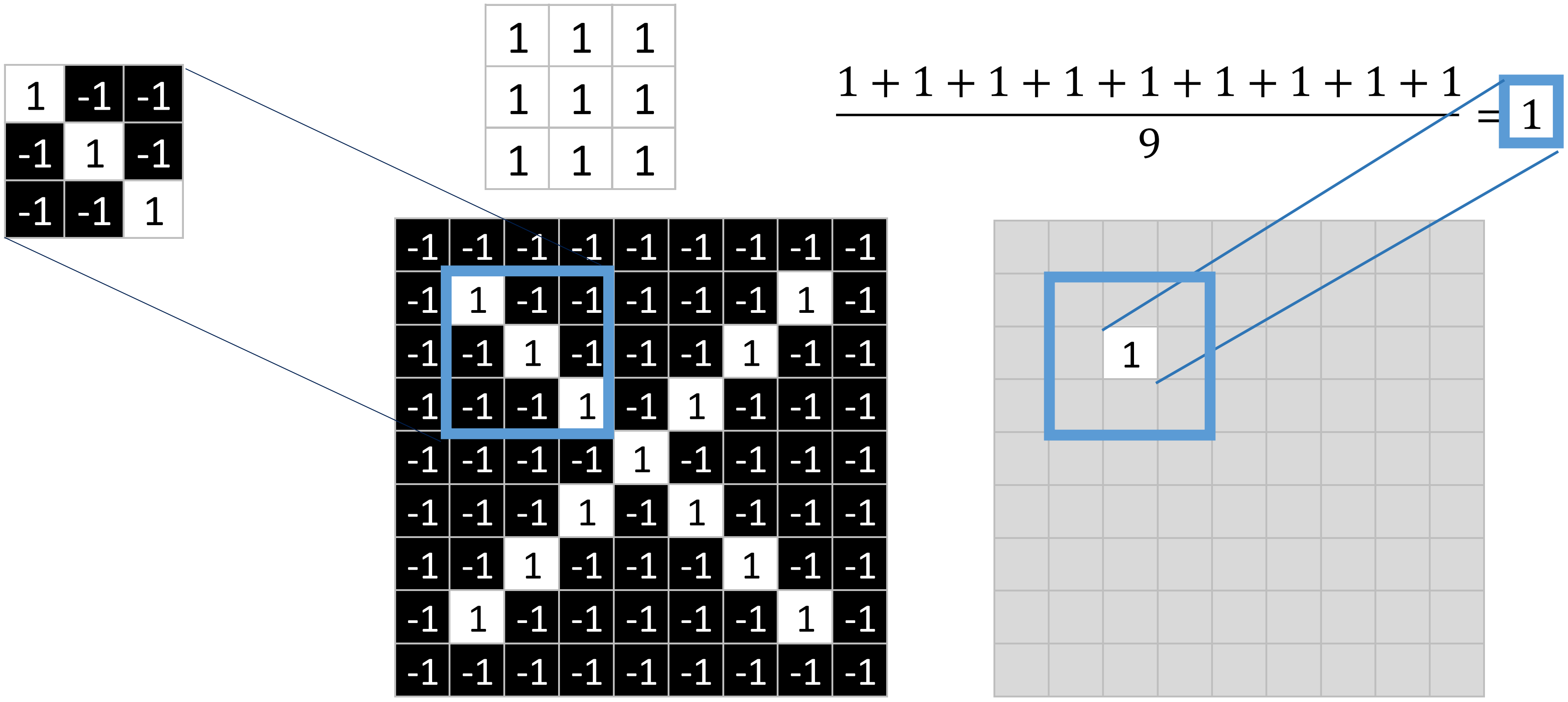


1	-1	-1
-1	1	-1
-1	-1	1

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

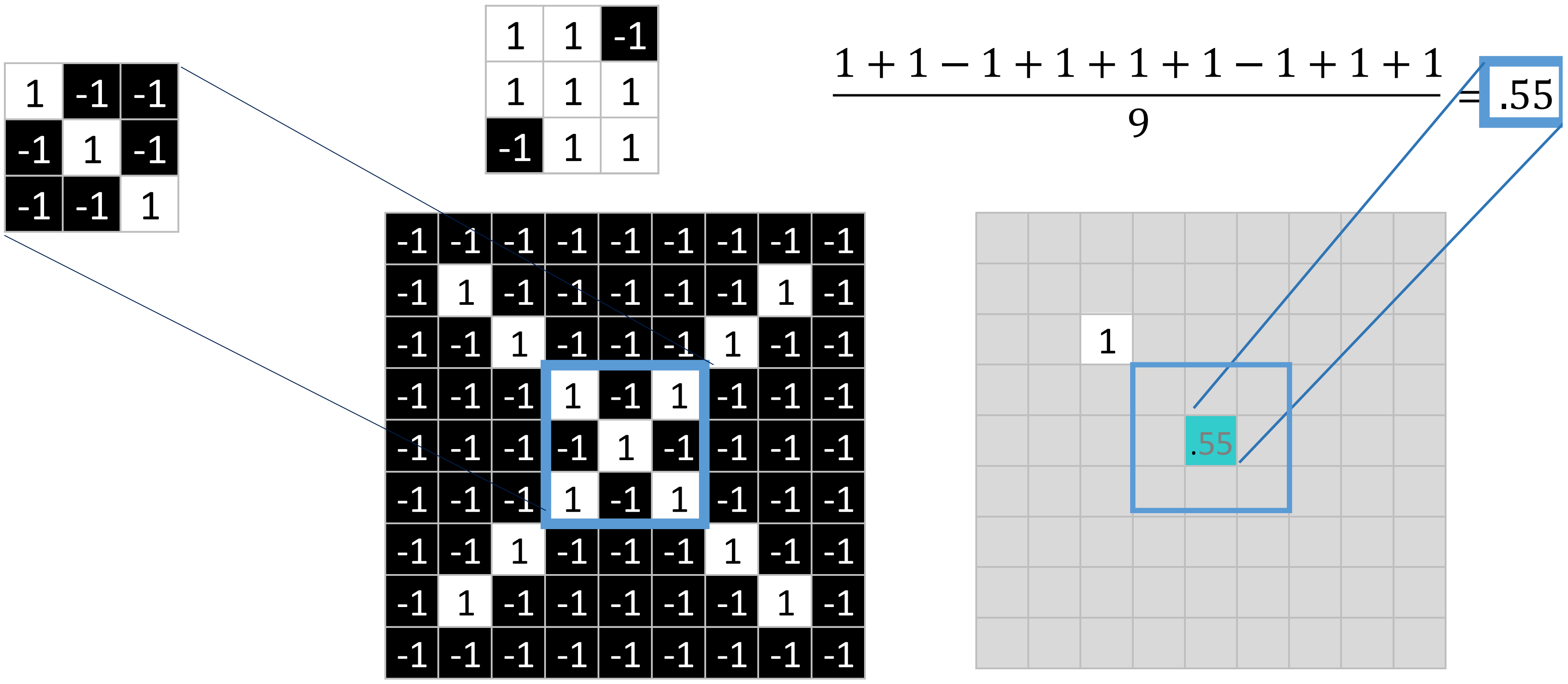
1		

Feature recognition



$$\frac{1+1+1+1+1+1+1+1+1}{9} = \boxed{1}$$

Feature recognition



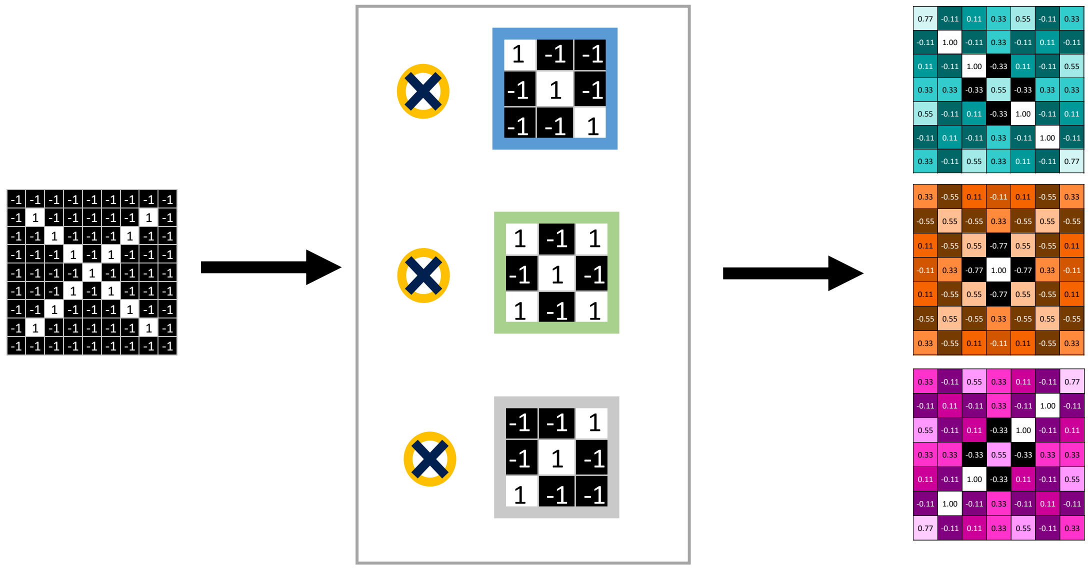
$$\frac{1+1-1+1+1+1-1+1+1}{9} = \boxed{.55}$$

Convolution: Search every possible match

$$
\begin{bmatrix}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\
-1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\
-1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1
\end{bmatrix}
\otimes
\begin{bmatrix}
1 & -1 & -1 \\
-1 & 1 & -1 \\
-1 & -1 & 1
\end{bmatrix}
=
\begin{bmatrix}
0.77 & -0.11 & 0.11 & 0.33 & 0.55 & -0.11 & 0.33 \\
-0.11 & 1.00 & -0.11 & 0.33 & -0.11 & 0.11 & -0.11 \\
0.11 & -0.11 & 1.00 & -0.33 & 0.11 & -0.11 & 0.55 \\
0.33 & 0.33 & -0.33 & 0.55 & -0.33 & 0.33 & 0.33 \\
0.55 & -0.11 & 0.11 & -0.33 & 1.00 & -0.11 & 0.11 \\
-0.11 & 0.11 & -0.11 & 0.33 & -0.11 & 1.00 & -0.11 \\
0.33 & -0.11 & 0.55 & 0.33 & 0.11 & -0.11 & 0.77
\end{bmatrix}
$$

Convolution Layer

- Using Multiple Filters: One image becomes a stack of features (filtered images)



-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	-1	-1
-1	1	-1
-1	-1	1

1	-1	1
-1	1	-1
1	-1	1

-1	-1	1
-1	1	-1
1	-1	-1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

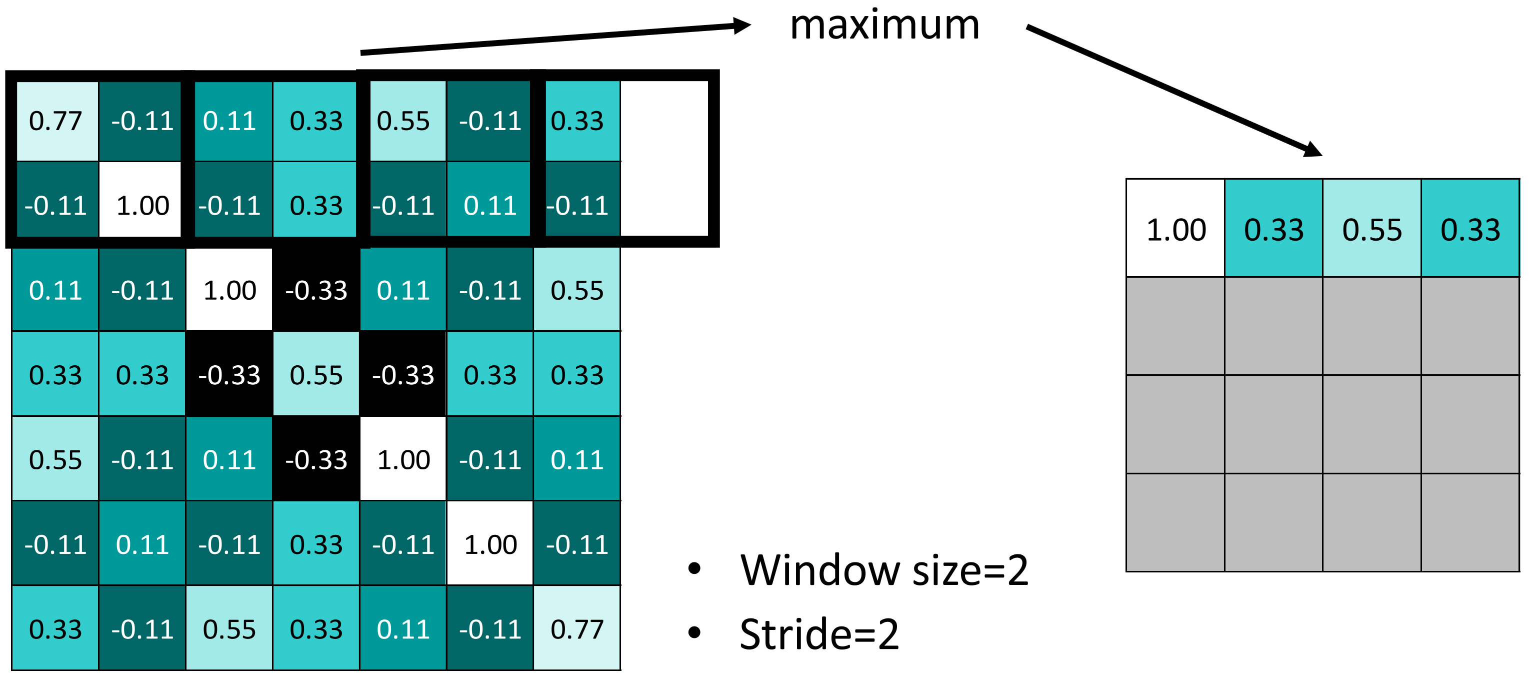
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

Pooling: Shrinking the image stack

1. Pick a window size (usually 2 or 3).
2. Pick a stride (usually the same window size, so there are no overlaps between windows).
3. Walk your window across your filtered images.
4. From each window, take the maximum (or average) value.

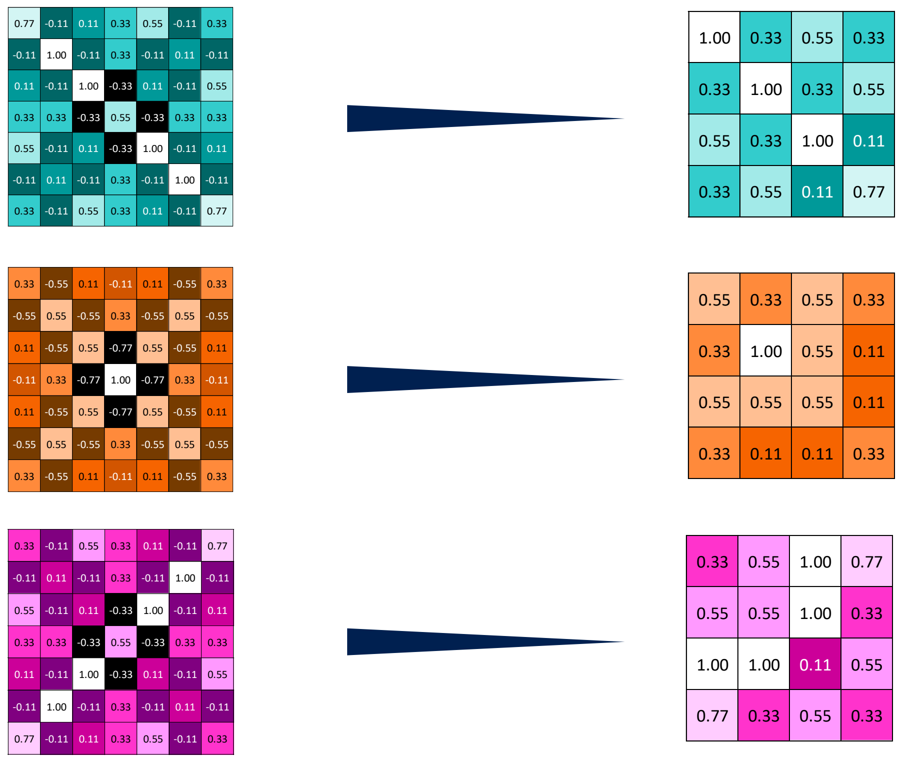
Max Pooling



- Window size=2
- Stride=2

Max Pooling

A stack of images becomes a stack of smaller images.



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

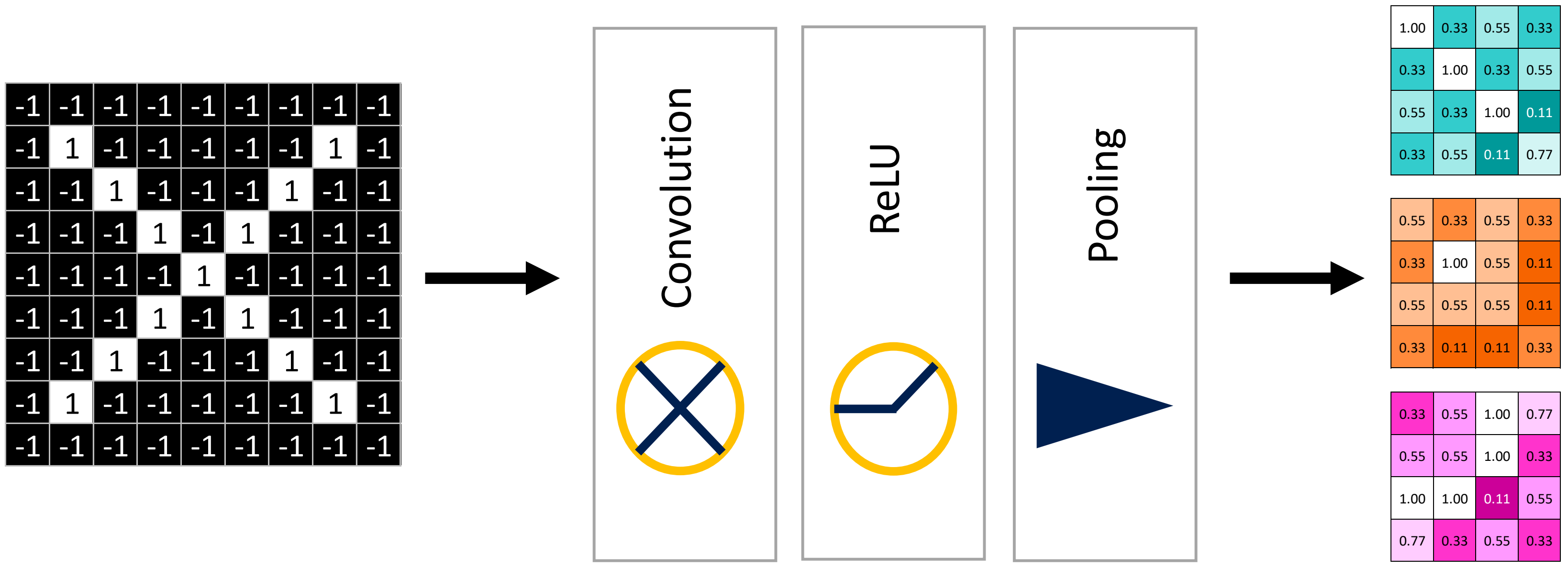
0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

Rectified Linear Units (ReLUs)

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

Layers get stacked



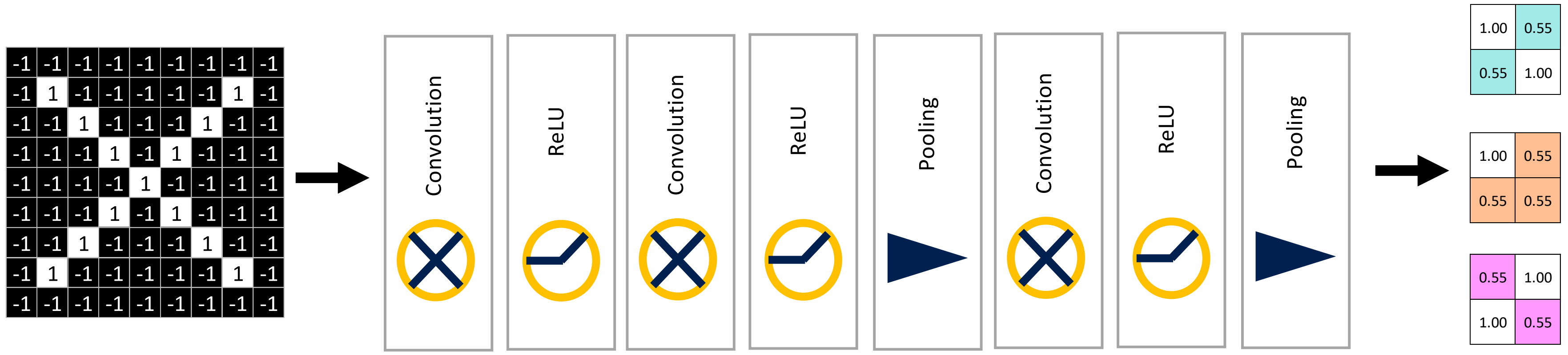
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

Layers get stacked



-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

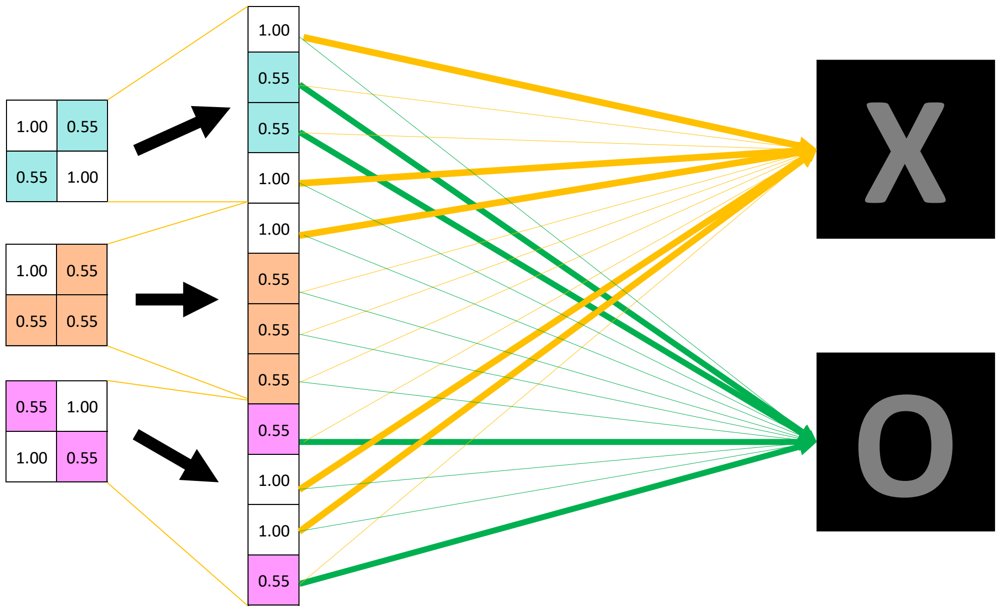
1.00	0.55
0.55	1.00

1.00	0.55
0.55	0.55

0.55	1.00
1.00	0.55

Classification with a Fully Connected Layer

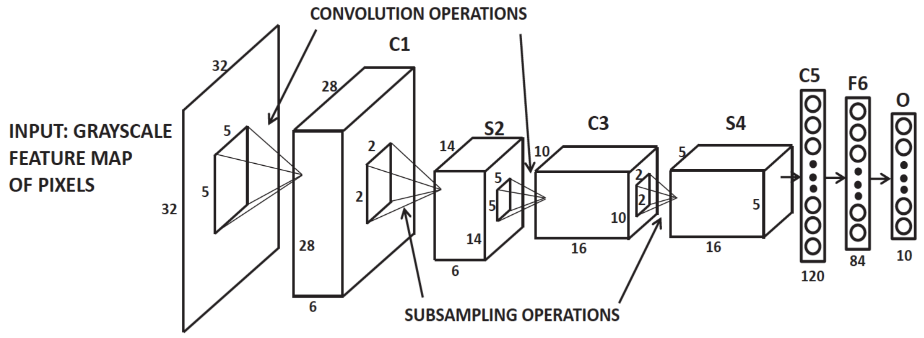
- Stack of images obtained with multiple filters can be the input of FC layer



Creating Convolutional Neural Networks

- Interleaving of convolution, pooling, and ReLU layers
- ReLU often follows the convolutional layers
- Max-pooling is often applied after convolution-ReLU combinations
- LeNet-5 (convolutional network for hand-written digits classification)
    - 7 layers, 3 convolutional layers (C1, C3, C5), 2 sub-sampling (pooling) layers (S2 and S4), 1 fully connected layer (F6), and an output (O) layer
    - Did not use ReLU activation functions
    - Used (scaled) hyperbolic tangent as activation function in hidden layers
    - C5 is basically a fully connected layer as each unit is mapped to all the 16 input activation maps and filter size is same as input maps
    - Pattern: Conv→Pool→Conv→ Pool→FullyConn→ FullConn→ Output

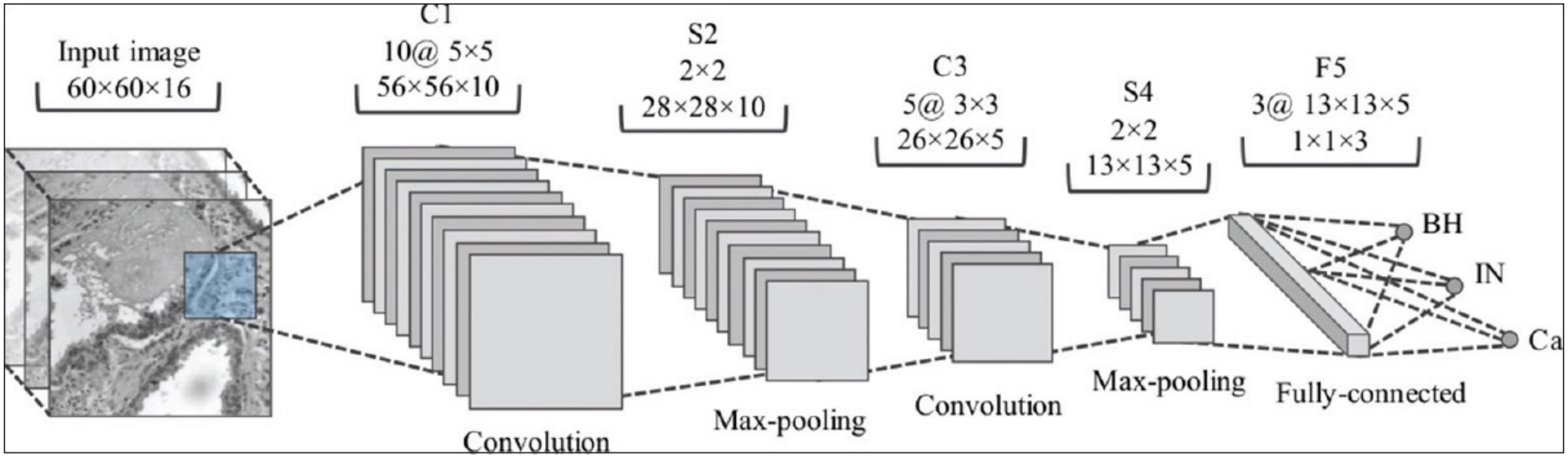
LeNet-5



CNN for MNIST classification

- https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/

CNN Application Example



- Classifications of multispectral colorectal cancer tissues using convolution neural network, **J Pathol Inform** 2017, **8**:1