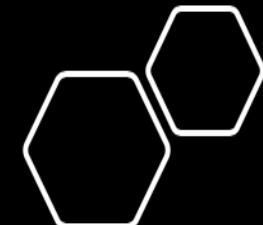


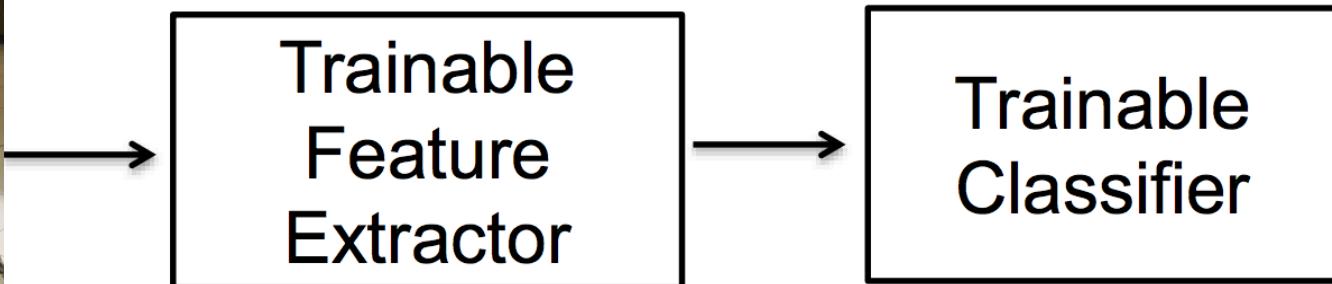
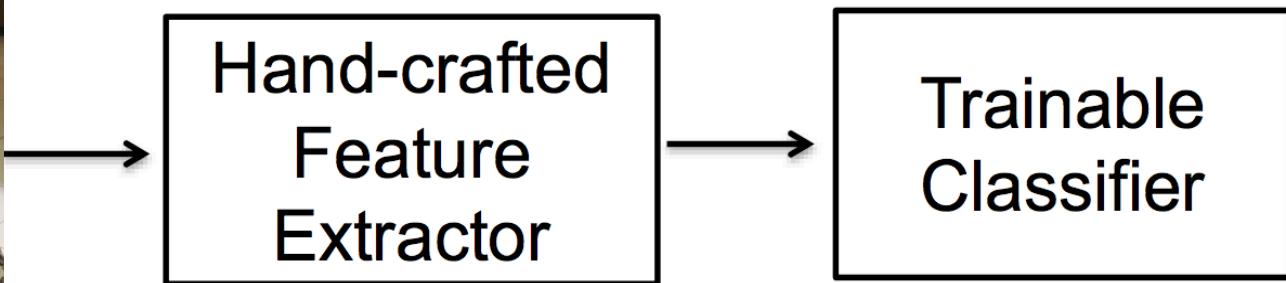
AMMI Review sessions

Deep Learning (4)

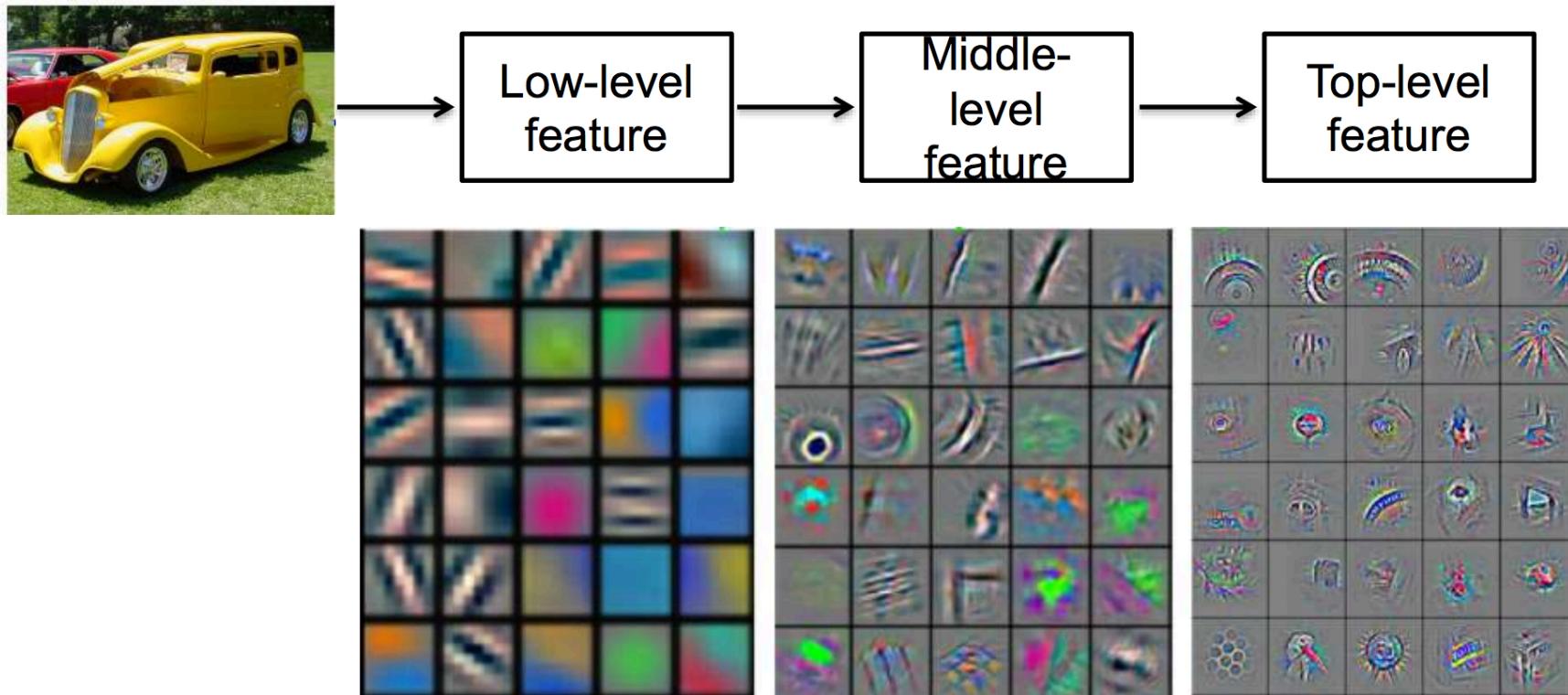
Introduction to Convolutional Neural Networks (CNNs) |



Traditional pattern recognition

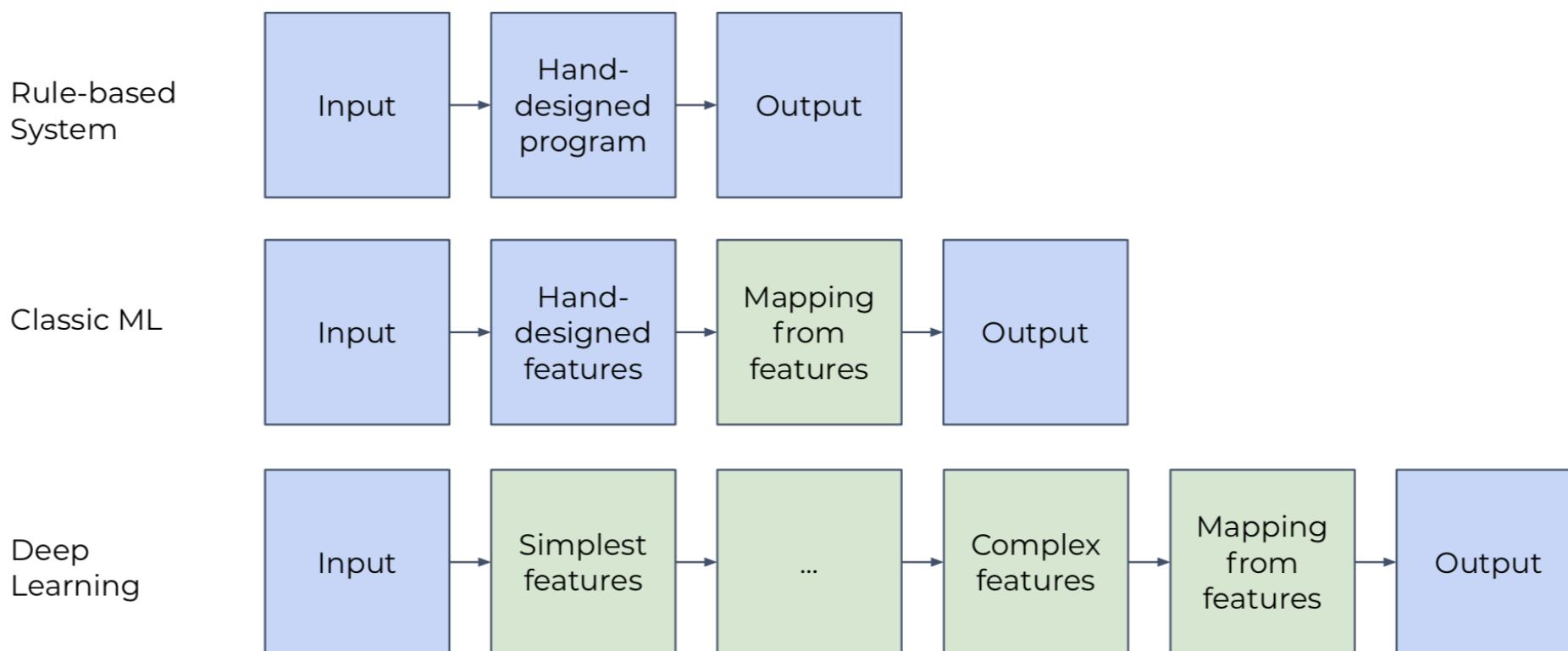


Learning feature representations



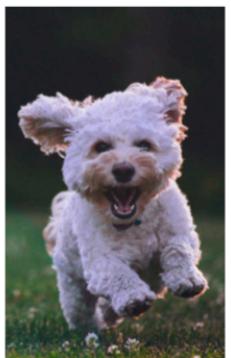
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Learning feature representations



What are good representations

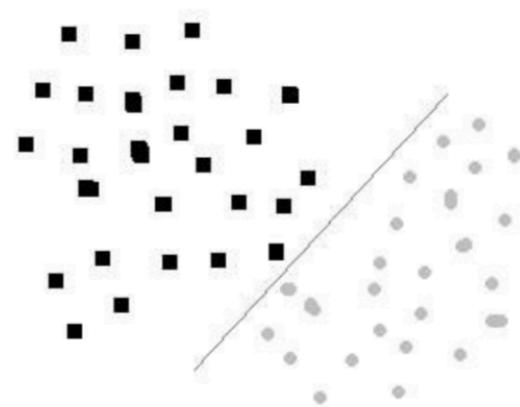
Source: Edgar Edgar, Unsplash



Input



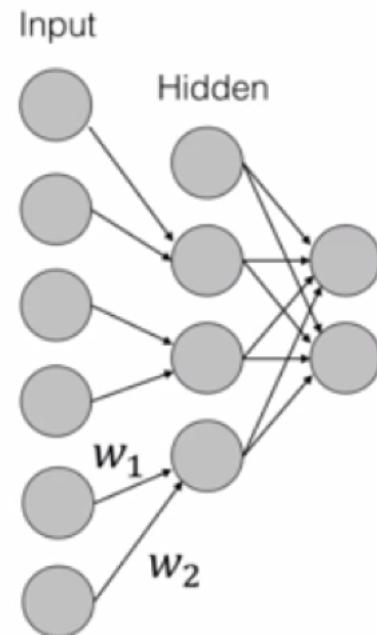
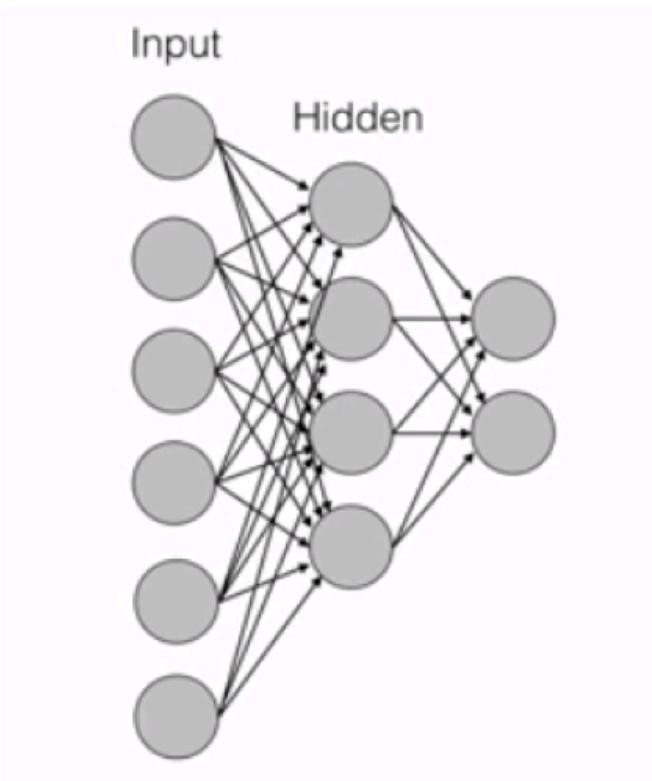
Source: Wikimedia, commons



Representations that
are linearly separable

Source: Joe Caione, Unsplash

Why convolution



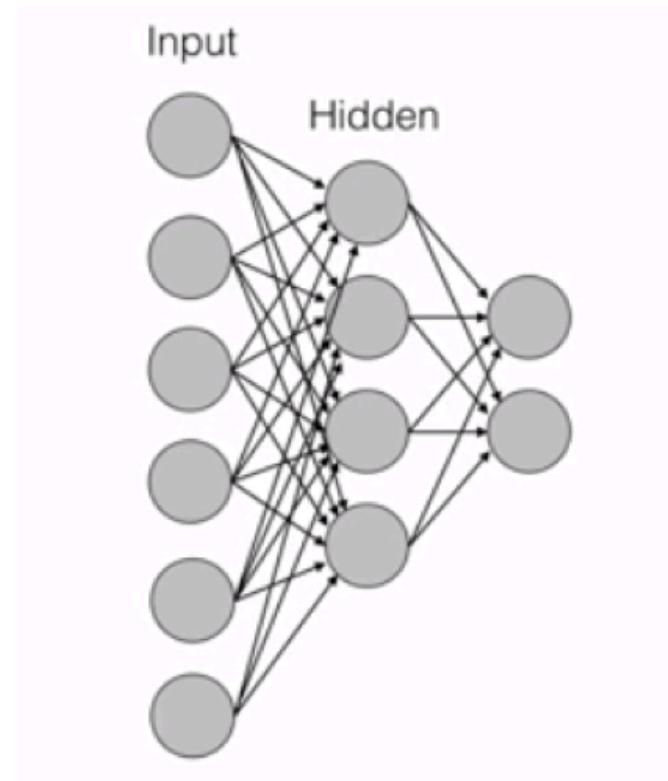
Why convolution

Too many parameters, unscalable to real images

Suppose the image has 256x256 RGB pixels,
and we have 1,000 classes.

How many parameters would a single layer have?

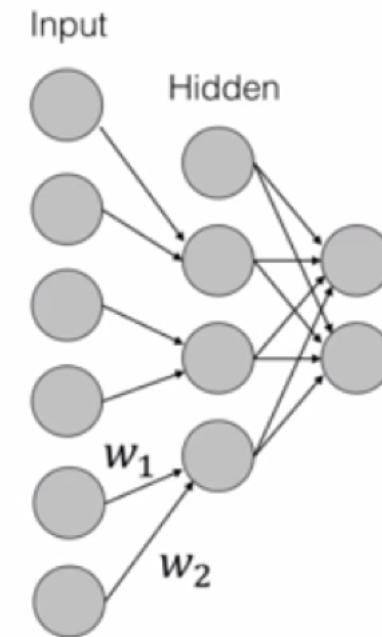
$$3 \times 256 \times 256 \times 1,000 = 196,608,000$$



Spatial features

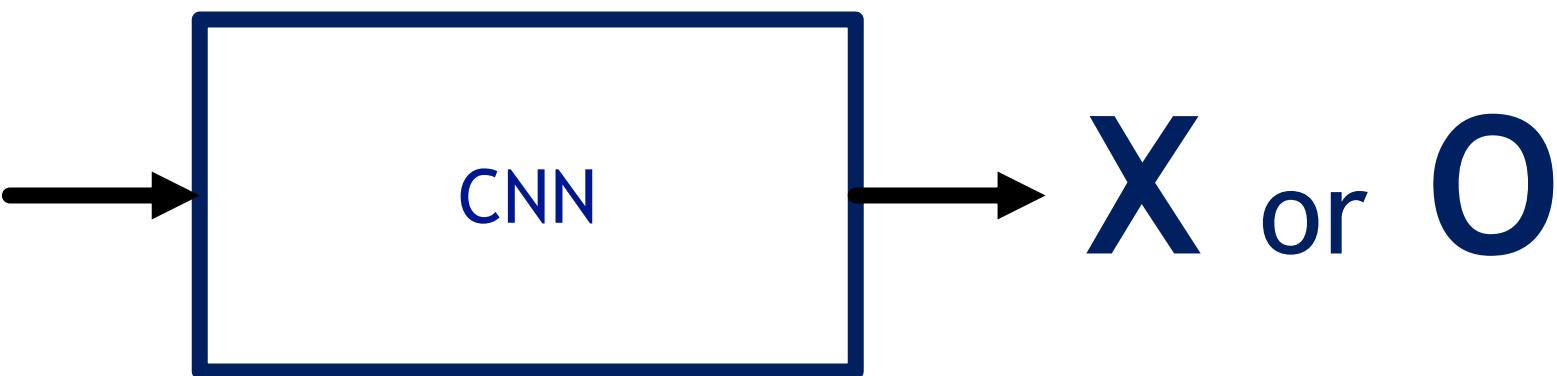
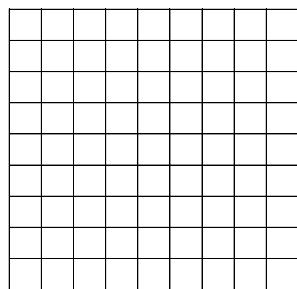


VS

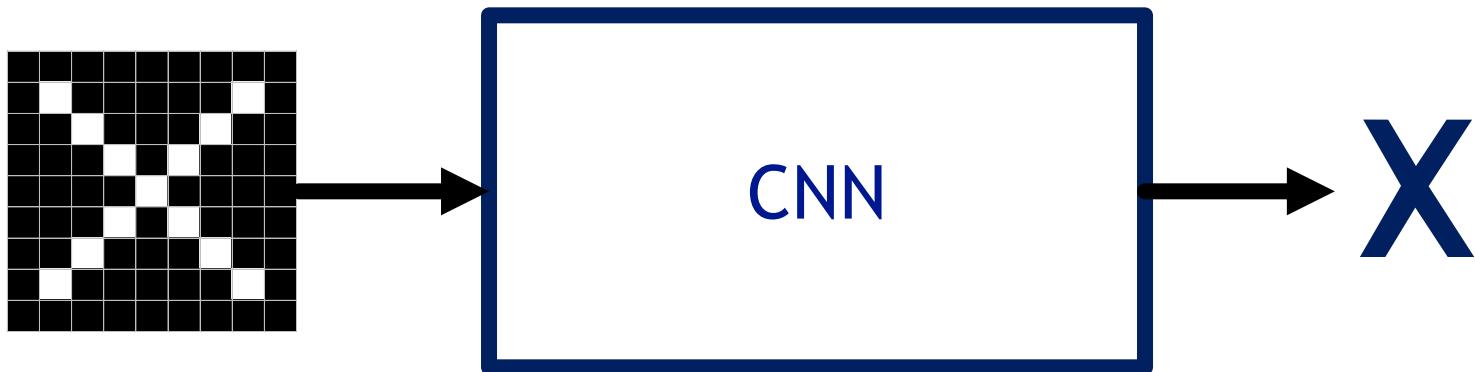


Why convolution

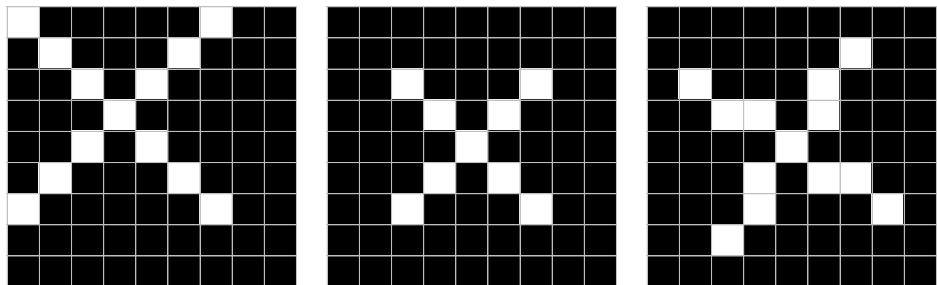
A two-dimensional
array of pixels



Why convolution



Trickier cases

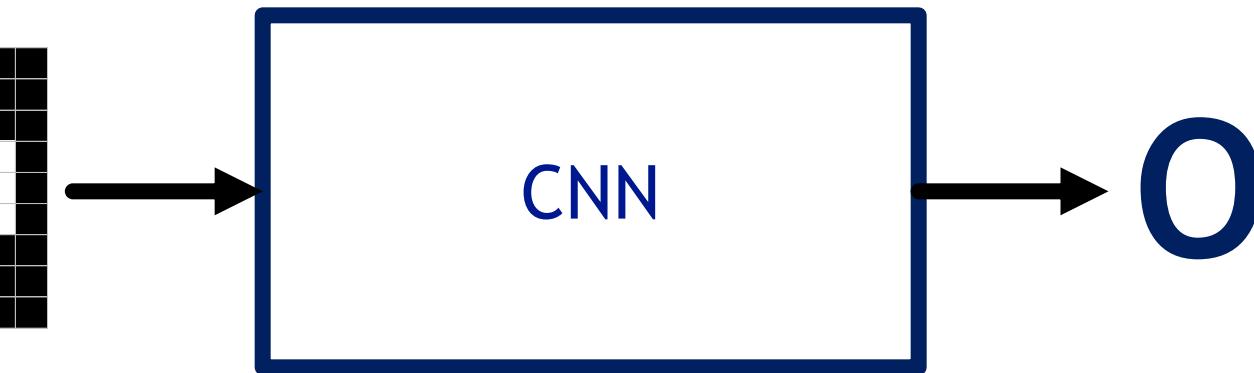
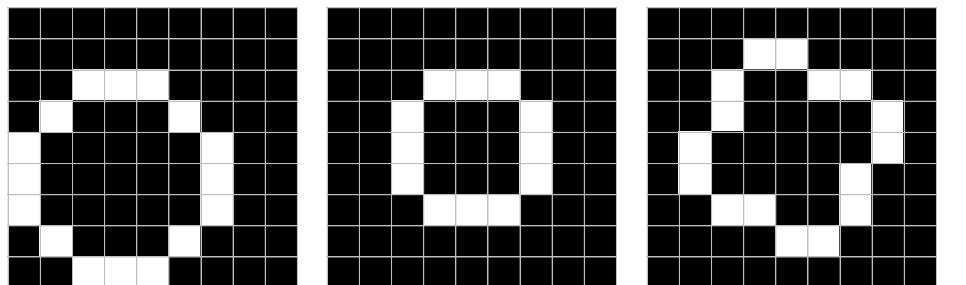


translation

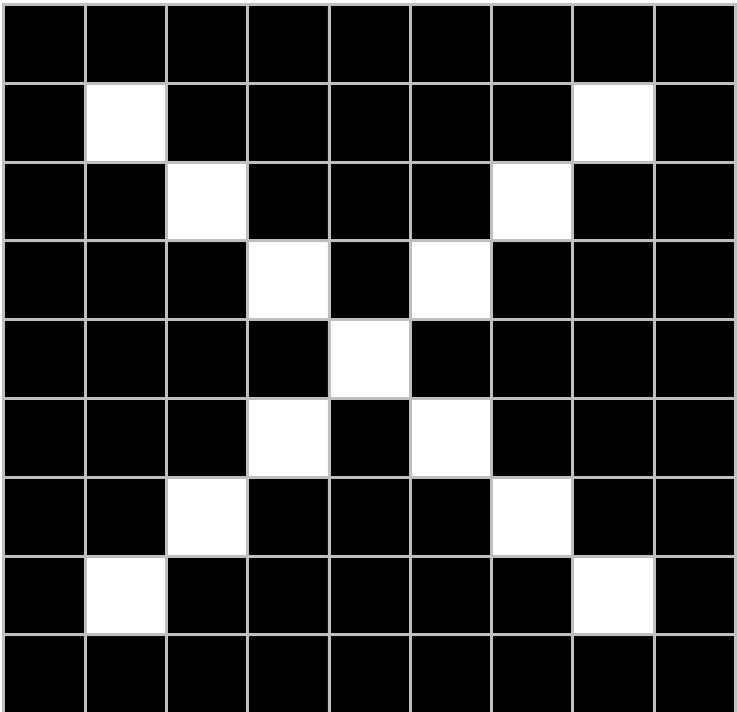
scaling

rotation

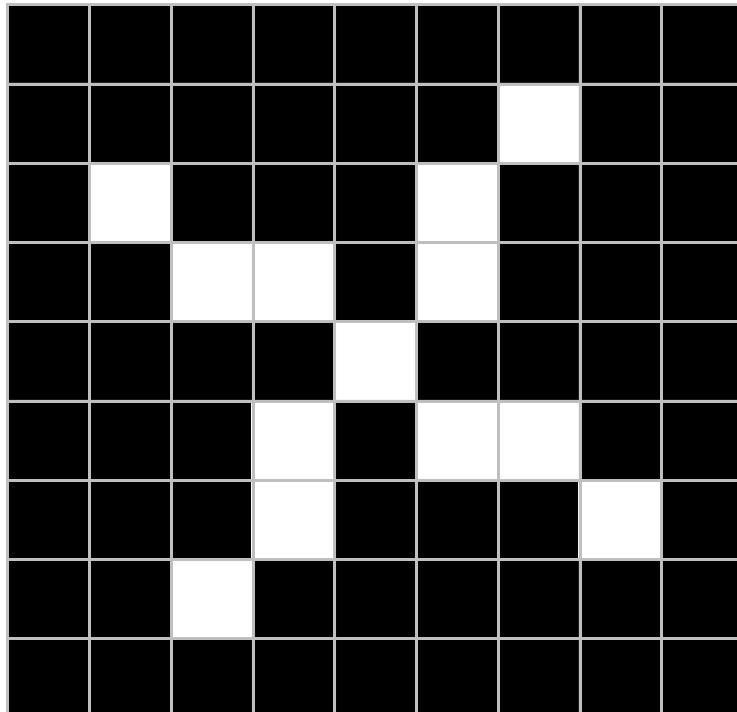
weight



Deciding is hard ?



?



Computers are literals



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

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

What computer see

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	X	-1	-1	-1	-1	X	X	-1	
-1	X	X	-1	-1	X	X	-1	-1	
-1	-1	X	1	-1	1	-1	-1	-1	
-1	-1	-1	-1	1	-1	-1	-1	-1	
-1	-1	-1	1	-1	1	X	-1	-1	
-1	-1	X	X	-1	-1	X	X	-1	
-1	X	X	-1	-1	-1	-1	X	-1	
-1	-1	-1	-1	-1	-1	-1	-1	-1	

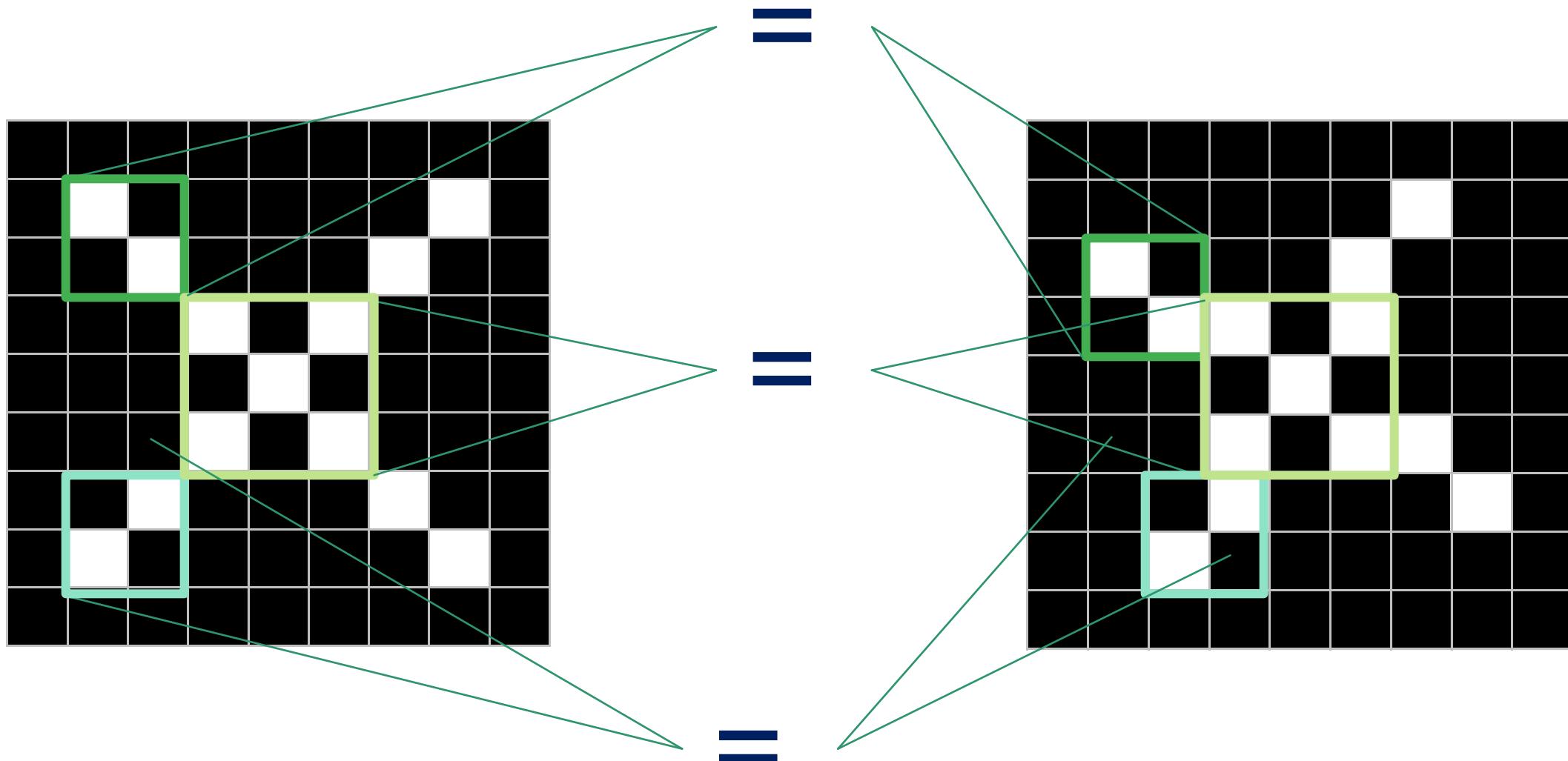
Computers are literals

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



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

Our target is to match image pieces!



Concept of features

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

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

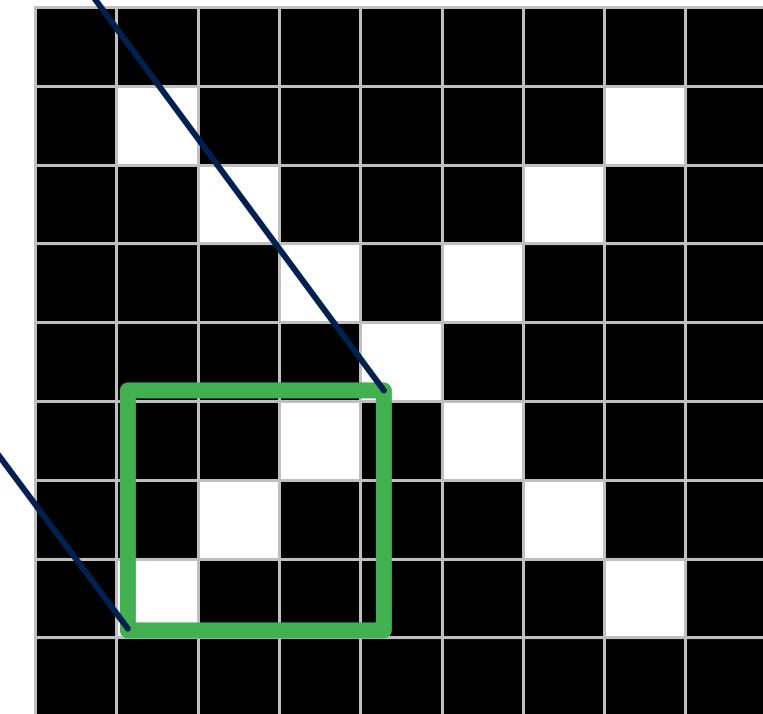
-1	-1	1
-1	1	-1
1	-1	-1

Concept of features

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

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

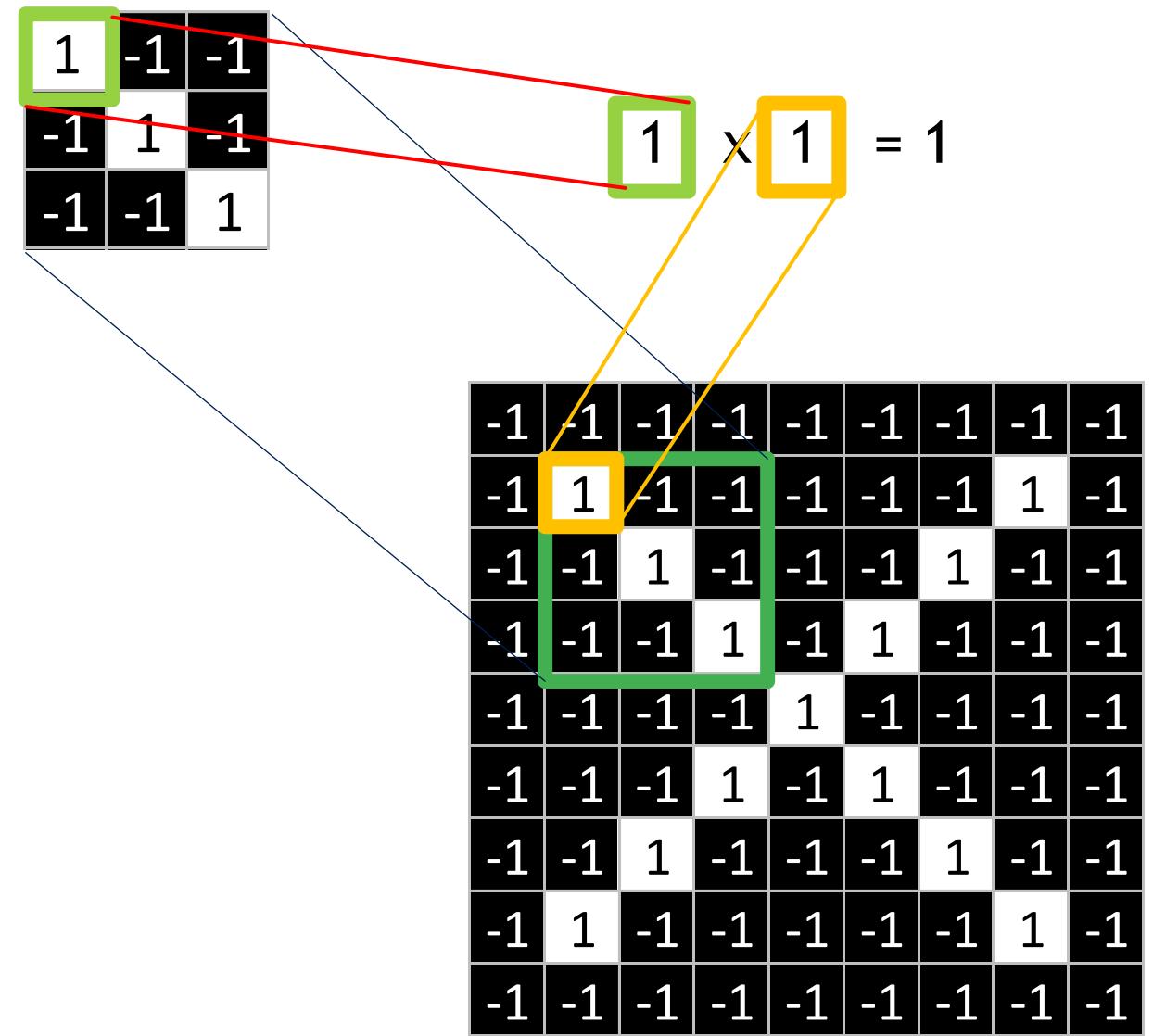
-1	-1	1
-1	1	-1
1	-1	-1



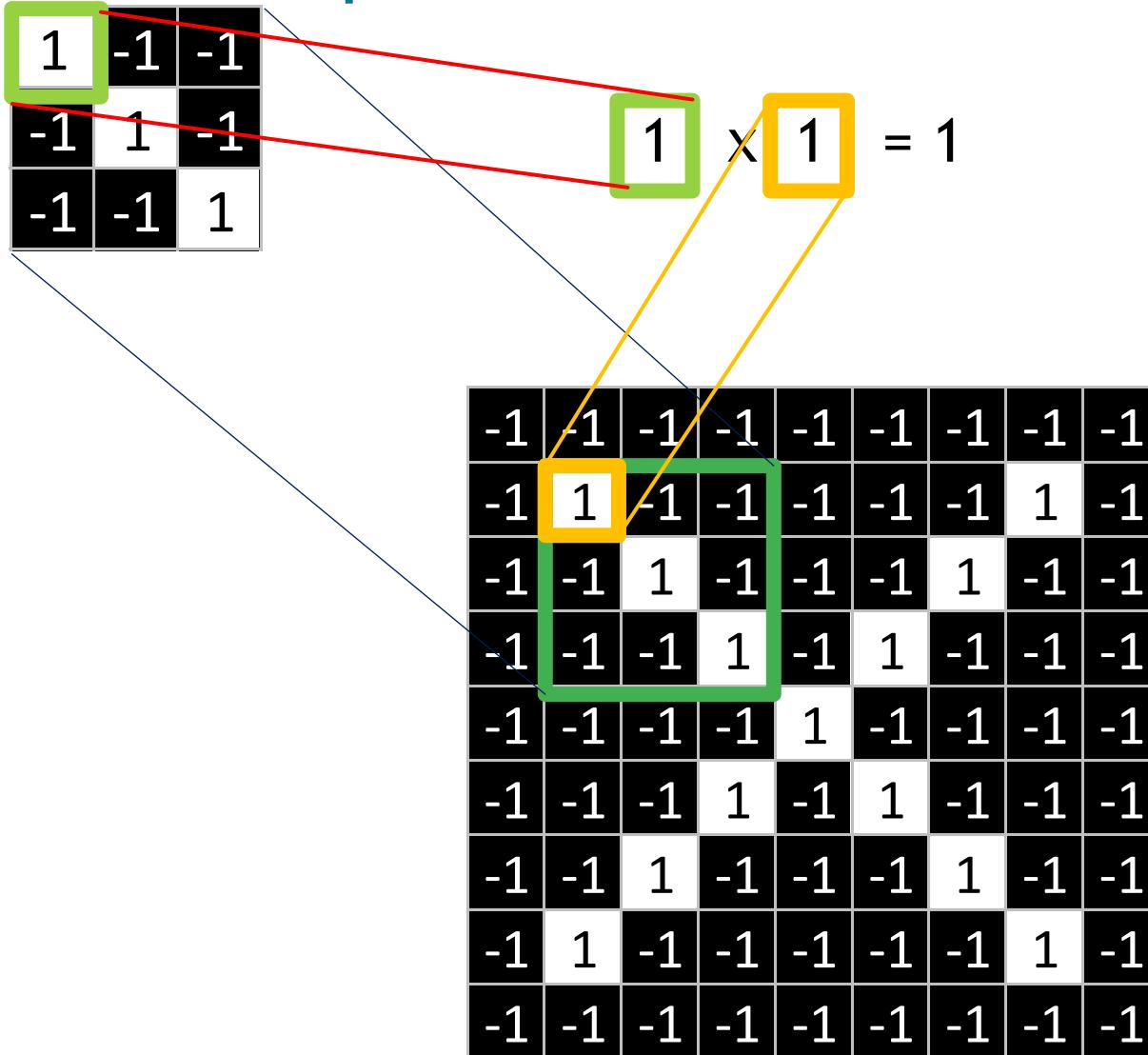
Filtering: The math behind the match

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.

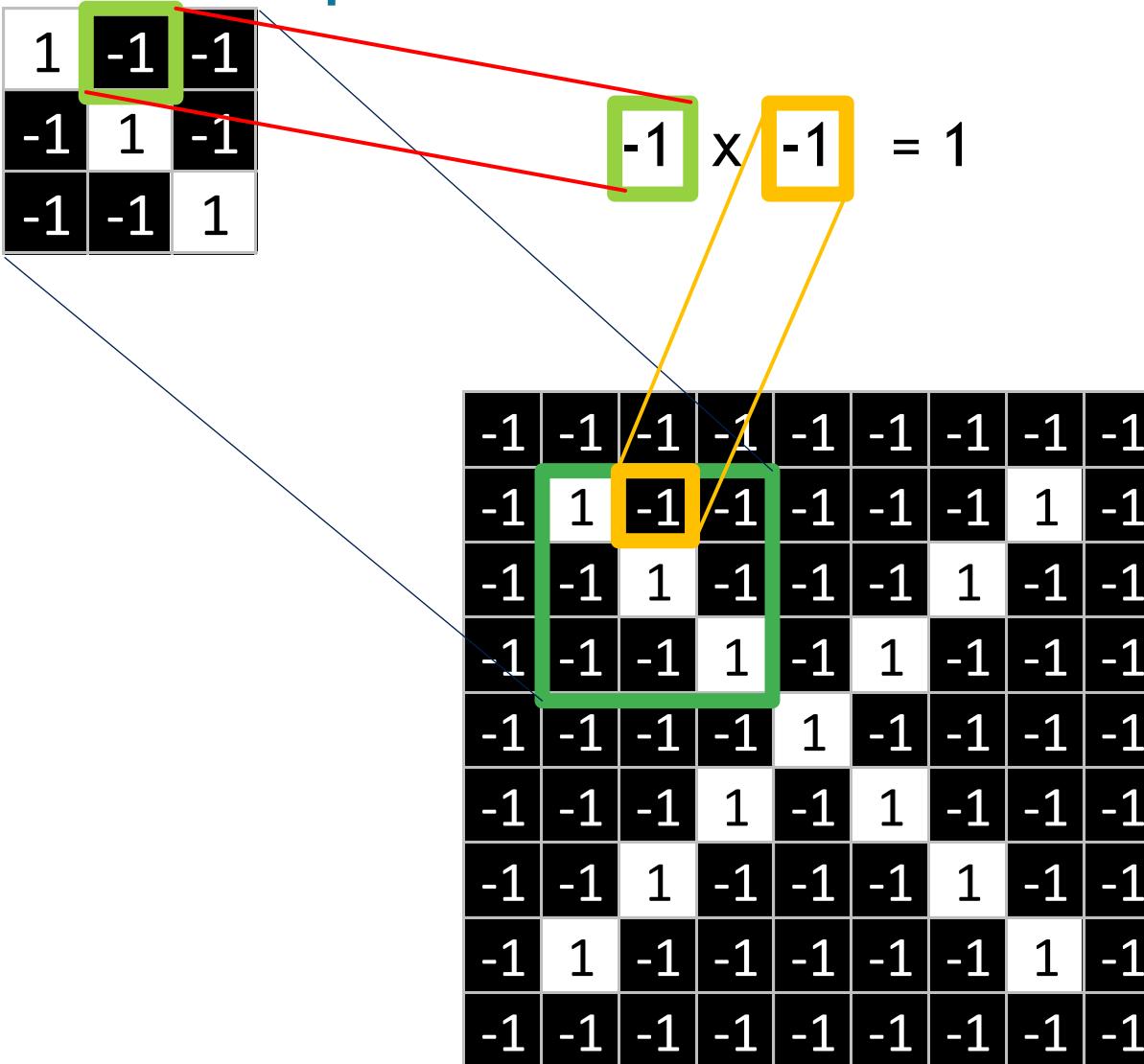
Convolution process



Convolution process



Convolution process



Convolution process

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

1	1	1
1	1	1
1	1	1

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

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

Convolution process

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

1	1	-1
1	1	1
-1	1	1

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

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

1								

Convolution process

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

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

[Demo 0](#)

Multi-features

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

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



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

=

0.33	-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

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



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

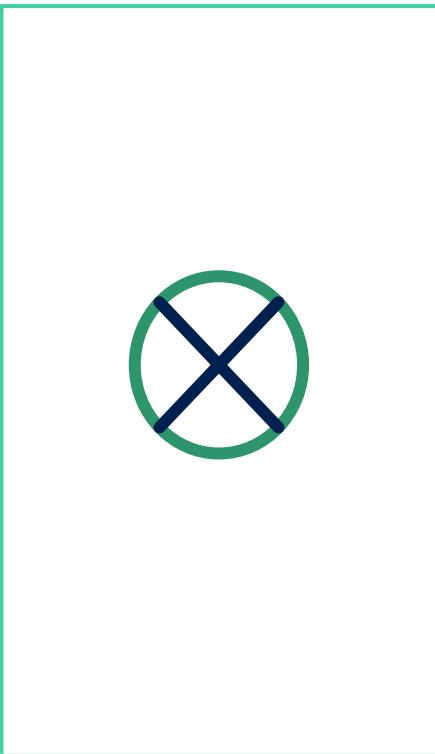
=

0.33	-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

Convolution layer

- One image becomes a stack of 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



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

Outline

1 Why Deep Learning ?

2 Why Convolution ?

3 Convolutional Networks

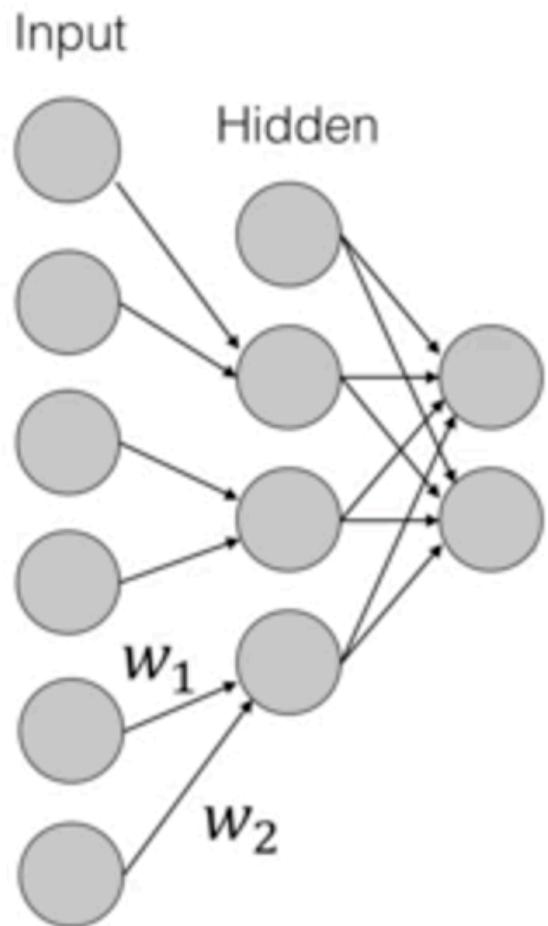
4 Applications

5 Popular architectures

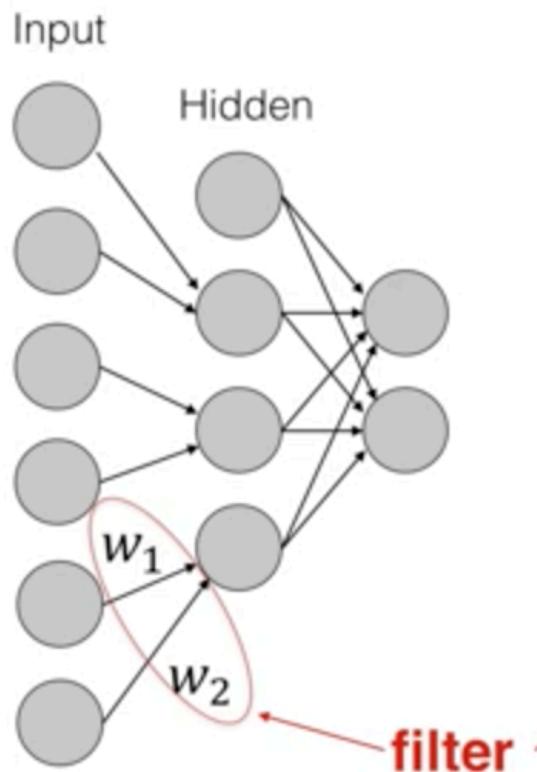
6 Demo

Q: So How we learn these filters using NN ?

Filter weights → NN weights



Filter weights → NN weights

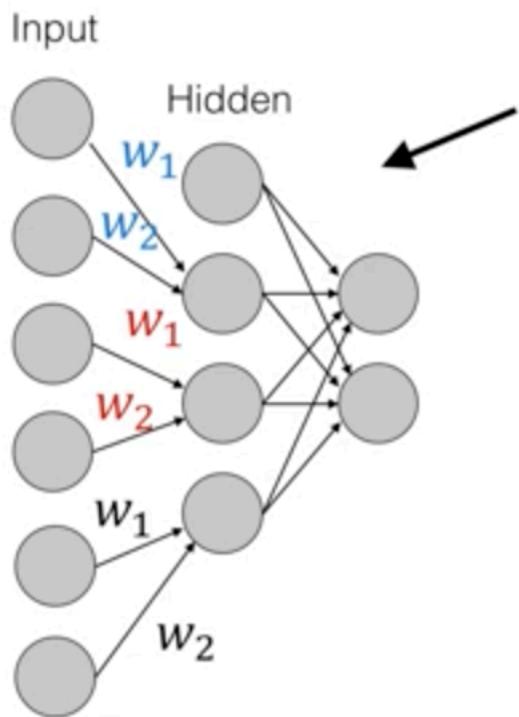


$$y = w_1x_1 + w_2x_2$$

If $(w_1, w_2) = (1, -1)$: $y = x_1 - x_2$

y maximal when $(x_1, x_2) = (1, 0)$

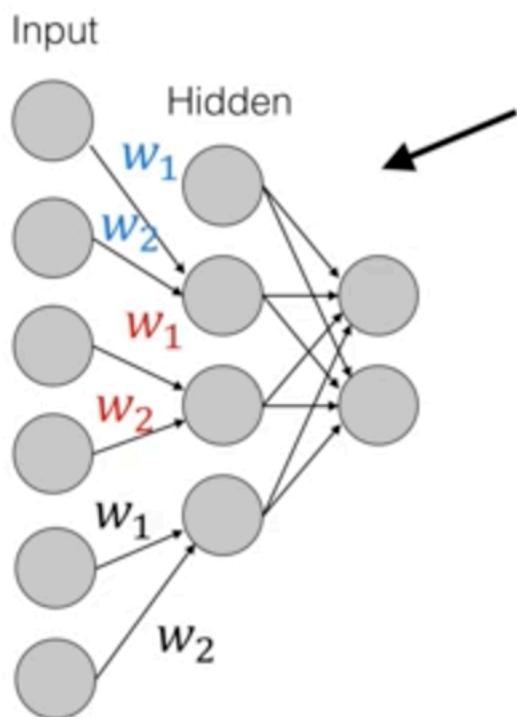
Filter weights → NN weights



1-d convolution with

- filters: 1
- filter size: 2
- stride: 2

Filter weights → NN weights

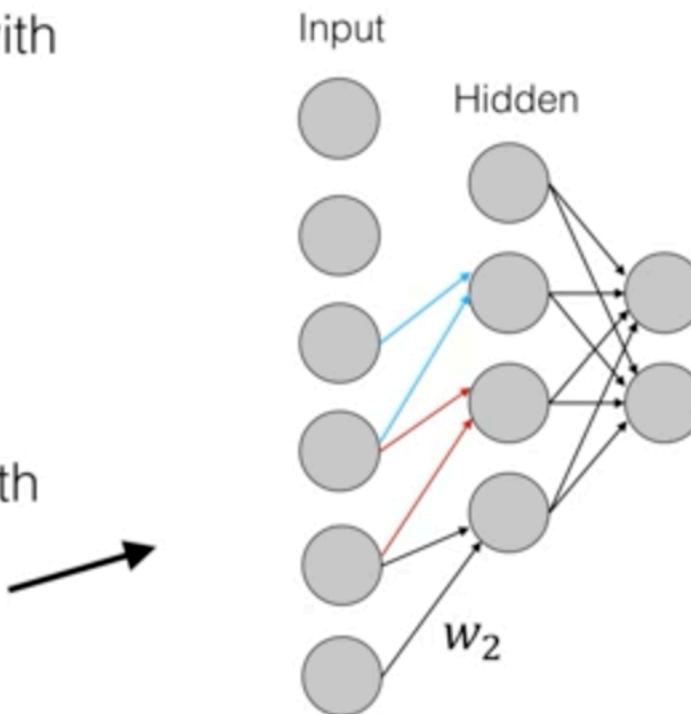


1-d convolution with

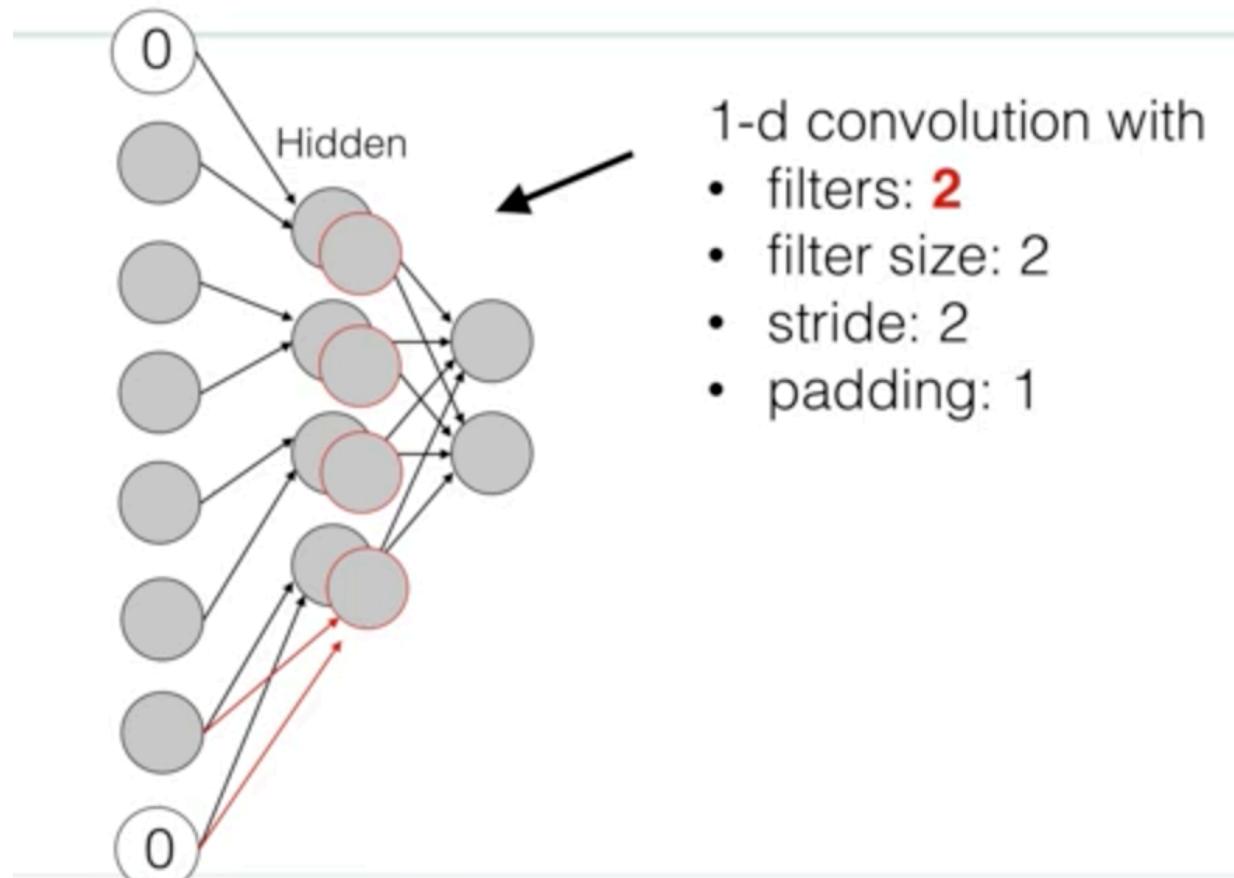
- filters: 1
- filter size: 2
- stride: 2

1-d convolution with

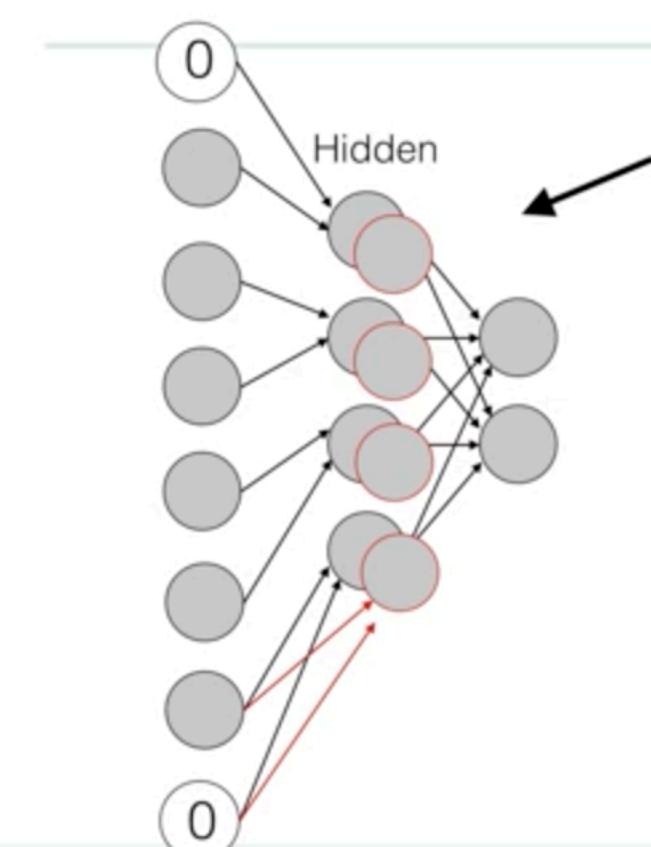
- filters: 1
- filter size: 2
- stride: 1



Filter weights → NN weights

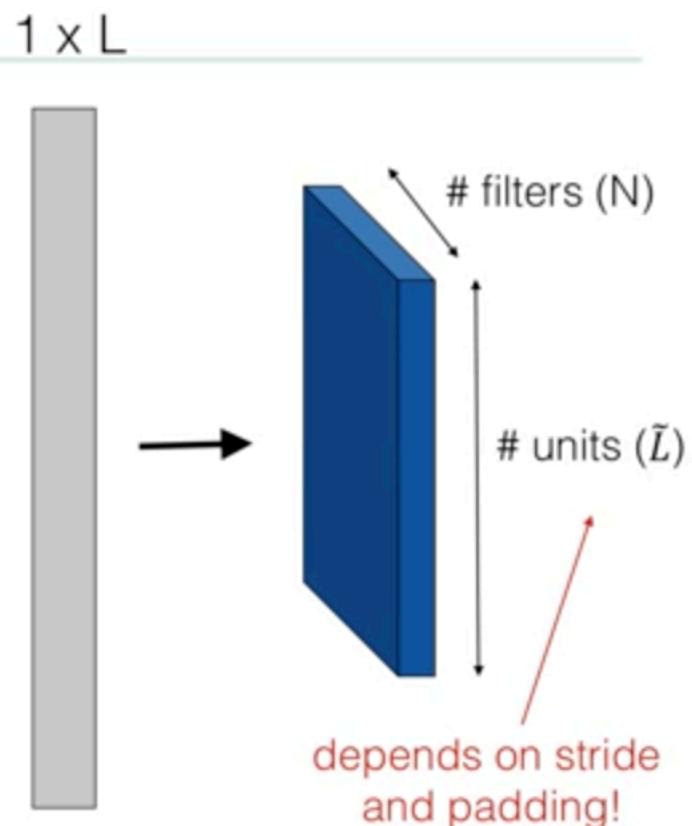


1-D convolution

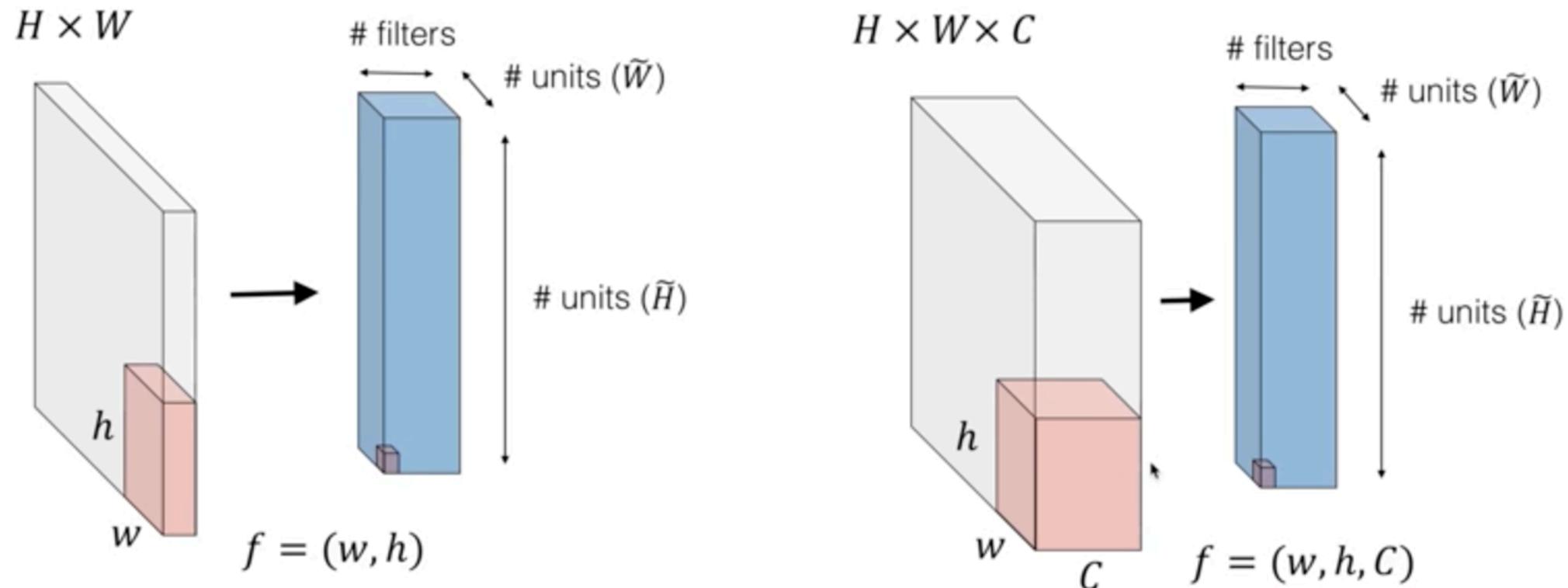


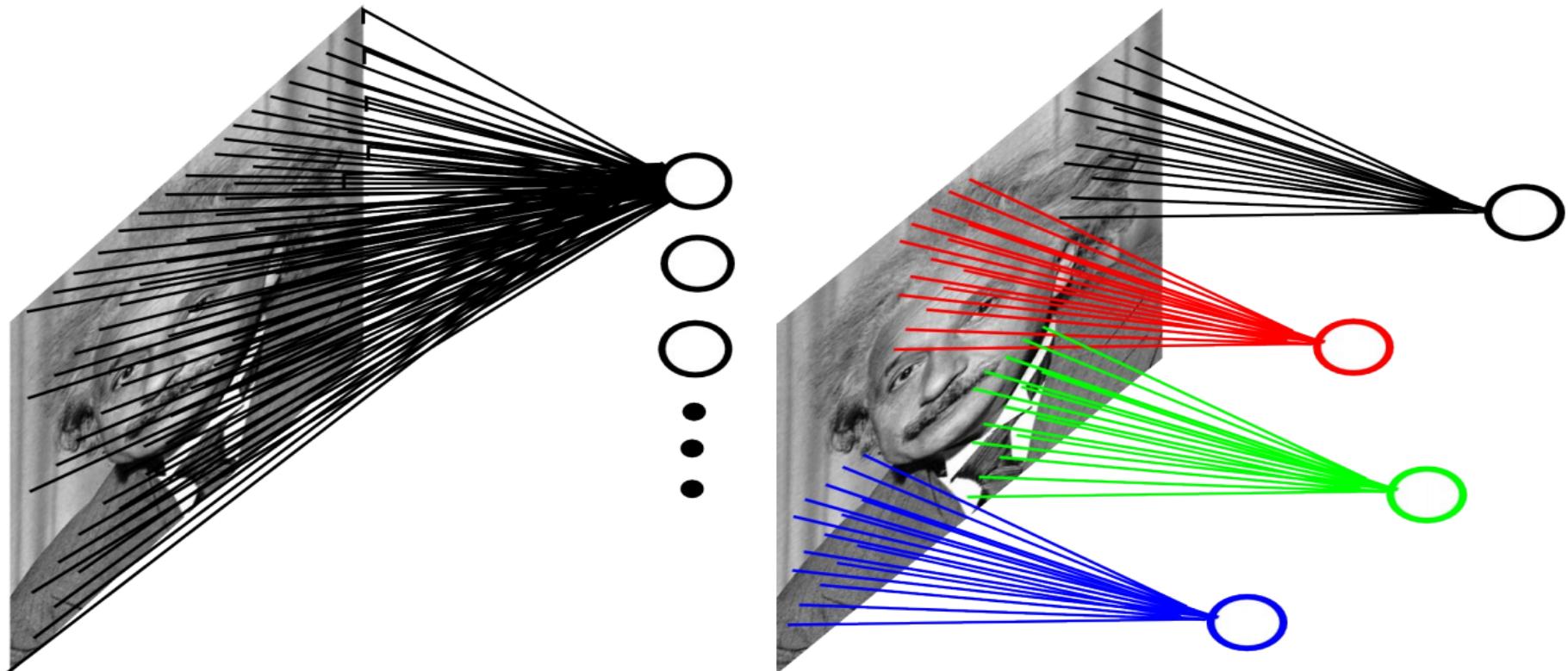
1-d convolution with

- filters: **2**
- filter size: 2
- stride: 2
- padding: 1



2-D convolution





Pooling

1. Pick a window size (usually 2 or 3).
2. Pick a stride (usually 2).
3. Walk your window across your filtered images.
4. From each window, take the maximum value.

Pooling

maximum

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			

Pooling

maximum

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		

Pooling

maximum

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		

Pooling

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	

maximum

1.00	0.33	0.55	0.33

Pooling

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

max pooling

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

Pooling

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

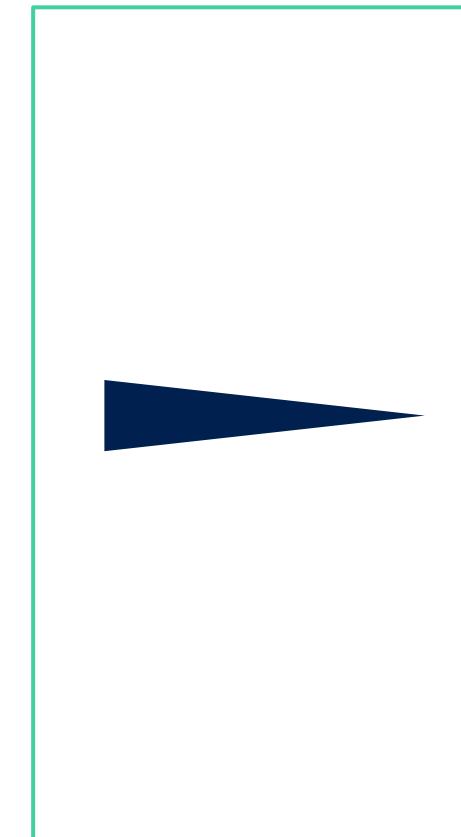
Pooling

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

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

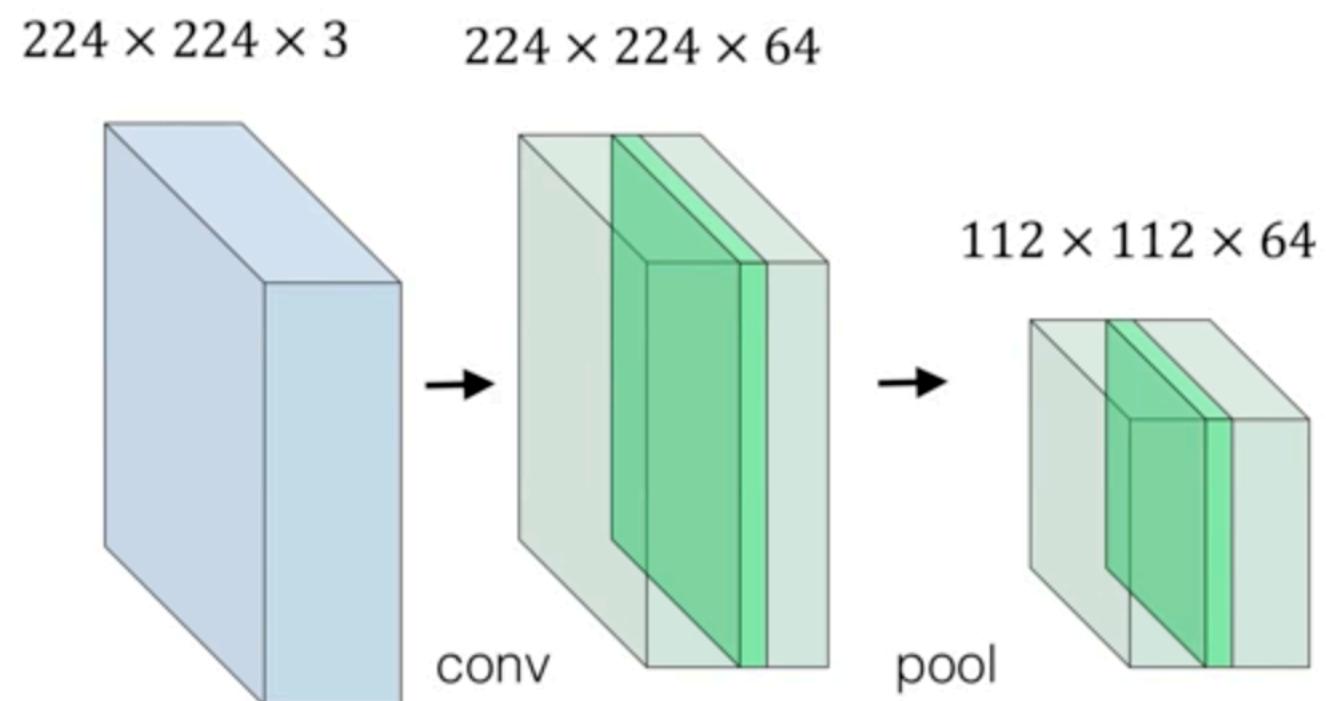


1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

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

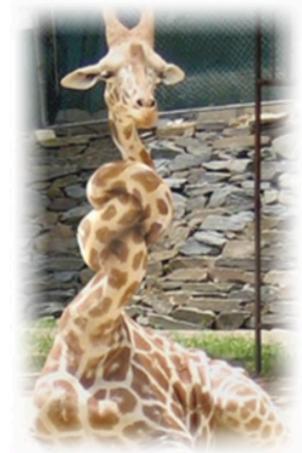
Pooling (sub-sampling)



Pooling - invariance



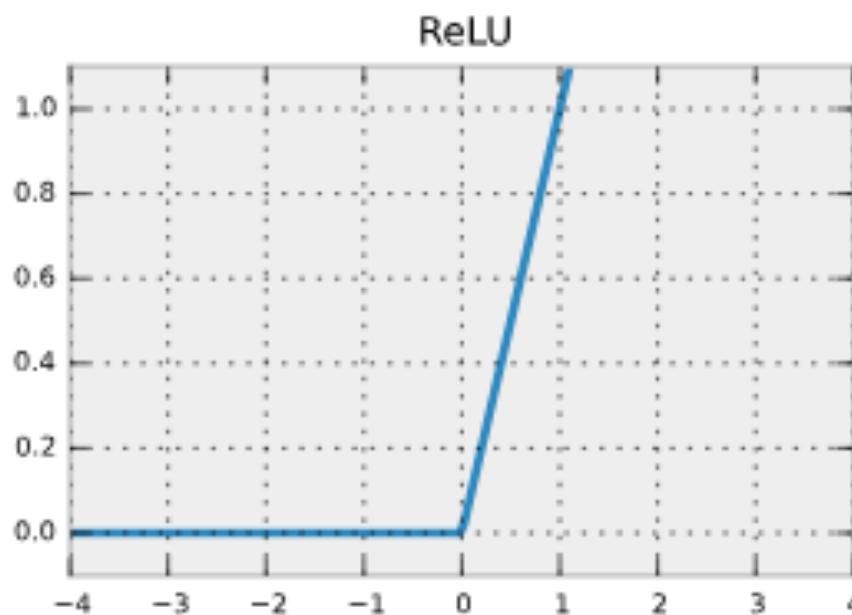
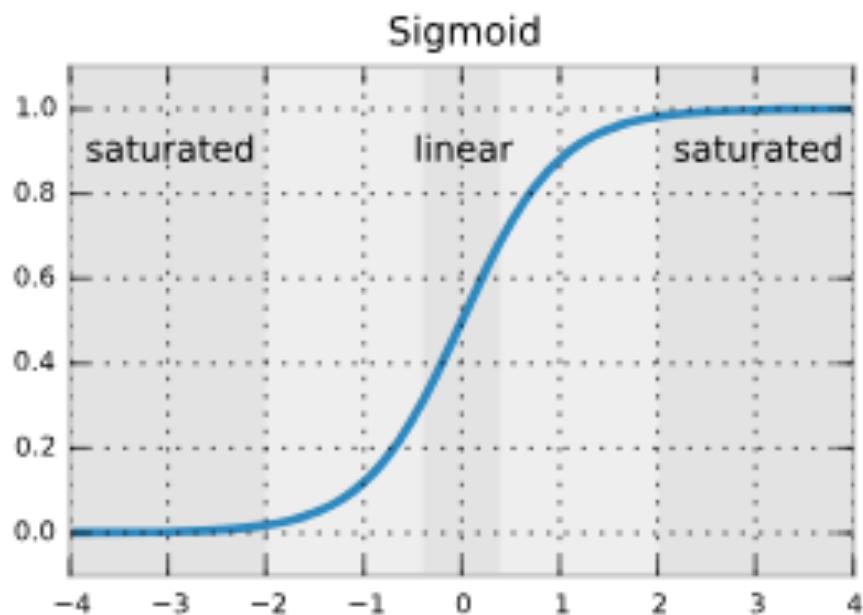
2



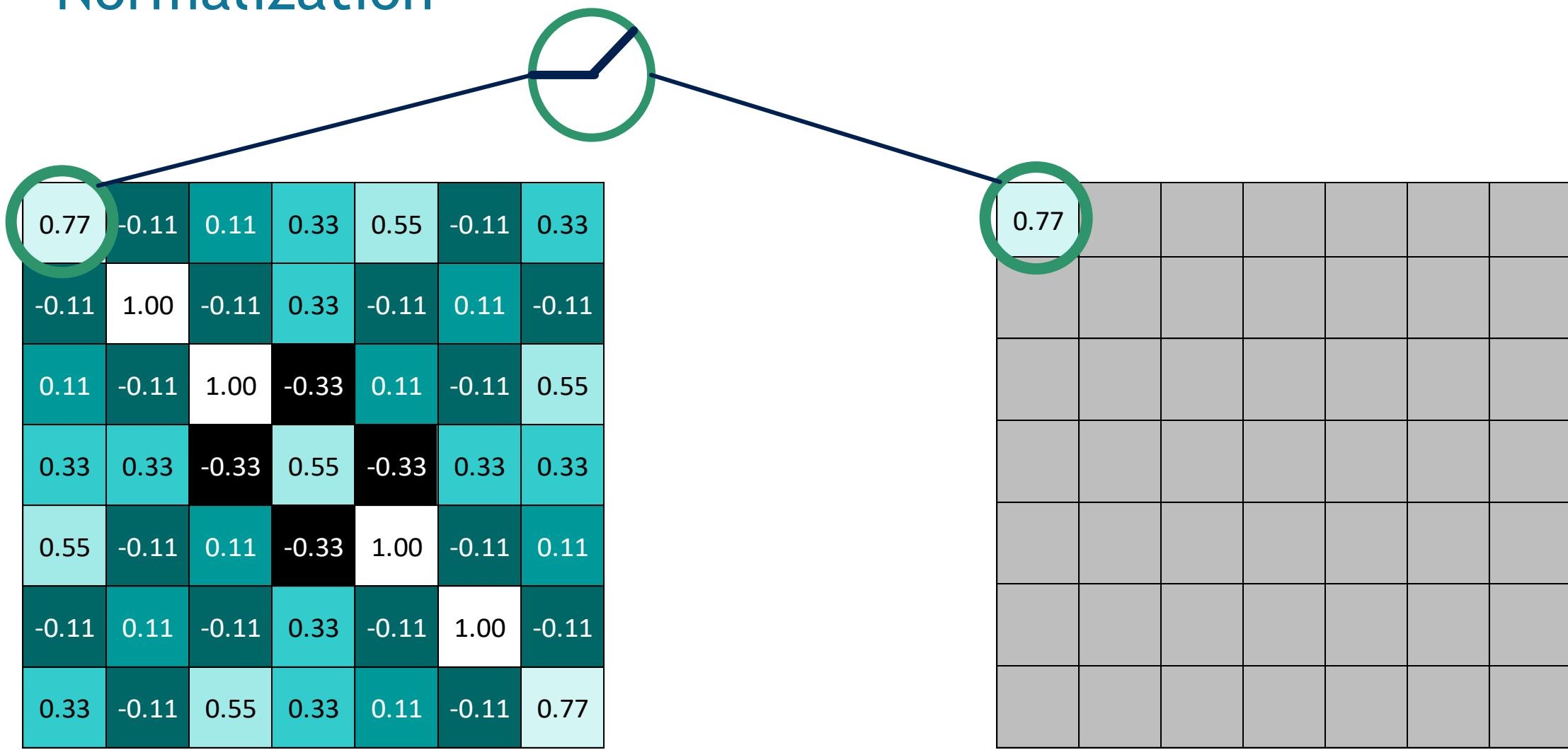
[Thomas Serre 2012]

Normalization

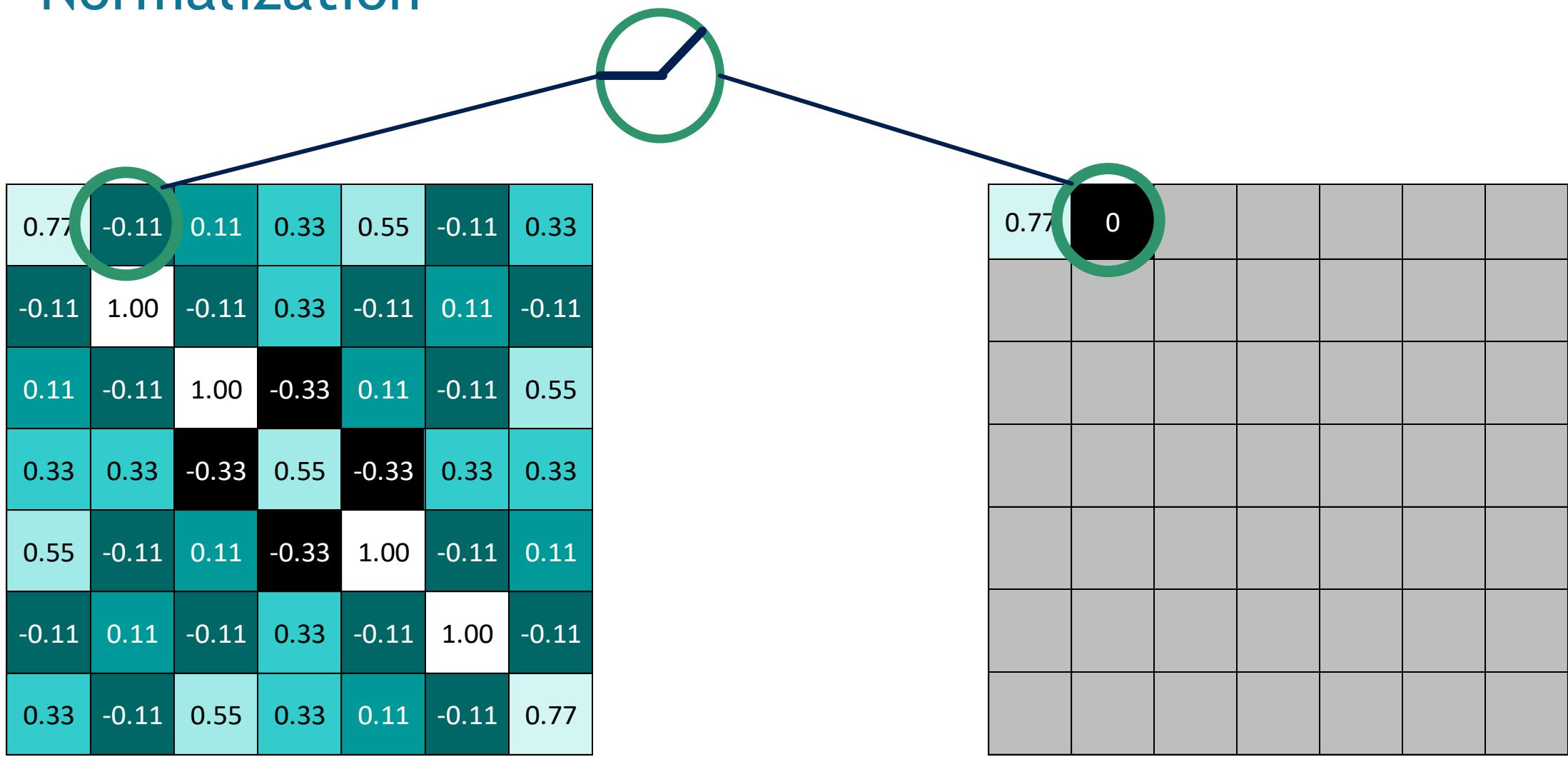
Normalization



Normalization



Normalization

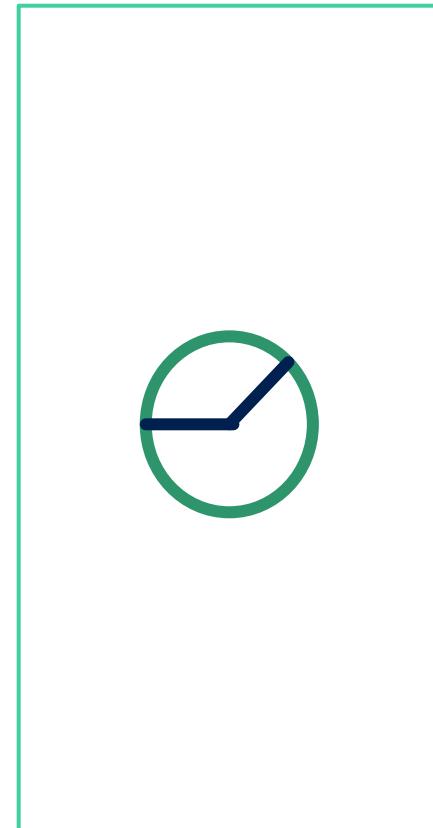


Normalization

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



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

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

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

Relu VS Sigmod

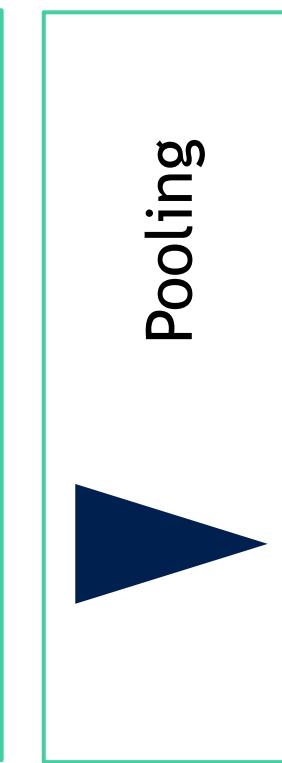
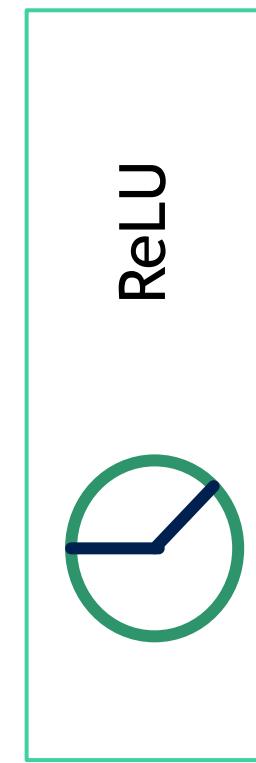
- Relu: Reduce the likelihood of the gradient to vanish
- Sigmoid: tend to vanish gradient
- Relu : tend to blow up activation (a is the output itself)
- Relu: More computationally efficient to compute than Sigmoid
- Relu : Dying Relu problem

[Demo 1](#)

[Demo 2](#)

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	



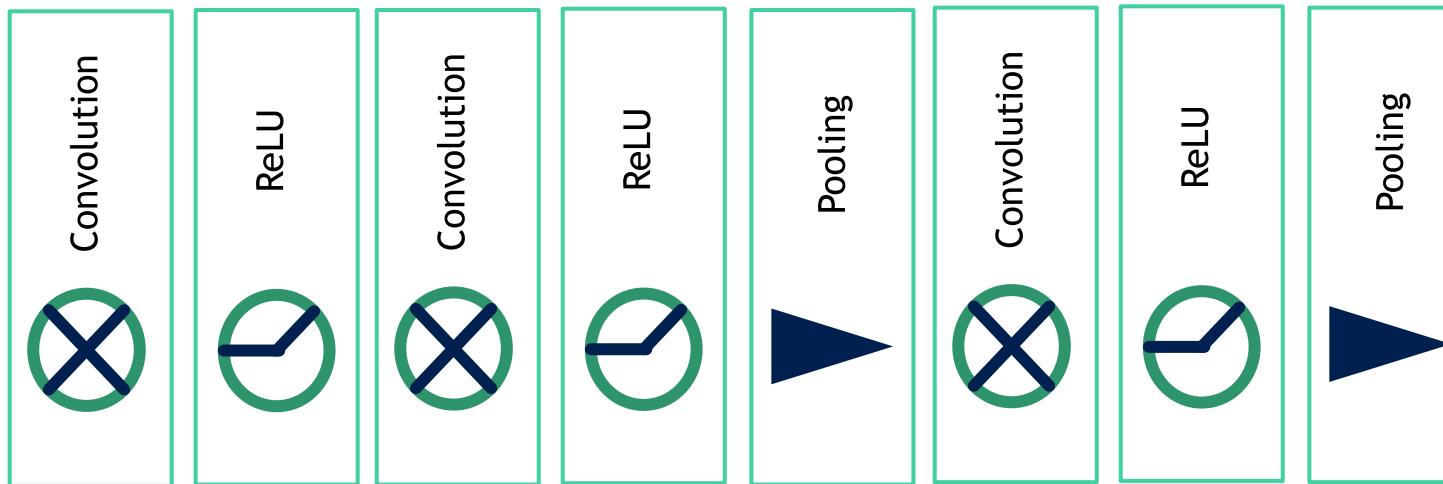
1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

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 deep stacked

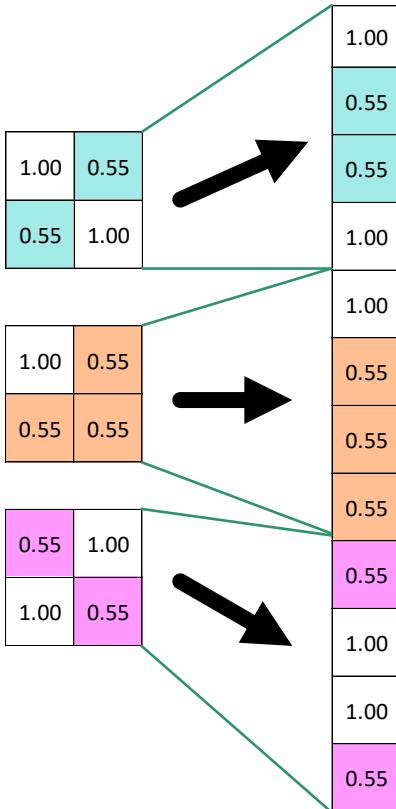
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	
-1	-1	1	-1	-1	-1	1	-1	-1	
-1	-1	-1	1	-1	1	-1	-1	-1	
-1	-1	-1	-1	1	-1	-1	-1	-1	
-1	-1	-1	-1	1	1	-1	-1	-1	
-1	-1	-1	1	-1	1	-1	-1	-1	
-1	-1	1	-1	-1	1	-1	-1	-1	
-1	1	-1	-1	-1	-1	-1	1	-1	
-1	-1	-1	-1	-1	-1	-1	-1	-1	



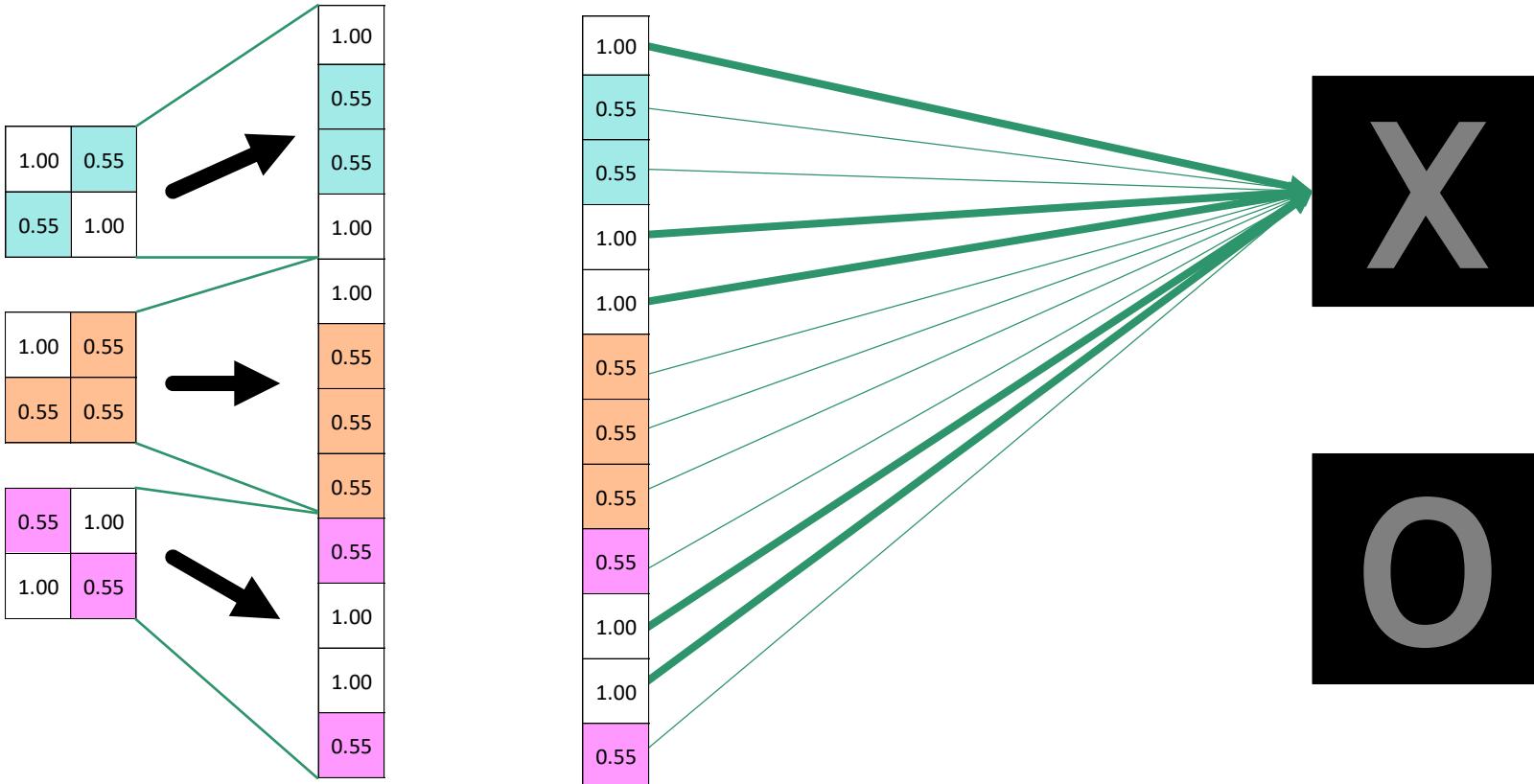
1.00	0.55
0.55	1.00
1.00	0.55
0.55	0.55
0.55	1.00
1.00	0.55

Learned features How to classify ?

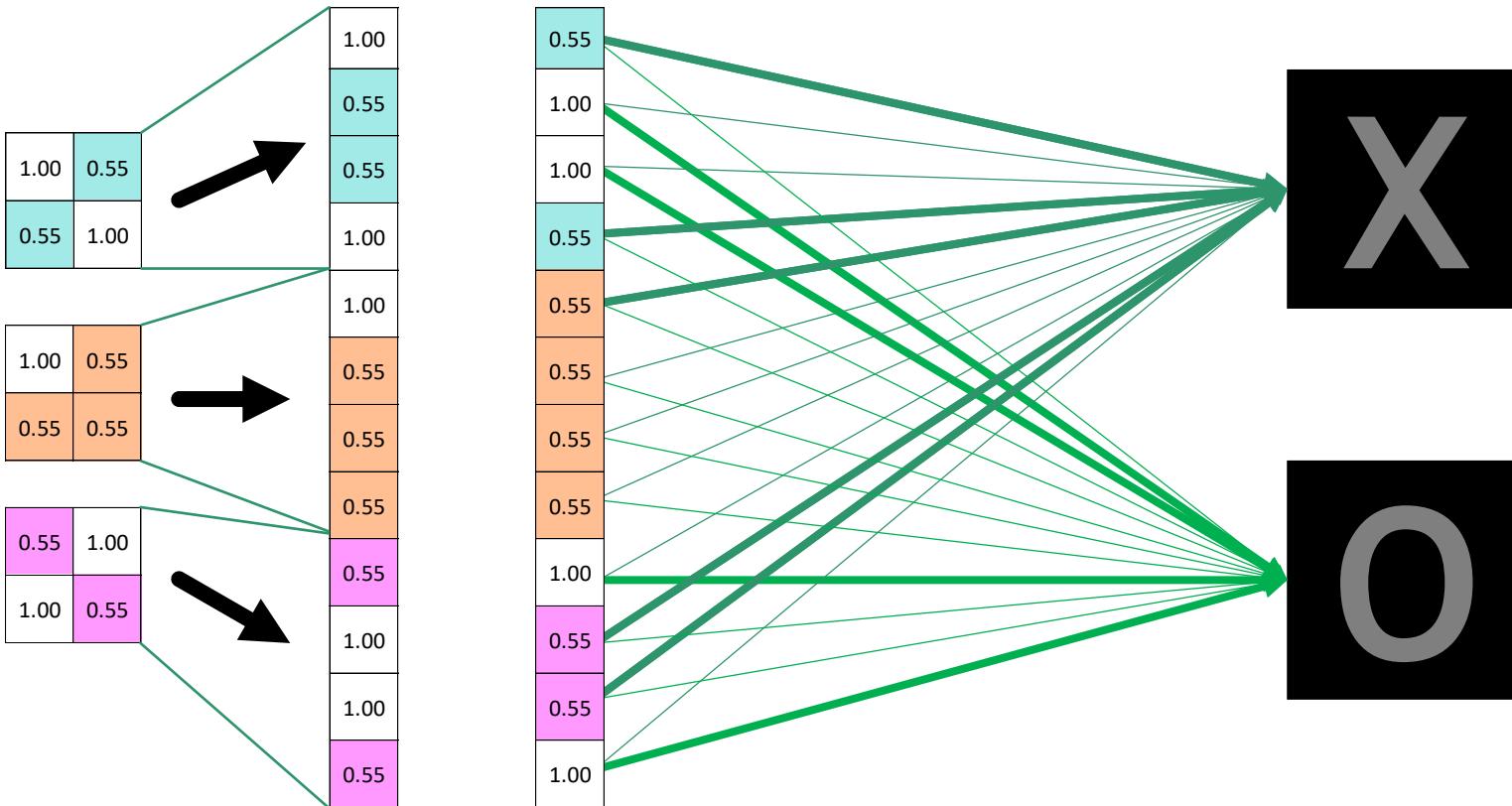
Fully connected layer



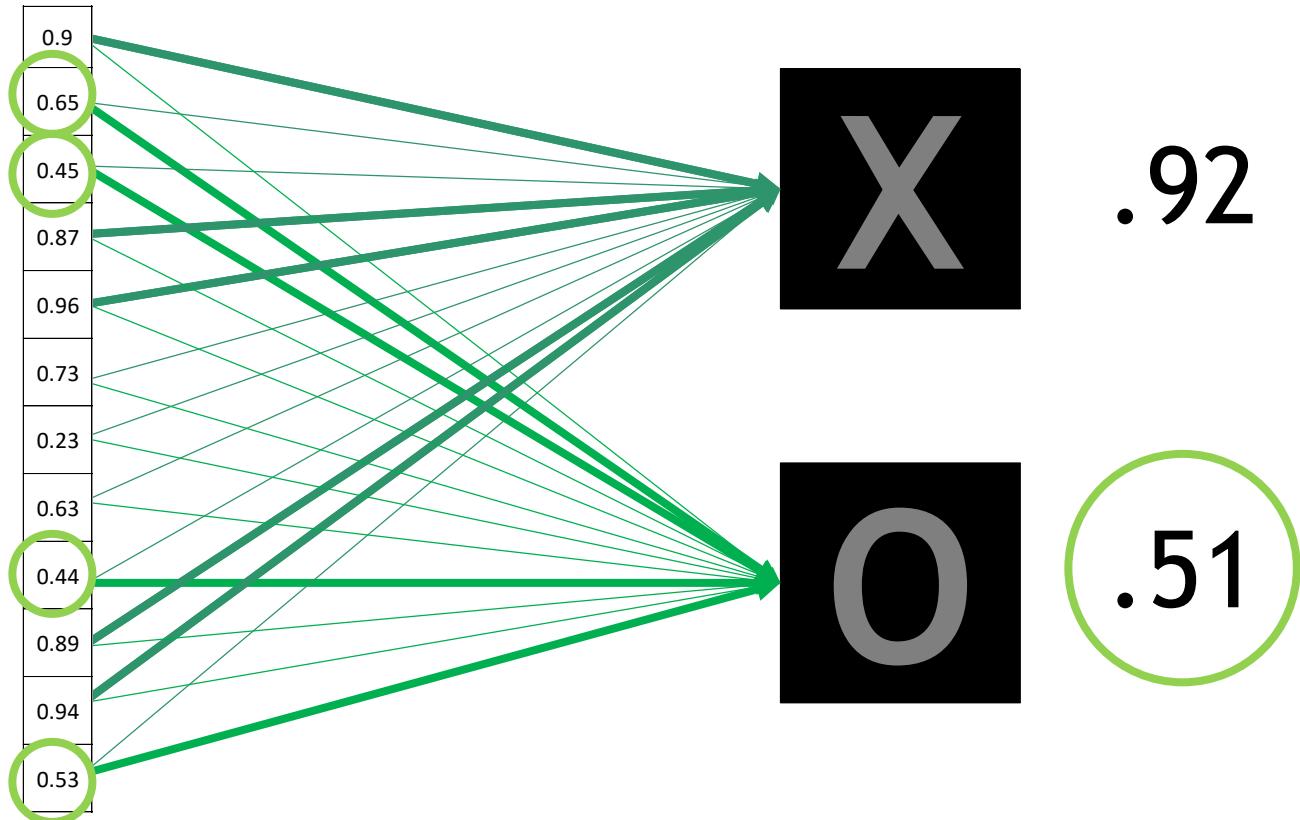
Fully connected layer



Fully connected layer

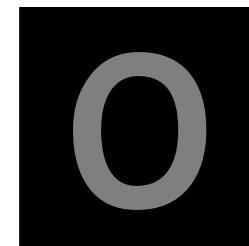
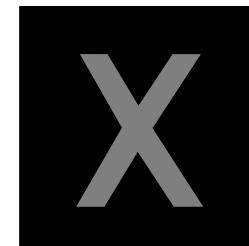
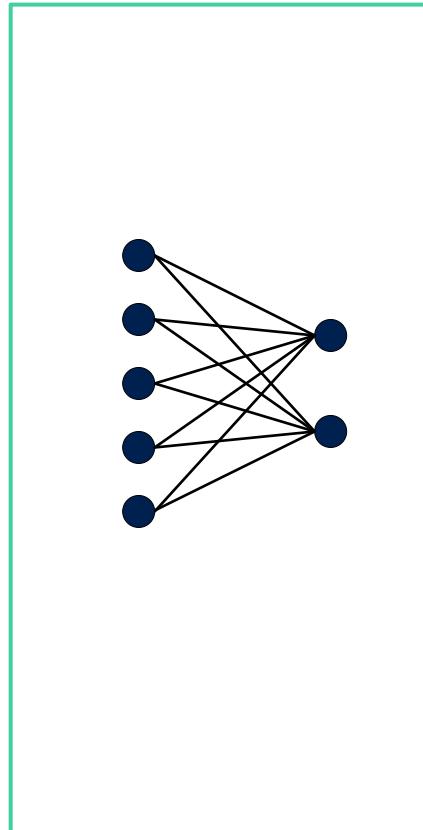


Fully connected layer



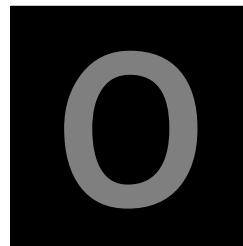
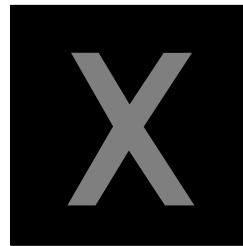
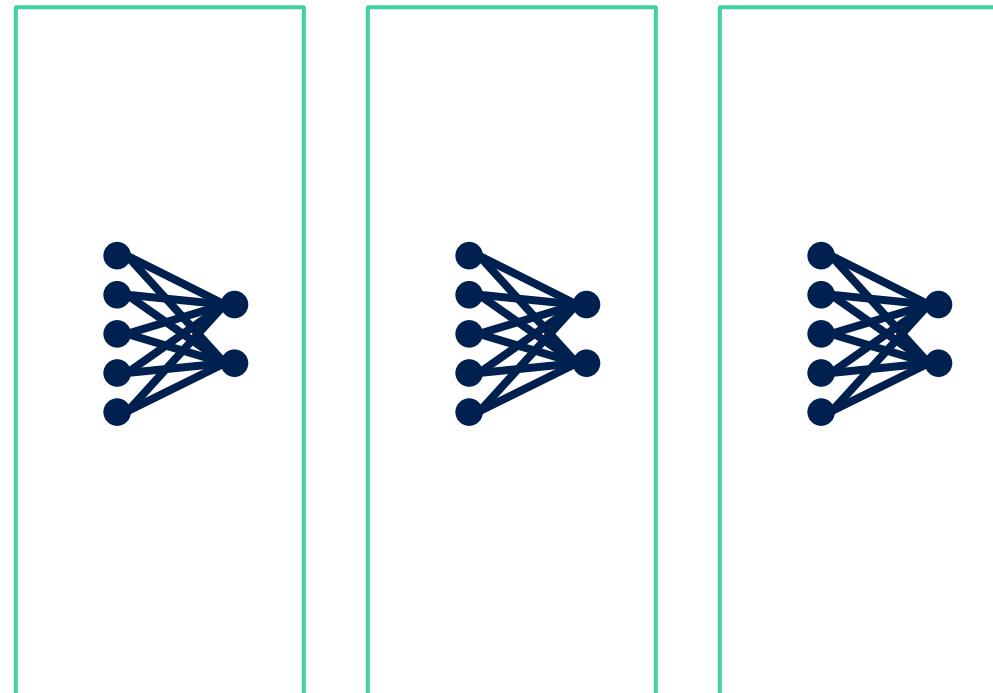
Fully connected layer

0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53



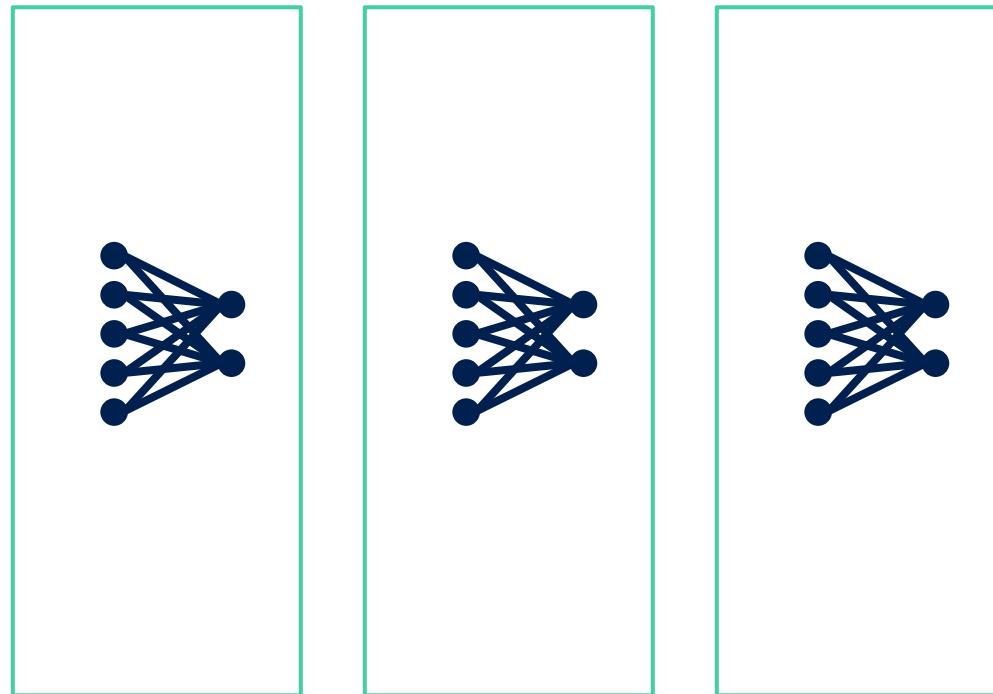
Fully connected layer

0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53

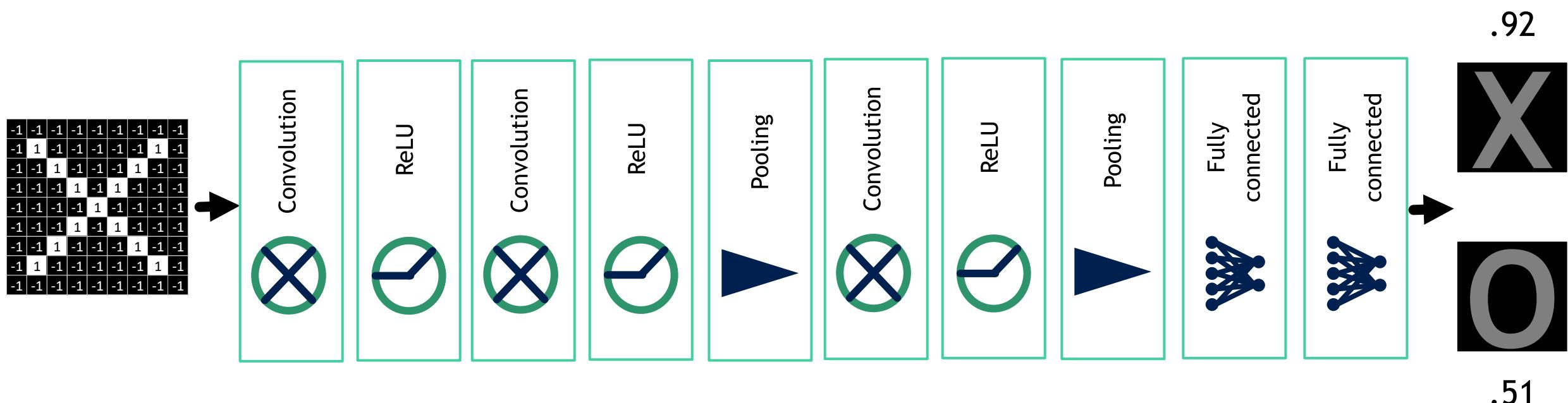


Fully connected layer

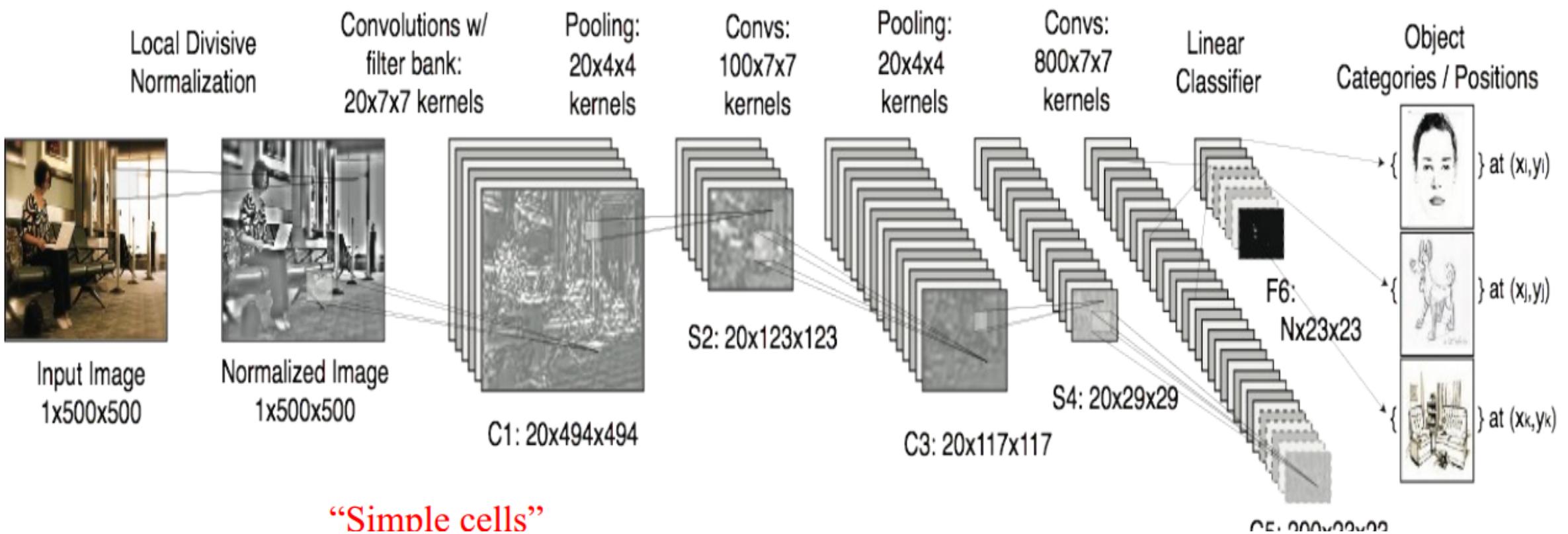
0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53



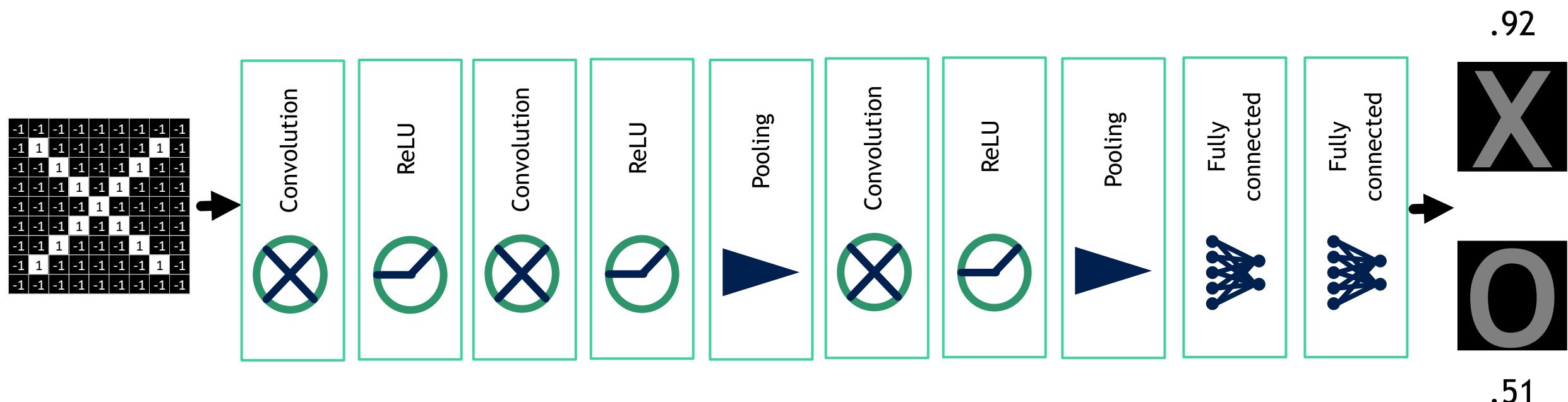
Putting it all together



Putting it all together

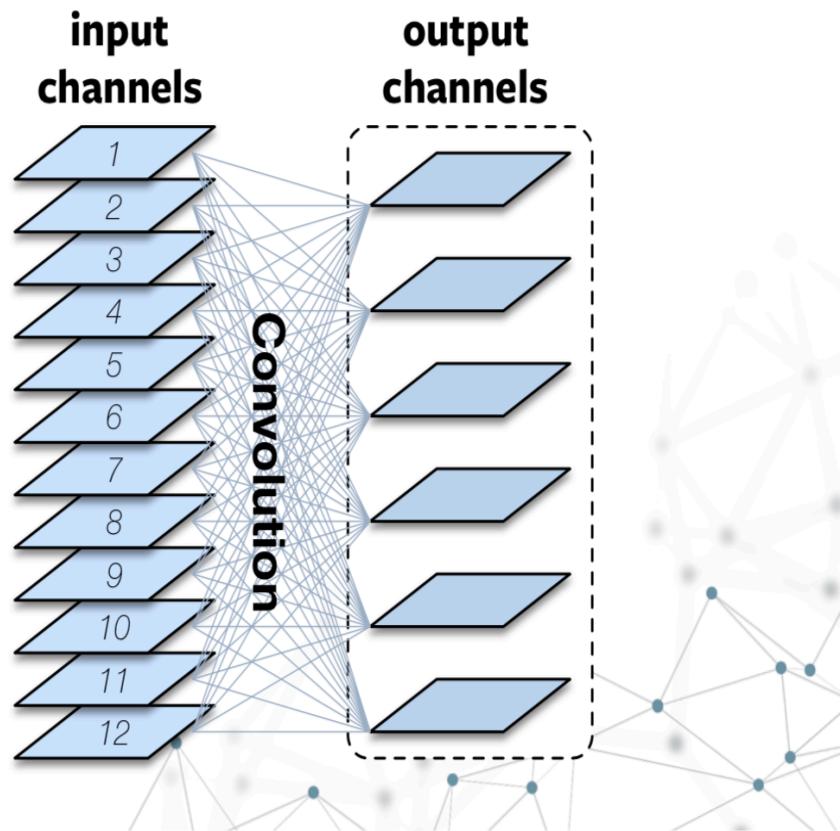


Putting it all together



Demo 3

- To produce a single output channel:
 - **Convolve** each input channel with some kernel
 - Note that the kernel is **different** for each input channel
 - **Sum** the convolved input channels to produce the output channel



Hyperparameters ?

Convolution

- Number of features
- Size of features

Pooling

- Window size
- Window stride

Fully Connected

- Number of neurons

Architecture

- How many of each type of layer?
- In what order?
- nonlinear transformations

Outline

1 Why Deep Learning ?

2 Why Convolution ?

3 Convolutional Networks

4 Applications

5 Popular architectures

6 Demo

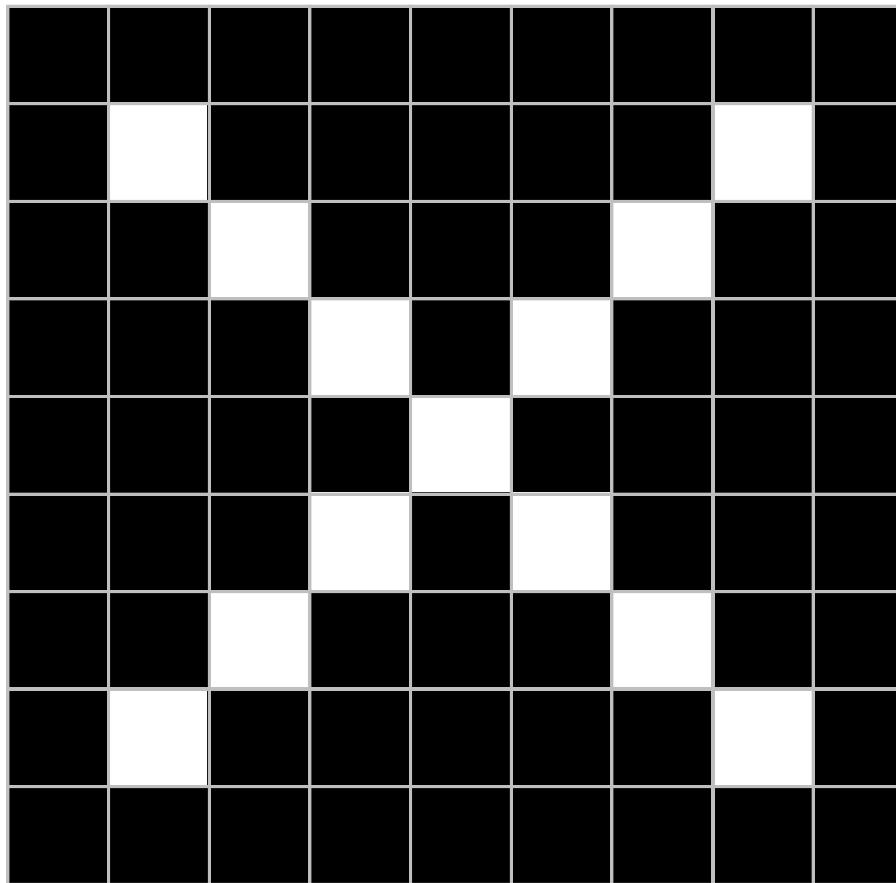
Not just images

- Any 2D (or 3D) data
- Spatial features

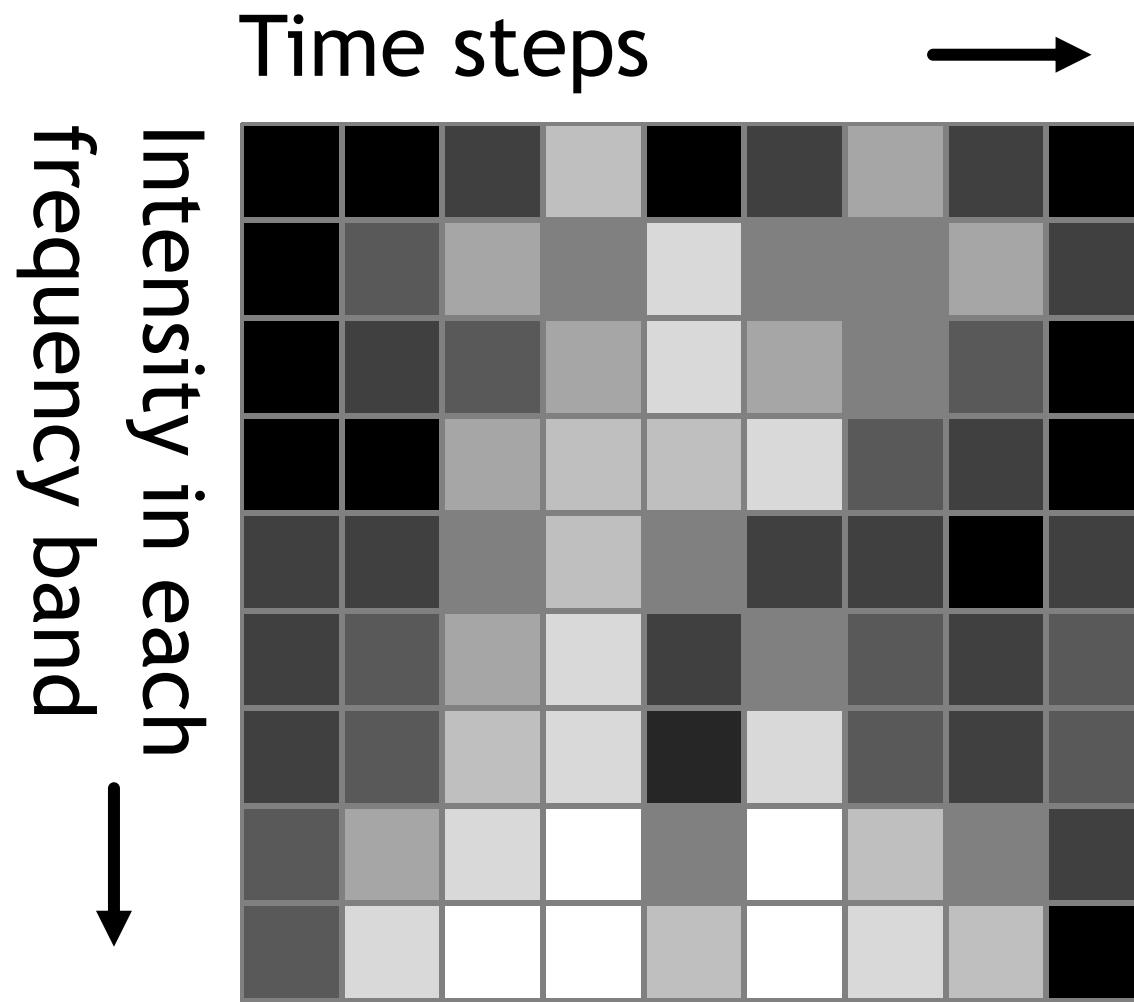
Images

Columns of pixels →

Rows of pixels



Sound

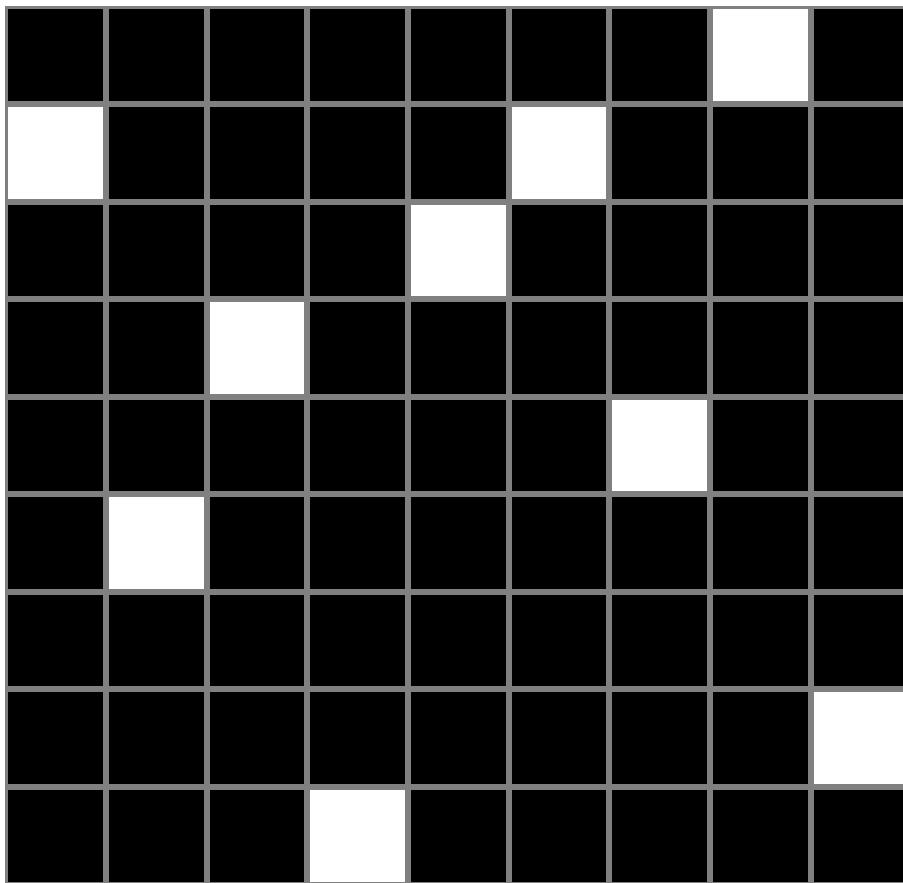


Text

Position in
sentence



Words in
dictionary



Spatial features become limitations themselves

How ?

Example

Name, age,
address, email,
purchases,
browsing activity,...



Customers



A	22	1A	a@a	1	aa	a1.a	123	aa1
B	33	2B	b@b	2	bb	b2.b	234	bb2
C	44	3C	c@c	3	cc	c3.c	345	cc3
D	55	4D	d@d	4	dd	d4.d	456	dd4
E	66	5E	e@e	5	ee	e5.e	567	ee5
F	77	6F	f@f	6	ff	f6.f	678	ff6
G	88	7G	g@g	7	gg	g7.g	789	gg7
H	99	8H	h@h	8	hh	h8.h	890	hh8
I	111	9I	i@i	9	ii	i9.i	901	ii9

Rule of thumb

If your data is just as useful after swapping any of your columns with each other, then you can't use Convolutional Neural Networks.

Summary

- Deep Learning for learning features representations
- Convolutional features are trained to match spatial features
- Very flexible models, many hyper parameters challenge
- many trade-offs
 - performance (memory and time)
 - Accuracy
 - Depth
 - Better yet simpler feature representations

References

- **Brandon Rohrer**, [Preview How convolutional neural networks \(CNNs\) work](#)
- [Ivado-mila Deep learning summer school 2019](#)
- [Intel AI introduction to CNNs](#)

References

- | | |
|------|--|
| [1] | • Brandon Roherer, Deep CNN for Image Classification |
| [2] | • I. S. G. E. H. Alex Krizhevsky, "ImageNet Classification with Deep Convolutional Neural Networks," NIPS, 2012. |
| [3] | • ImageNet, "ILSVRC results," 2016. [Online]. Available: http://www.image-net.org/challenges/LSVRC/ . |
| [4] | • L. D. a. D. Yu, Deep Learning Methods and Applications, 2013. |
| [5] | • G. I. Quo c V. Le Go ogle Brain, "A Tutorial on Deep Learning,Part 1: Nonlinear Classifiers and The Backpropagation Algorithm," December 2015. |
| [6] | • Y. LeCun, "Gradient-Based Learning Applied to document recognition". |
| [7] | • J. Schmidhuber, "Deep learning in neural networks: An overview," 12 September. |
| [8] | • B. Y. Bengio, "Learning Deep Architectures for AI". |
| [9] | • R. F. (NYU), "Deep Learning Methods for Vision CVPR 2012". |
| [10] | • S.-Y. L. Hyun Ah Song, Hierarchical Representation Using NMF. |
| [11] | • C. Szegedy, "Going deeper with convolutions," 2014. |
| [12] | • Dr.karpathy, "CS231n Convolutional Neural Networks for Visual Recognition," 26 11 2016. [Online]. |
| [13] | • D. o. C. S. C. I. N. Y. U. Matthew D. Zeiler, "Visualizing and Understanding Convolutional Networks," 2013. |
| [14] | • V. N. a. G. Hinton, "Rectified Linear Units improve Restricted Boltzmann Machines," International Conference on Machine Learning, 2010. |