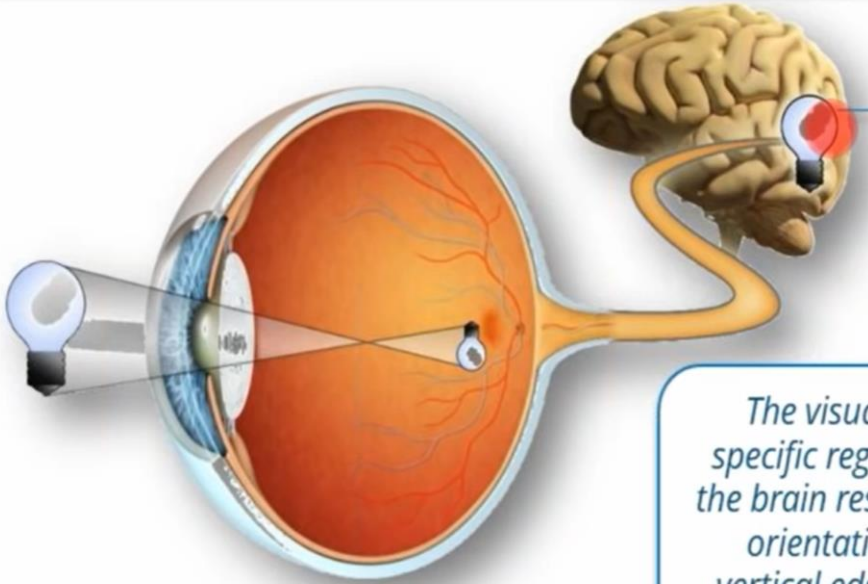
Convolutional Neural Networks_Ex

Dr. Dinesh Kumar Vishwakarma
Associate Professor, Department of Information Technology
Delhi Technological University, Delhi
Email: dinesh@dtu.ac.in
Web page: http://www.dtu.ac.in/Web/Departments/InformationTechnology/faculty/dkvishwakarma.php
Biometric Research Laboratory
http://www.dtu.ac.in/Web/Departments/InformationTechnology/lab_and_infra/bml/

What is CNN?

***Convolutional Neural Network** (**CNN**, or **ConvNet**)* is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex.



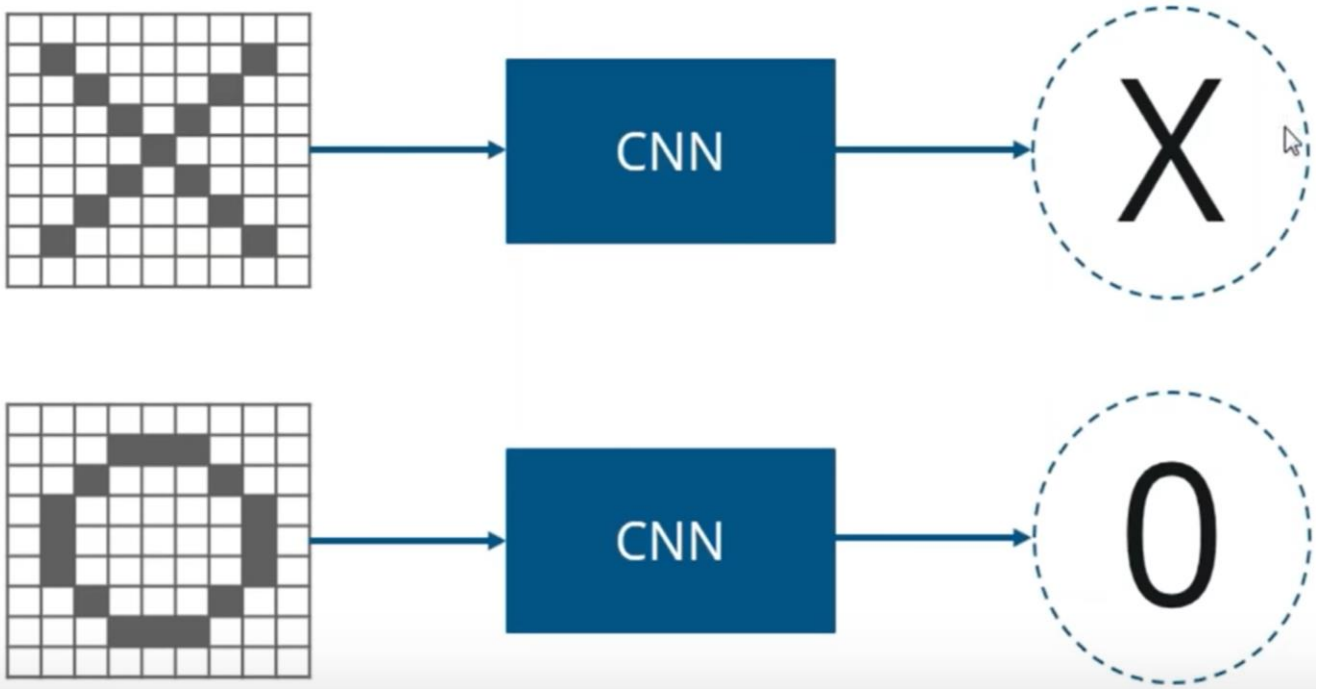
Visual Cortex

The visual cortex has small regions of cells that are sensitive to specific regions of the visual field. Some individual neuronal cells in the brain responds (or fires) only in the presence of edges of a certain orientation. For example, some neurons fires when exposed to vertical edges and some when shown horizontal or diagonal edges

How CNN Works?

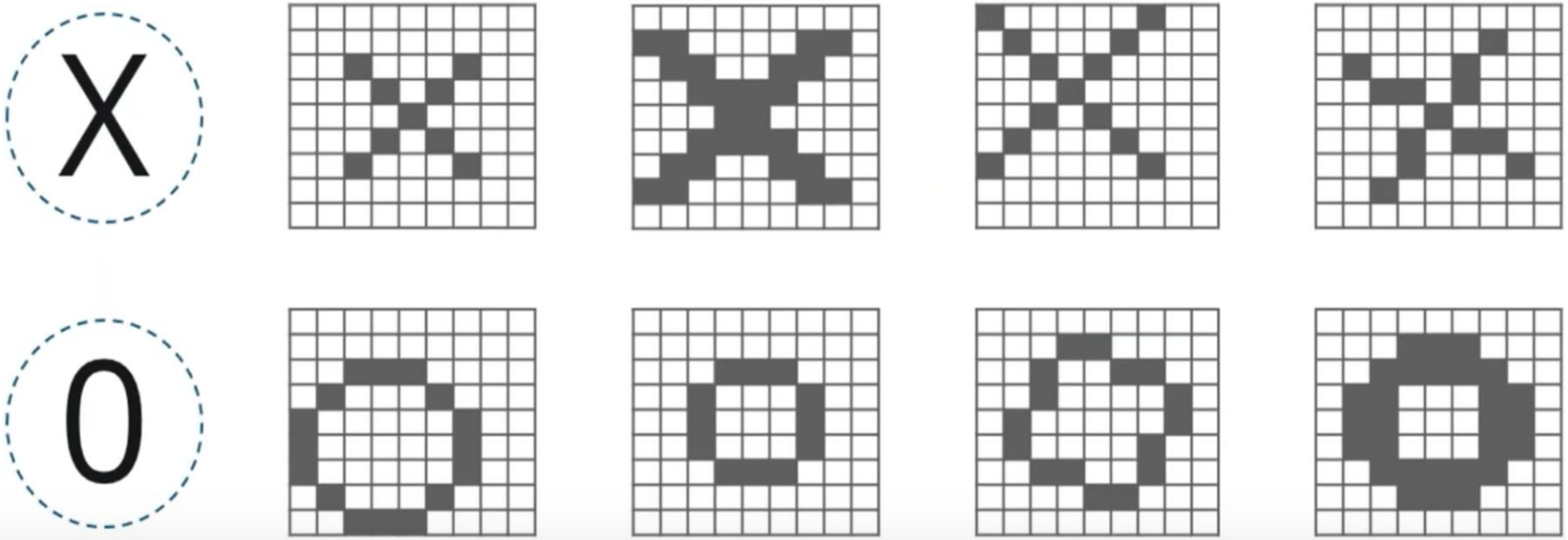
Convolutional Neural Networks have following layers:

- ✓ Convolution
- ✓ ReLU Layer
- ✓ Pooling
- ✓ Fully Connected

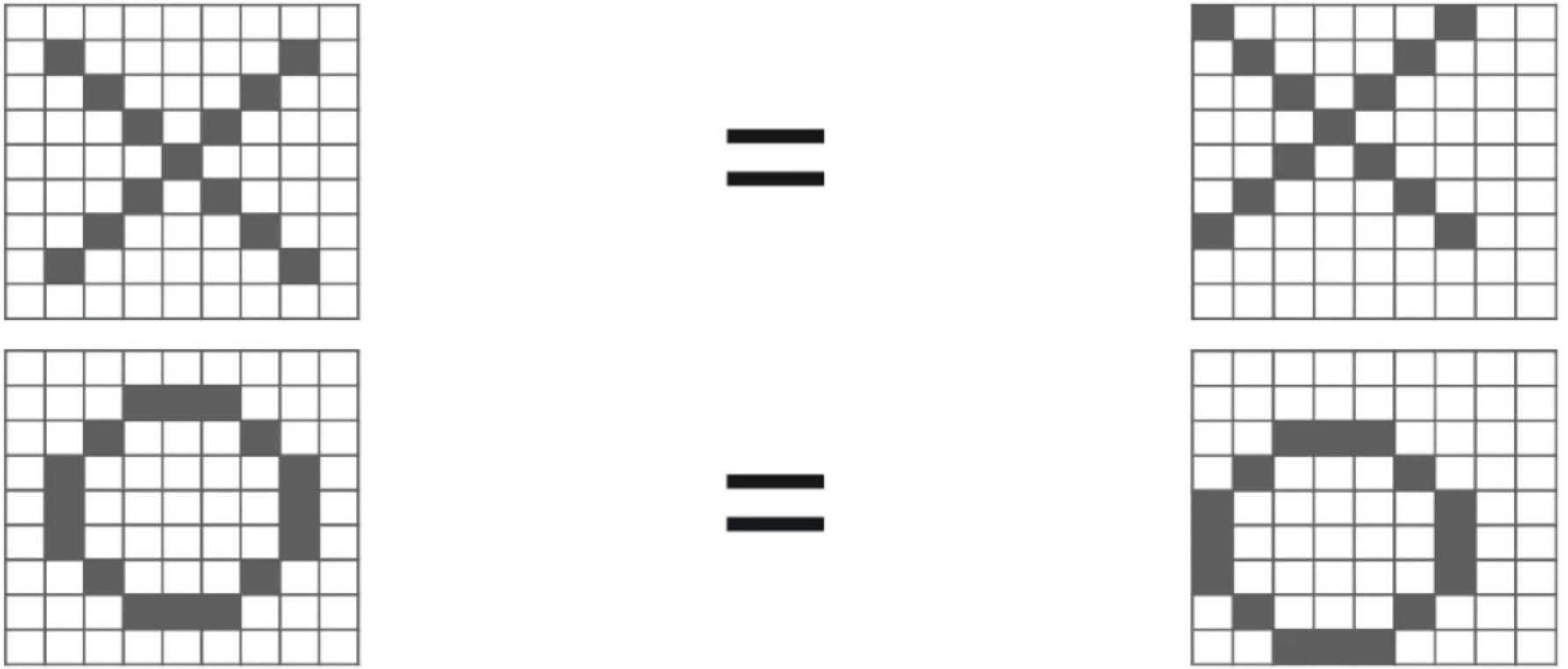


Possible Case

Here, we will have some problems, because X and O images won't always have the same images.
There can be certain deformations. Consider the diagrams shown below:



How CNN Works?



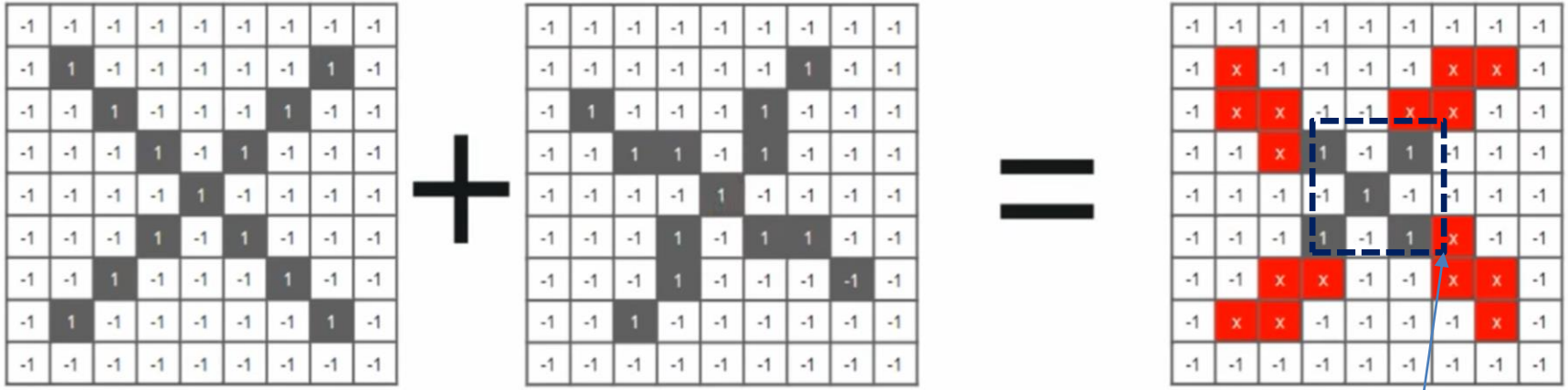
How CNN Works?

A computer understands an image using numbers at each pixels.
In our example, we have considered that a black pixel will have value 1 and a white pixel will have -1 value.

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

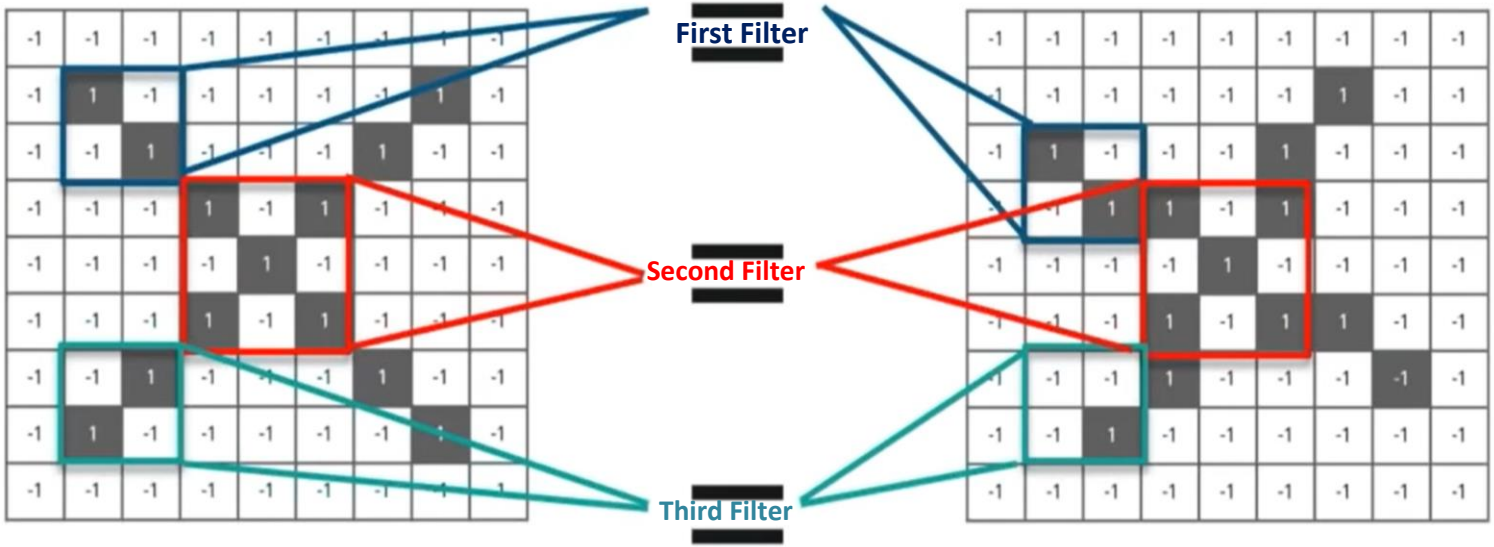
How CNN Works?

Using normal techniques, computers compare these images as:



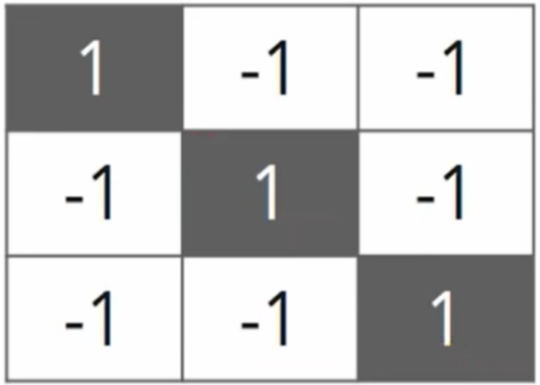
How CNN Works?

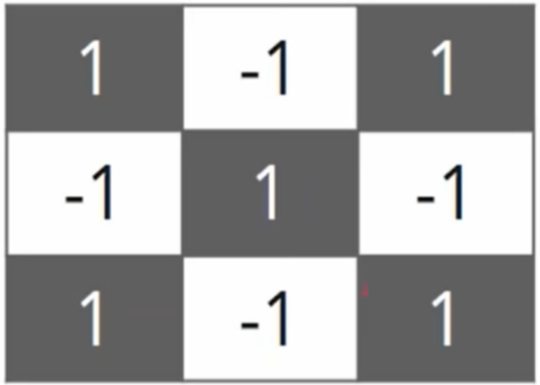
CNN compares the images piece by piece. The pieces that it looks for are called features.
By finding rough feature matches, in roughly the same position in two images, CNN gets a lot better at seeing similarity than whole-image matching schemes.

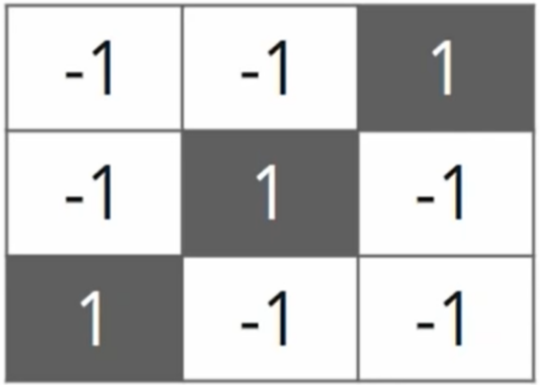


How CNN Works?

We will be taking three features or filters, as shown below:







How CNN Works?

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

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

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

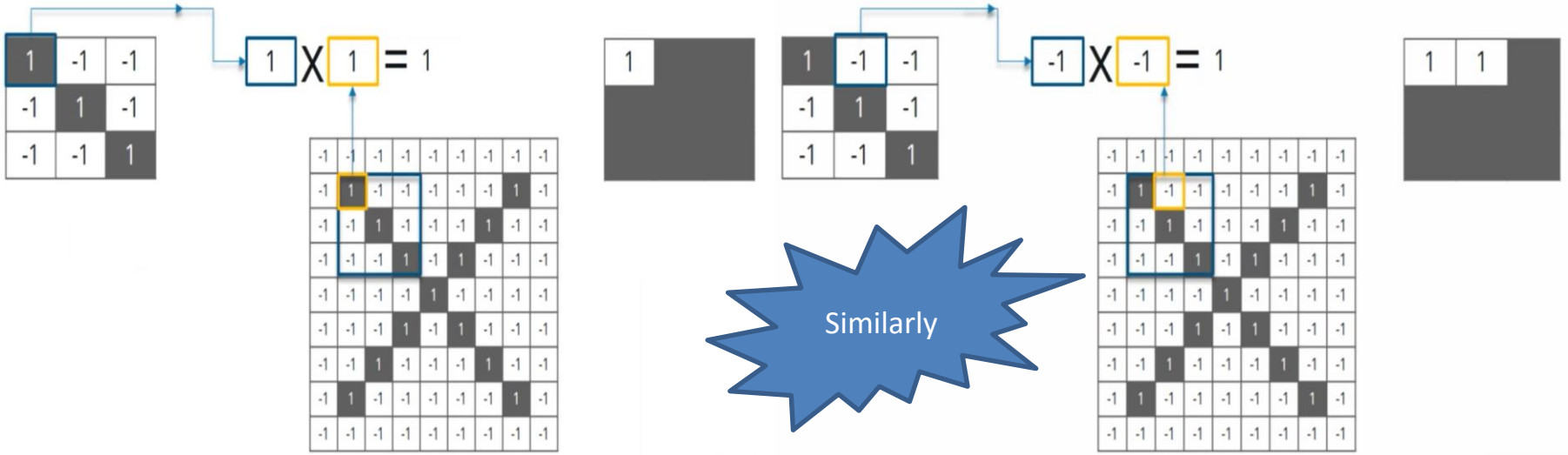
These are small pieces of the bigger image. We choose a feature and put it on the input image, if it matches then the image is classified correctly.

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

Convolutional Layer

Operations Involved in Convolution Layer

- Move filter over the image at every possible position.
- Multiply each image pixel by corresponding feature pixel.



Operations Involved in Convolution Layer.

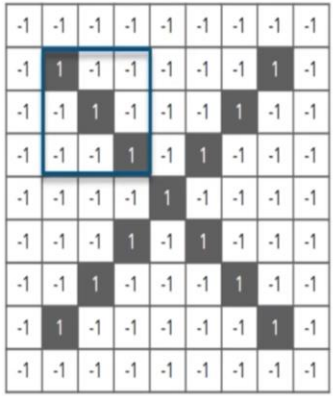
O/P of Conv Layer for first filter

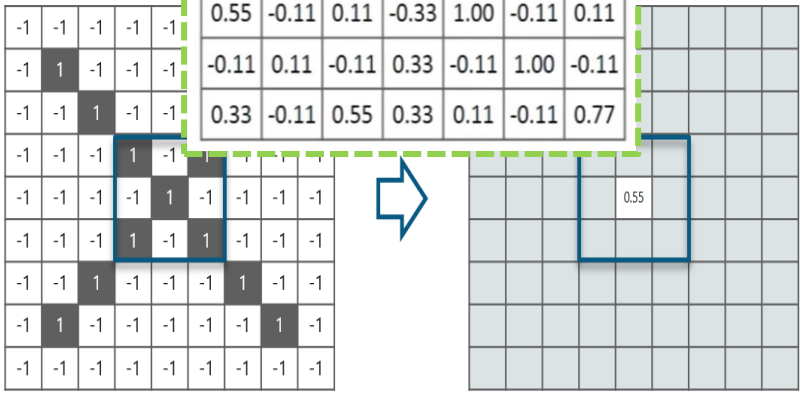
- Adding all the multiplied values of filter at every move.
- Divide the sum by total number of pixels.

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

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

1	1	1
1	1	1
1	1	1

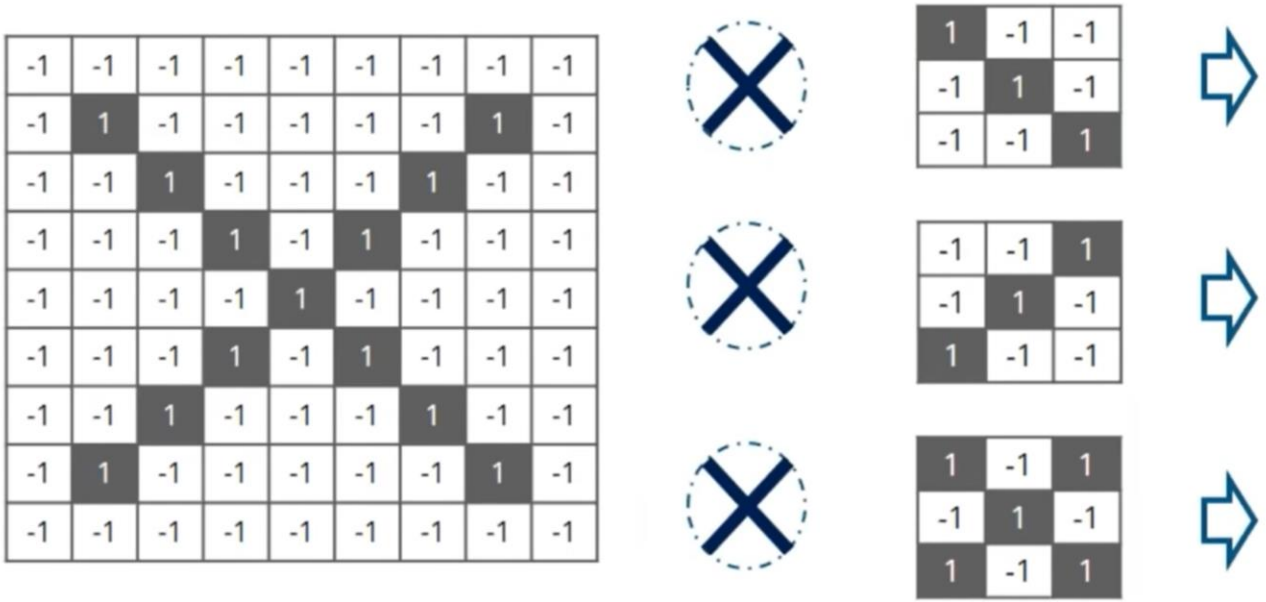




0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.0	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Output of Convolutional Layer

Similarly, we will perform the same convolution with every other filters



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.0	-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	-0.11	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

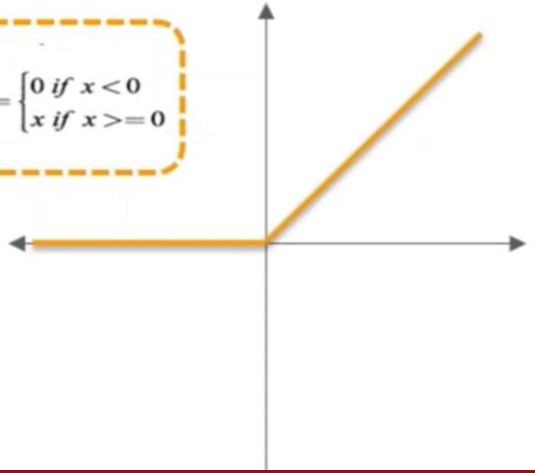
Activation Layer: ReLU

- ✓ In this layer we remove every negative values from the filtered images and replaces it with zero's
- ✓ This is done to avoid the values from summing up to zero

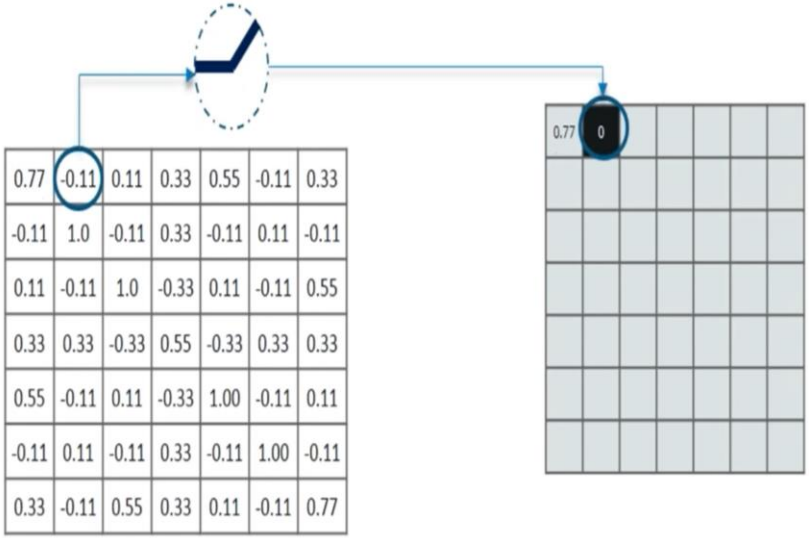
Rectified Linear Unit (ReLU) transform function only activates a node if the input is above a certain quantity, while the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship with the dependent variable

x	f(x)=x	F(x)
-3	f(-3) = 0	0
-5	f(-5) = 0	0
3	f(3) = 3	3
5	f(5) = 5	5

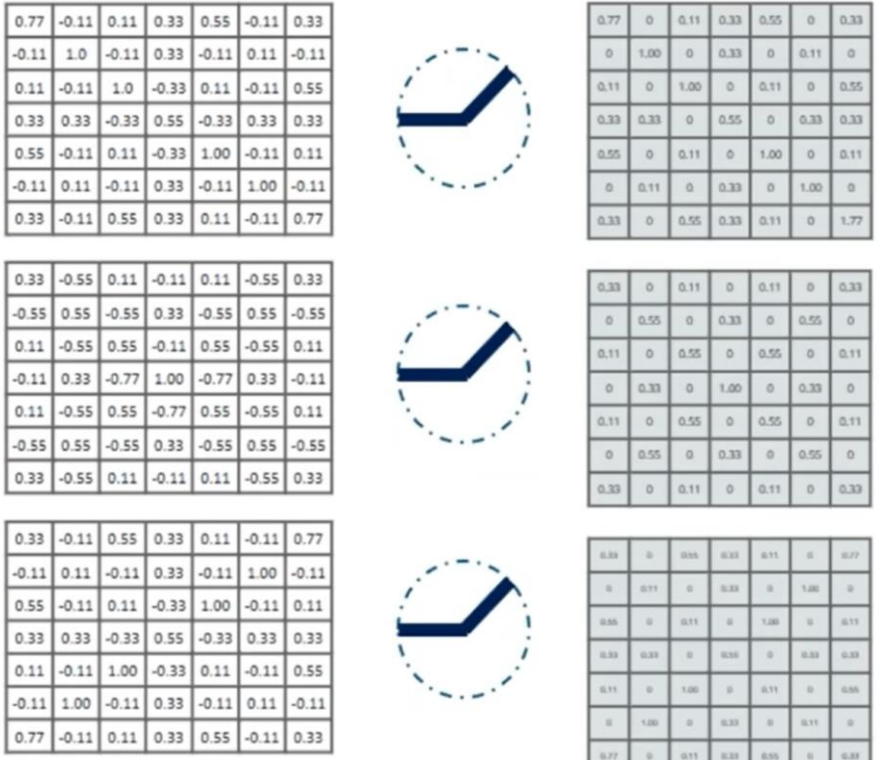
$$f(x) = \begin{cases} 0 \text{ if } x < 0 \\ x \text{ if } x >= 0 \end{cases}$$



ReLU: Removes the Negative Values



First Feature



Pooling Layer

In this layer we shrink the image stack into a smaller size

Steps:

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.

Symbol:

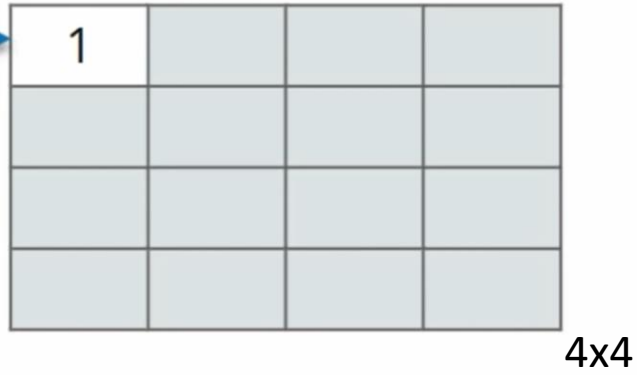
Let's perform pooling with a window size 2 and a stride 2

Max Pooling of size: 2x2, Stride : 2

Let's start with our first filtered image
In our first window the maximum or highest value is 1, so we track that and move the window two strides

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	.77

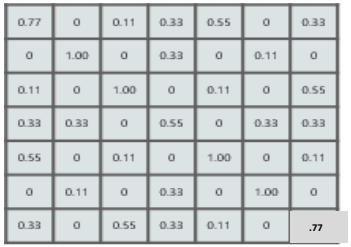
7x7



4x4

Convolutional Neu

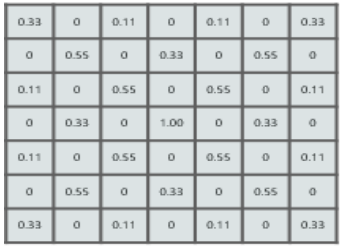
Max Pooling of size: 2x2, Stride : 2



7x7

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

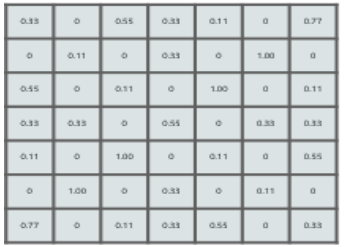
4x4



7x7

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

4x4

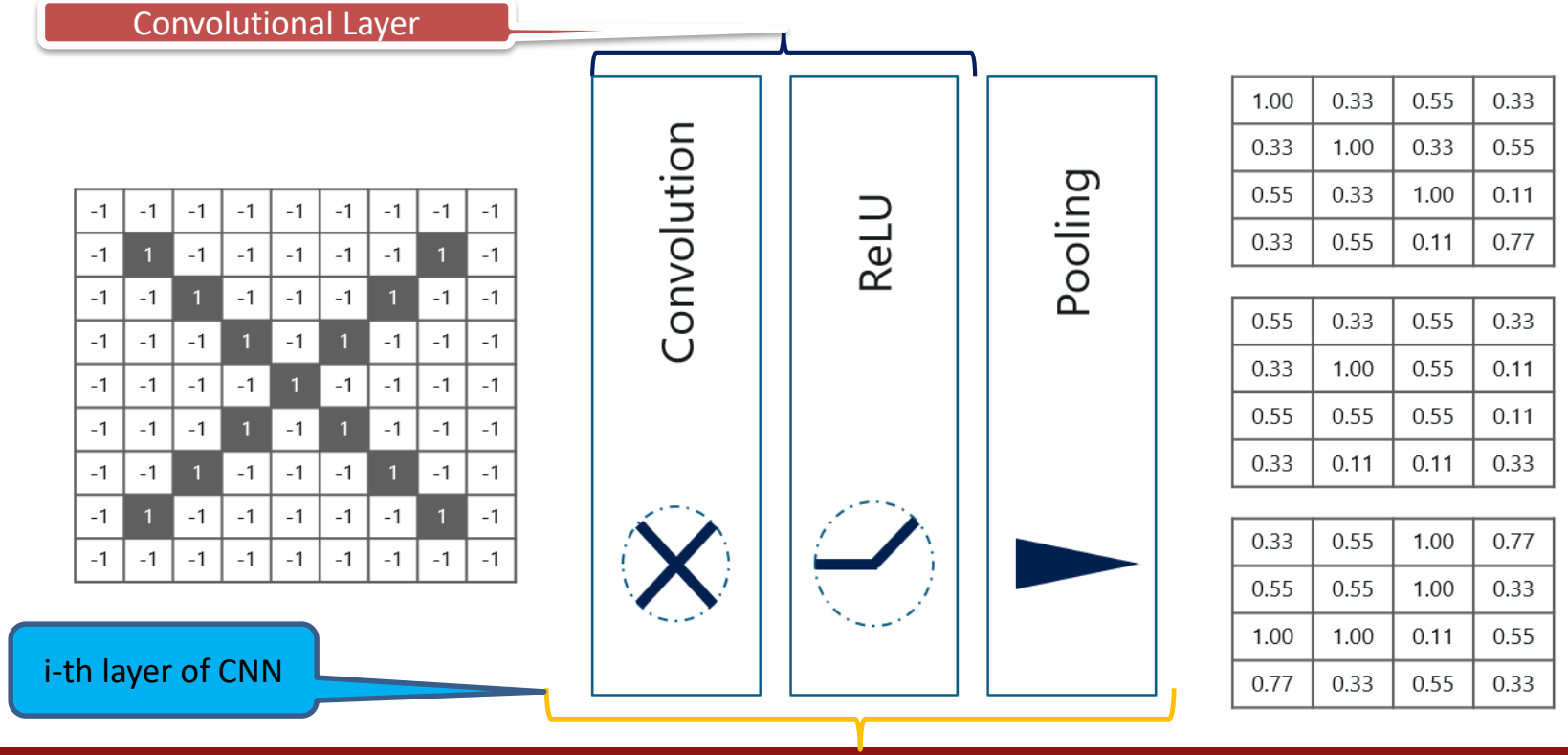


7x7

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

4x4

Stacking of Layers



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

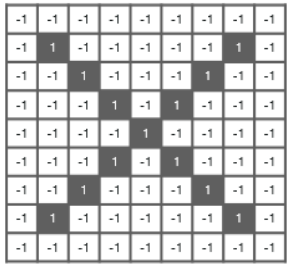
1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

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

Practical CNN Model

- Most of the practical CNN models have more number of layers.



1	0.55
0.55	1.00

2x2

1	0.55
0.55	0.55

2x2

0.55	1.00
1.00	0.55

2x2

Fully Connected Layer

This is the final layer where the actual classification happens

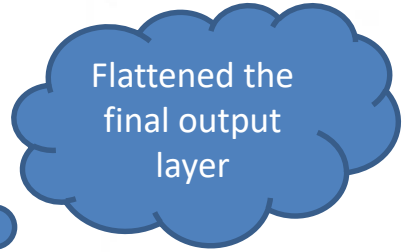
Here we take our filtered and shrinked images and put them into a single list

1	0.55
0.55	1.00

1	0.55
0.55	0.55

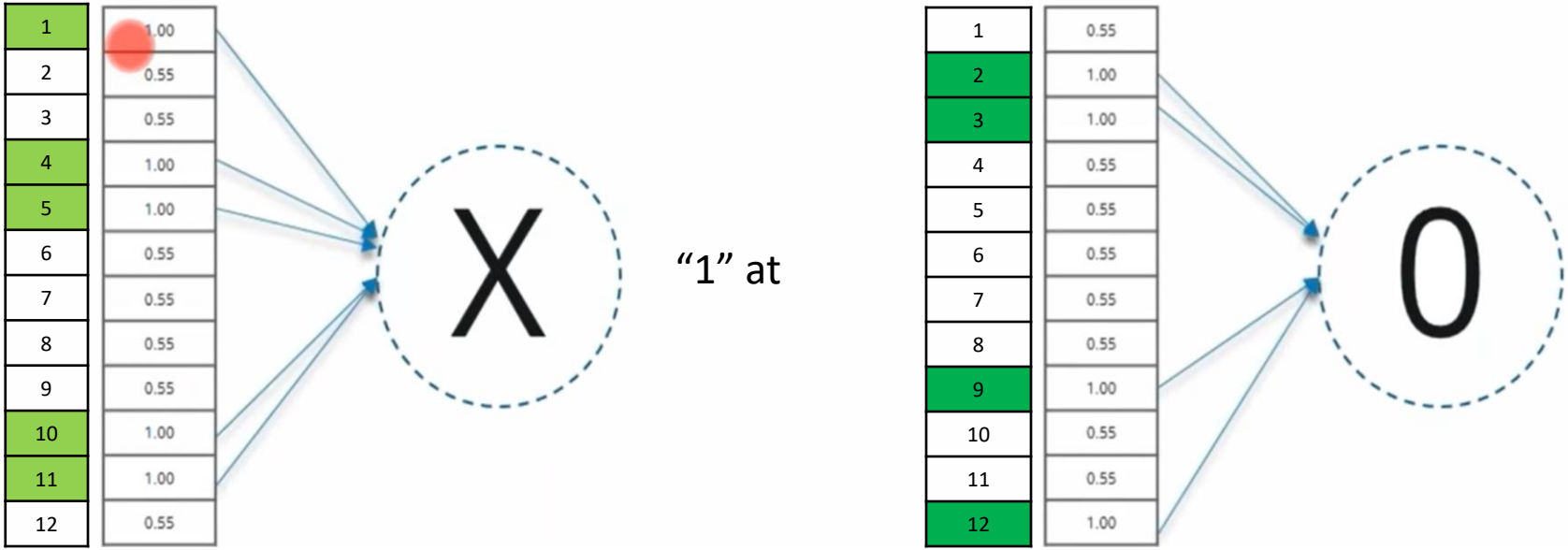
0.55	1.00
1.00	0.55

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



Output

When we feed in, 'X' and 'O'. Then there will be some element in the vector that will be high. Consider the image below, as you can see for 'X' there are different elements that are high and similarly, for 'O' we have different elements that are high.



Prediction

Consider the below list of a new input image:

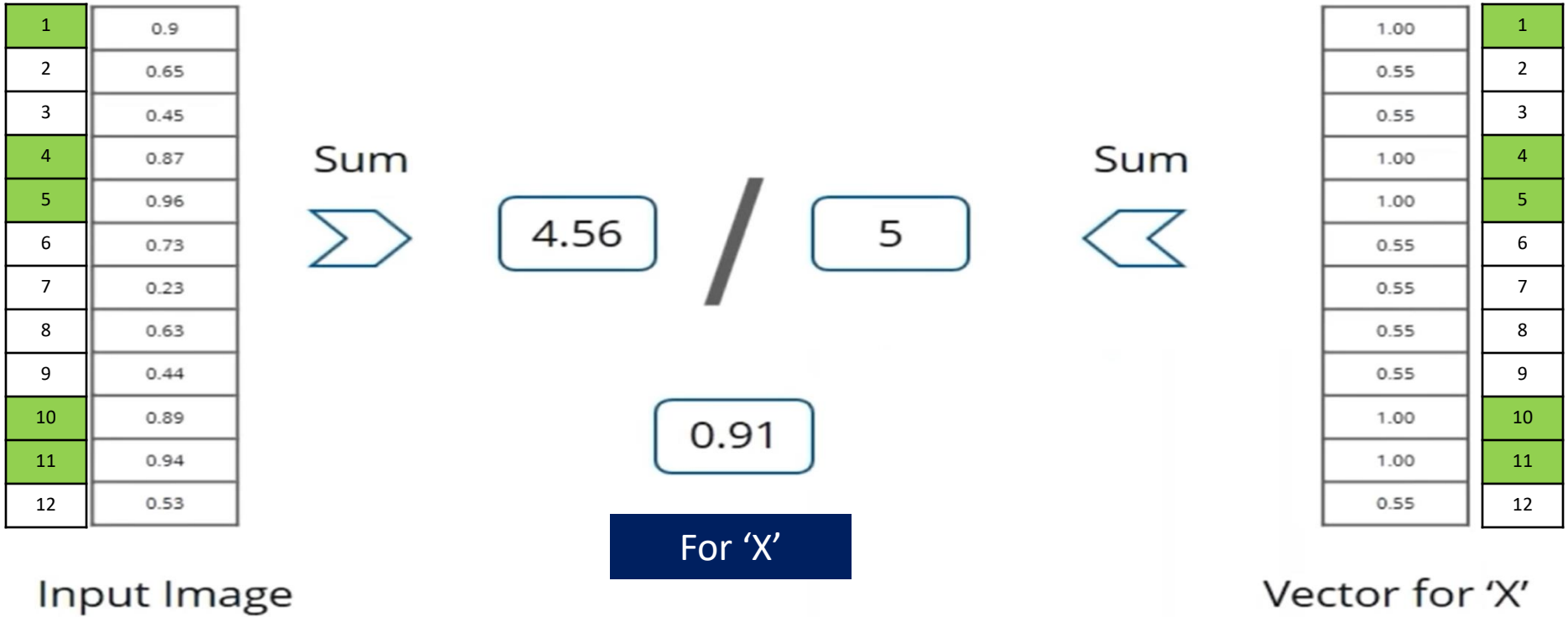
1	0.9
2	0.65
3	0.45
4	0.87
5	0.96
6	0.73
7	0.23
8	0.63
9	0.44
10	0.89
11	0.94
12	0.53

Now Network is training and gives prediction for a Test sample

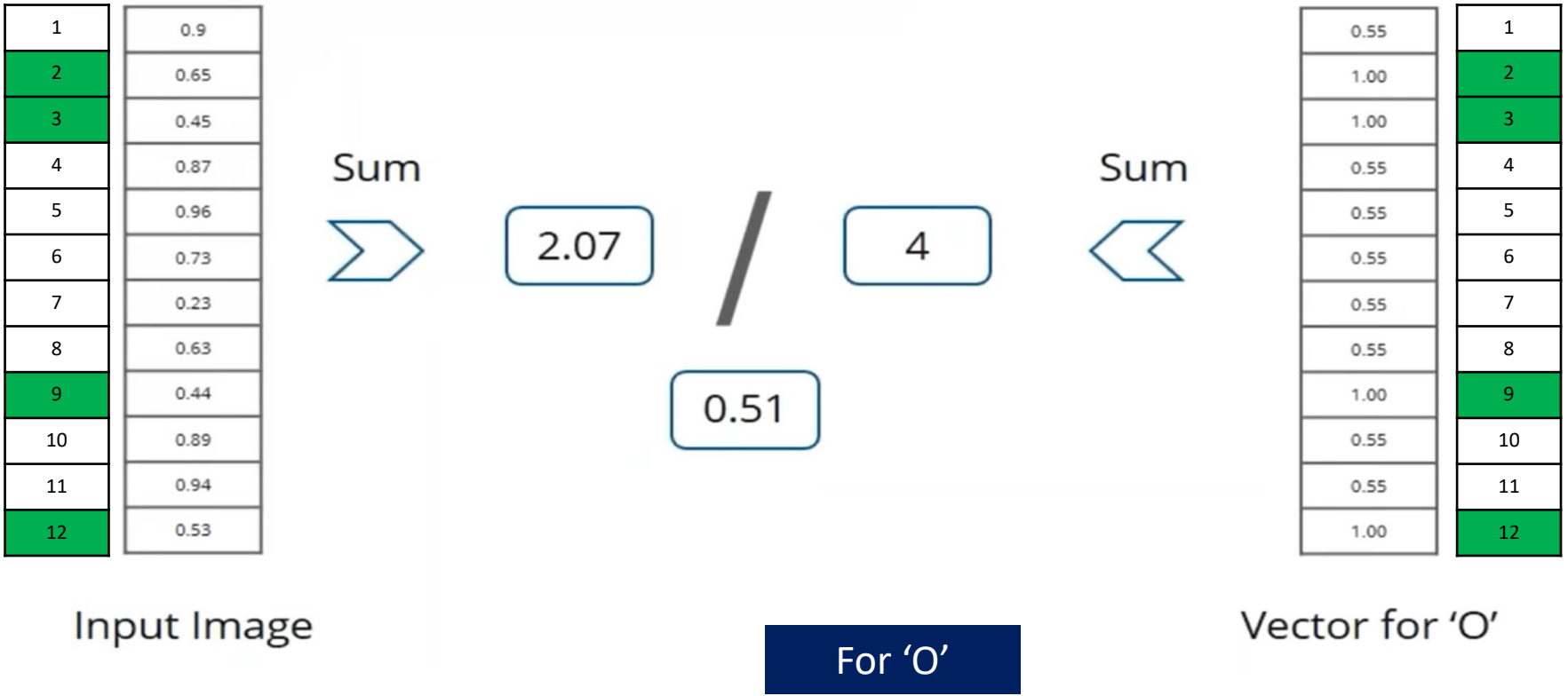
Is is “X” or “O”?

Lets Compare

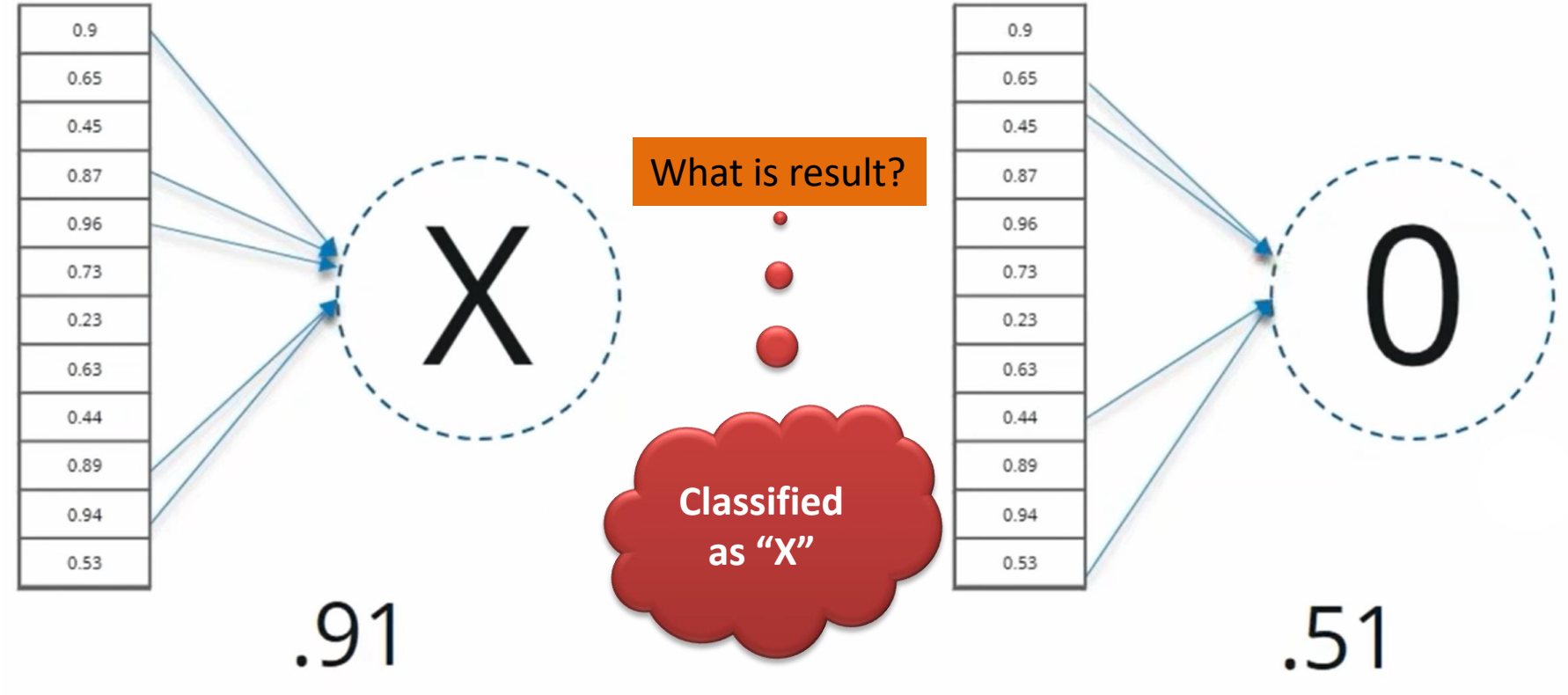
Prediction: Compare with ‘X’ and ‘O’



Prediction: Compare with ‘X’ and ‘O’



Result: Compare with ‘X’ and ‘O’



References

- https://www.youtube.com/watch?v=umGJ30-15_A