Convolutional Neural Networks

RAMAN LAB @PLS

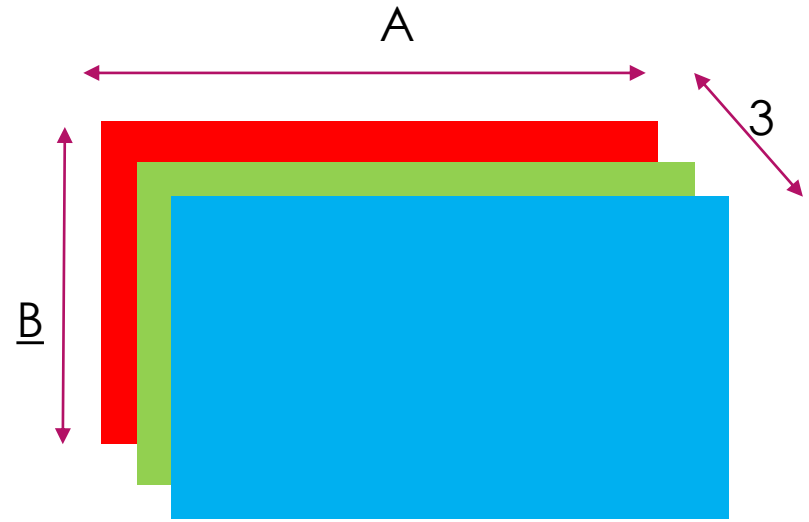
Agenda

- How Computer Reads an Image?
- Why not fully connected Networks for Image Recognition?
- What is Convolutional Neural Network?
- How Convolutional Neural Network works?
- Real world Applications

How computer reads an image

Channels

Size of Image B x A x 3



Pixels value of RGB

B: Rows | A: Columns | 3: Channels

How computer reads an image

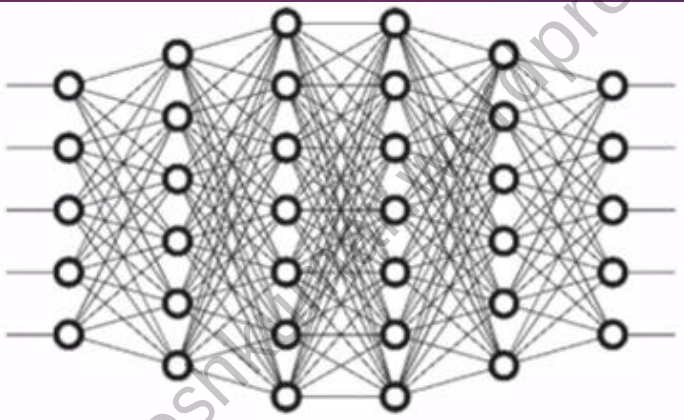
What We See

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

What Computers See

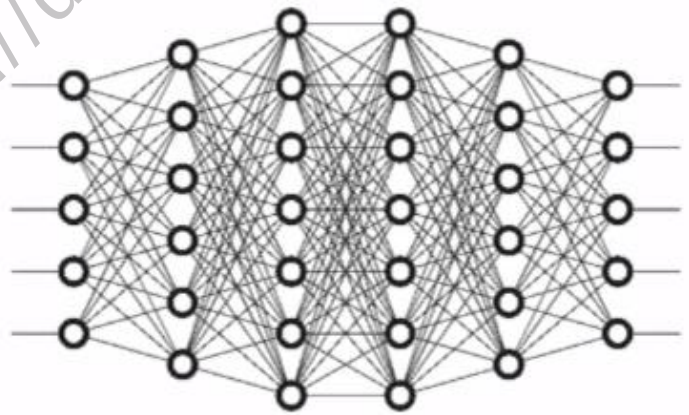
Why not fully connected networks

Image with 28 x 28 x 3 Pixels (MNIST)



Number of weights in the first hidden layer will be 2352

Image with 200 x 200 x 3 pixels

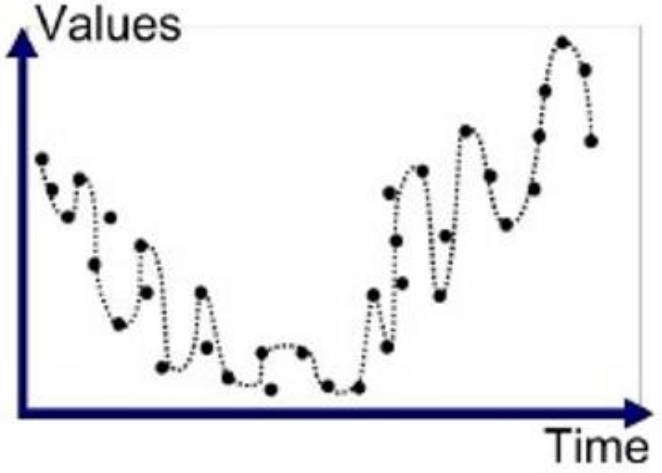


Number of weights in the first hidden layer will be 120,000

Why not fully connected networks

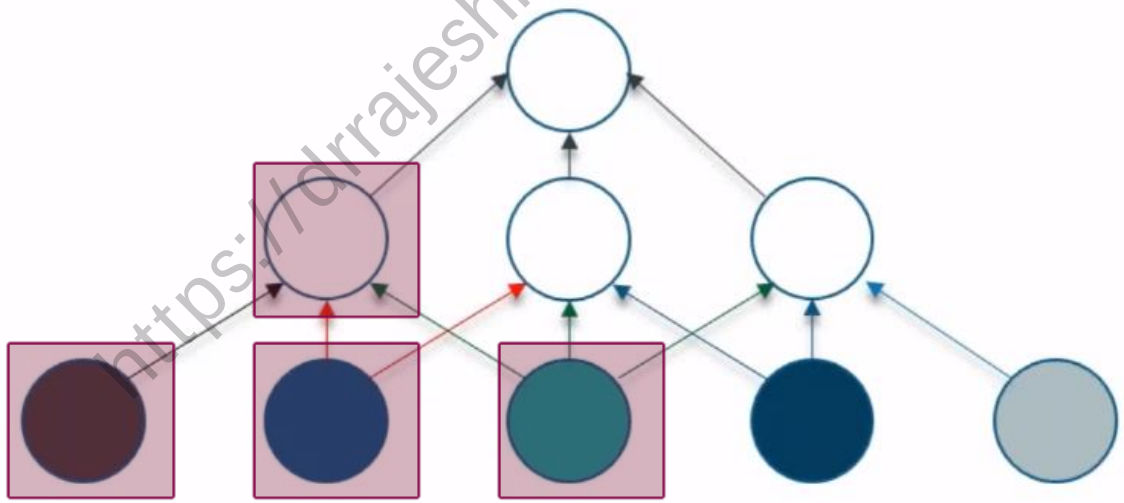
High computational Resources

Overfitting



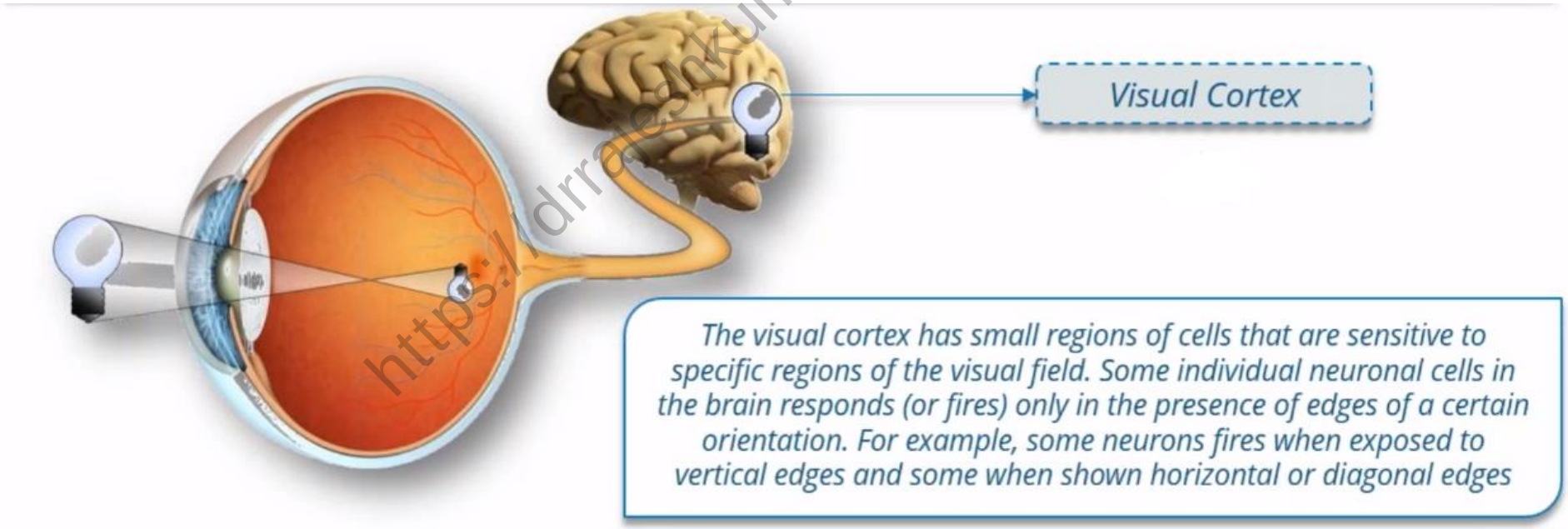
Why Convolutional Neural Networks

In case of CNN, the neuron in a layer will only be connected to a small region of layer before it, Instead of all neurons in a fully-connected manner.



What is Convolutional Neural Network

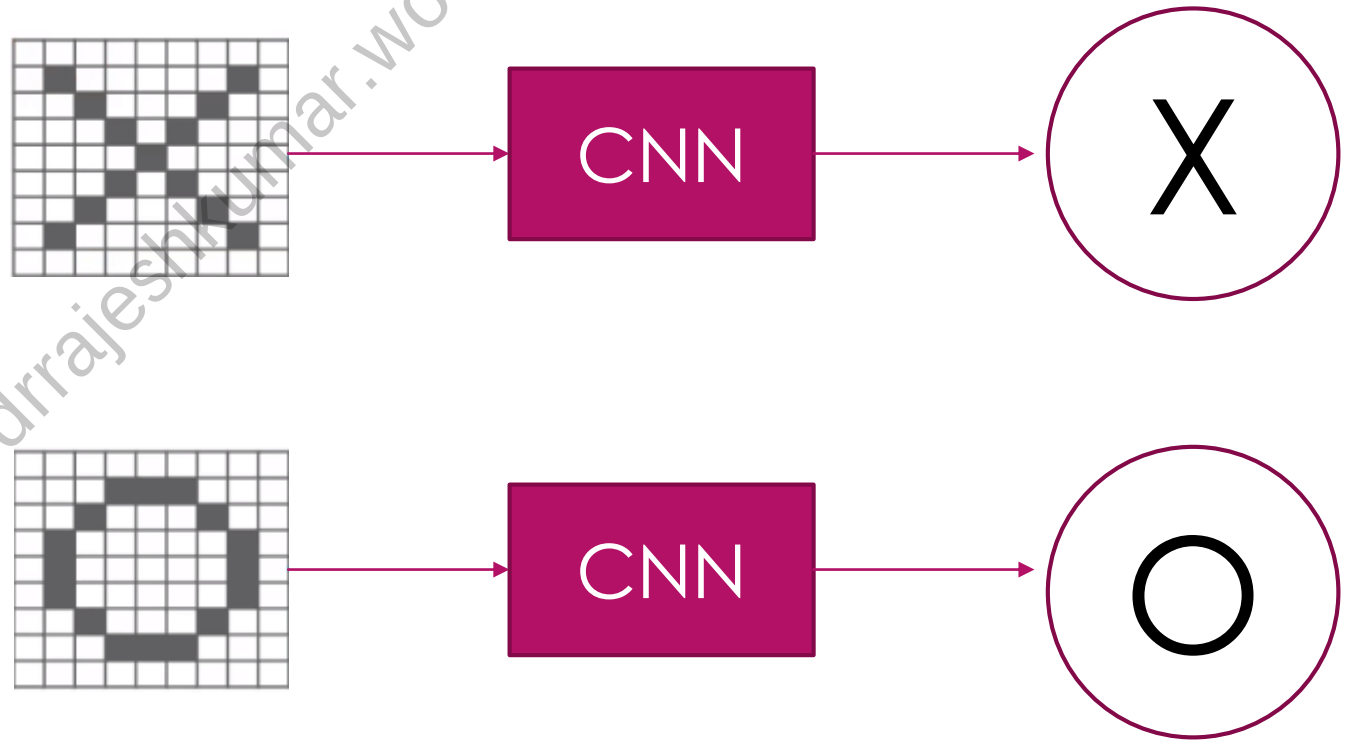
Convolutional Neural Network (CNN) is a type of feed-forward artificial neural network in which the Connectivity pattern between its neuron is inspired by organization of animal visual cortex.



How CNN works?

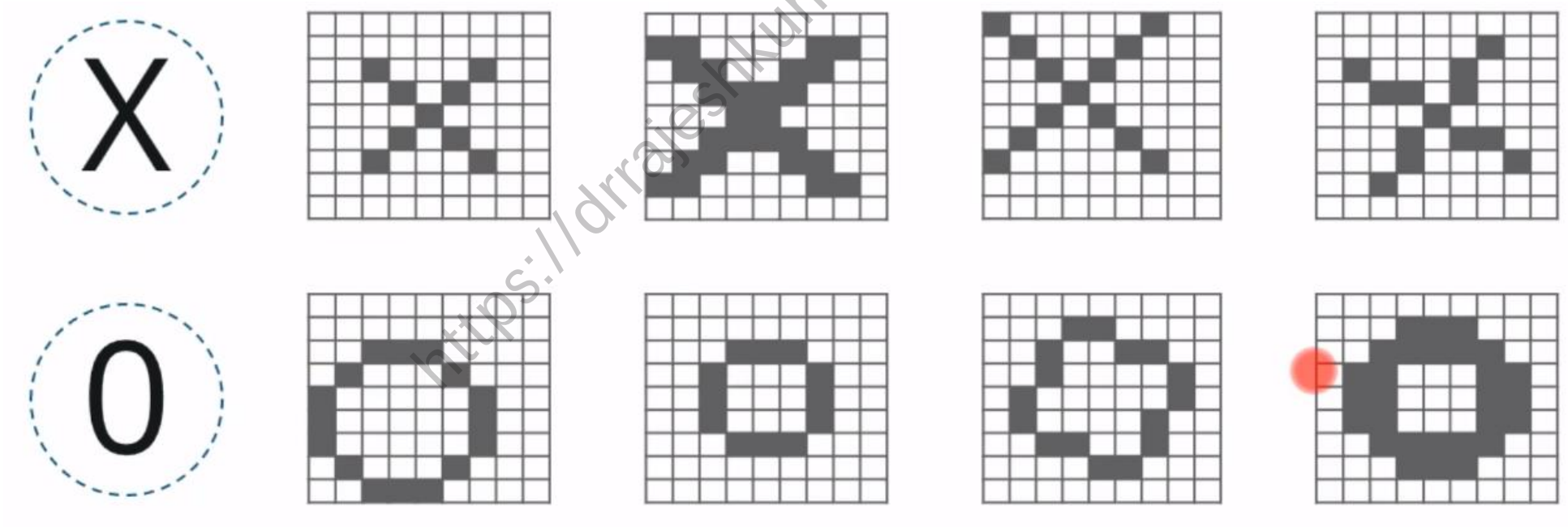
Convolutional Neural Network has following layers:

- ✓ Convolutional Layer
- ✓ ReLU Layer
- ✓ Pooling Layer
- ✓ Fully Connected Layer



Trickier Case

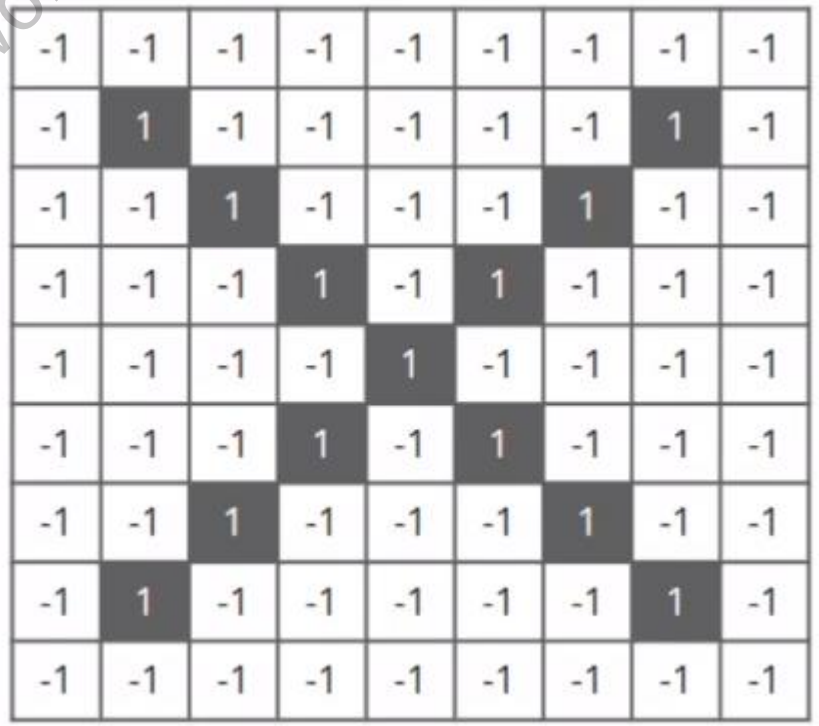
Here we will have some problems, because X and O images won't always have the same images. There can be certain deformations. Consider the diagram below



How CNN works?

A Computer understand an image using numbers at each pixels.

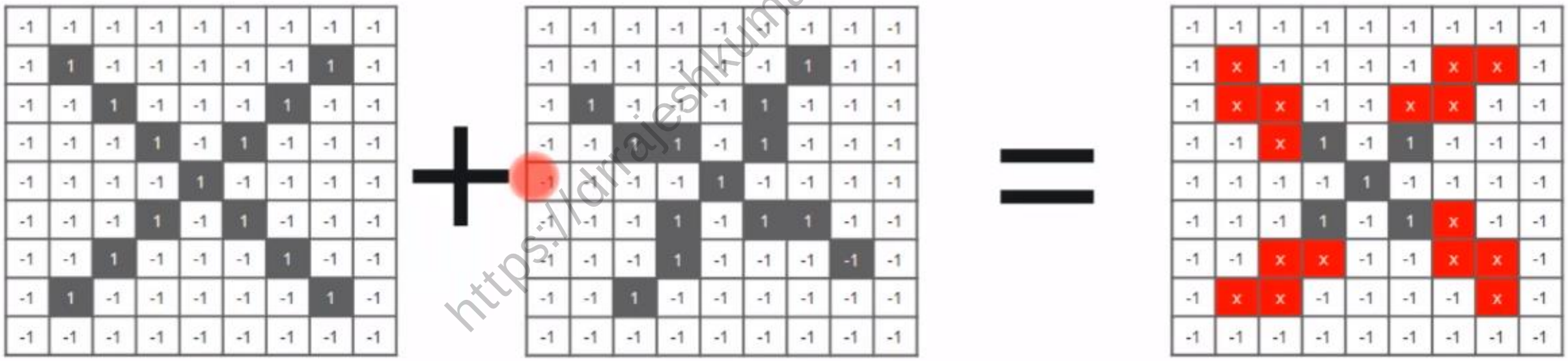
In our example. We have considered that black pixel will have 1 and a white pixel will have -1 value.



-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:



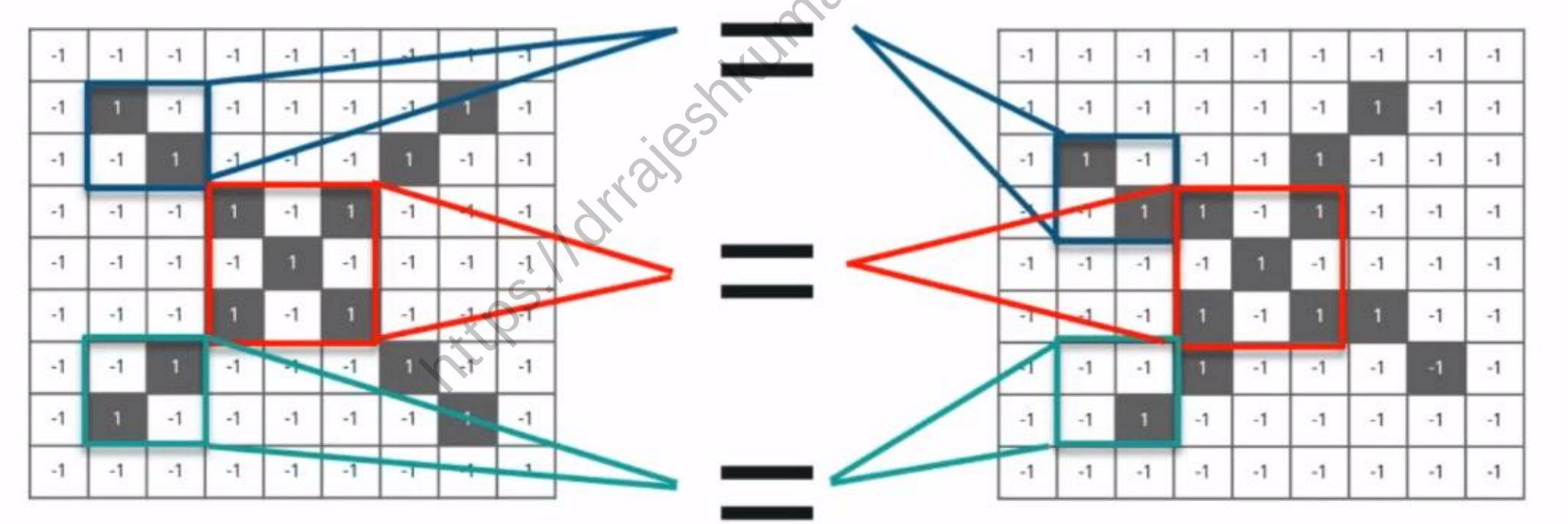
Correct X

Deformed X

Computer unable to recognize if it is X or not

How CNN works?

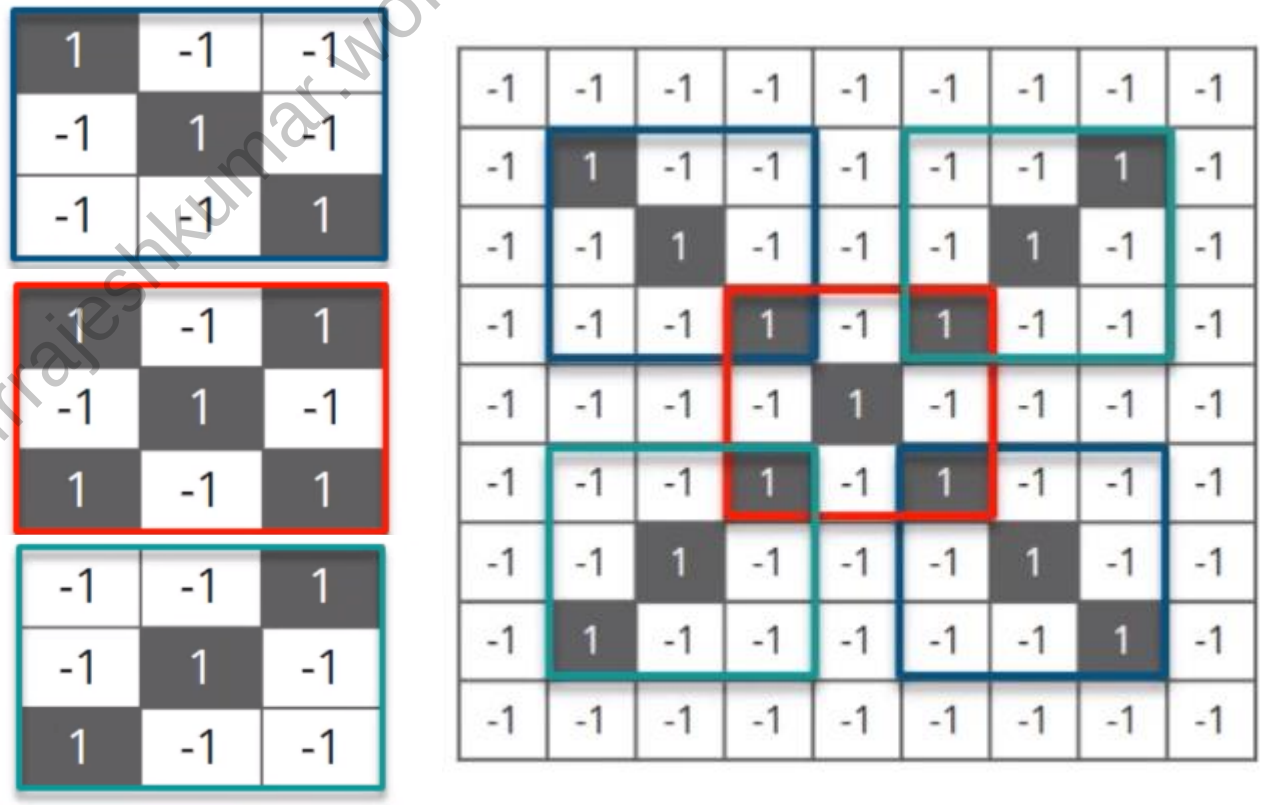
CNN compares the images piece by piece. The pieces that it looks for are called features. By finding rough feature matches in roughly the same position in two images, CNN gets a lot better at seeing similarity than whole-image matching scheme.



How CNN works?

We will be taking three features or filters as shown:

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

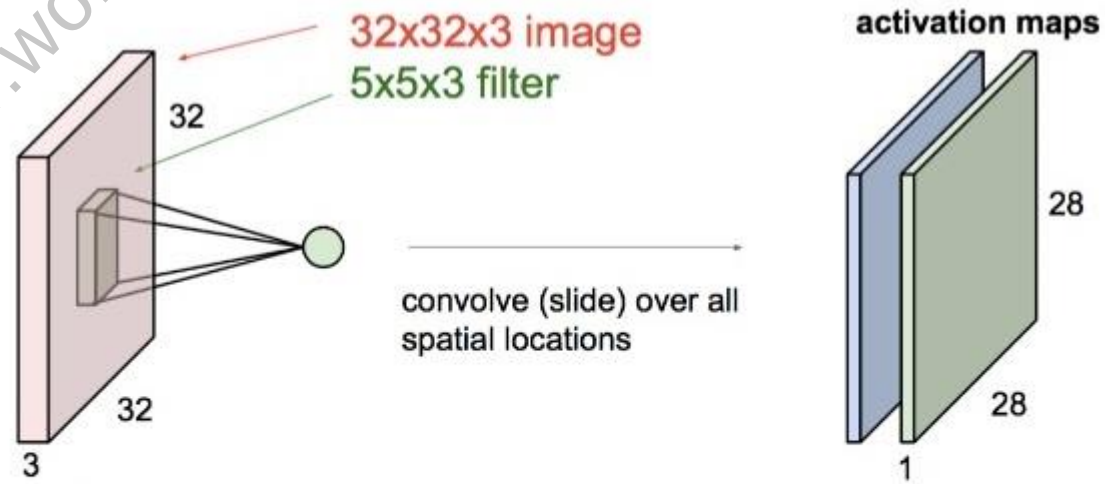


Convolutional Layer

CONVOLUTE THROUGHOUT THE IMAGE

Steps involved in Convolutional Layer

- Here we move the feature/filter to every possible position on the image.
- Steps involved in Convolutional Layer are as follows
  1. Line Up the image.
  2. Multiply each image pixel by corresponding feature pixel.
  3. Add them up.
  4. Divide by total number of pixel in the feature.



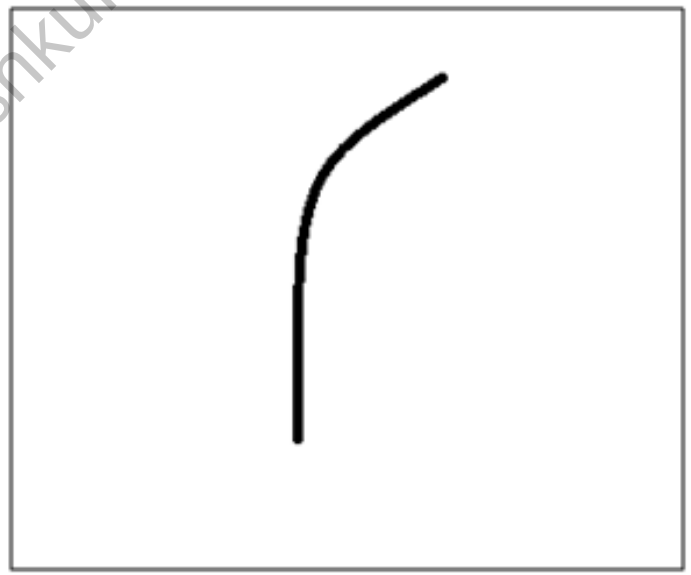
- **Convolution** (3-dim dot product) image and filter
- Stack filter in one layer (See blue and green output, called **channel**)

What are filters?

- Each of these filters can be thought of as **feature identifiers**
- By Feature, we here means things like straight edges, simple colors, and curves.

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

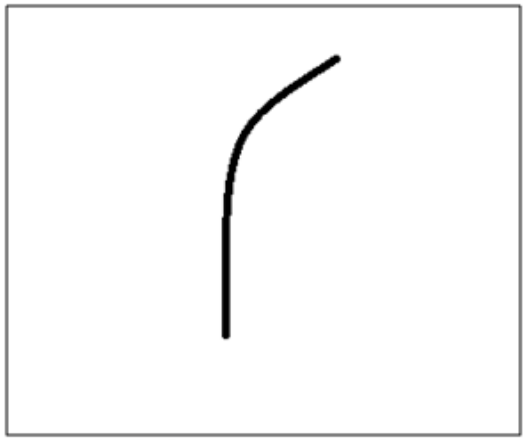


Visualization of a curve detector filter

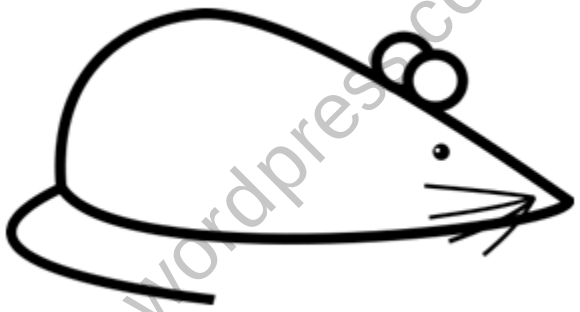
What are filters?

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

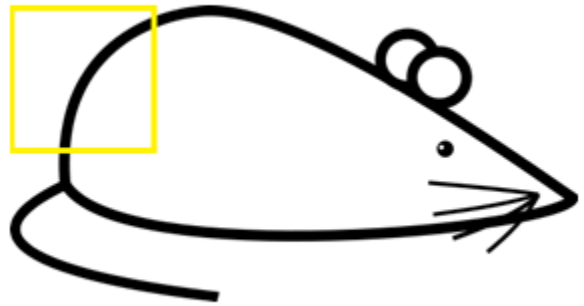
Pixel representation of filter



Visualization of a curve detector filter



Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

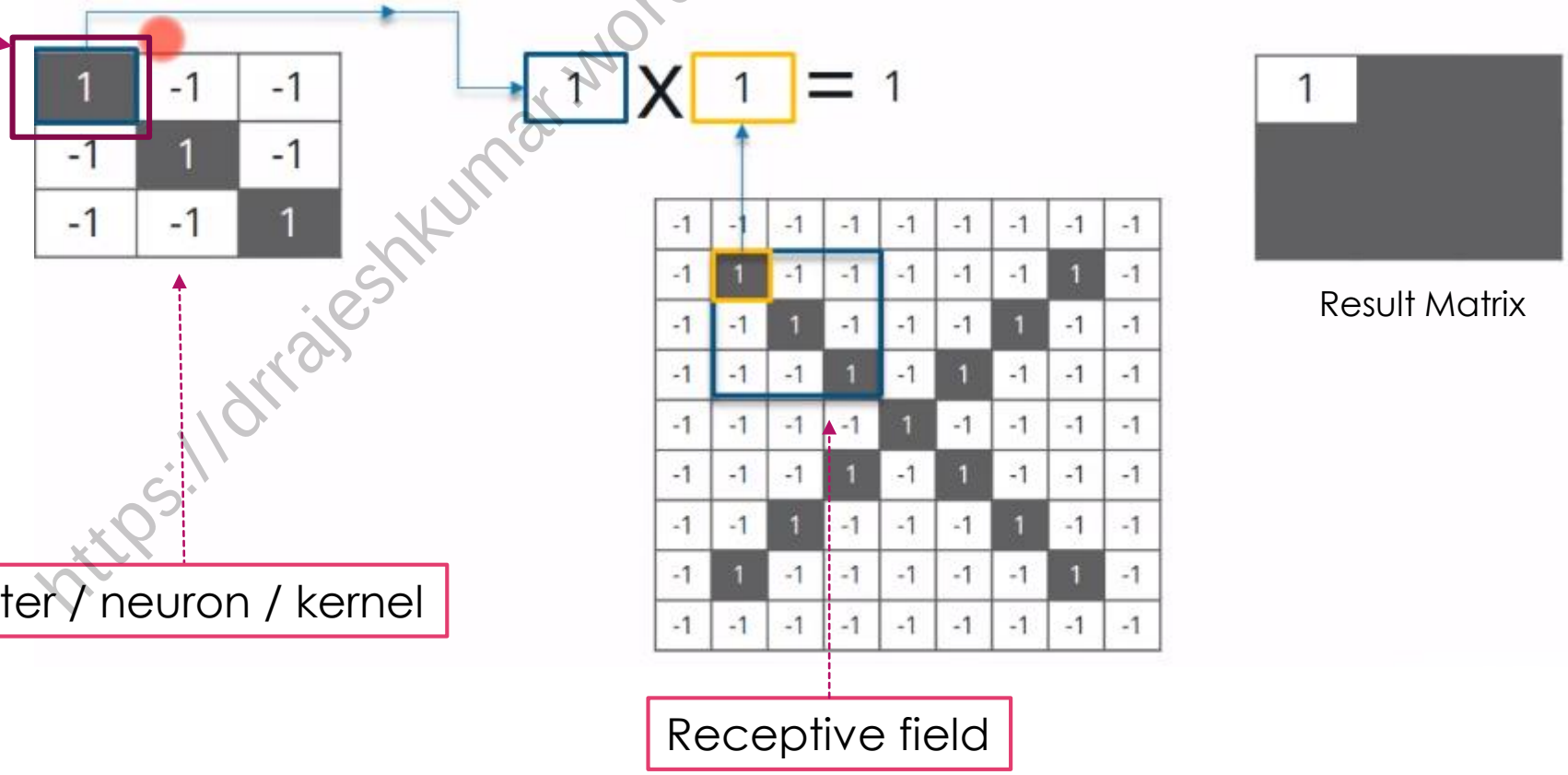
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Convolutional Layer

Weights/parameters

Filter pixels value are multiplied with the corresponding input image pixel value and is saved in another result matrix.



Convolutional Layer

Add and divide by total number of pixel.

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

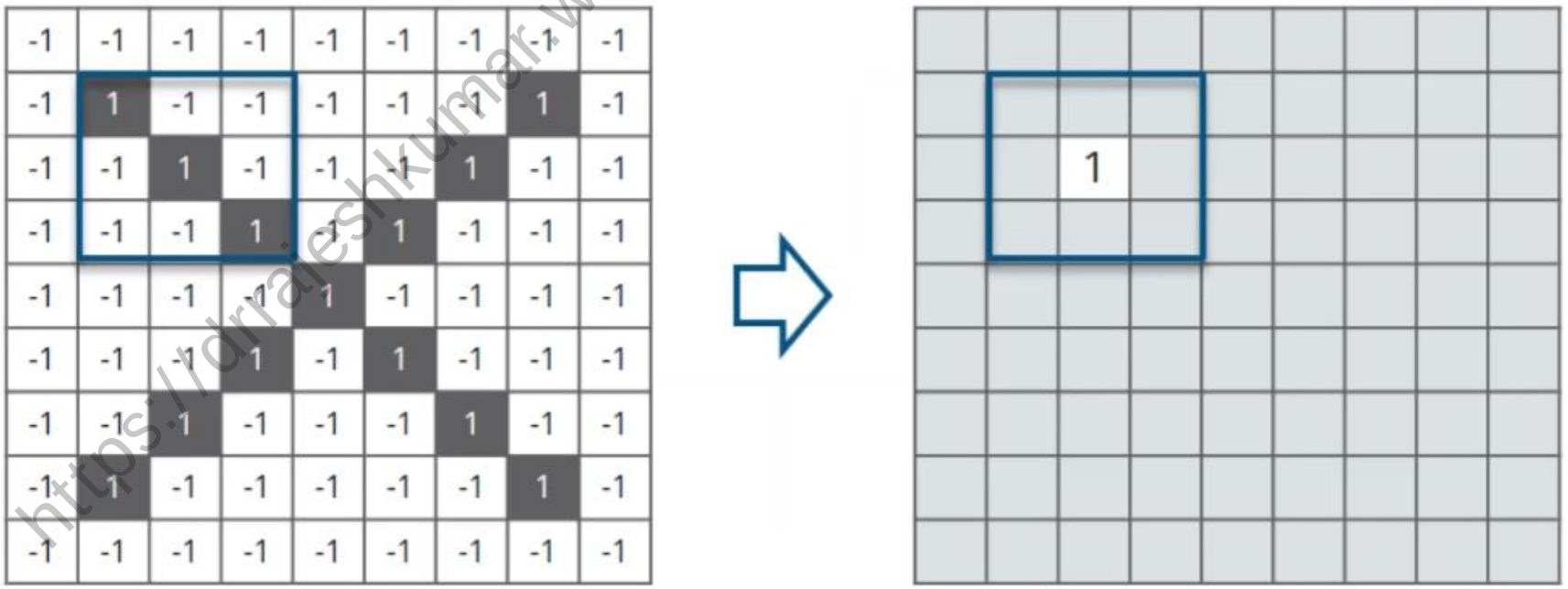
1	1	1
1	1	1
1	1	1

Result of the above Filter: 1

Convolutional Layer

Create a map to put value of filter

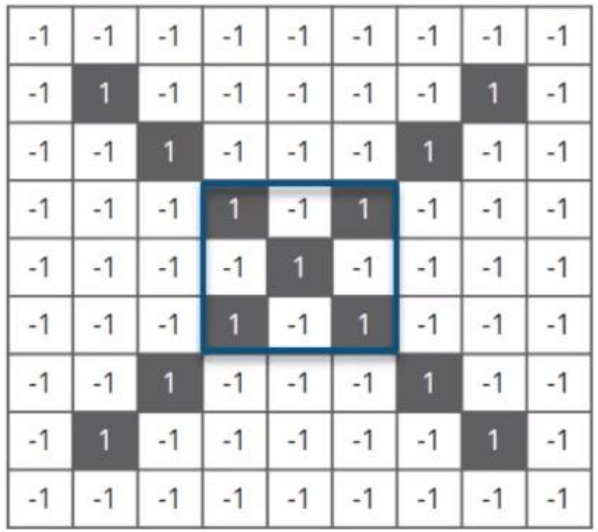
Now to keep track of where that feature was, we create a map and put value of the filter at the place.



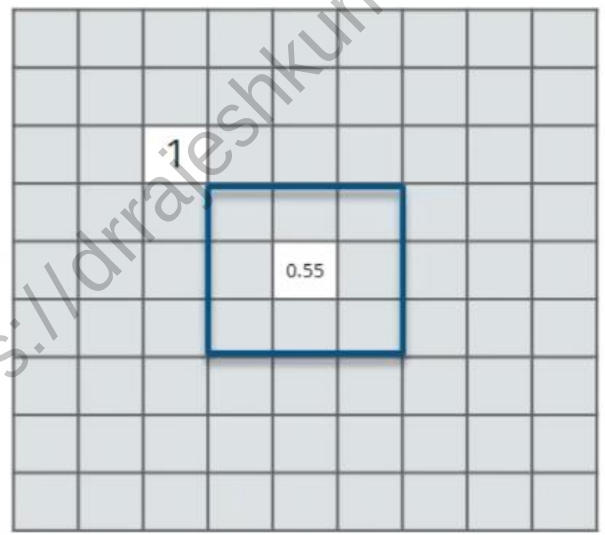
Feature Map

Convolutional Layer

Similarly we move the filter to every other positions of the image and will how the feature matches the area.



9 x 9 matrices

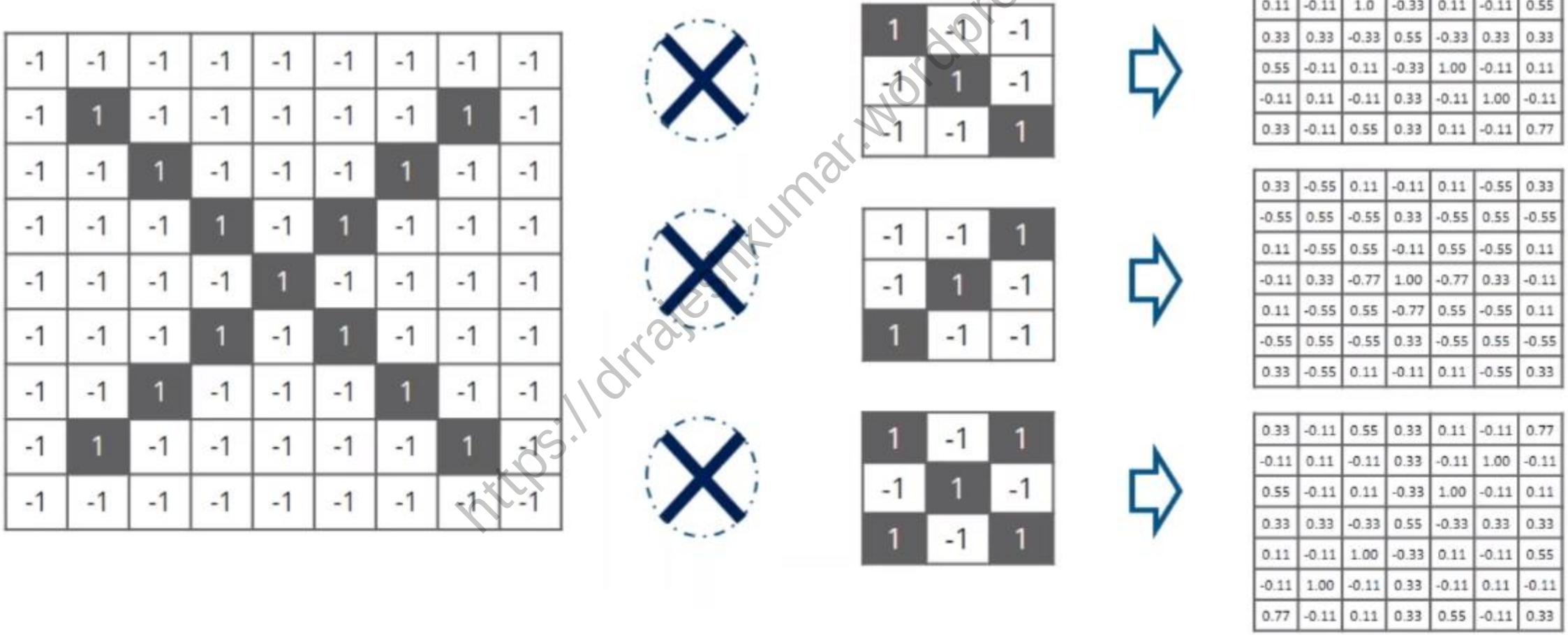


0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.0	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

(Activation/Feature Map : 7 x 7 Matrices)

Convolution Layer Output

After performing the same convolution, with ever filter, we get all the filter/feature maps.



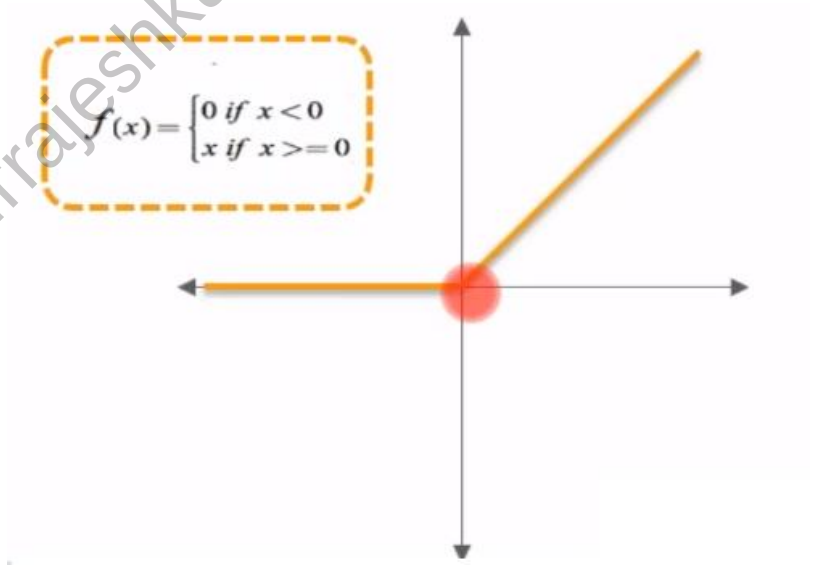
ReLU Layer

ACTIVATION FUNCTION

ReLU Layer – Activation Function

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

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



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

ReLU function applied on One feature

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

0.77	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	1.77

Negative values transformed to 0. We have to perform ReLU function for all features.

Pooling Layer

IN THIS LAYER WE SHRINK THE IMAGE STACK INTO SMALLER SIZE.

Pooling Layer

- POOL layer will perform a down sampling operation along the spatial dimensions (width, height), resulting in lower volume.
- In this layer we shrink the image stack into smaller size.
- Steps:
  1. Pick a window size (usually 2-3)
  2. Pick a stride (usually 2)
  3. Walk your window across filtered images.
  4. From each window, take maximum value

Symbol:

Pooling Layer

- Choose the highest value in the window 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	1.77

Max value

1			

Pooling Layer

Moving the windows across entire image

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	1.77

7 x 7 matrix

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

4 x 4 matrix

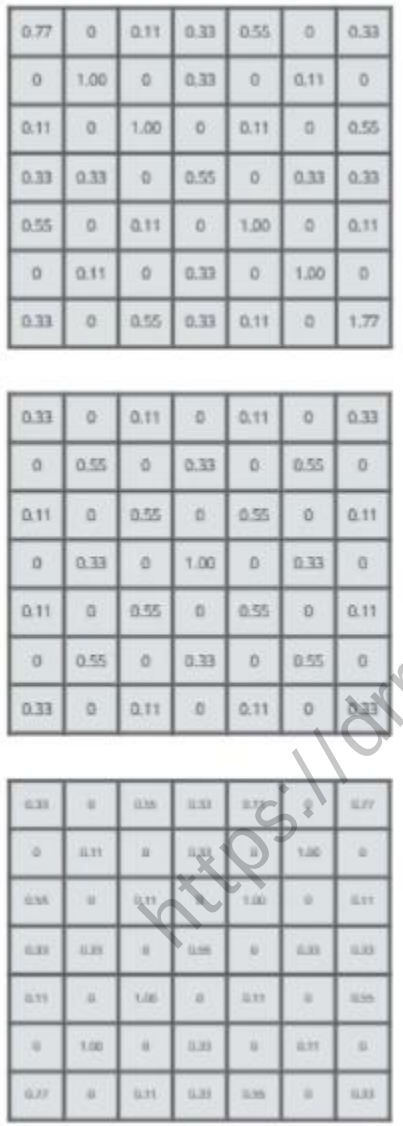
Output after passing through pooling layer.

The basic role of pooling layer is to shrink the size of our image matrix.

Here we have converted a 7x7 matrix to a 4x4 matrix.

Since we took 3 features in the beginning, we have 3 outputs after pooling layer.

Now next we have to stack up all the layers.

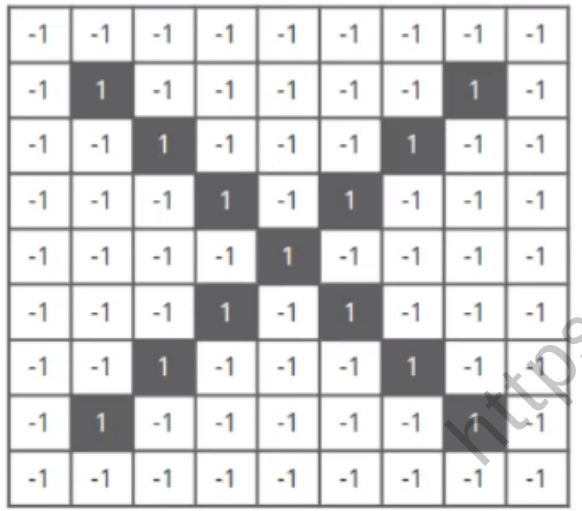


1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

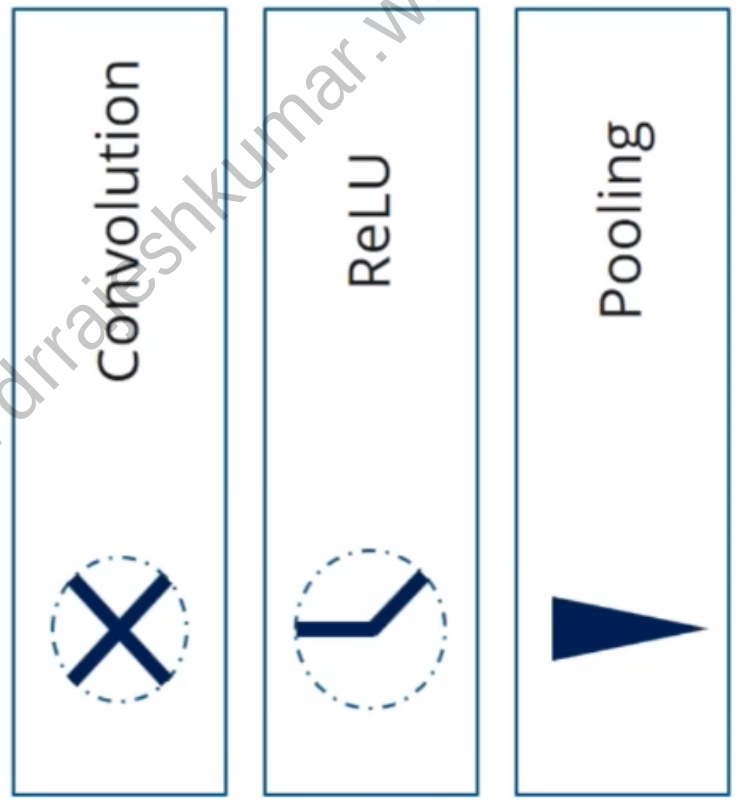
0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

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

Stacking up the layers



Input Image



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

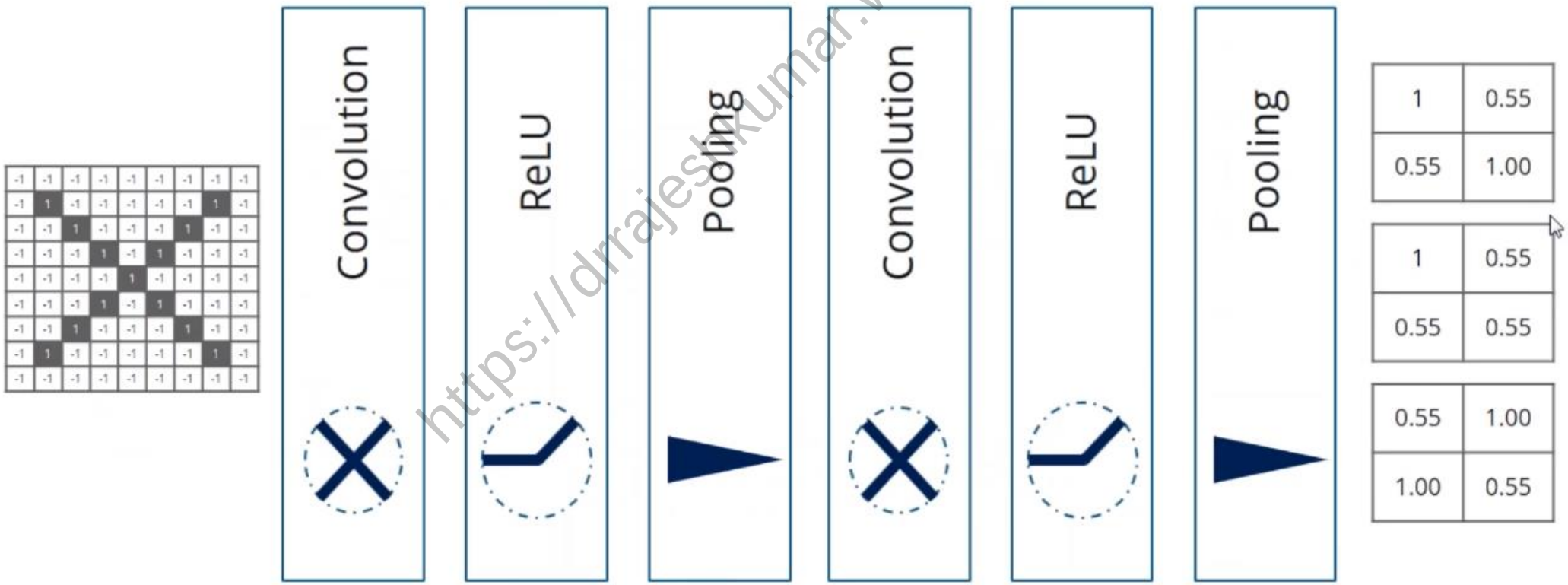
0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

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

4x4 matrices

Stacking up the layers

Here we have applied one more layer of each convolution, ReLU and pooling, in order to downsize 4x4 matrix to 2x2 matrices.



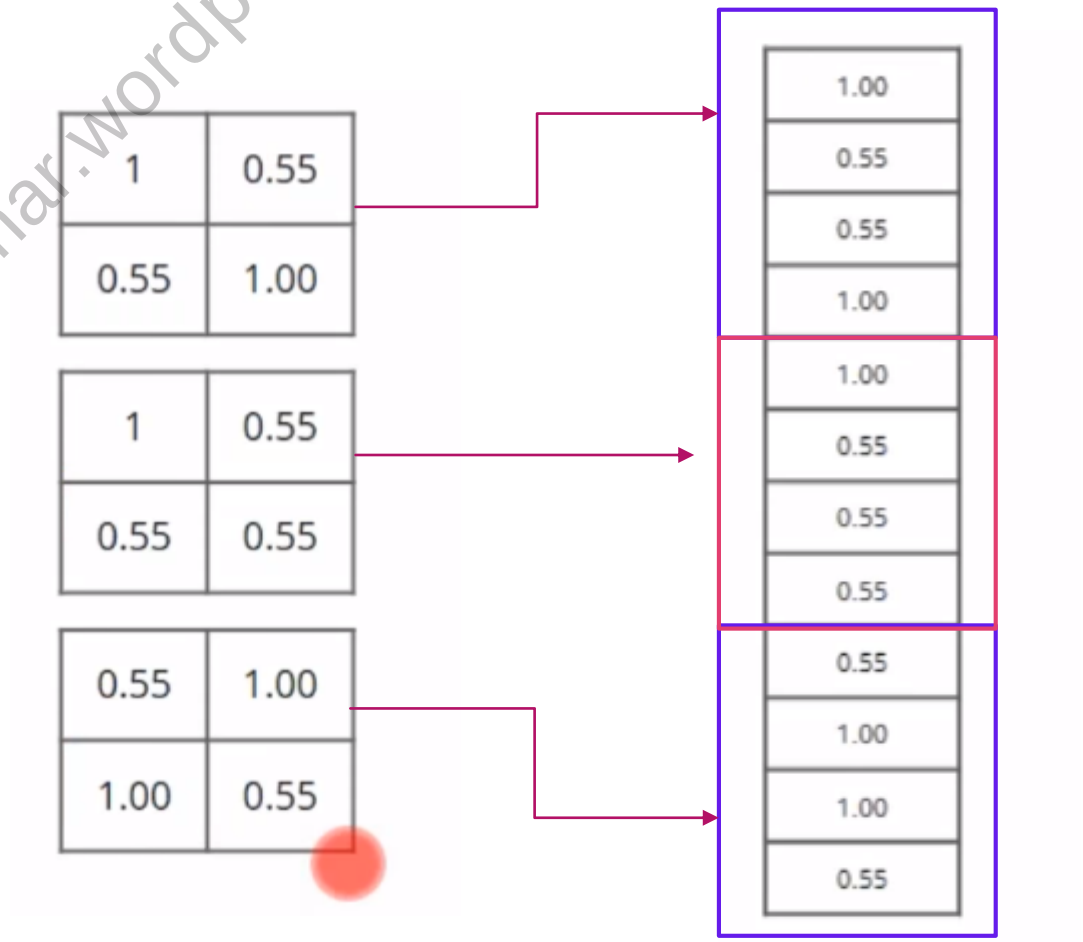
Fully Connected Layer

ACTUAL CLASSIFICATION TAKE PLACE IN FULLY CONNECTED LAYER.

Fully Connected layer

This is the final layer where the actual classification happens.

Here we take our filter and shrinked image and put them in into single list.



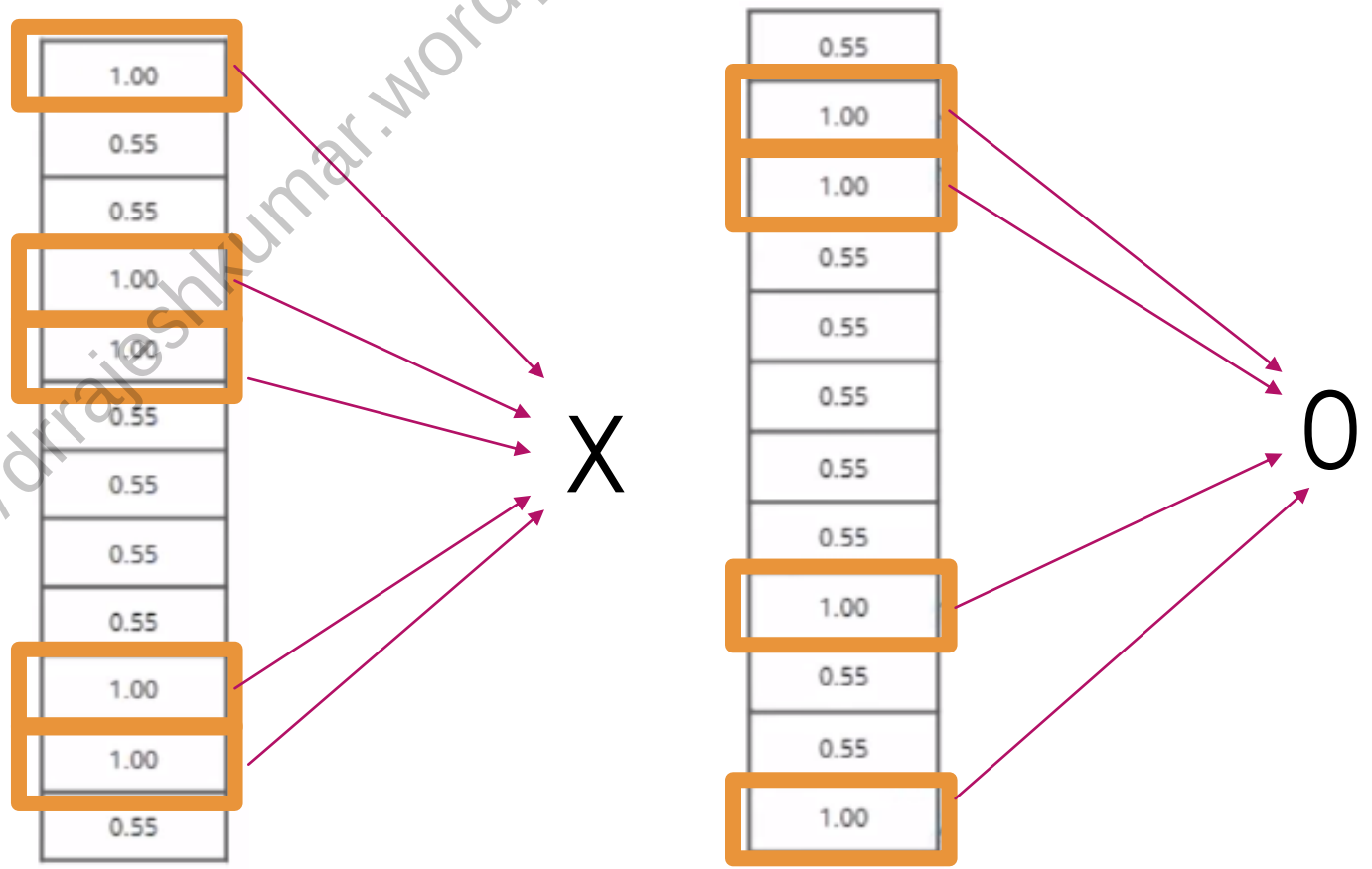
Output

When we feed in, ‘X’ and ‘O’. Then there will be some element that will be high.

Now if we have an input image with 1st, 4th,5th,10th,11th value high we can say that the image is X

Similarly, if we have an image with 2nd,3rd,9th and 12th value high, we can say that it is O

This completes the training of our model. Now lets see results on some unseen images.



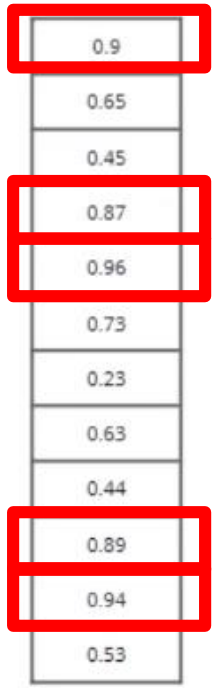
Prediction

After passing an input image through 4 layers of CNN, we have received the list of the image, we need to classify the image as X or O based on the model we learned earlier.

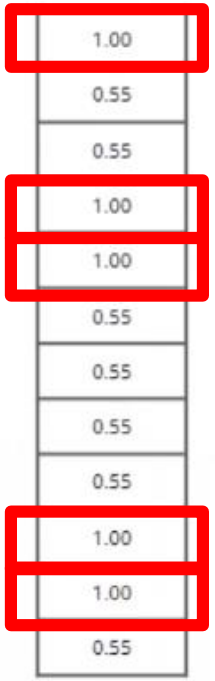
0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53

Let's compare with the list of 'X' and 'O'

Compare the input Vector with X

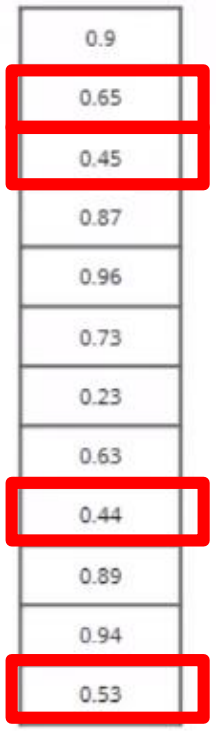


Input Image

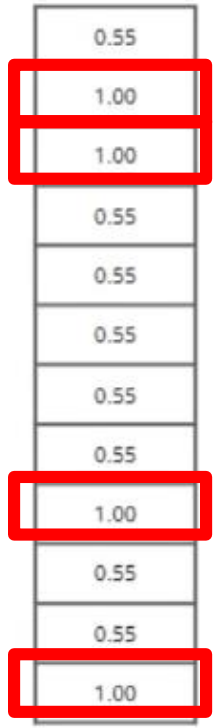


Vector for 'X'

Compare the input Vector with O

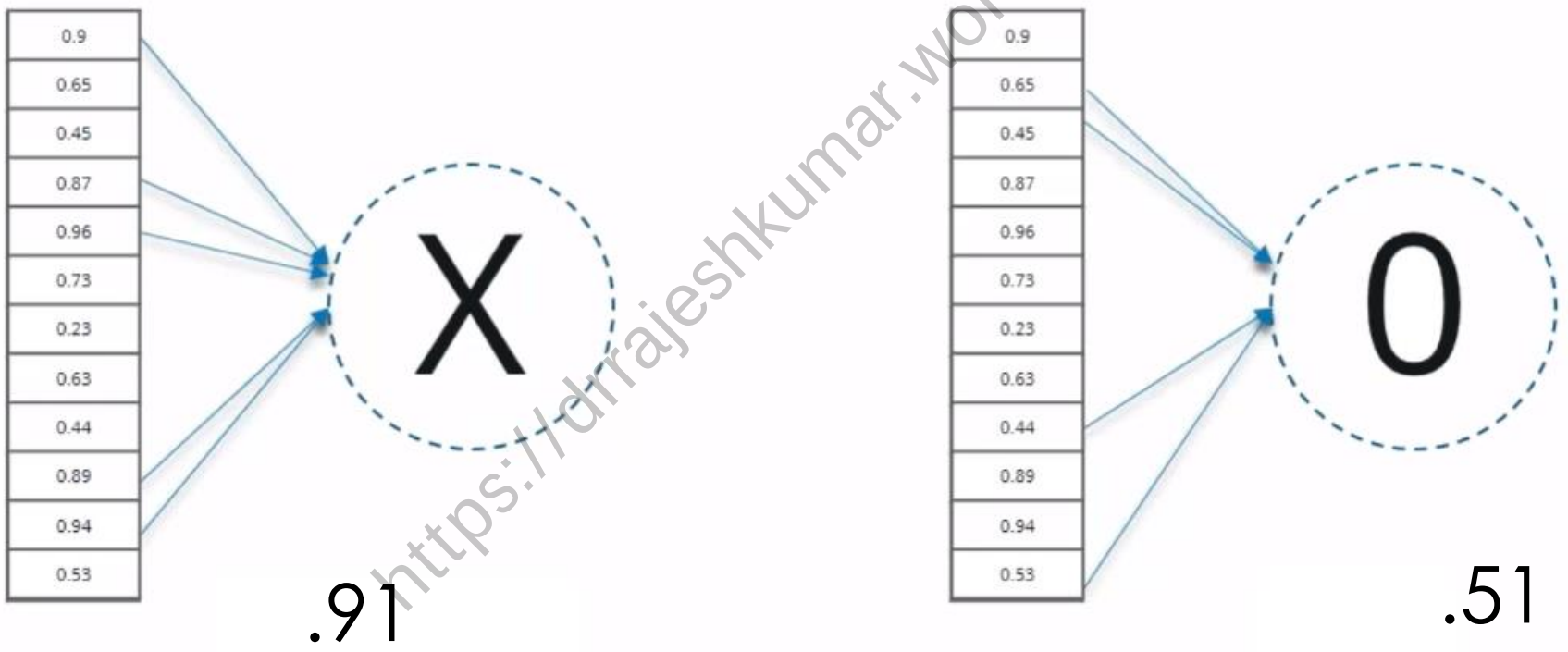


Input Image



Vector for 'O'

Result



The new input image is classified as 'X'

Use case

Similarly, If we train our model with dataset of Dogs and cats. Once training is done, our model will be able to predict a new unlabeled input correctly.

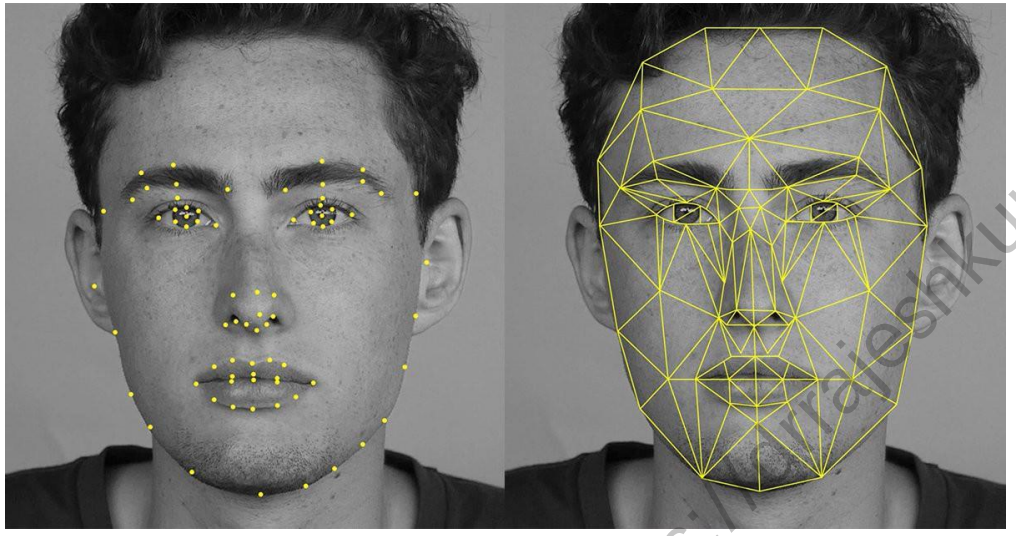
Applications of CNN

- Image processing and Character Recognition.

2

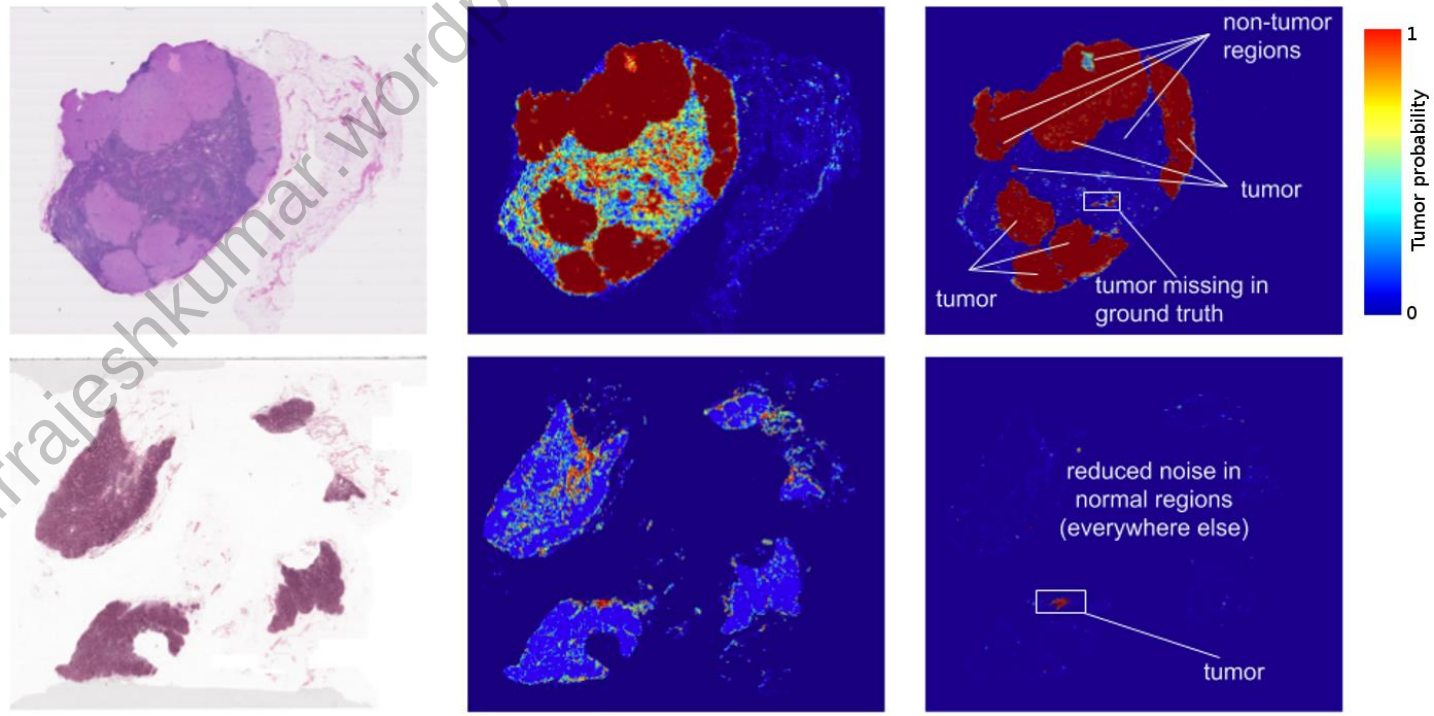
Applications of CNN

- Face Detection



Applications of CNN

- Cancer Detection

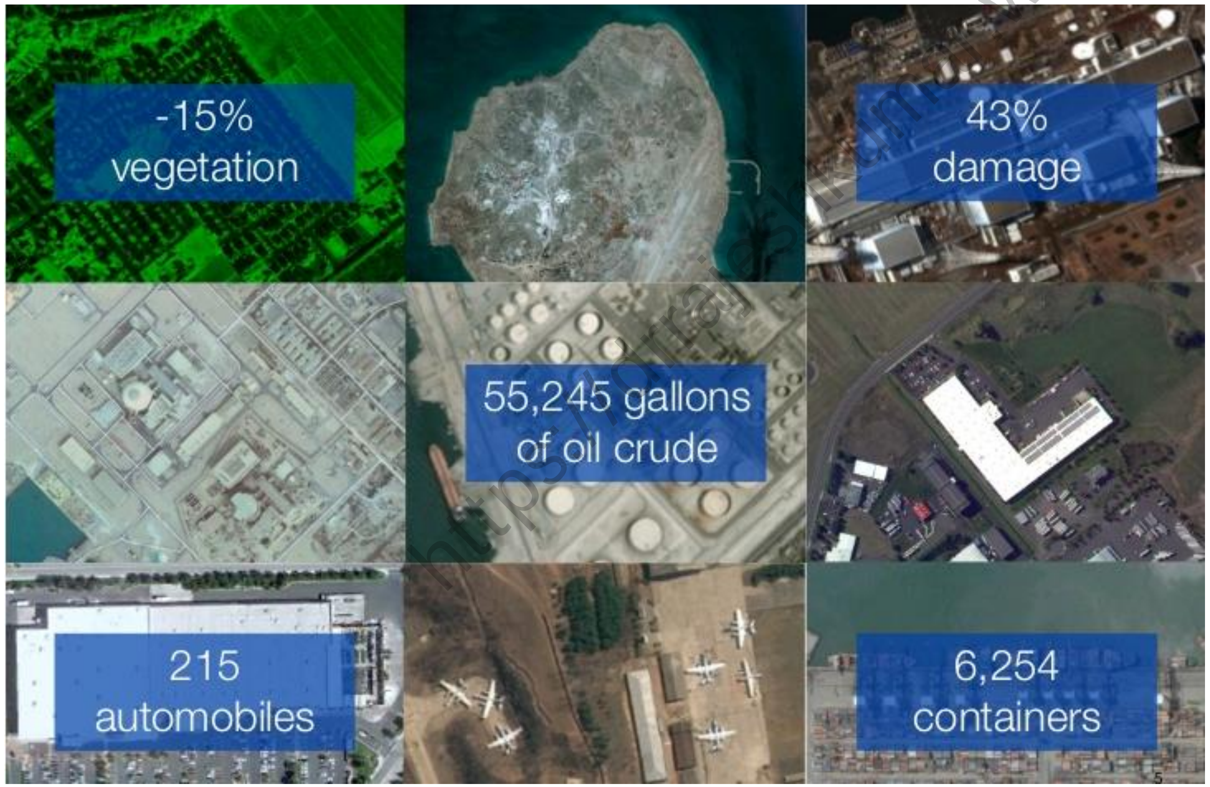


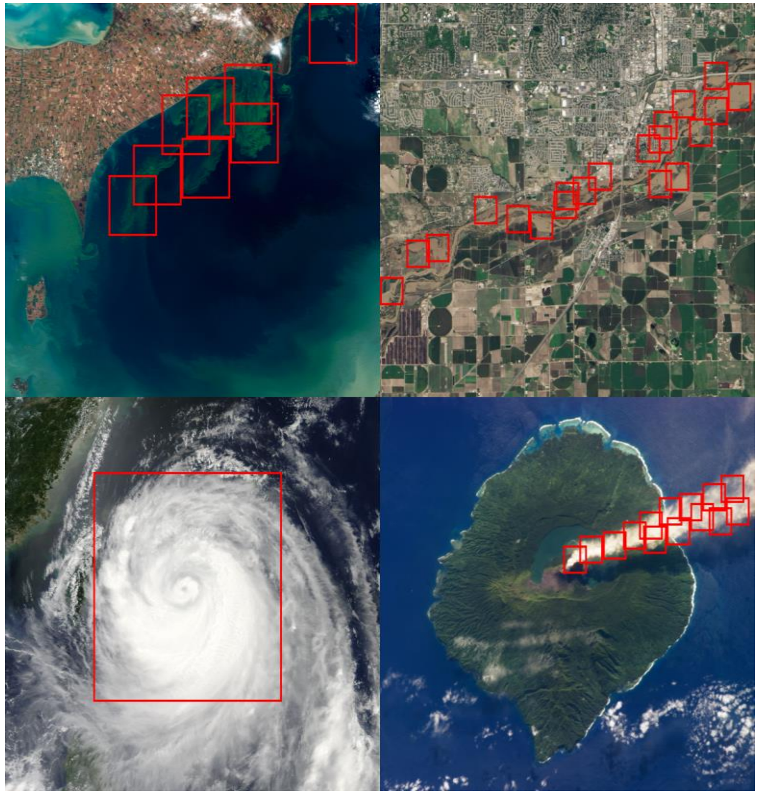
siliconANGLE

Google uses machine learning to detect breast cancer better than pathologists

Applications of CNN

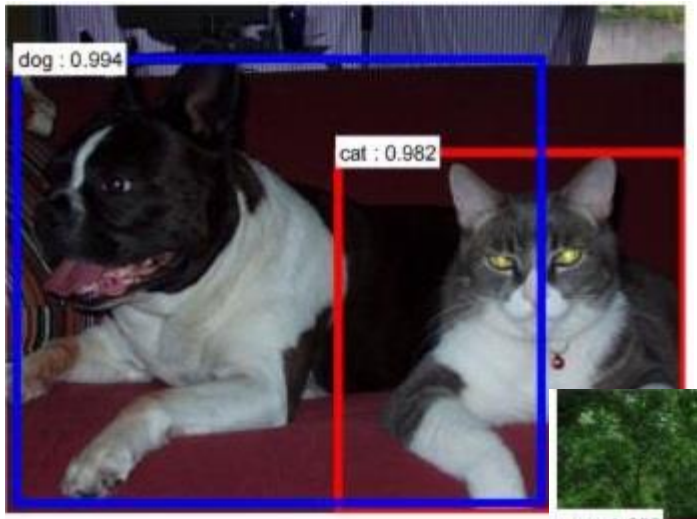
- Satellite imagery processing

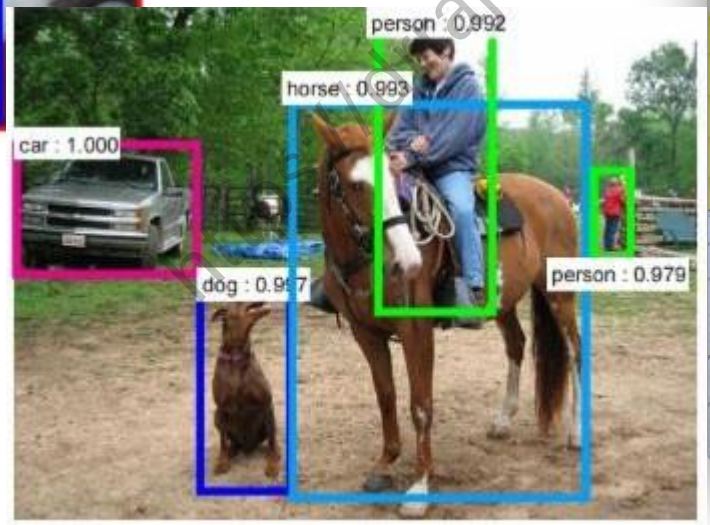


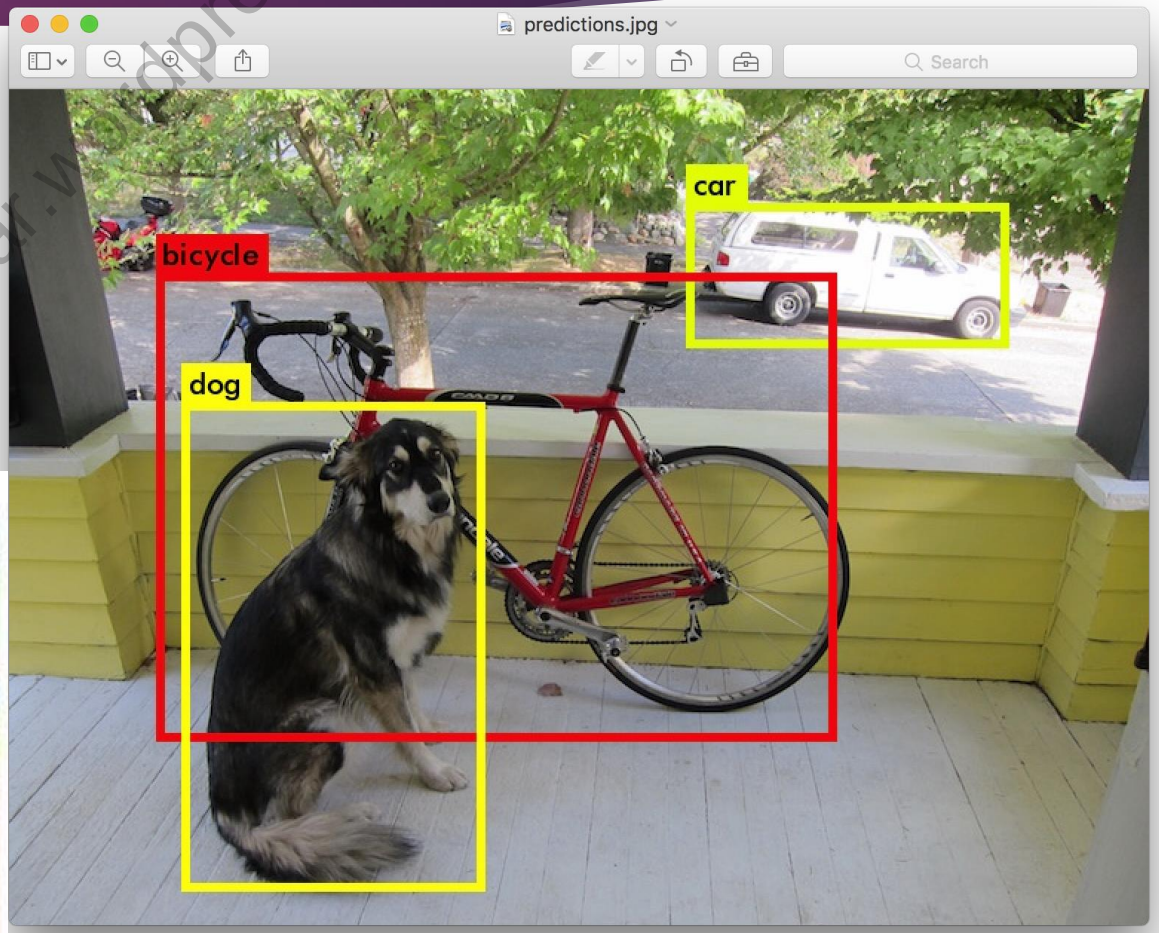


Applications of CNN

- Object detection







Thank you