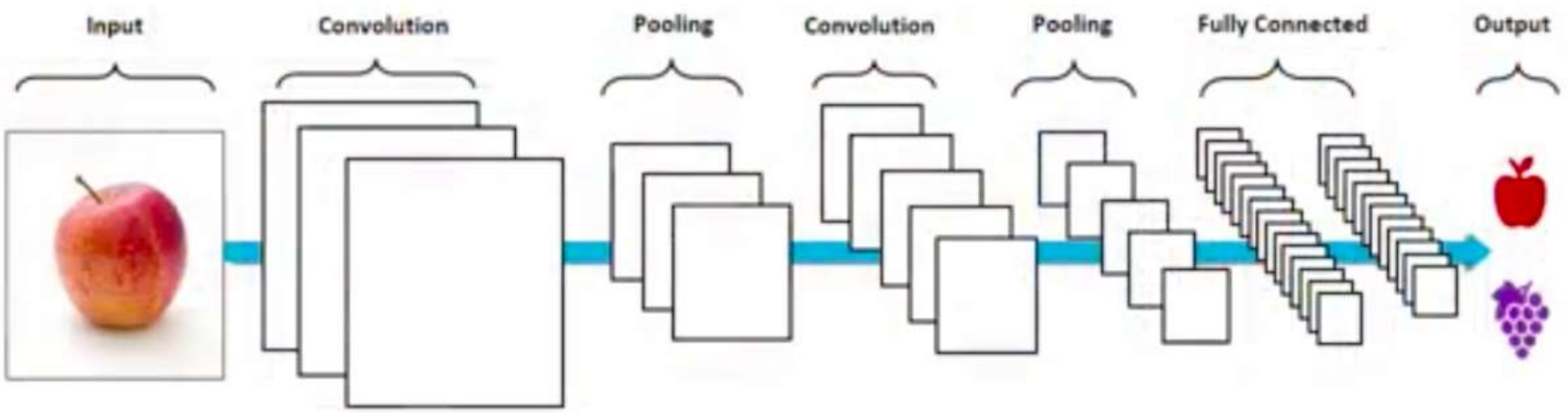
Convolutional Neural Network (CNN)

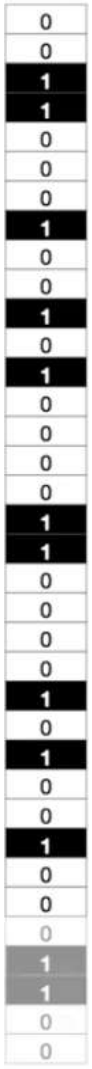
Lecture 08



6x6 grid of pixels into a single column of 36 input nodes

This image is so small, just 6 pixels by 6 pixels

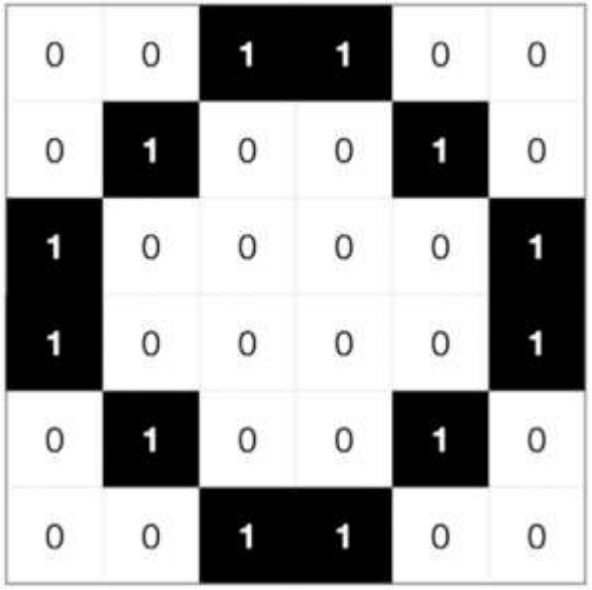
0	0	1	1	0	0
0	1	0	0	1	0
1	0	0	0	0	1
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0

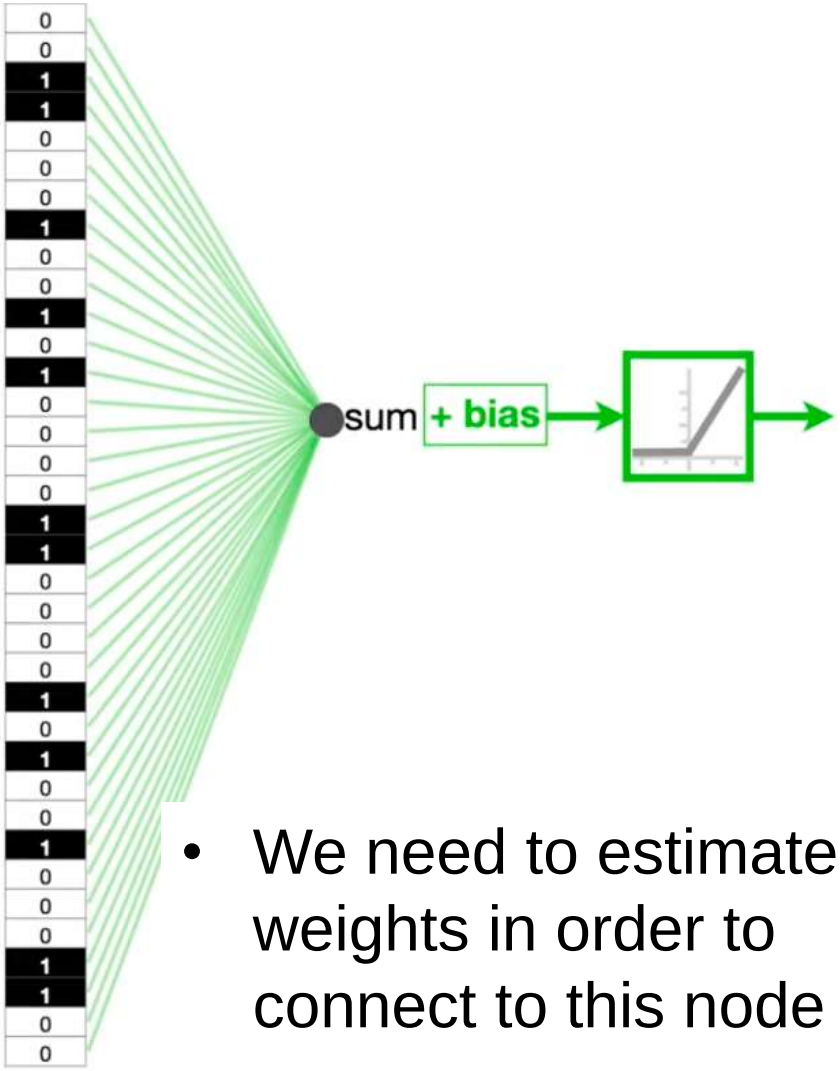


We simply convert this 6x6 grid of pixels into a single column of 36 input nodes

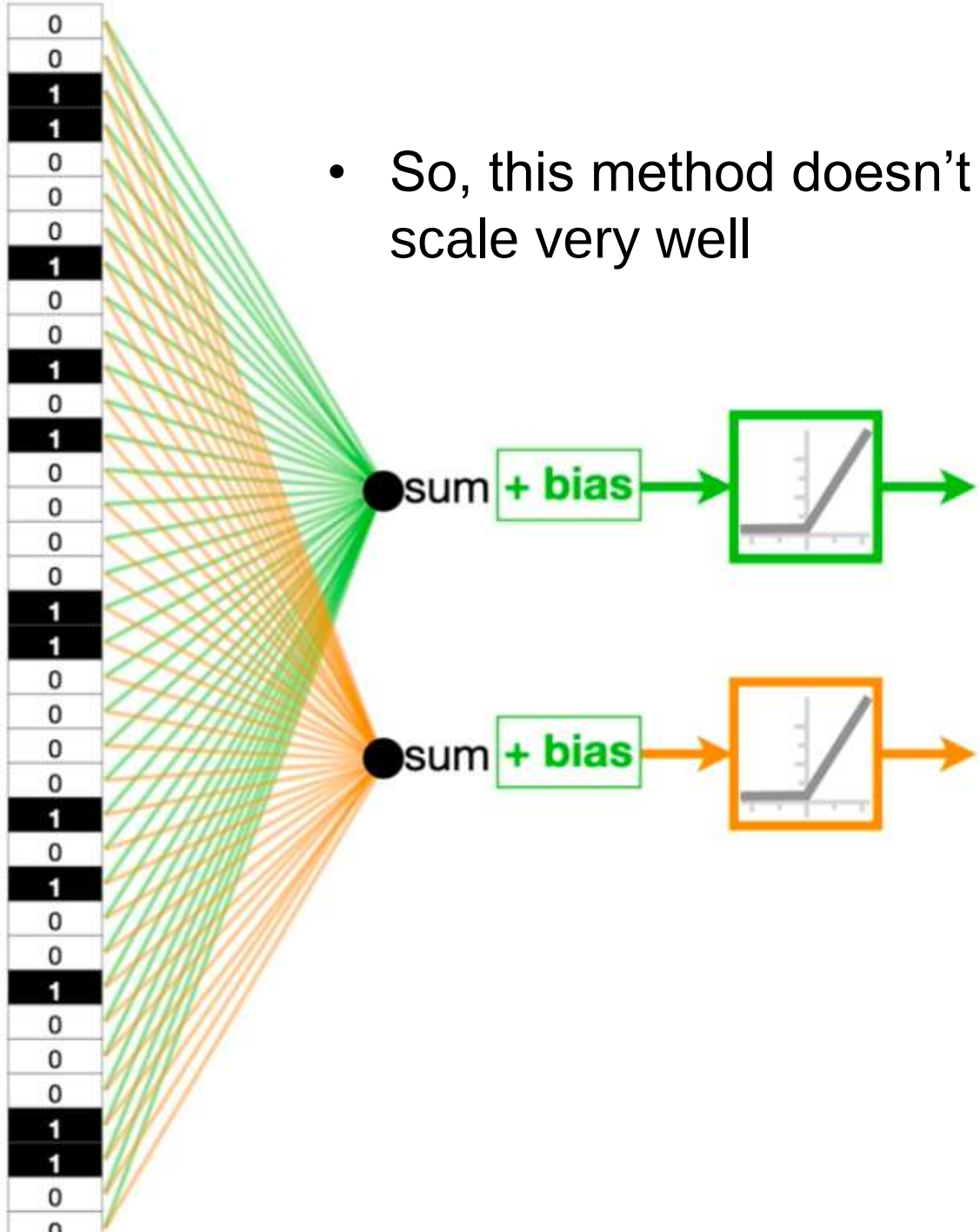
36 connections from the 36 input nodes to this node in the Hidden Layer

- Remember each connection has a weight that we have to extimate with backpropagation





- We need to estimate 36 weights in order to connect to this node



- So, this method doesn't scale very well

- Usually the first hidden layer has more than one node and each additional node adds an additional 36 Weights that we need to estimate
- The original image is small (6x6) and black and white
- However, if we had a larger image, like 100 pixels which is still pretty small compared to real world pictures.
- we would end up estimating 10,000 Weights per node in the Hidden Layer

Complicated images like this teddy bear tend to have correlated pixels

- For example, any brownish pixel in this image tends to be close to other brown pixels
- And any white pixel tends to be near other white pixels
- And it might be helpful if we can take advantage of the correlation that exists among each pixel
- Thus, classification of large and complicated images is usually done using something called convolution Neural Network

Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) do 3 things to make image classification practical:

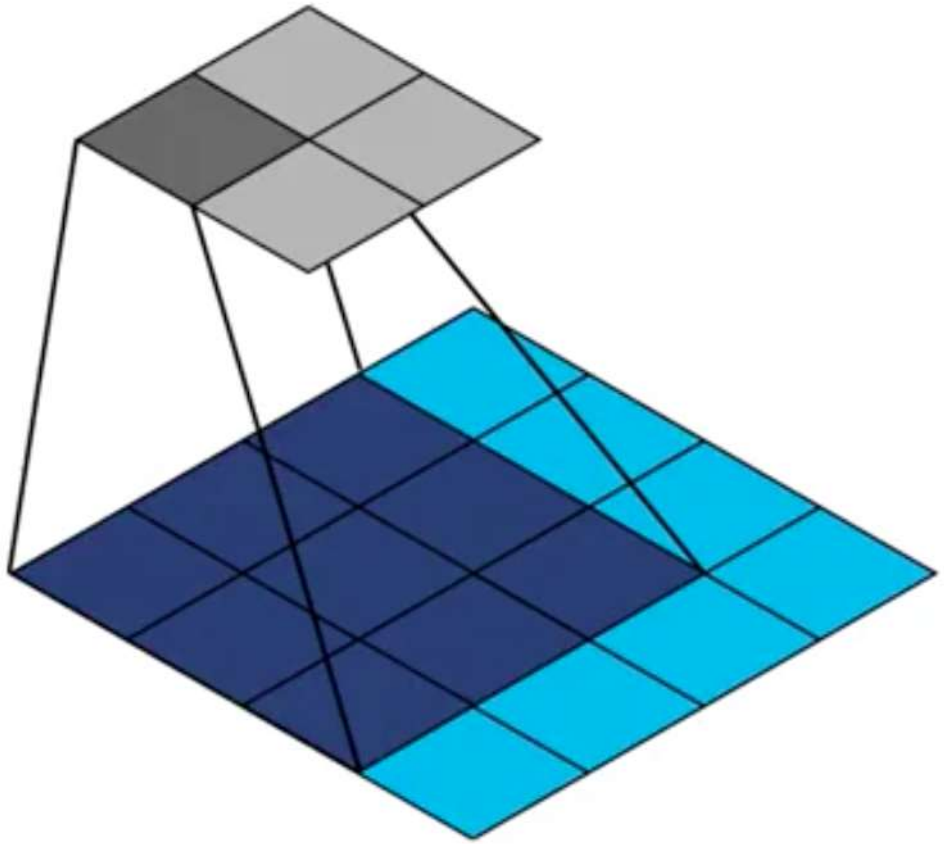
- Reduce the number of input nodes
- Tolerate small shifts in where the pixels are in the image
- Take advantage of the correlations that we observe in complex images

Convolutional layer of the CNN

- The convolutional layer of the CNN will use a filter to scan a portion of the receptive field for a particular visual features
- For example, a filter might search for a horizontal or vertical lines or other shapes within the image. There will also be a separate filter for each color channel
- A feature map is created after the input image has been scanned by a convolutional filter. The areas of the input image that match the features in the filter will be highlighted.
- There's normally going to be multiple filters that will be applied in parallel. When there is more than one filter a separate feature map is created for each filter and then they're stacked to create an output matrix of three dimensions.
- The feature map is then going to be scanned by the next convolutional layer to find even more complex patterns

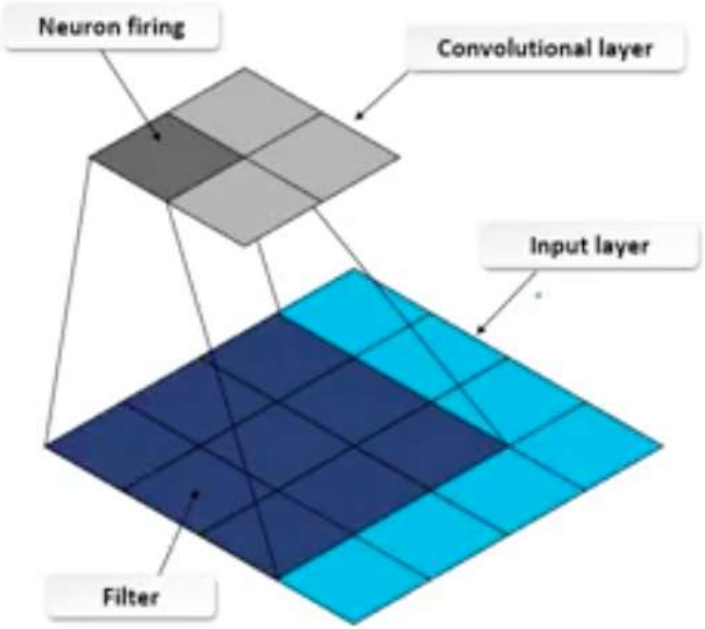
CNN Filters

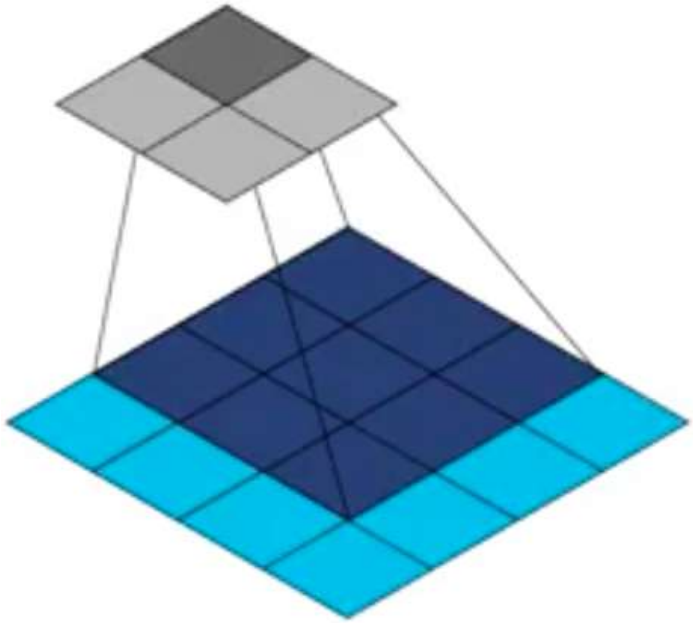
- Our convolutional layer is going to scan that image using a filter that has been defined as having a dimension of three by three.
- When that filter scans the input pixels, and each one of these boxes really represents a pixel, what will happen is we are going to be adding up weights. Those weights are going to be passed up to the convolutional layer where we're building our feature map.



CNN Filters

- We will move that filter across the image and as we do that we will be building up that feature map at the convolutional layer.
- If we look at this one step at a time you can see we start by scanning the top left corner of the image. That fires the neuron in the feature map of the convolutional layer in the top left corner. Then we move the filter to the right, and it triggers the neuron in the top right hand corner.





CNN Filters

- Looking at the graphic you can see that our input layer is made up of a five by five matrix, and the pixels in the layer are either turned off or on.
- CNN is going to use a filter to attempt to really detect the edges and the shapes within that image. Let's say that a filter has been generated through training
- When we apply this filter progressively to the input image what's going to happen is we're going to start to collect weights from that image that can then later be analyzed.

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

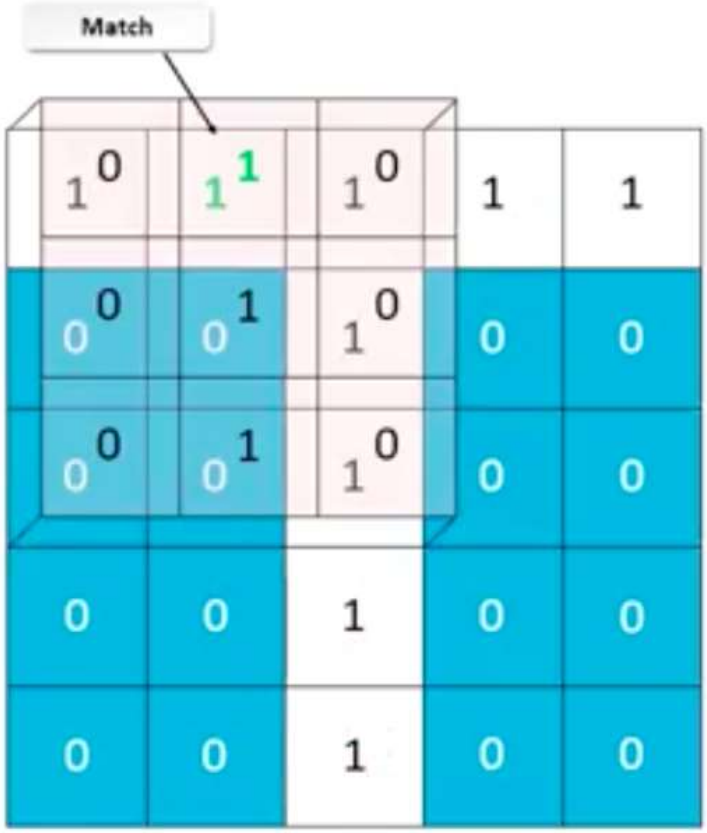
Input Layer

0	1	0
0	1	0
0	1	0

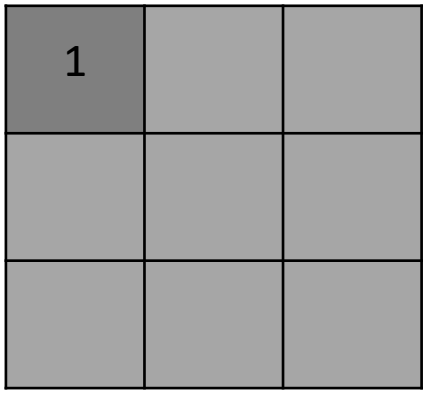
Filter

CNN Filters

- The convolutional layer is going to then scan the input image using that filter. You can see that we've got our filter here that has now been laid over top of the original input image.
- What we're looking for are any areas where a value of one in the filter matches a value of one in the original input image.



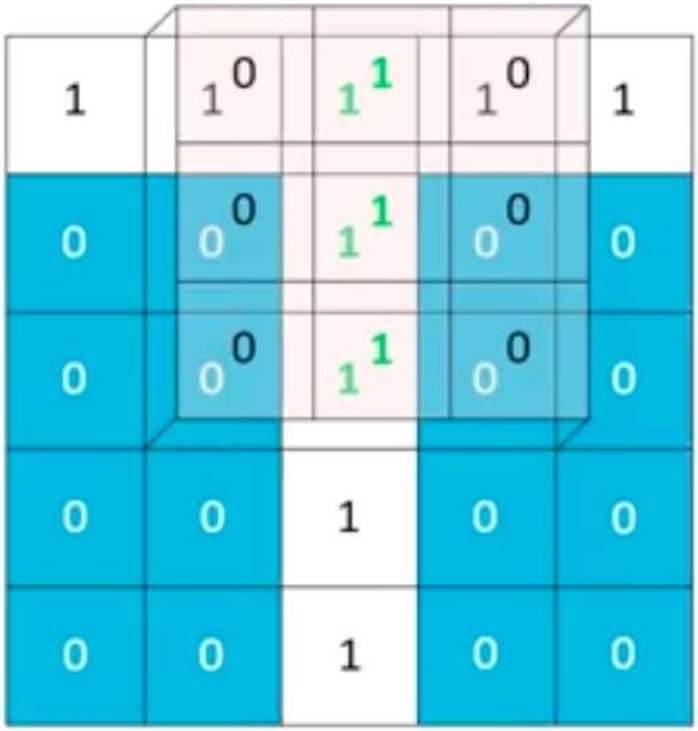
Convolution Layer with Neuron Firing



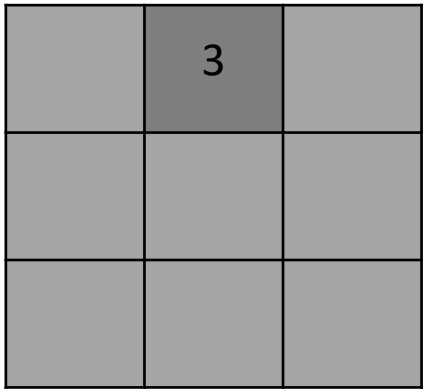
Feature Map

CNN Filters

- This time you can see that there are three pixels that match the filter, and so this neuron would end up with a weight of three associated with it.
- Once the scanning process completes we would end up with a feature map really that would be based upon those weights
- Once the filter has been applied to the entire convolutional layer the values of each pixel and the inputs have been combined together, we pass that all through an activation function to calculate our feature weights.

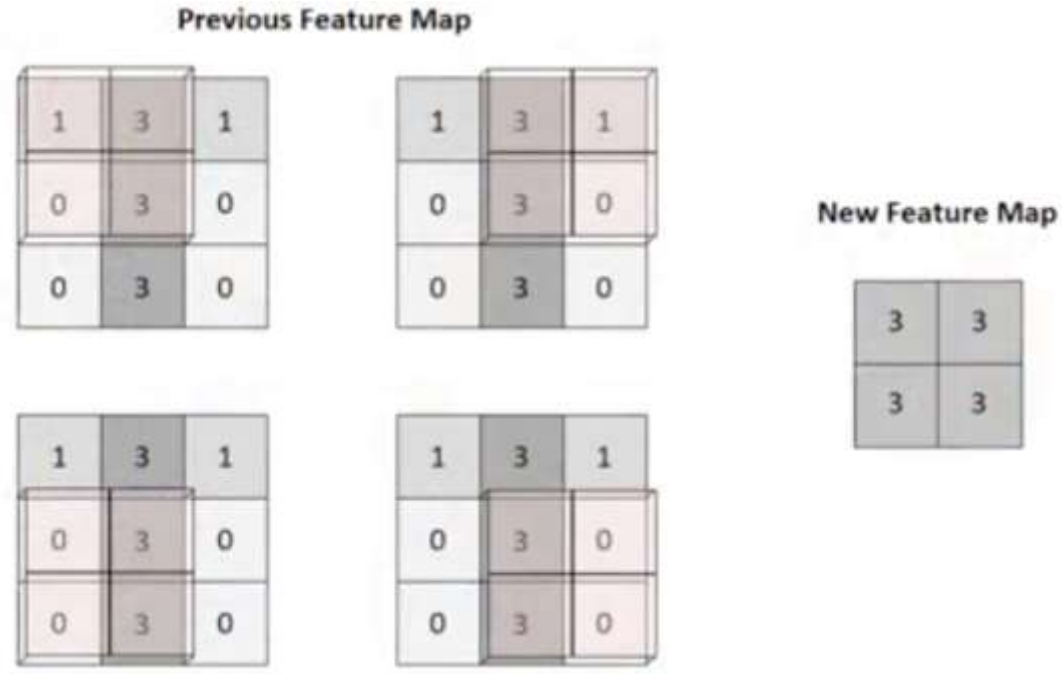


Convolution Layer with Neuron Firing



Feature Map

Max Pooling


Previous Feature Map
1 3 1
0 3 0
0 3 0
1 3 1
0 3 0
0 3 0
1 3 1
0 3 0
0 3 0
1 3 1
0 3 0
0 3 0
New Feature Map
3 3
3 3

Edge detection in CNN

let's take a 6 X 6 grayscale image (i.e. only one channel):

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Next, we convolve this 6 X 6 matrix with a 3 X 3 filter:

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	**1**	3
0	**1**	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

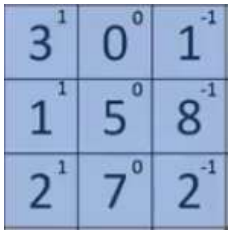
6 X 6 image

*

1	0	-1
1	0	-1
1	0	-1

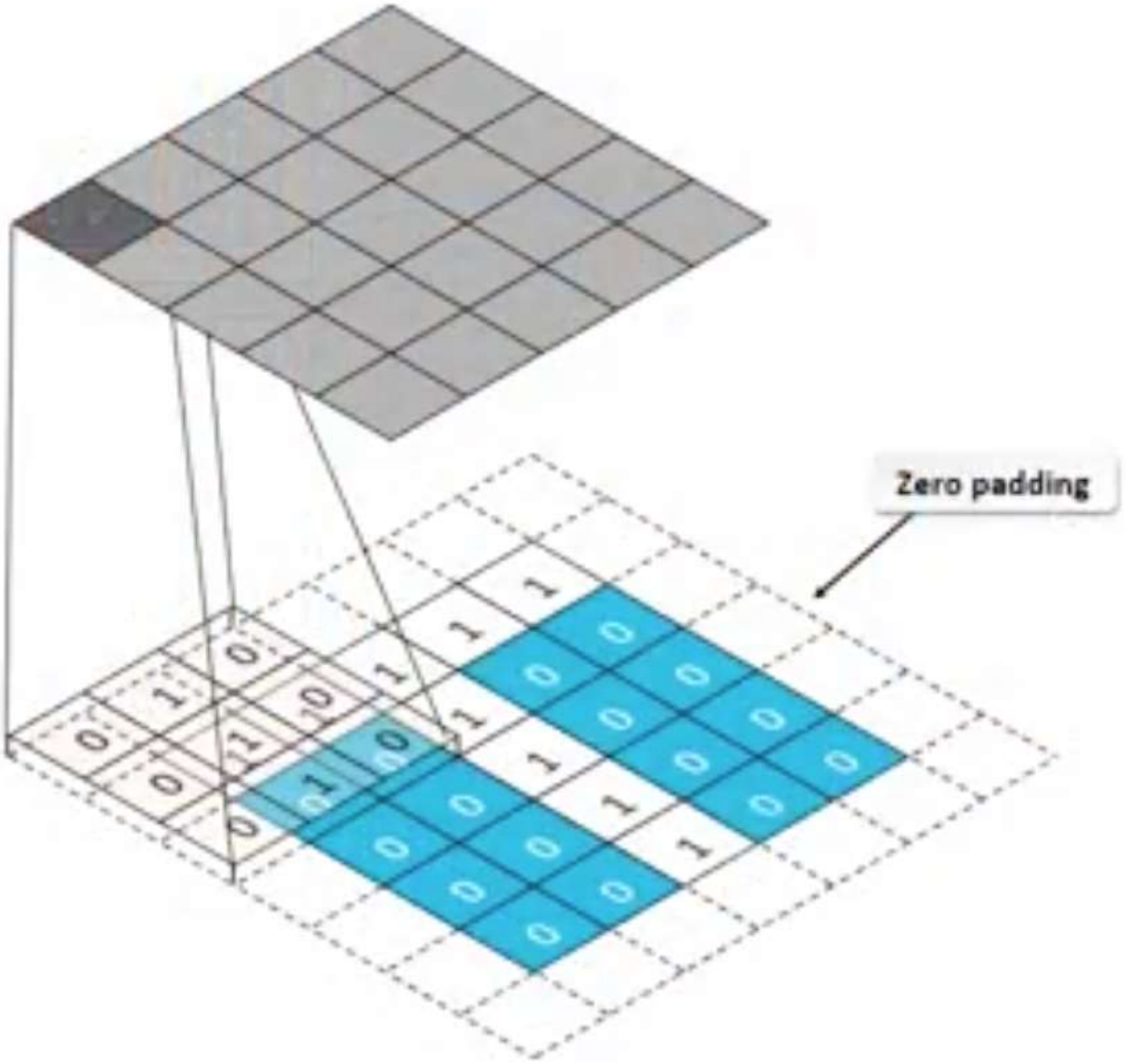
3 X 3 filter

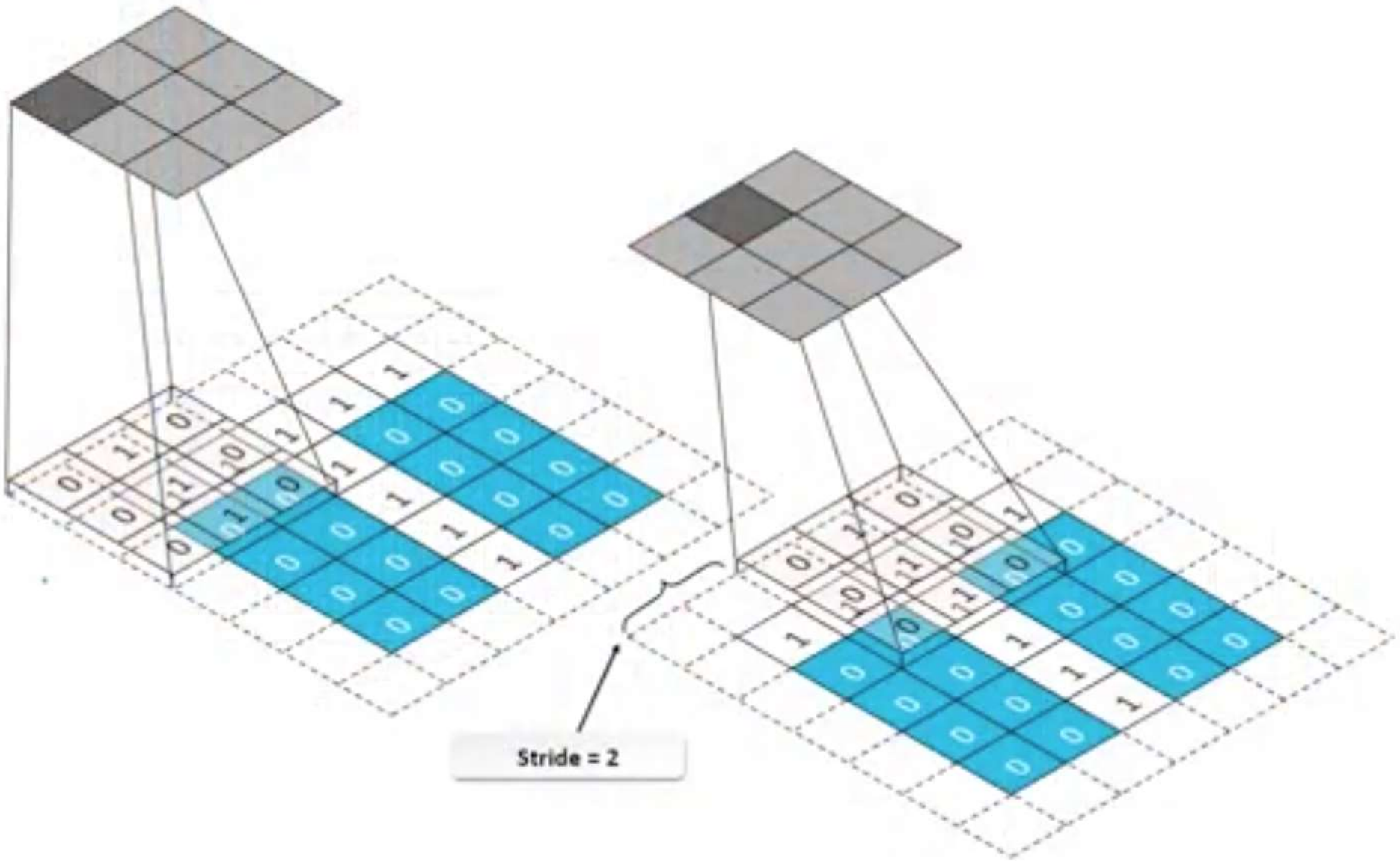
After the convolution, we will get a 4 X 4 image. The first element of the 4 X 4 matrix will be calculated as:



Similarly, we will convolve over the entire image and get a 4 X 4 output:

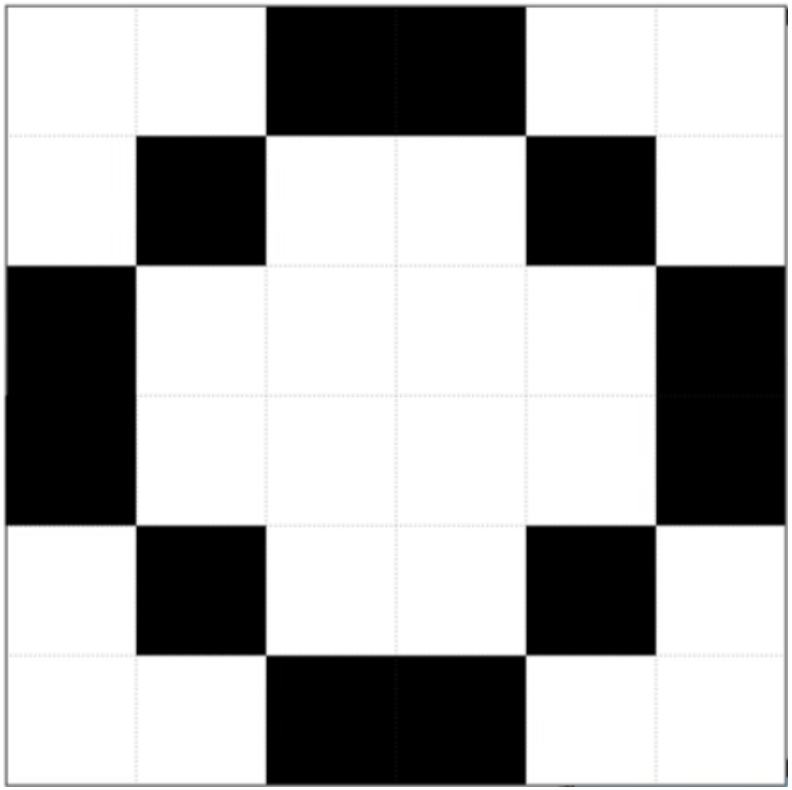
-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16


Zero padding

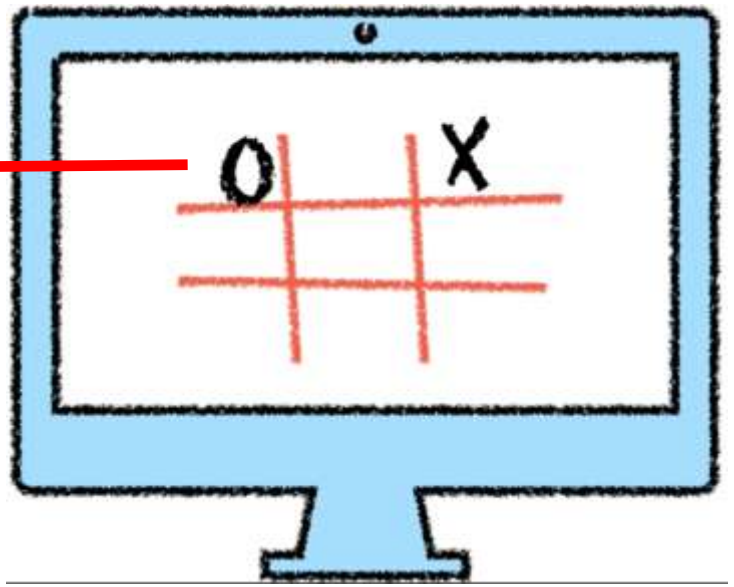

Stride = 2

How a Convolution Neural Network can recognize this letter “O”

The letter “O” zoomed in



Tick Tack Toe game



Convolutional Neural Networks (CNN)

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

Input Image

1	0	1
0	0	1
0	1	0

Filter

Apply Filter to the input image, we overlay the filter onto the image and then we multiply together each overlapping pixel and then we add each product together

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

0	0	1
0	1	0
1	0	0

(0 x 0) + (0 x 0) + (0 x 0) + (1 x 1)
+ (0 x 0) + (1 x 1) + (0 x 0) +
(1 x 1) + (0 x 0) +(0 x 0)
= 3

By computing the Dot Product between the input and Filter, we can say that the filter is convolved with the input and that's what gives Convolution Neural Network

Convolutional Neural Networks (CNN)

Now we add a Bias term to the output of the Filter and put the final value into something called a Feature Map

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

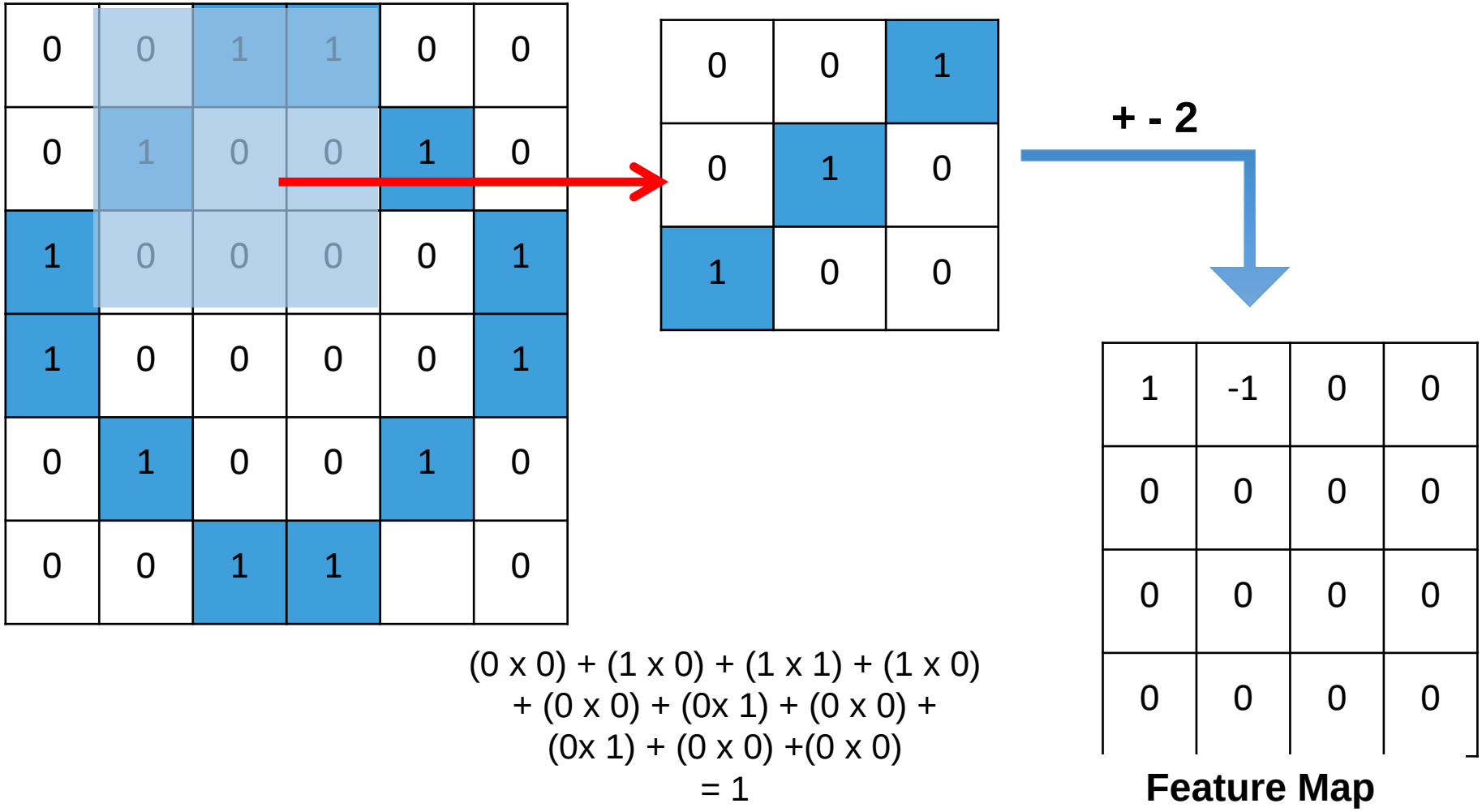
0	0	1
0	1	0
1	0	0

+ (-2)

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

(0 x 0) + (0 x 0) + (0 x 0) + (1 x 1)
+ (0 x 0) + (1 x 1) + (0 x 0) +
(1 x 1) + (0 x 0) +(0 x 0)
= 3

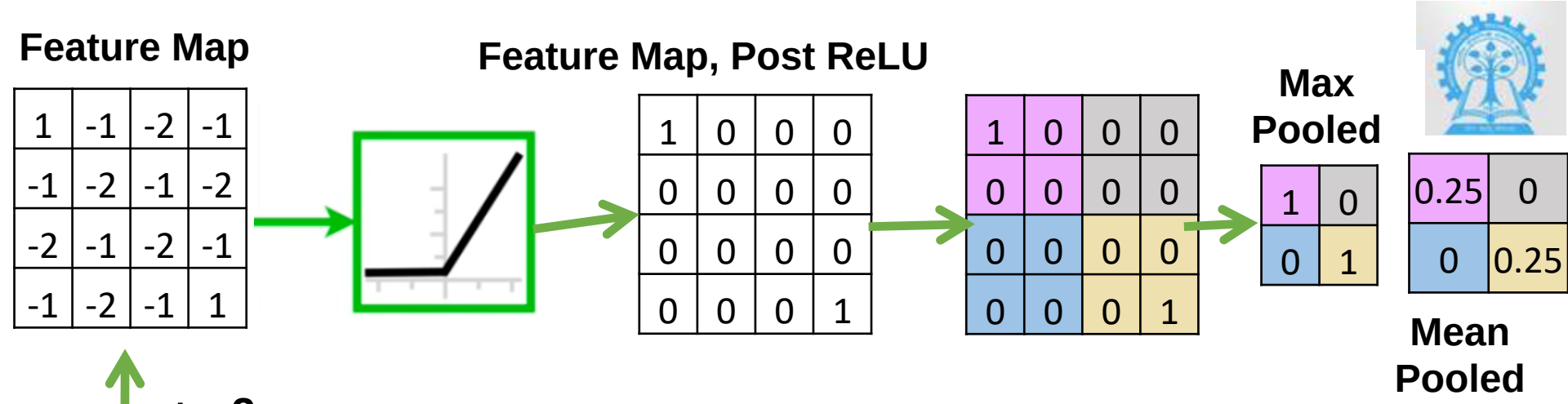
- However, other Convolution Neural Networks might move over 2 or more pixels but in this example, we just move over one pixel and calculate the Dot Product of the Filter and overlapping pixels in the image, add the Bias term and put the final value into the Feature Map
- Then we shift the Filter over again and repeat until we have filled up the Feature Map



Convolutional Neural Networks (CNN)

1	-1	-2	-1
-1	-2	-1	-2
-2	-1	-2	-1
-1	-2	-1	1

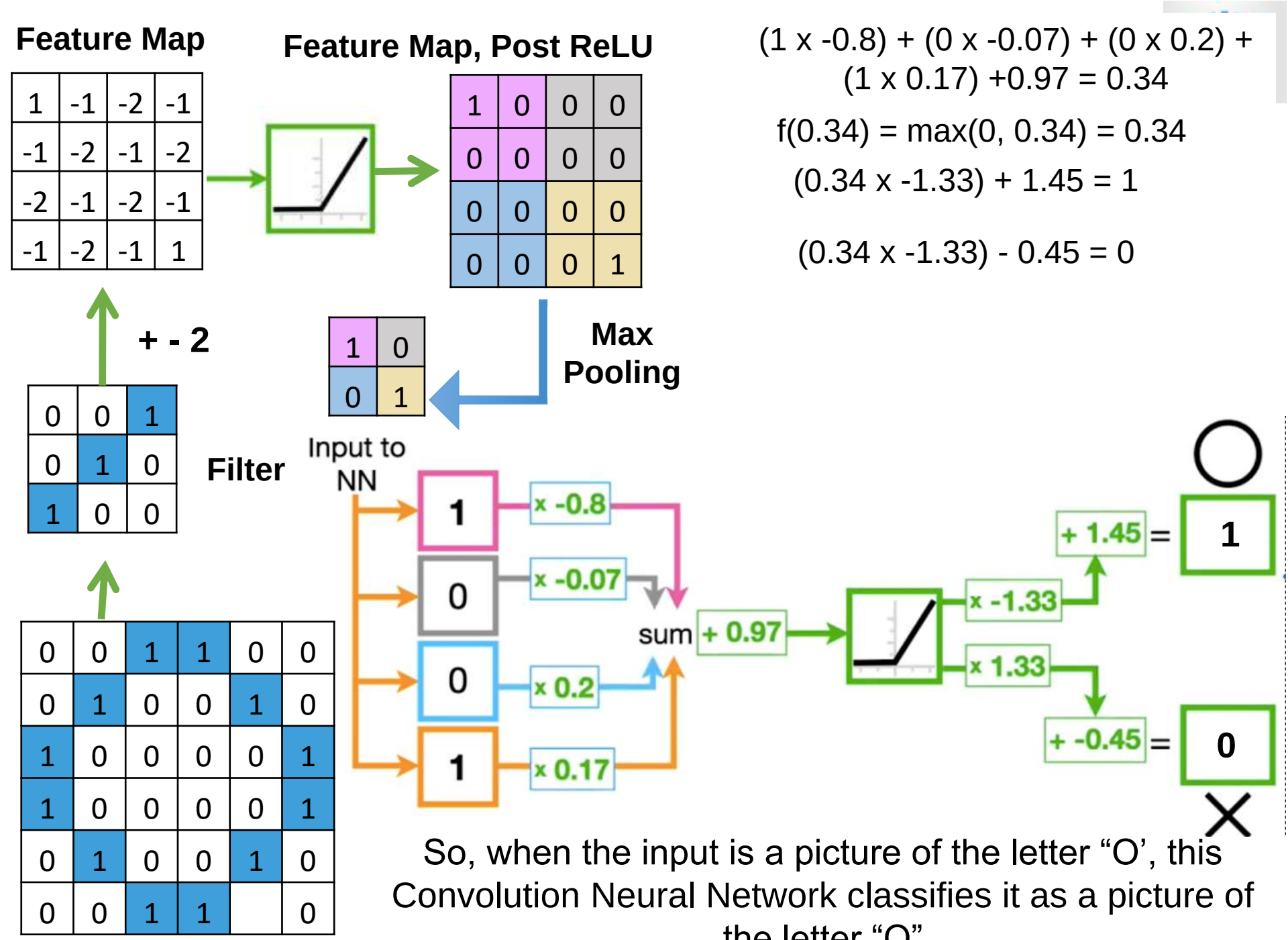
Feature Map

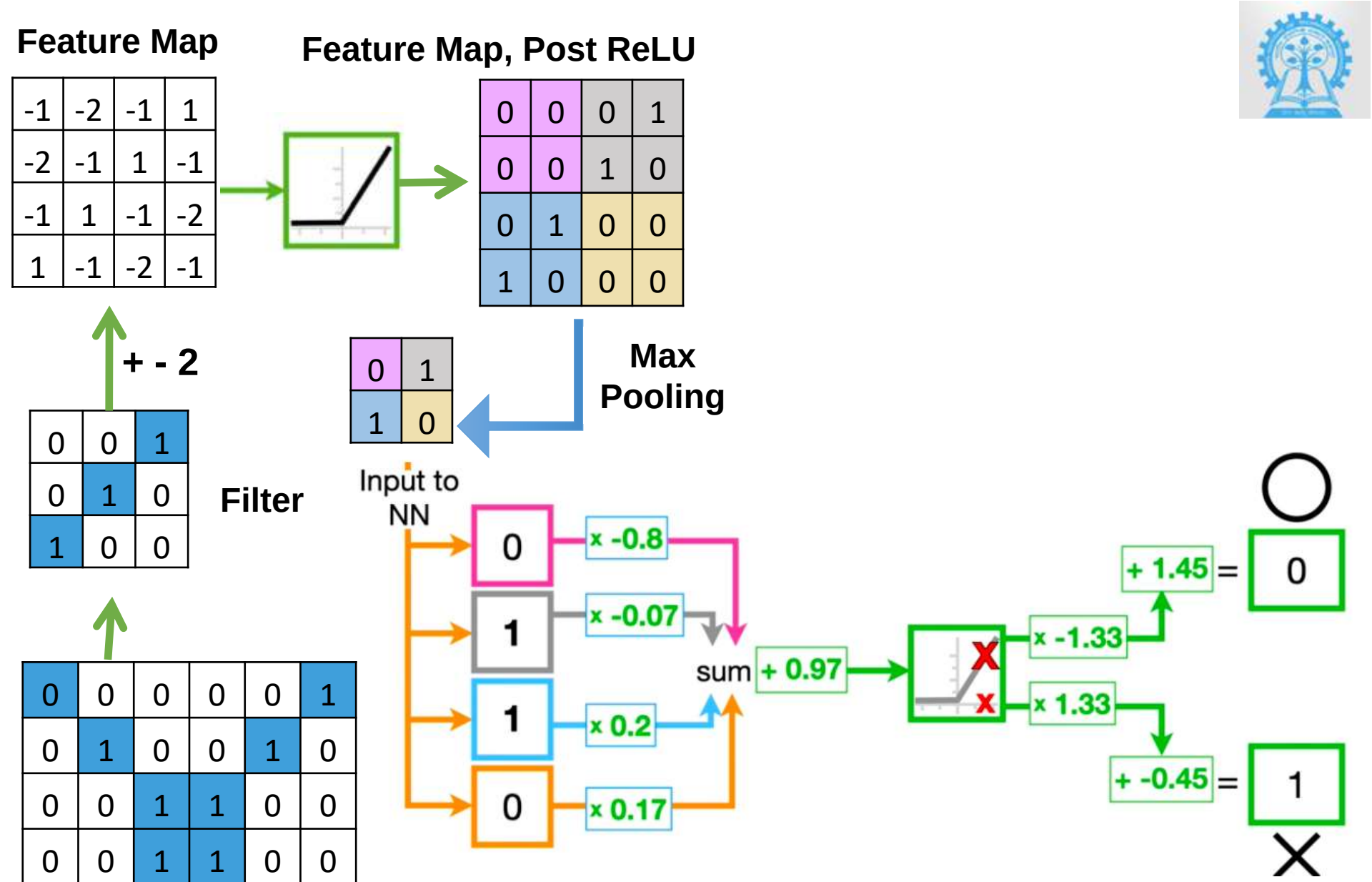


- To summarize we started with an input image of the letter “O” and then applied a Filter to it. We Convolved the Filter and with the input and added a Bias term to the values and that gave us a Feature Map. Because each cell in the Feature Map corresponds to a group of neighboring pixels
- Typically we run the Feature Map through a ReLU Activation Function and that means that all of the negative values are set to 0 and positive values are same as before.
- The Feature Map helps take advantage of any correlations there might be in the image
- Now we apply another filter to the new Feature Map. Unlike before, we simply select the maximum value

Convolutional Neural Networks (CNN)

- To summarize we started with an input image of the letter “O” and then applied a Filter to it. We Convolved the Filter and with the input and added a Bias term to the values and that gave us a Feature Map. Because each cell in the Feature Map corresponds to a group of neighboring pixels
- The Feature Map helps take advantage of any correlations there might be in the image
- Typically we run the Feature Map through a ReLU Activation Function and that means that all of the negative values are set to 0 and positive values are same as before.
- Now we apply another filter to the new Feature Map. Unlike before, we simply select the maximum value and this filter usually maoves in such a way that it does not overlap itself.
- When we select the maximum value in each region, we are applying Max Pooling
- Alternatively, we could calculate the average value for each region and that would be called Average or Mean Pooling


Feature Map
Feature Map, Post ReLU
(1 x -0.8) + (0 x -0.07) + (0 x 0.2) + (1 x 0.17) +0.97 = 0.34
f(0.34) = max(0, 0.34) = 0.34
(0.34 x -1.33) + 1.45 = 1
(0.34 x -1.33) - 0.45 = 0
+ - 2
Max Pooling
Filter
Input to NN
x -0.8
x -0.07
x 0.2
x 0.17
sum
+ 0.97
x -1.33
x 1.33
+ 1.45 =
+ -0.45 =
1
0
So, when the input is a picture of the letter “O’, this Convolution Neural Network classifies it as a picture of the letter “O”



So, when the input is a picture of the letter “X’, this Convolution Neural Network classifies it as a picture of the letter “X”

Numerical

- Given an input matrix (image) and a filter (kernel), perform convolution and max pooling operations (2*2).

1. Input Matrix:

$$\begin{bmatrix} 1 & 2 & 0 & 3 \\ 4 & 5 & 1 & 0 \\ 1 & 2 & 3 & 4 \\ 0 & 1 & 1 & 2 \end{bmatrix}$$

2. Filter (Kernel):

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

3. Padding: None

Summary

- Convolutional Neural Networks help reduce the number of input nodes into the Neural Network
- In this case, we started with a 6x6 image, or 36 potential inputs and compressed those down to just 4 inputs into the Neural Network
- Convolutional Neural Networks can tolerate small shifts in where the pixels are in the image
- Convolutional Neural Networks take correlations into account. This is accomplished by the Filter, which looks at a region of pixels, instead of just one pixel at a time