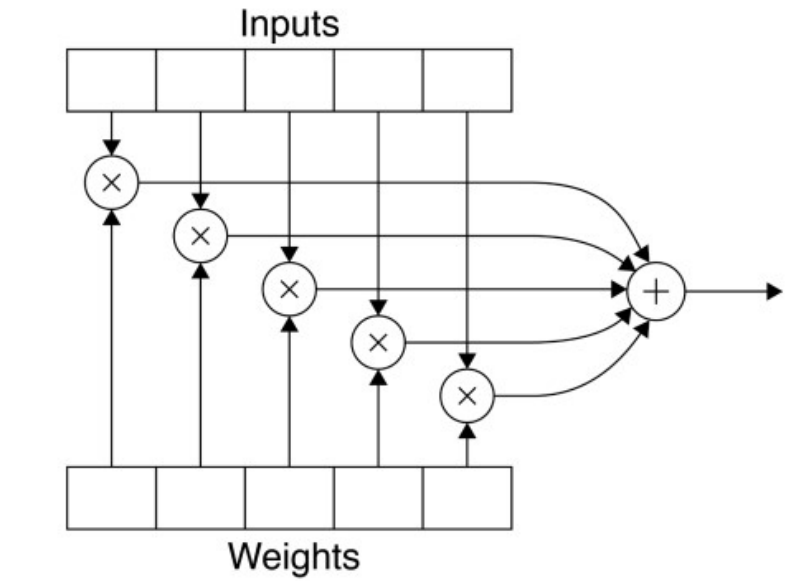
Components of CNN Architecture

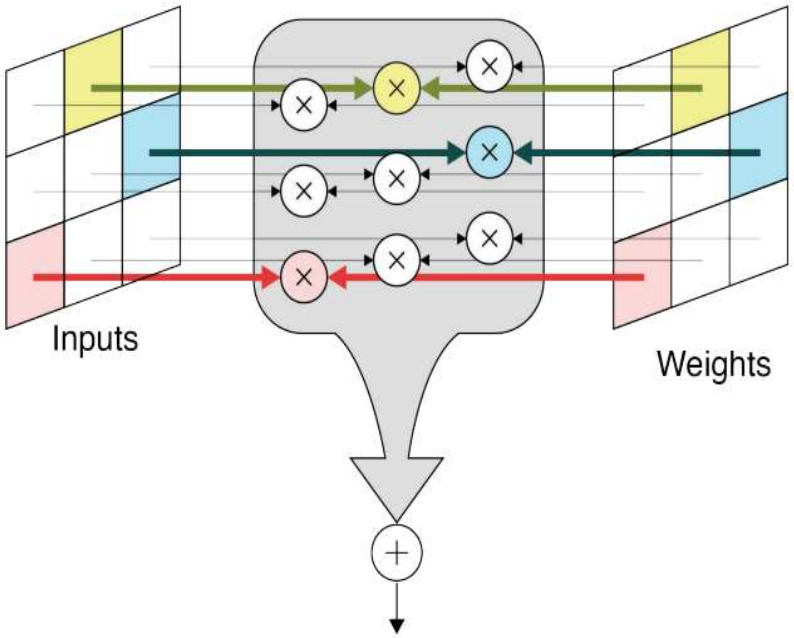
1. Convolution
2. Padding
3. Stride
4. Pooling
5. Fully Connected Layer

1. Convolution
Convolution in 1-D



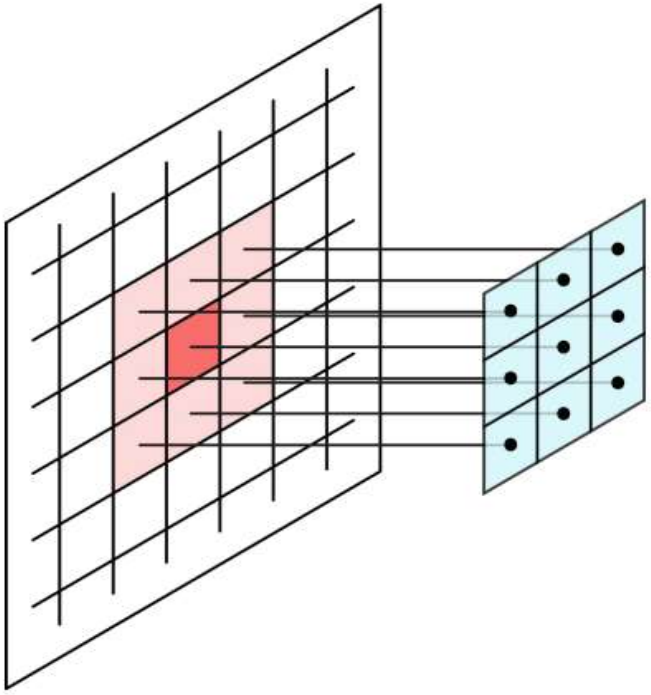
A step of convolution involves two lists, which we can call “inputs” and “weights,” though both lists are treated the same way.
Corresponding values are multiplied together pairwise, and then
the results are added together.
This is what an artificial neuron does, ignoring the activation function.

Convolution in 2-D



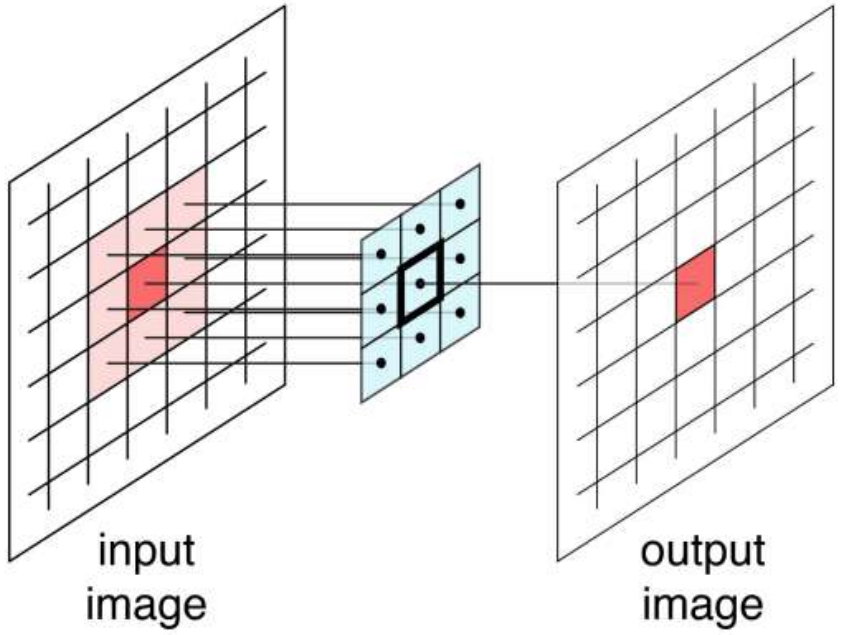
Convolution in 2D works just like in 1D, though the picture gets more cluttered. Each pair of corresponding values gets multiplied together,
and
then all of those products are added together

Convolution in 2-D



The neuron gets inputs in this case from a 3 by 3 square around the pixel. The location of the pixel in the image is highlighted in bright red. The set of all 9 pixels is called the local receptive field for the neuron. Our input image here has one channel, so the neuron receives 9 inputs. Each is multiplied by a corresponding weight, shown in blue.

Convolution in 2-D

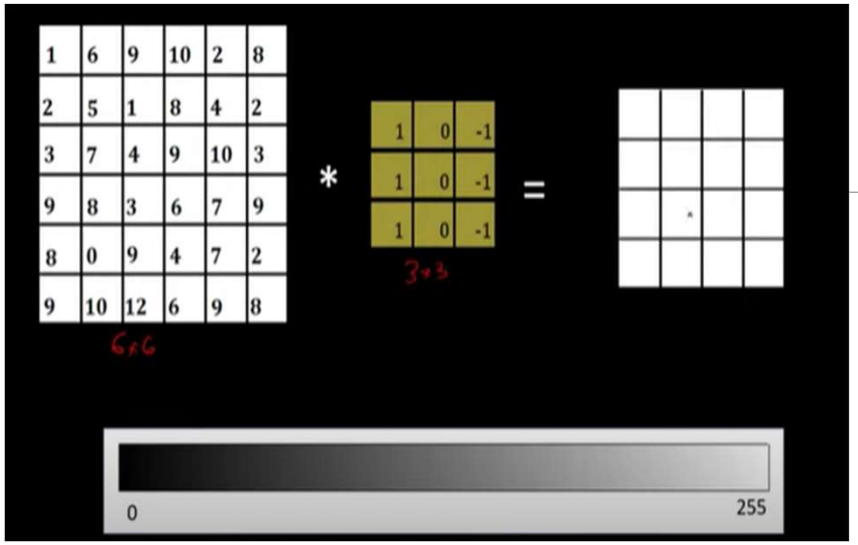


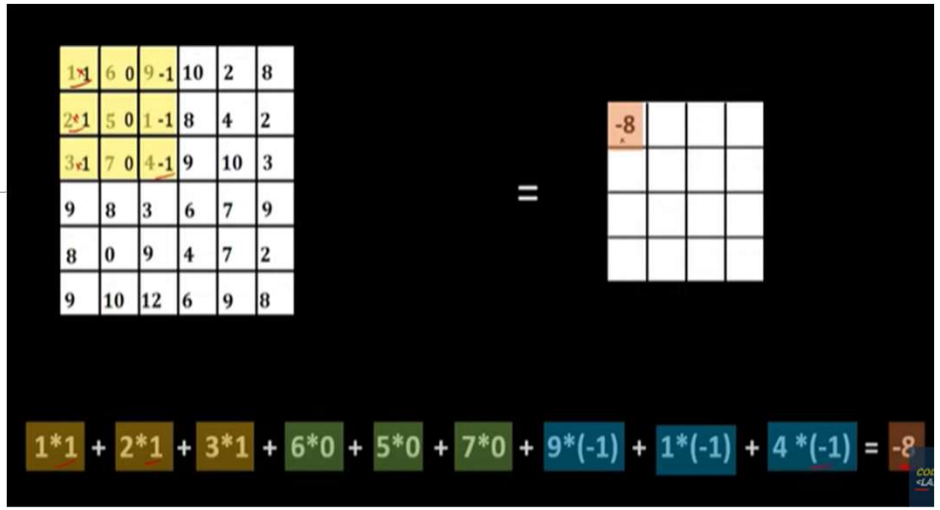
The neuron has a 3 by 3 receptive field, with the highlighted anchor at the center.

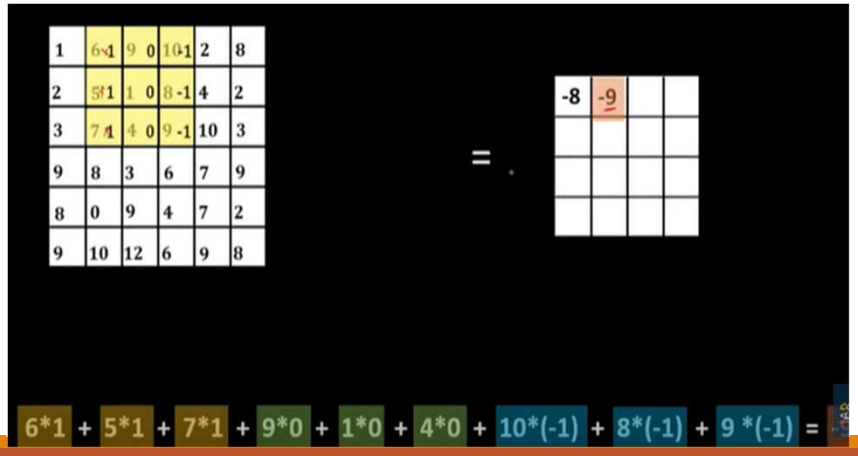
The bright red pixel in the input image, called the focus pixel, shows where the anchor is currently located.

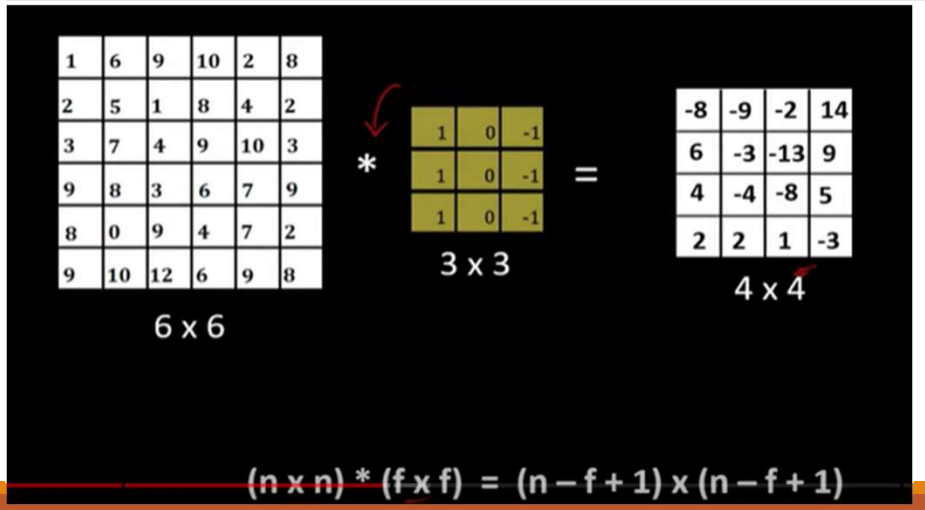
The output of the neuron goes into the output image at the same location as the focus pixel.

Convolution

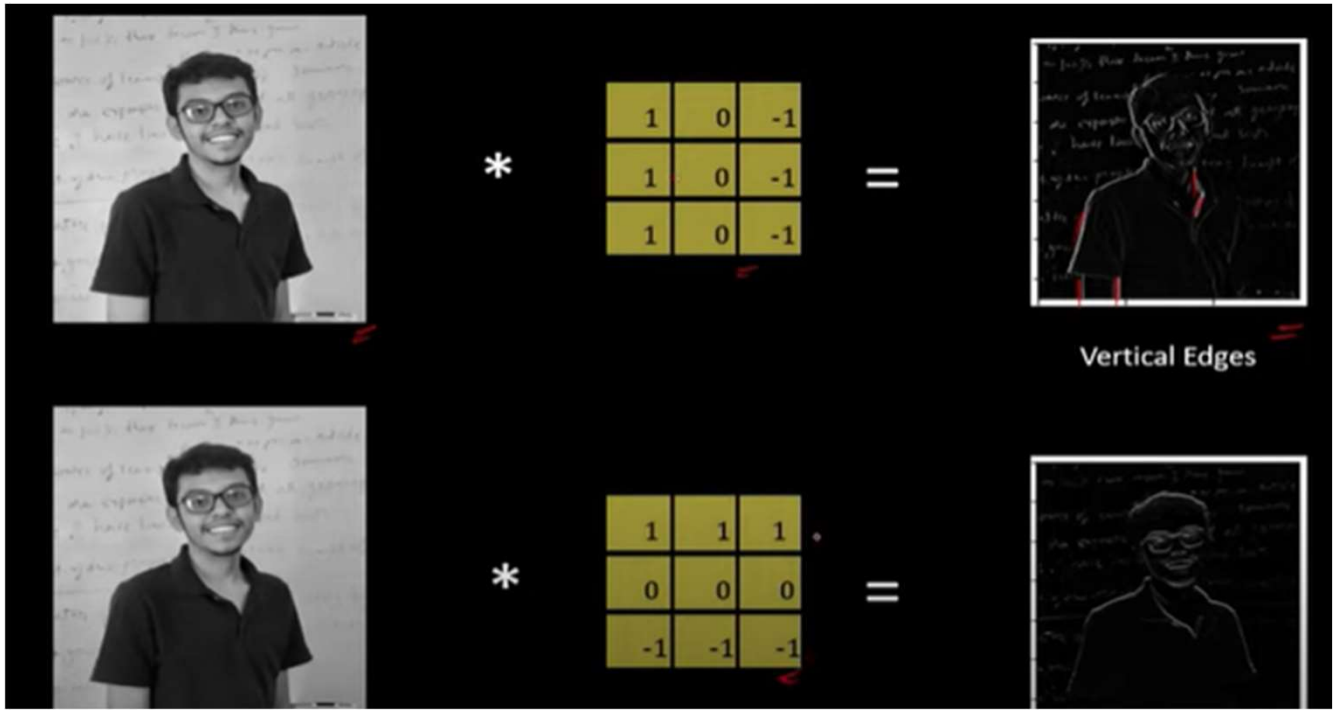




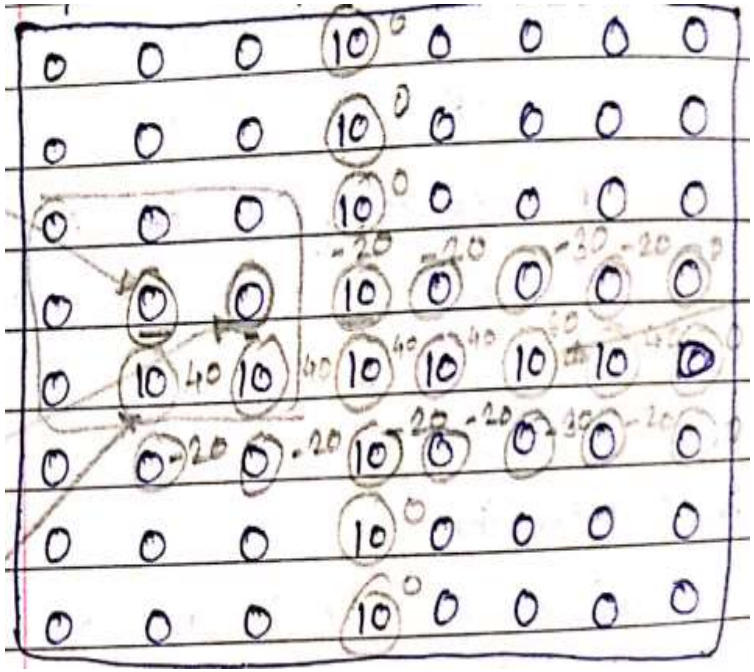




Convolution- Detects the edges and features

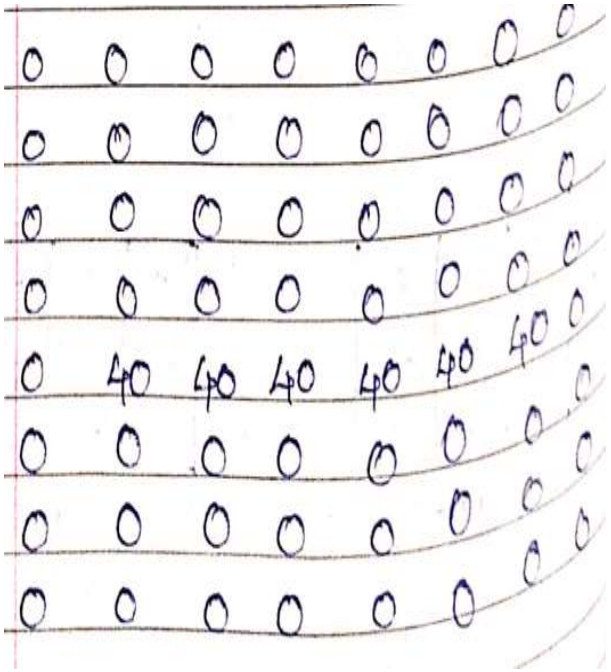


Convolution- Detects the edges and features

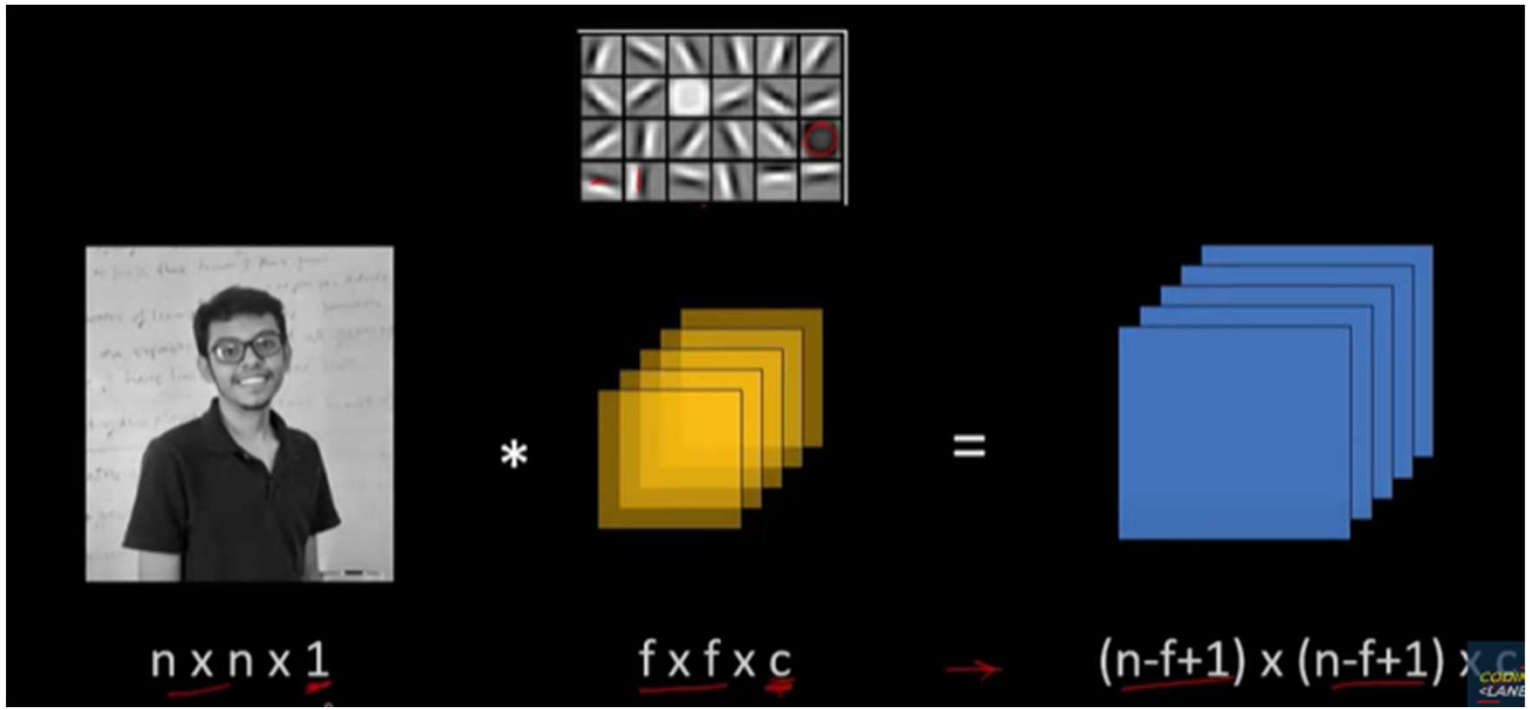


-1	-1	-1
2	2	2
-1	-1	-1

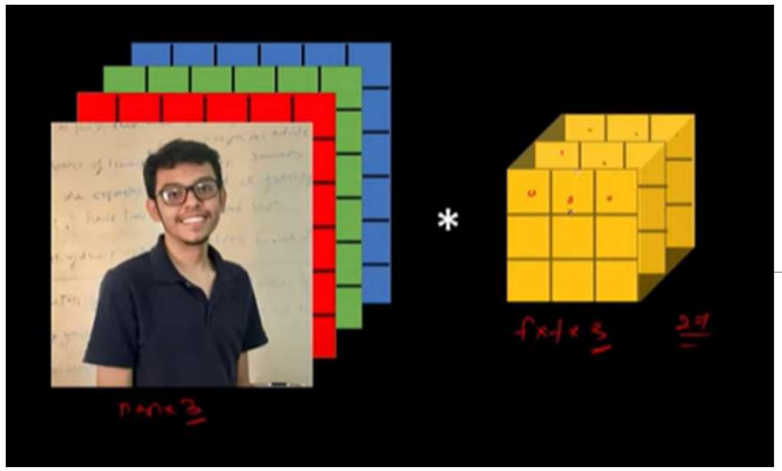
Horizontal

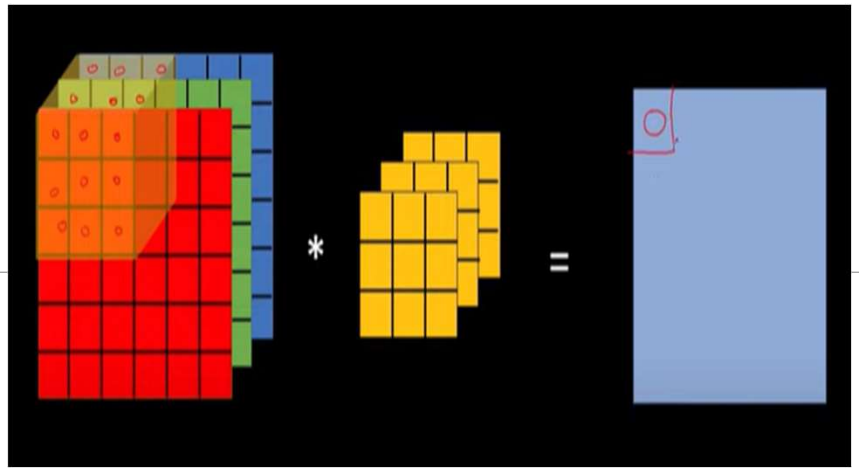


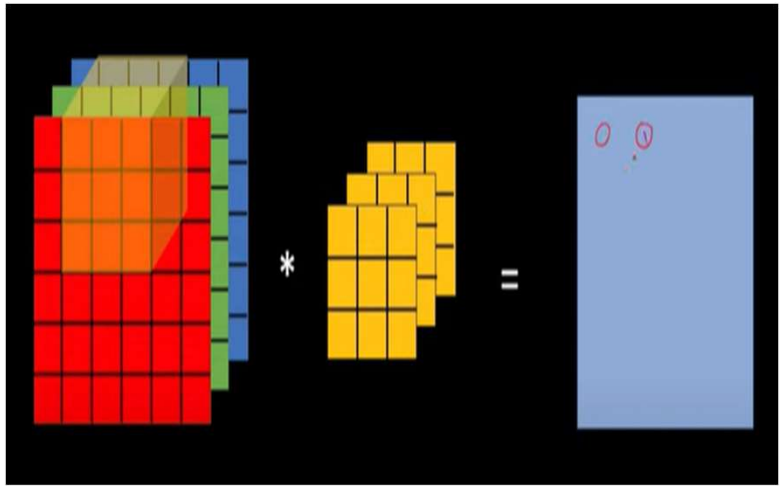
Single layer of CNN-multiple filters.. horizontal…. vertical edges..circles..etc

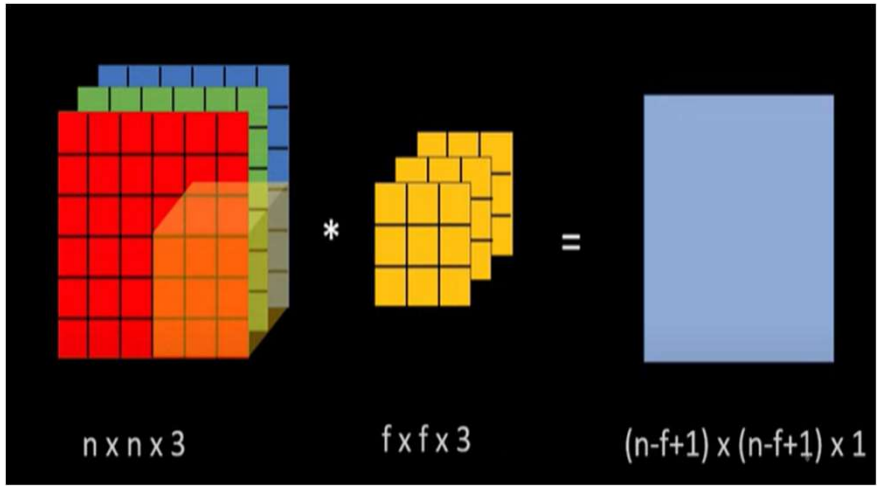


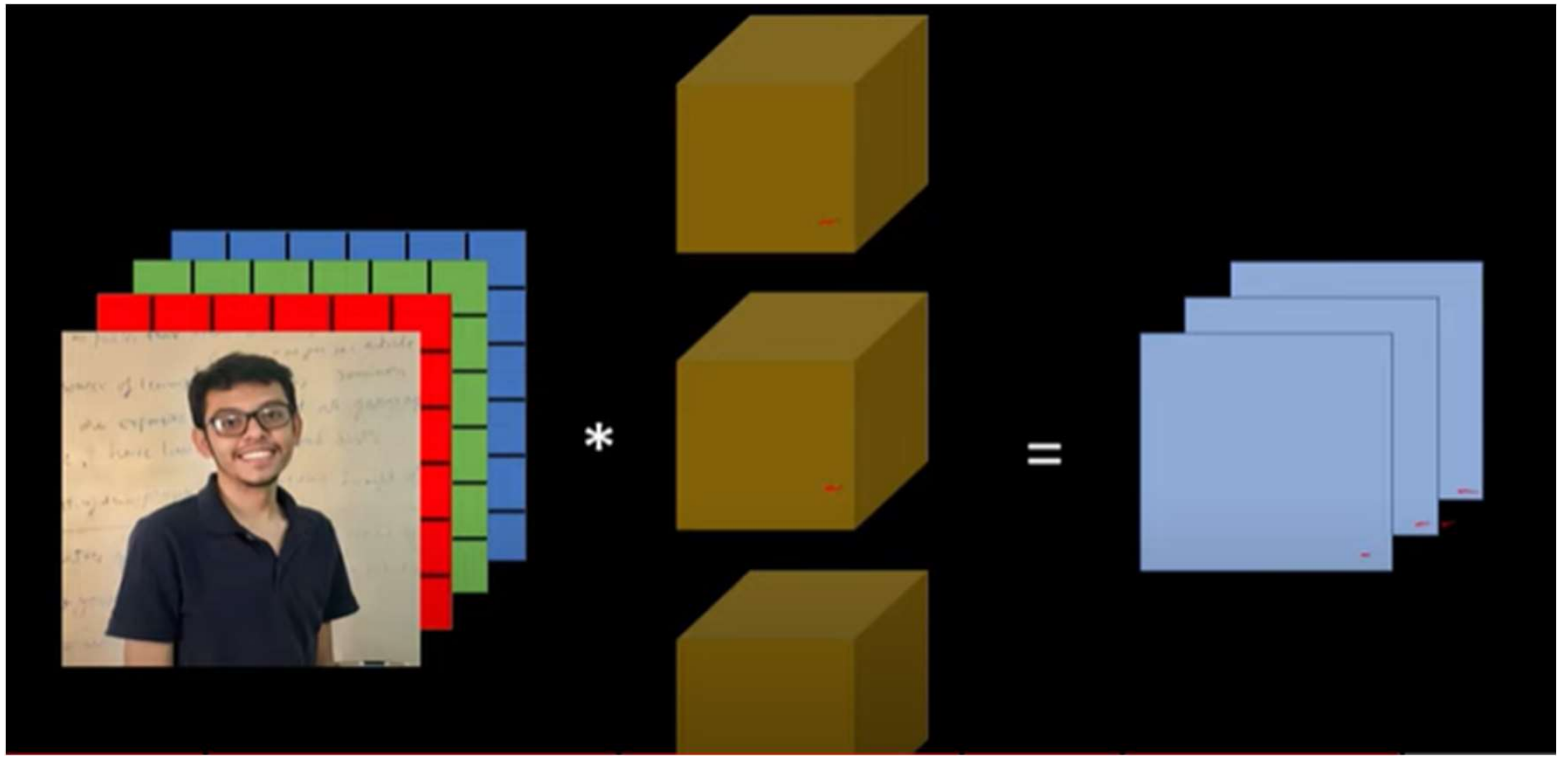
Black and white Image



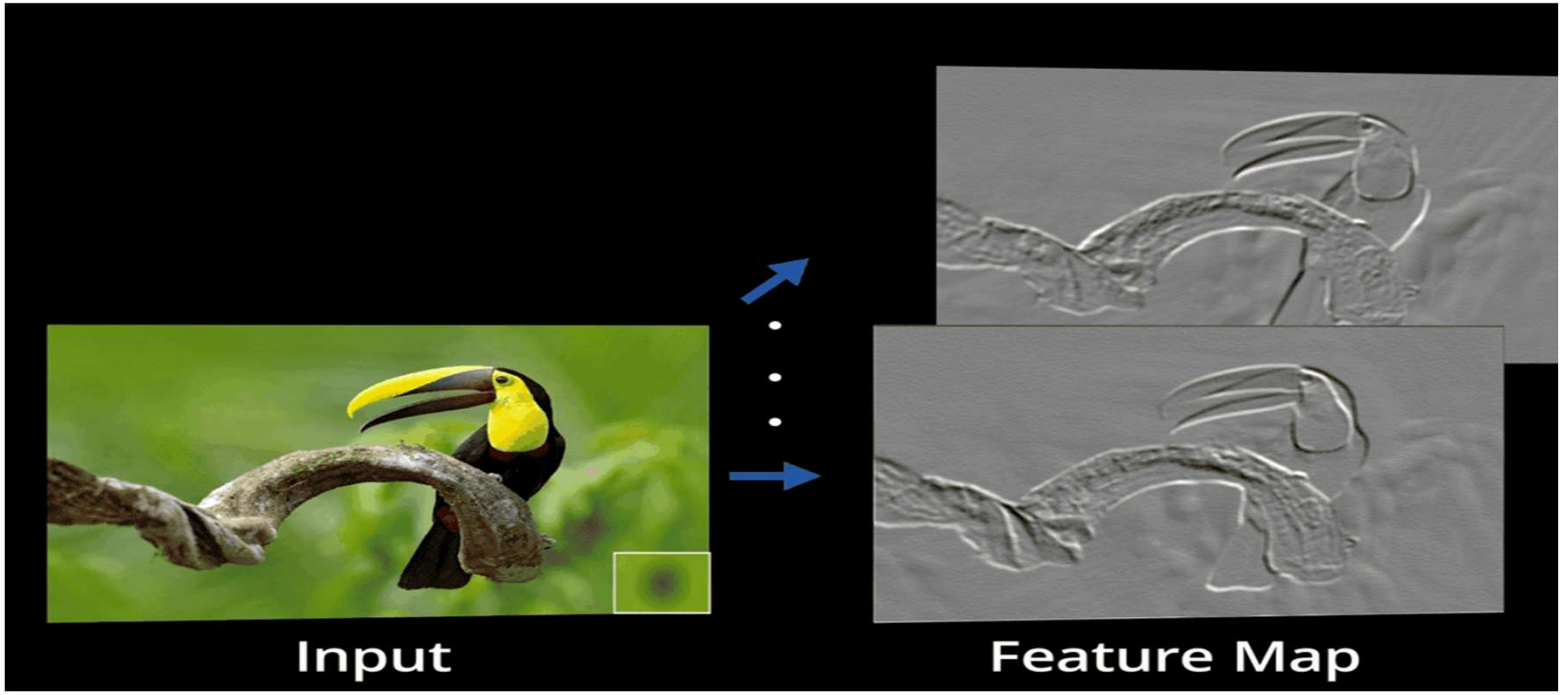





n x n x 3
f x f x 3
(n-f+1) x (n-f+1) x 1



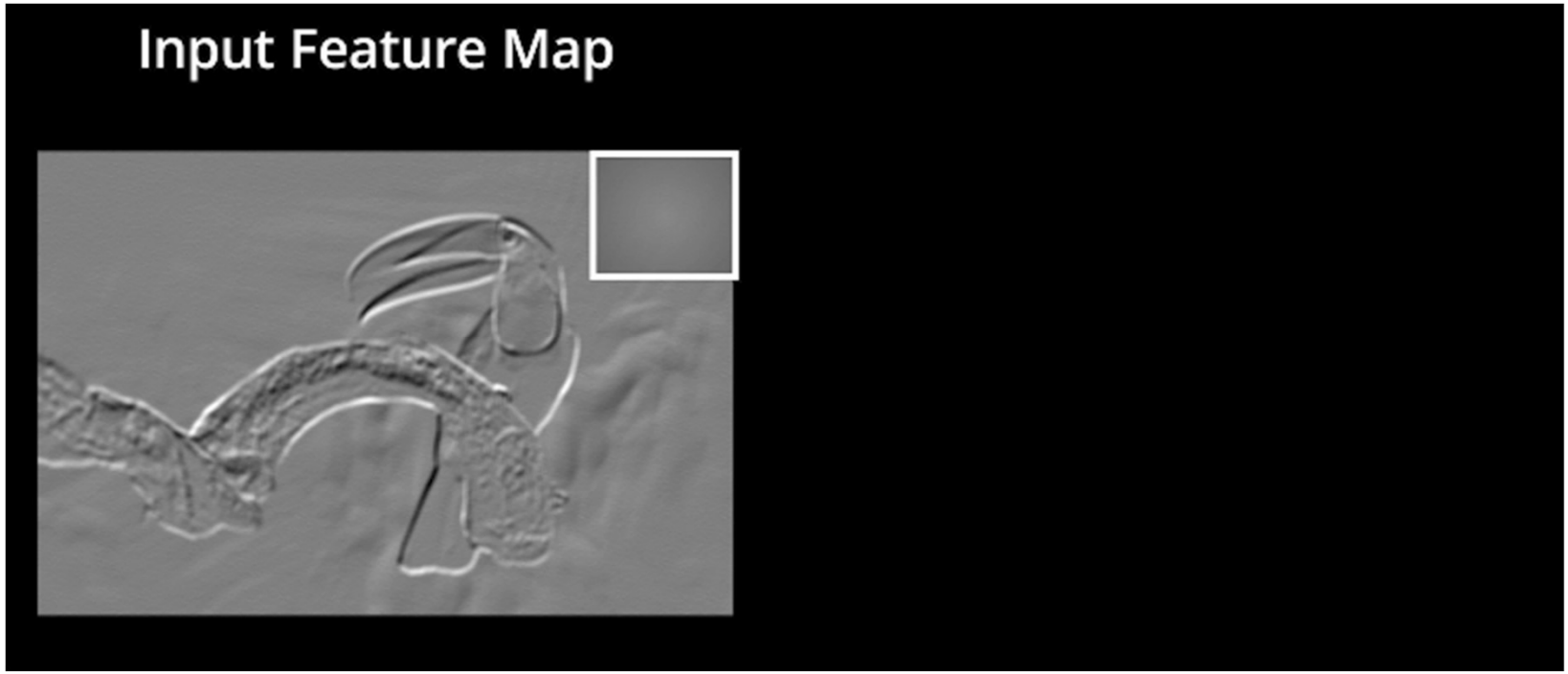
Convolution Layer – output



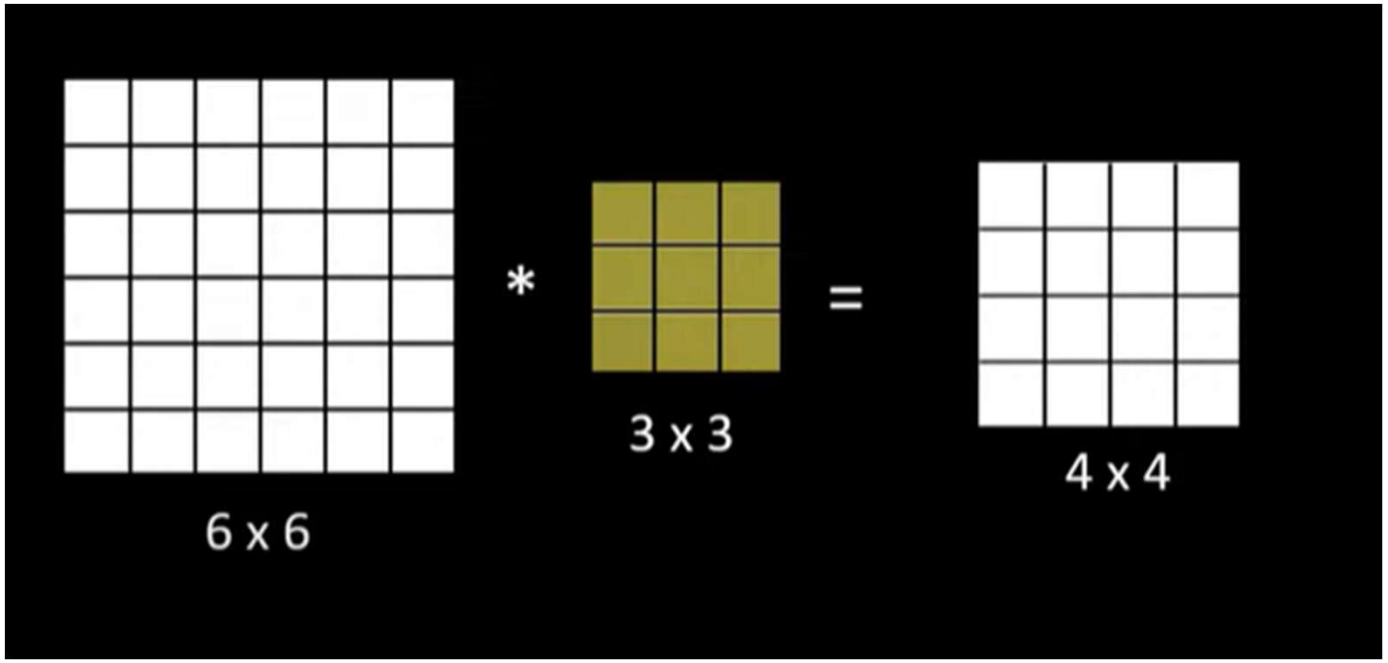
2. Sharpen / Rectify feature map

- Once the feature maps are extracted, the next step is to move them to a ReLU layer.
- ReLU performs an element-wise operation and sets all the negative pixels to 0.
- It introduces non-linearity to the network, and the generated output is a rectified feature map.

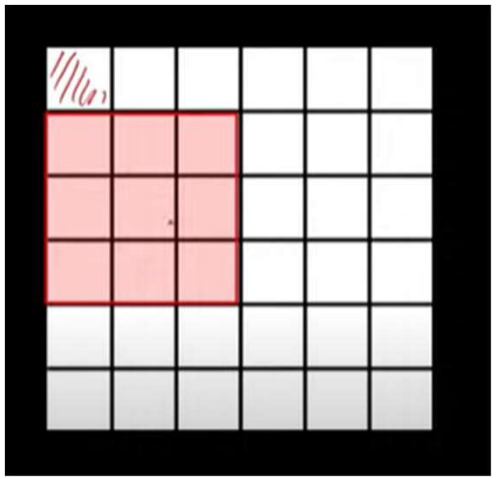
ReLU - Sharpen / Rectify feature map



Limitations of Convolution- Reduction in image size-might loose valuable information

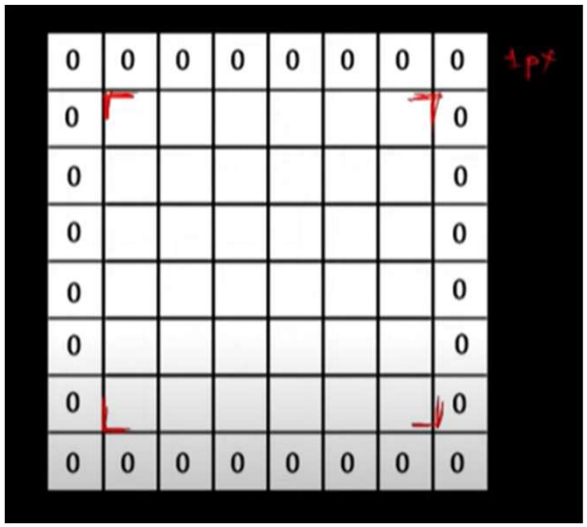


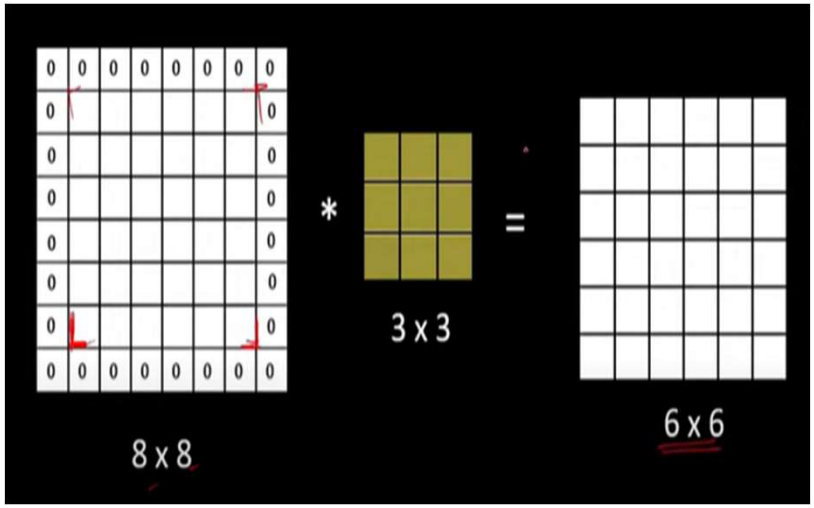
Limitations of Convolution-Corner pixels gets less attention as compared to others

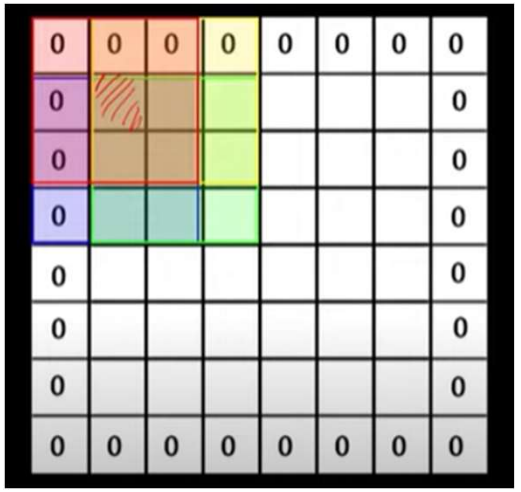


Solution-Padding

3. Padding





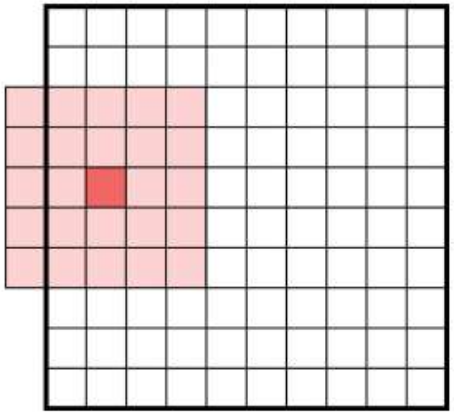


Padding

The idea is that we add a border of “extra” pixels around the outside of the image.

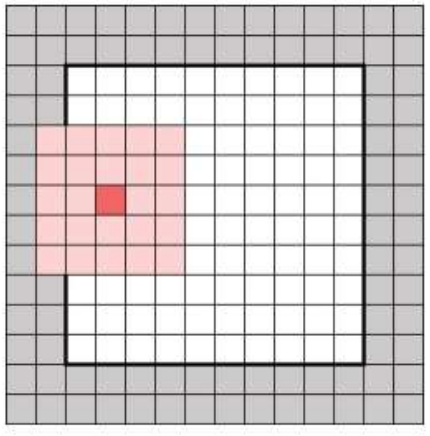
By far the most common choices is simply zero.

This choice is called **zero-padding.**



Near the edge, the filter’s receptive field can fall off the side of the image.

What values do we use for these missing pixels?



A better way to solve the “falling off the edge” problem is to add padding, or extra pixels, around the border of the image.

Here we’ve added a 2-pixel border, so every pixel in the original image (shown in white) can be used as the center of the filter. Usually, the padded pixels are given the value 0.

"VALID" and "SAME" convolution

1. Valid Convolution

No padding

2. Same Convolution

$n' = n + 2p$

$(n' - f + 1) = n$

$(n + 2p - f + 1) = n$

$p = \left(\frac{f-1}{2}\right)$

$f \Rightarrow$ usually odd

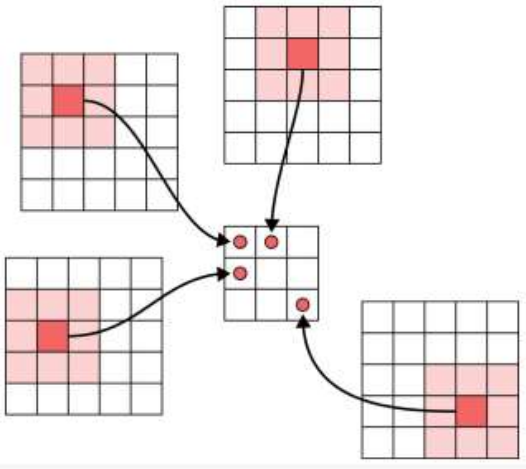
0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0

Ipx $n=6$, $p=1$, 8 2px

```
tf.nn.conv2d(X, W1, strides = [1,1,1,1], padding = 'VALID')
```

Strides

4.Stride



Sweeping a 3 by 3 filter over a 5 by 5 image, without padding. Each step of the filter moves it to the right by one pixel in the input, and then it moves down by one pixel.
The diagram shows 4 of the 9 filter positions that would make up this output.

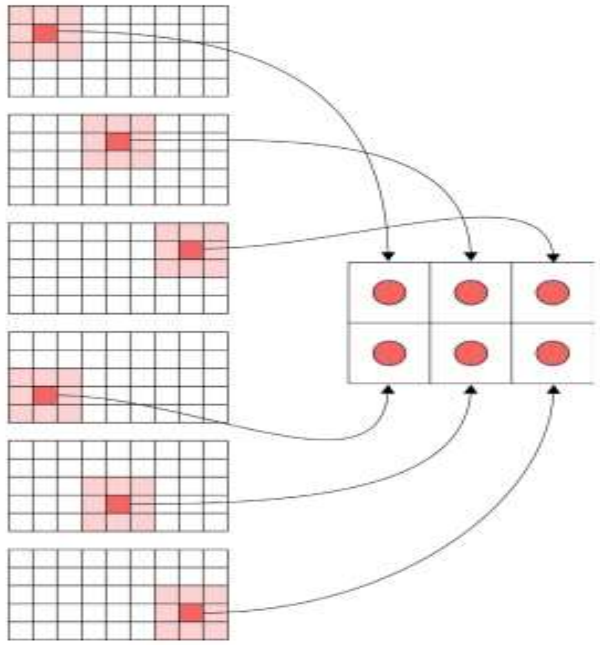
Stride

Our input scanning can skip over pixels.

In this example, we look only at every third pixel horizontally, and every other pixel vertically.

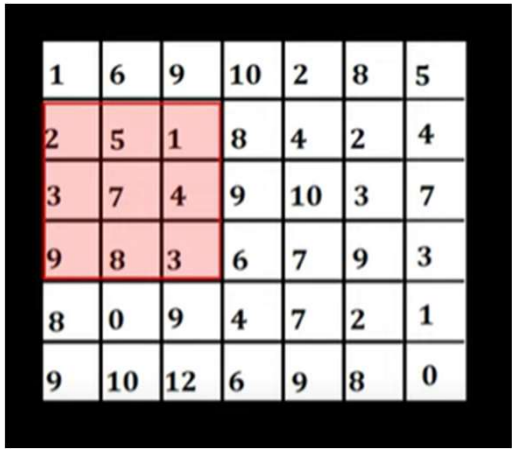
In other words, we use a stride of 3 horizontally and 2 vertically. For clarity, we're not using padding in this figure.

The 5 by 9 input image turns into a 2 by 3 output.

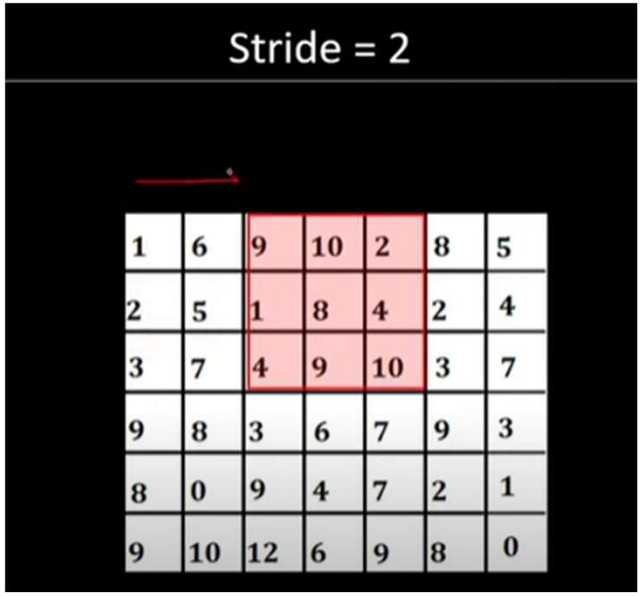


The number of rows and columns traversed per slide are referred as - Stride

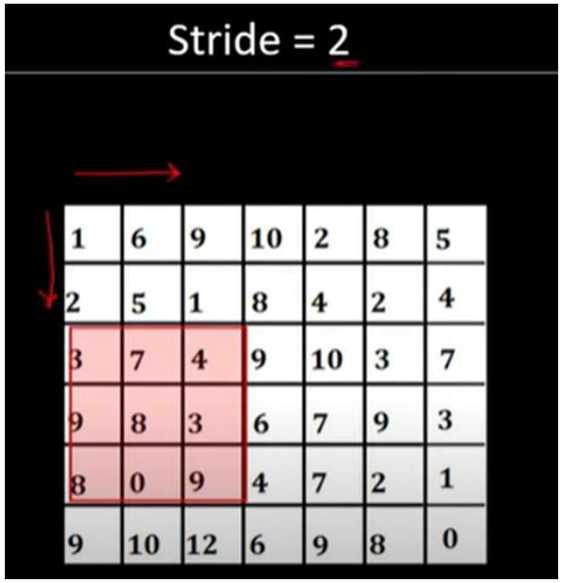
Stride



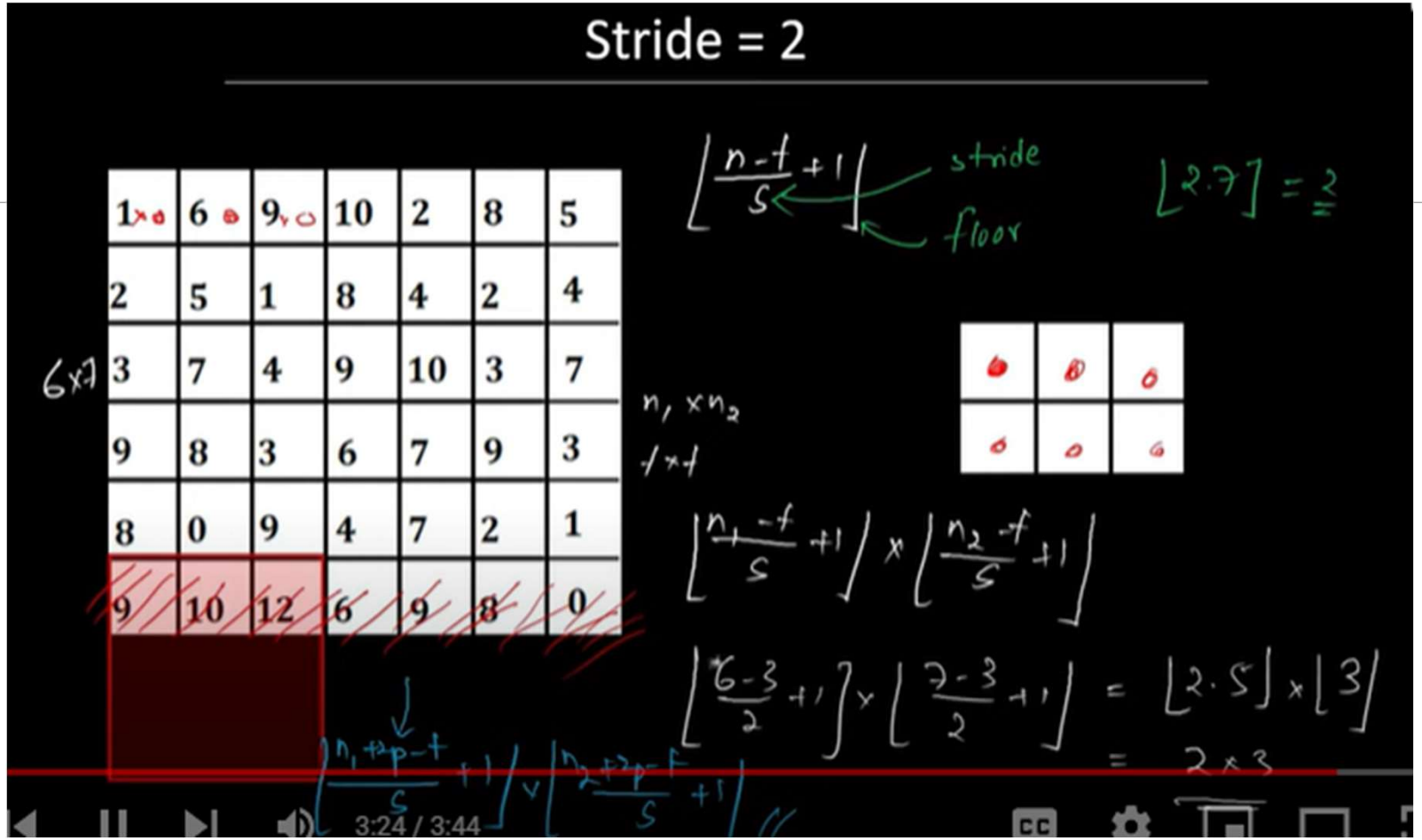
1	6	9	10	2	8	5
2	5	1	8	4	2	4
3	7	4	9	10	3	7
9	8	3	6	7	9	3
8	0	9	4	7	2	1
9	10	12	6	9	8	0



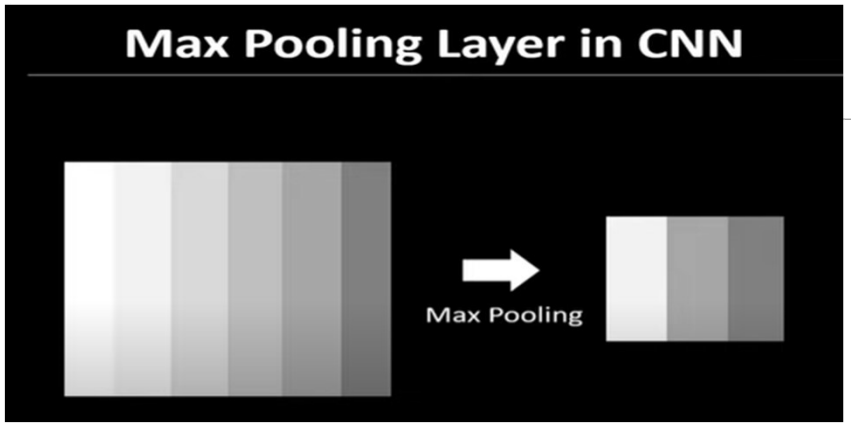
1	6	9	10	2	8	5
2	5	1	8	4	2	4
3	7	4	9	10	3	7
9	8	3	6	7	9	3
8	0	9	4	7	2	1
9	10	12	6	9	8	0

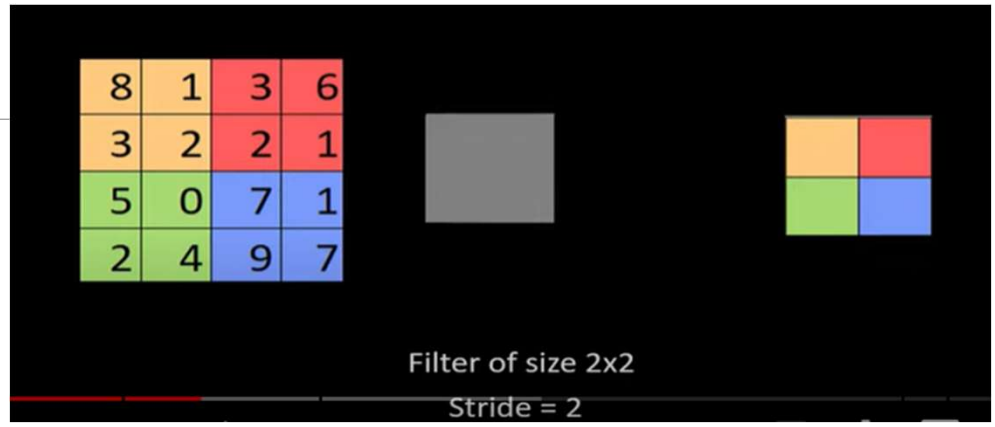


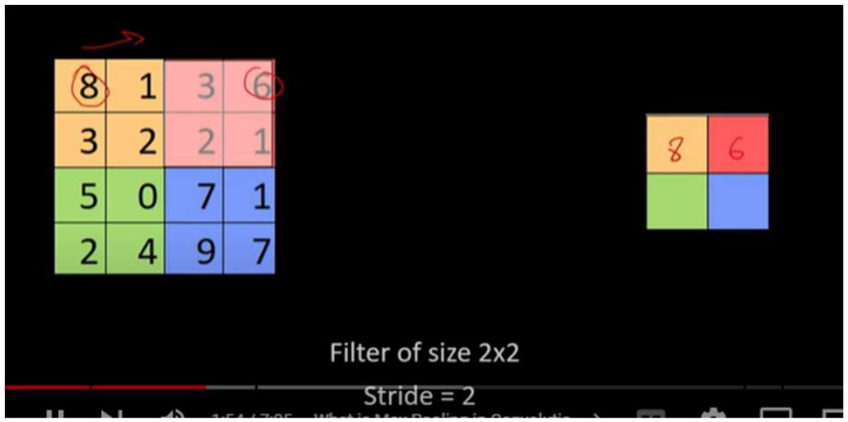
1	6	9	10	2	8	5
2	5	1	8	4	2	4
3	7	4	9	10	3	7
9	8	3	6	7	9	3
8	0	9	4	7	2	1
9	10	12	6	9	8	0

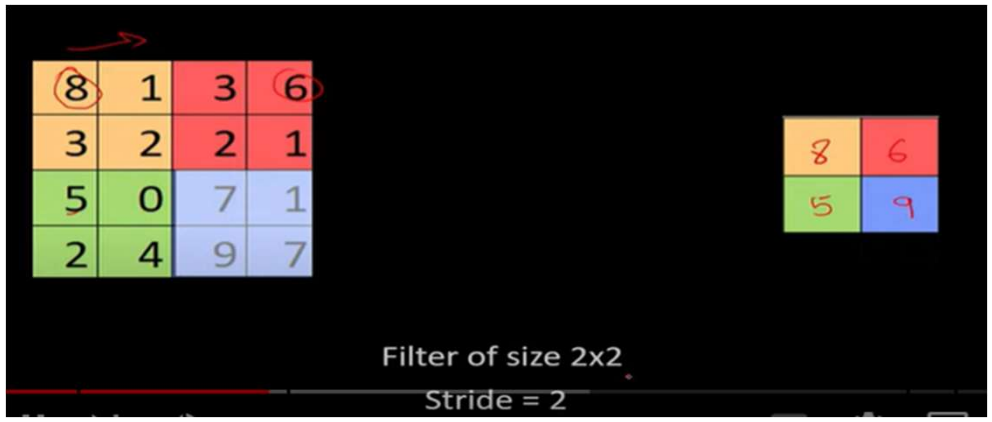

Stride = 2
6×7
1 6 9 10 2 8 5
2 5 1 8 4 2 4
3 7 4 9 10 3 7
9 8 3 6 7 9 3
8 0 9 4 7 2 1
9 10 12 6 9 8 0
stride
floor
[2.7] = 2
n1 × n2
1 × 1
3:24 / 3:44

5. Pooling


Max Pooling Layer in CNN
Max Pooling


8 1 3 6
3 2 2 1
5 0 7 1
2 4 9 7
Filter of size 2x2
Stride = 2


8 1 3 6
3 2 2 1
5 0 7 1
2 4 9 7
8 6
Filter of size 2x2
Stride = 2


8 1 3 6
3 2 2 1
5 0 7 1
2 4 9 7
8 6
5 9
Filter of size 2x2
Stride = 2

Pooling layer plays an important role in pre-processing of an image.

Pooling layer reduces the number of parameters when the images are too large.

Pooling is "**downscaling**" of the image obtained from the previous layers.

It can be compared to shrinking an image to reduce its pixel density.

Spatial pooling is also called downsampling or subsampling, which reduces the dimensionality of each map but retains the important information.

There are the following types of spatial pooling: Max Pooling and Average pooling

Why do we need Max Pooling?

1. Reduce image size, thus reduce computational cost

2. Enhances Features

0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5
0	0.1	0.2	0.3	0.4	0.5

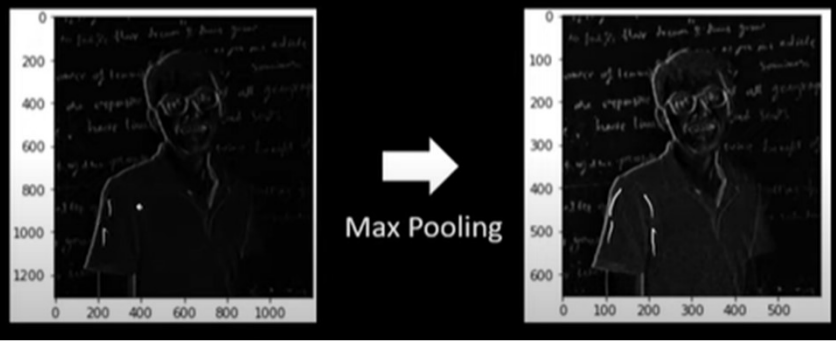
0.1	0.3	0.5
0.1	0.3	0.5
0.1	0.3	0.5

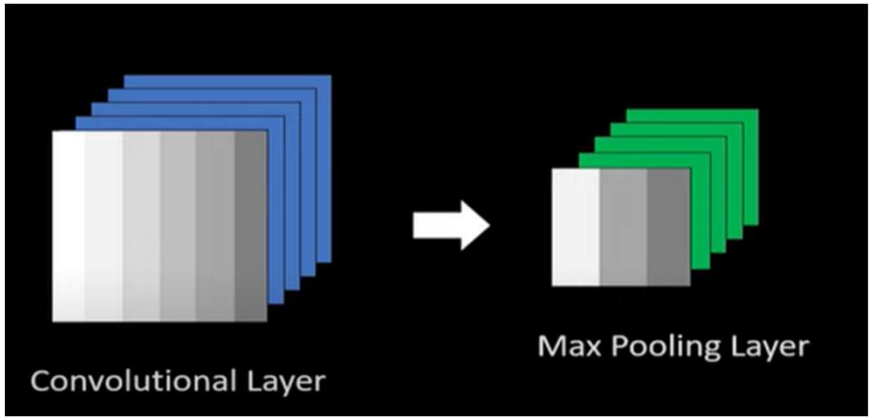
Output

Why do we need Max Pooling?

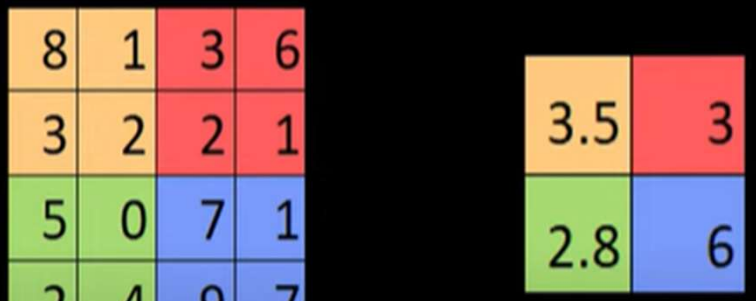
1. Reduce image size, thus reduce computational cost

2. Enhances Features





Average Pooling



Summary

Why do we need Max Pooling?

1. Reduce image size, thus reduce computational cost
2. Enhances the Features of the image

Where?

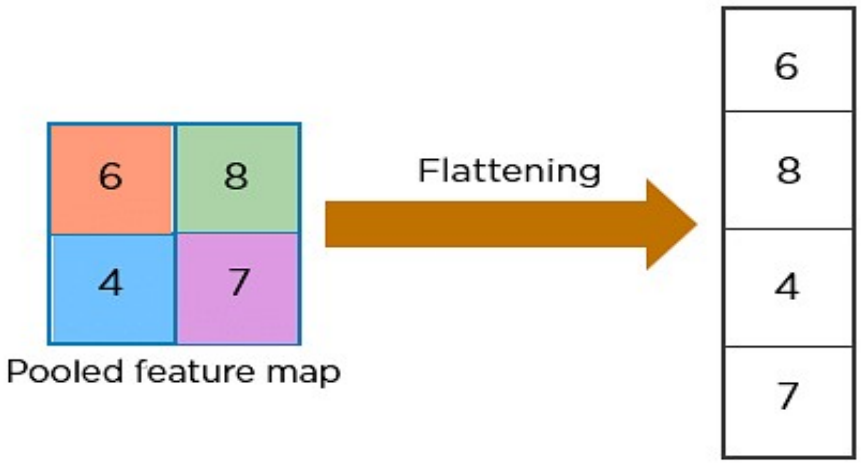
After Convolutional layer

Other points

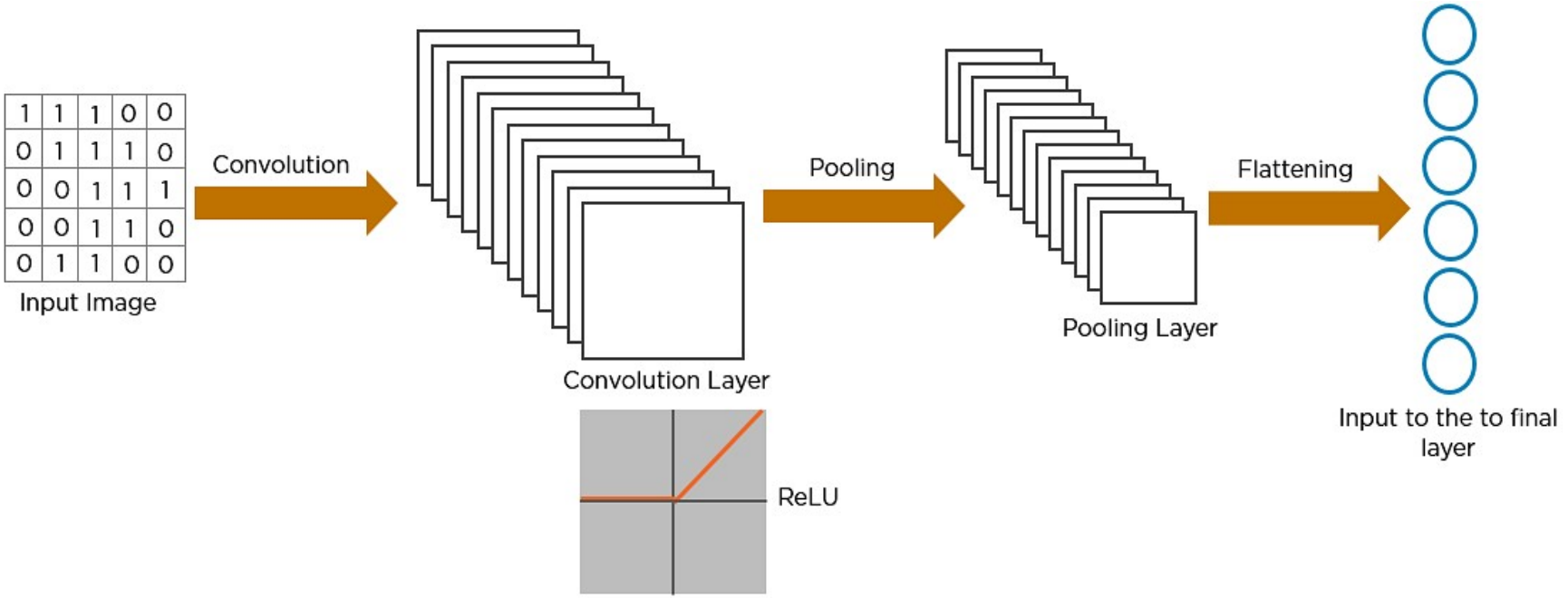
1. No parameters involved, thus no training
2. Same number of channels in output as input

6. Flattening operation

- The next step in the process is called flattening.
- Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector.



Flattening operation



7.Fully Connected Layer

- The flattened matrix is fed as input to the fully connected layer to classify the image.
- Fully connected layer looks like a regular neural network connecting all neurons and forms the last few layers in the network.
- The output from flatten layer is fed to this fully-connected layer.
- The feature vector from fully connected layer is further used to classify images between different categories after training.
- All the inputs from this layer are connected to every activation unit of the next layer.

Fully Connected Layer

- The flattened matrix is fed as input to the fully connected layer to classify the image.