# Convolutional neural networks (CNN)

- Cnn are one of the most popular models used today. This neural network computational model uses a variation of multilayer perceptrons and contains one or more convolutional layers that can be either entirely connected or pooled.
- These convolutional layers create feature maps that record a region of image which is ultimately broken into rectangles and sent out for nonlinear processing.
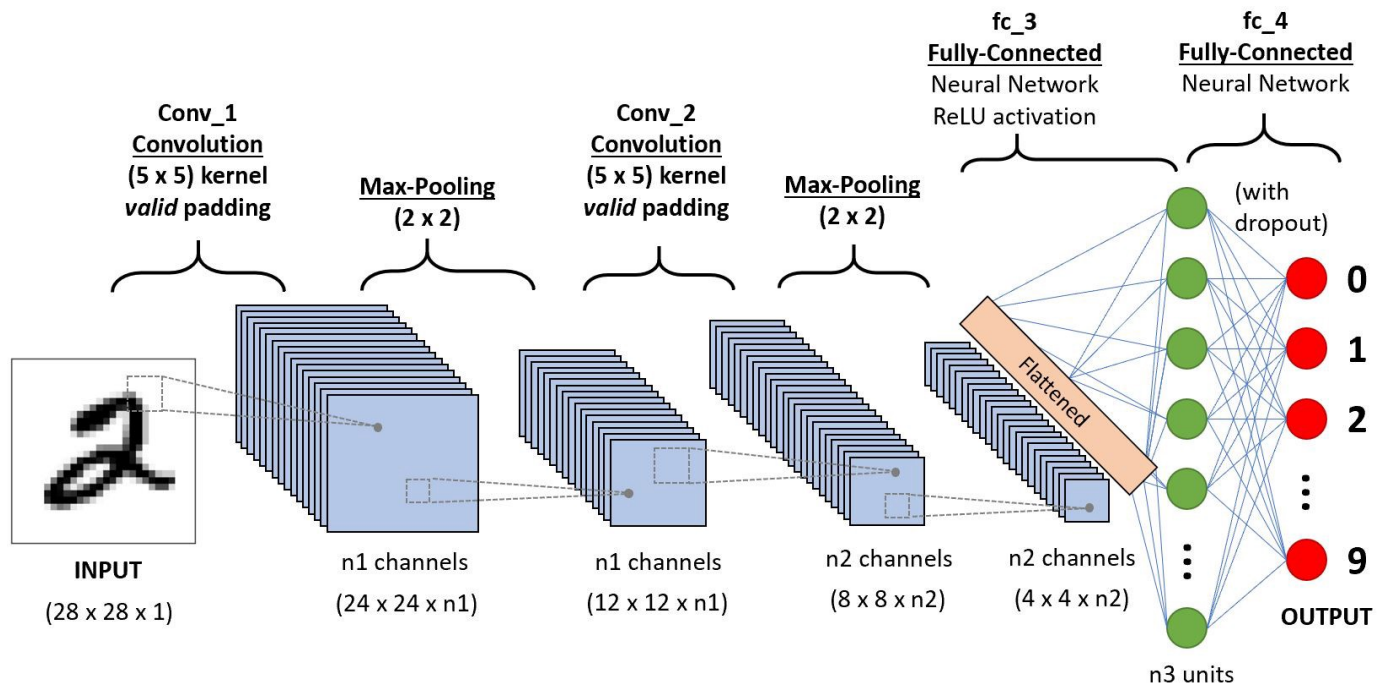
- Let us suppose this in the input matrix of 5×5 and a filter of matrix 3X3, for those who don't know what a filter is a set of weights in a matrix applied on an image or a matrix to obtain the required features, please search on convolution if this is your first time!

Note: We always take the sum or average of all the values while doing a convolution.

Steps Involve in CNN

STEP 1: Convolution

⬇

STEP 2: Max Pooling
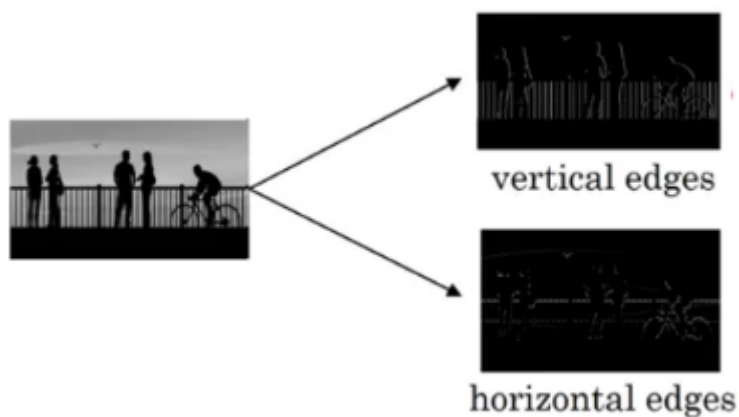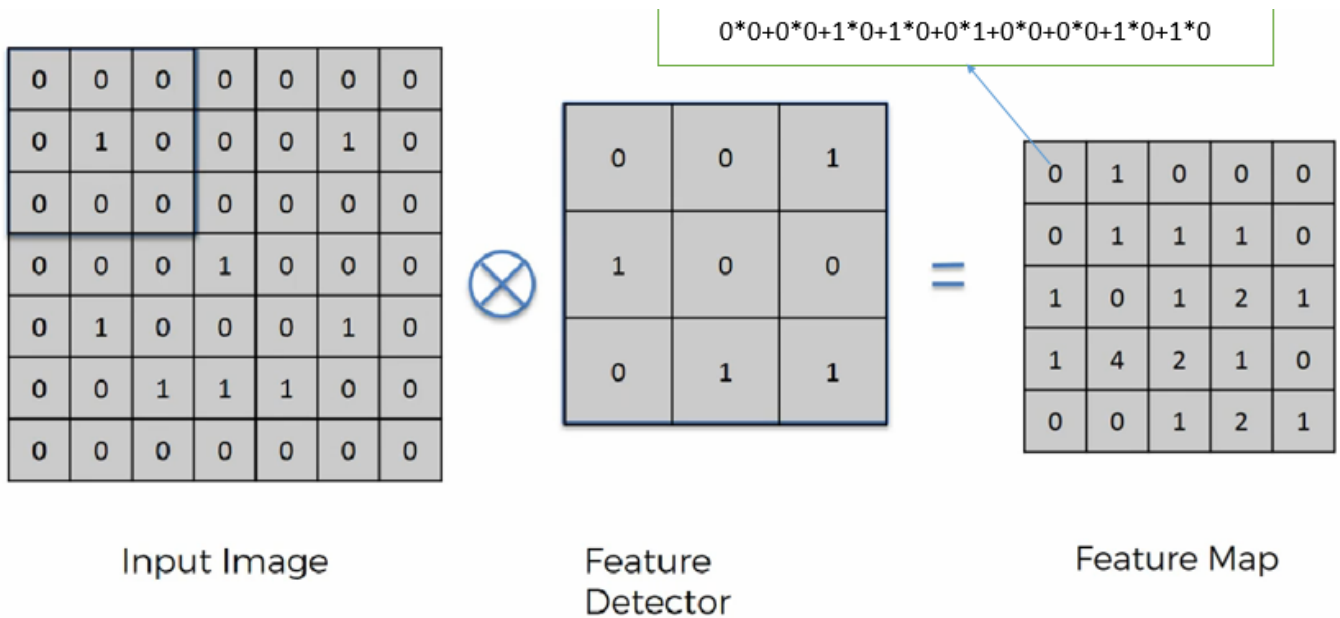
⬇

STEP 3: Flattening

⬇

STEP 4: Full Connection

1. Edge Detection (Convolution)

- In the previous article, we saw that the early layers of a neural network detect edges from an image. Deeper layers might be able to detect the cause of the objects and even more deeper layers might detect the cause of complete objects (like a person's face).

In this section, we will focus on how the edges can be detected from an image. Suppose we are given the below image: As you can see, there are many vertical and horizontal edges in the image. The first thing to do is to detect these edges:

vertical edges

horizontal edges

$$0*0+0*0+1*0+1*0+0*1+0*0+0*0+1*0+1*0$$

Input Image

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

⊗

Feature Detector

0	0	1
1	0	0
0	1	1

=

Feature Map

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

- So, we take the first 3 X 3 matrix from the 6 X 6 image and multiply it with the filter. Now, the first element of the 4 X 4 output will be the sum of the element-wise product of these values, i.e. _00+00+10+10+01+00+00+10+1*0_ =0. To calculate the second element of the 4 X 4 output, we will shift our filter one step towards the right and again get the sum of the element-wise product:
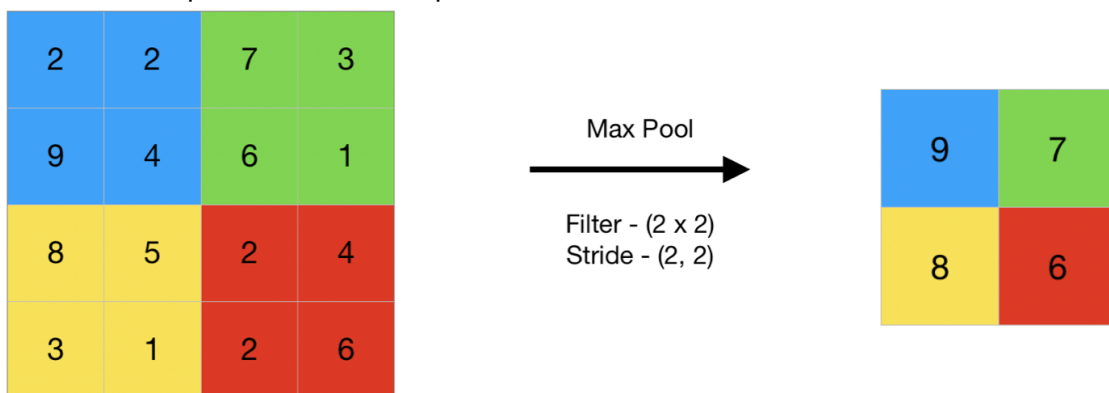
2. Pooling

- A pooling layer is another building block of a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently. The most common approach used in pooling is max pooling.
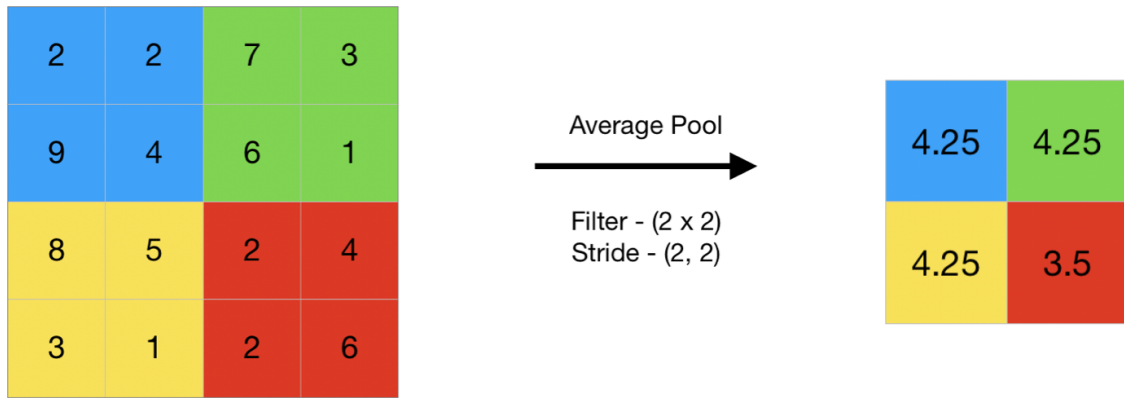
Types of Pooling Layers :-

1. Max Pooling

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

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

Max Pool →

Filter - (2 x 2)
Stride - (2, 2)

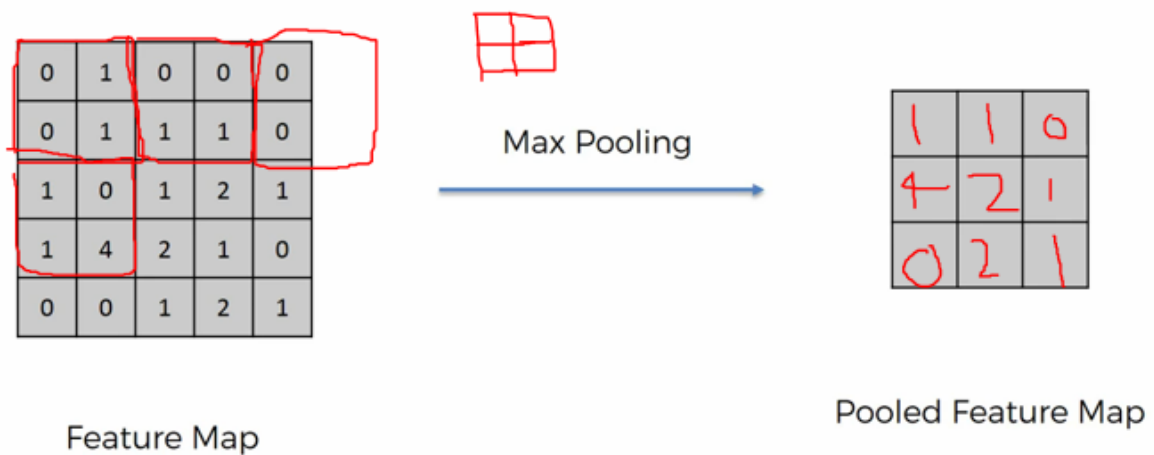| 9	7
8	6

2. Average Pooling

Average pooling computes the average of the elements present in the region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.

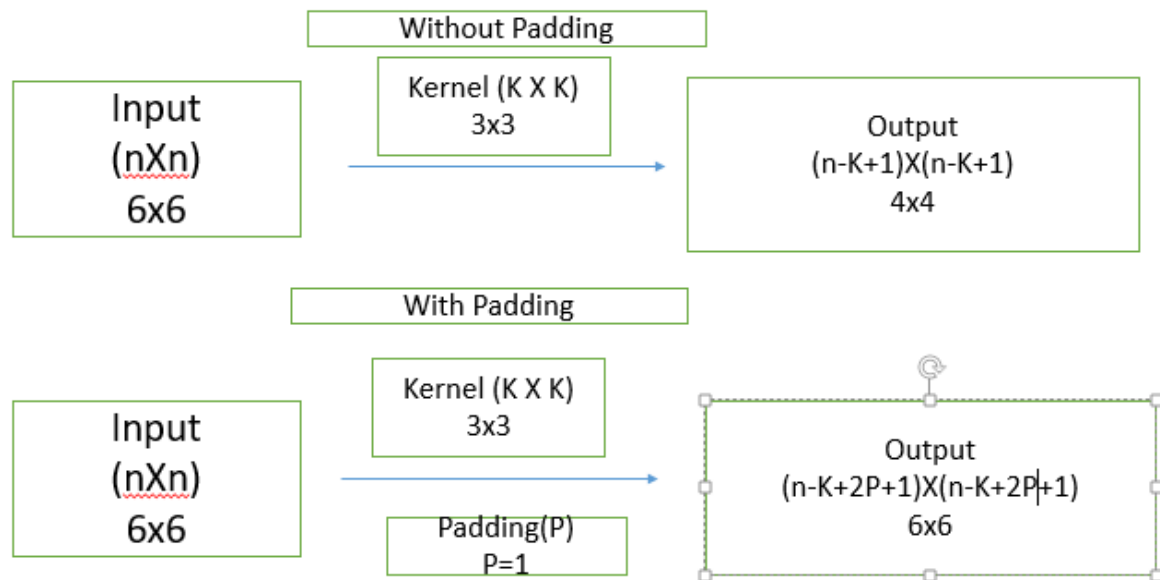- More On Pooling https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/ (https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/)

Now Apply Pooling in our above Feature Map

Problem with Simple Convolution Layers

- While applying convolutions we will not obtain the output dimensions the same as input we will lose data over borders so we append a border of zeros and recalculate the convolution covering all the input values.

     1. Padding
     2. Striding

1. Padding

Without Padding

| Input (nXn) 6x6 | → Kernel (K X K) 3x3 → | Output (n-K+1)X(n-K+1) 4x4 |

With Padding

| Input (nXn) 6x6 | → Kernel (K X K) 3x3, Padding(P) P=1 → | Output (n-K+2P+1)X(n-K+2P+1) 6x6 |

- See In without padding our input is 6x6 but output image goes down into 4x4 . so by using padding we got the same result.Padding is simply a process of adding layers of zeros to our input images so as to avoid the problems mentioned above.

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

Zero-padding added to image

* So padding prevents shrinking as, if p = number of layers of zeros added to the border of the image, then our (n x n) image becomes (n + 2p) x (n + 2p) image after padding. So, applying convolution-operation (with (f x f) filter) outputs (n + 2p – f + 1) x (n + 2p – f + 1) images. For example, adding one layer of padding to an (8 x 8) image and using a (3 x 3) filter we would get an (8 x 8) output after performing convolution operation.

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

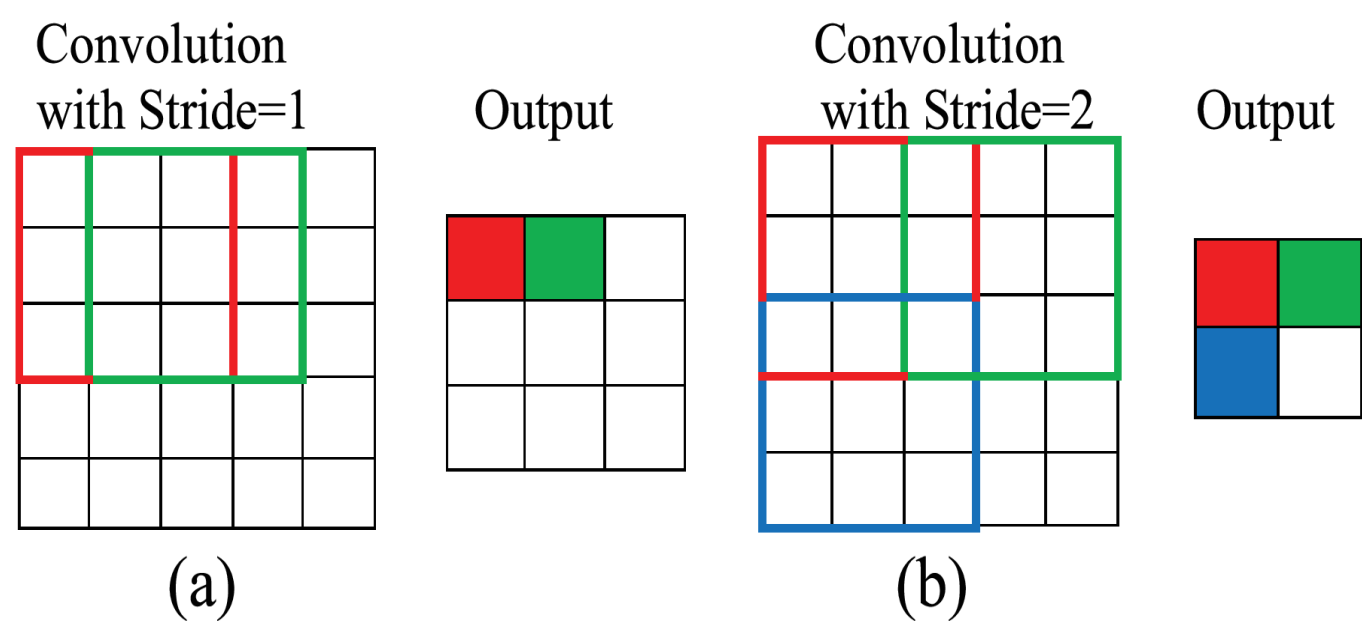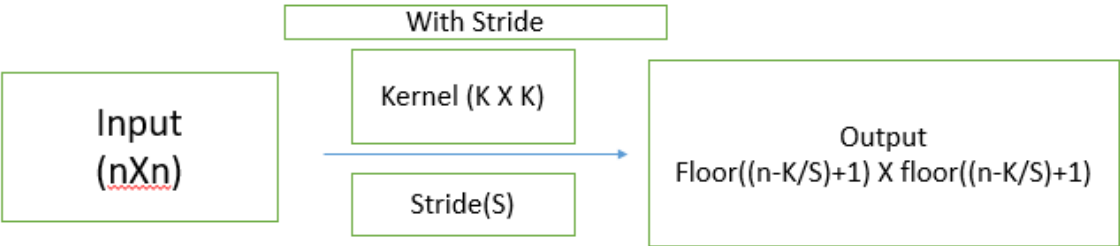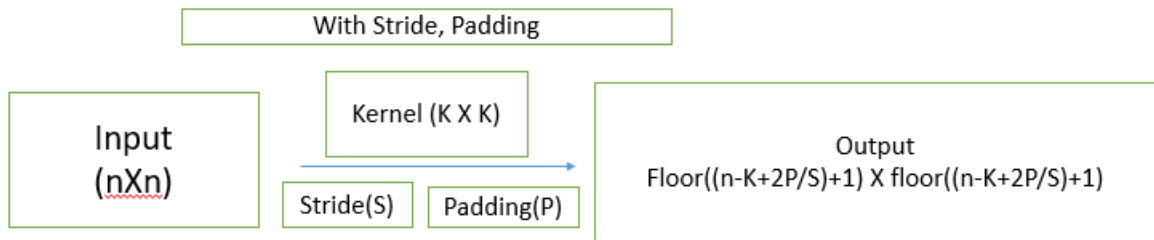| 114				

2. Strides

- It uses to reduce the size of matrix. if we sfited by 1 then we called stride=1 and if we sfited by 2 means stride = 2 so on.

With Stride

Kernel (K X K)

Input
(nXn)

Stride(S)

Output
Floor((n-K/S)+1) X floor((n-K/S)+1)

Convolution
with Stride=1

Output

Convolution
with Stride=2

Output

(a)

(b)

Padding,Stride Put in One Equation

With Stride, Padding

Input (nXn)

Kernel (K X K)

Stride(S)   Padding(P)

Output
Floor((n-K+2P/S)+1) X floor((n-K+2P/S)+1)

Step3 : Flattening

- Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.

Pooled Feature Map
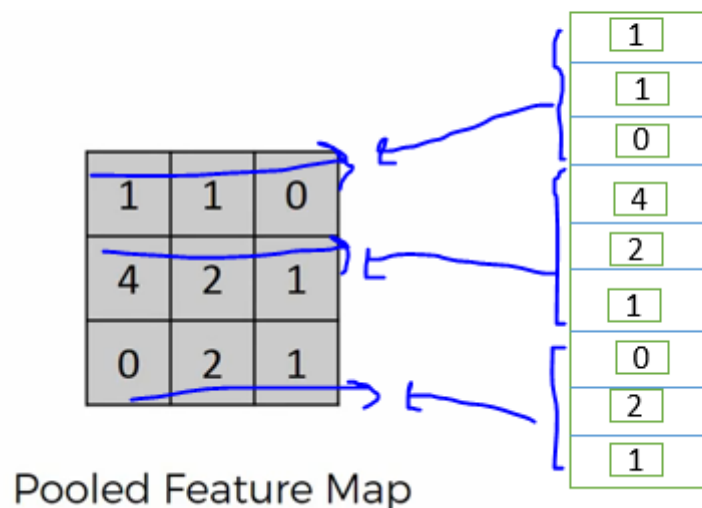
Step 4

Complete CNN in one View

here in last step we use full connection network