why not ANN

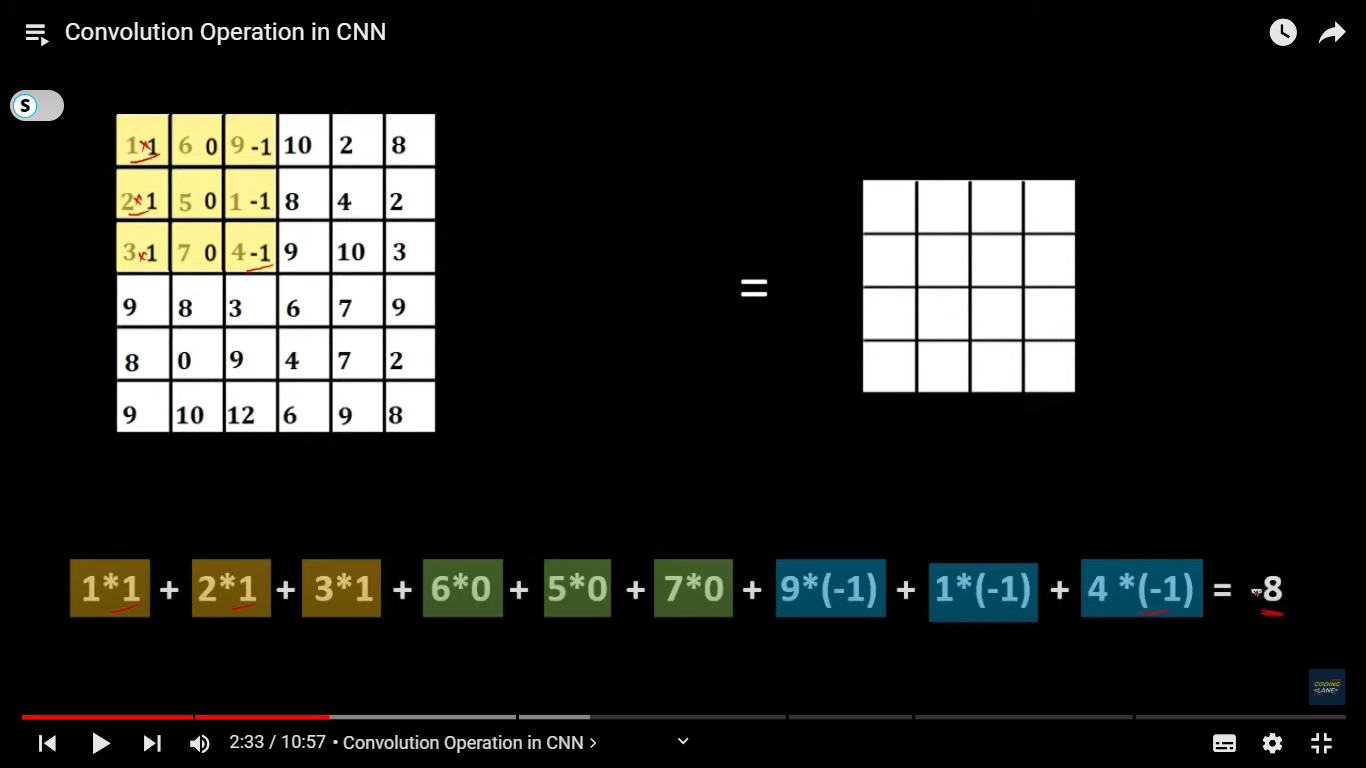
lets say we are dealing with a image dataset and we are working on RGB colors having size of 1024 x 1024 this means it will have 3 million input features. lets say first hidden layer has 100 input feature this means total no.. of weight parameters will be 3 billion.

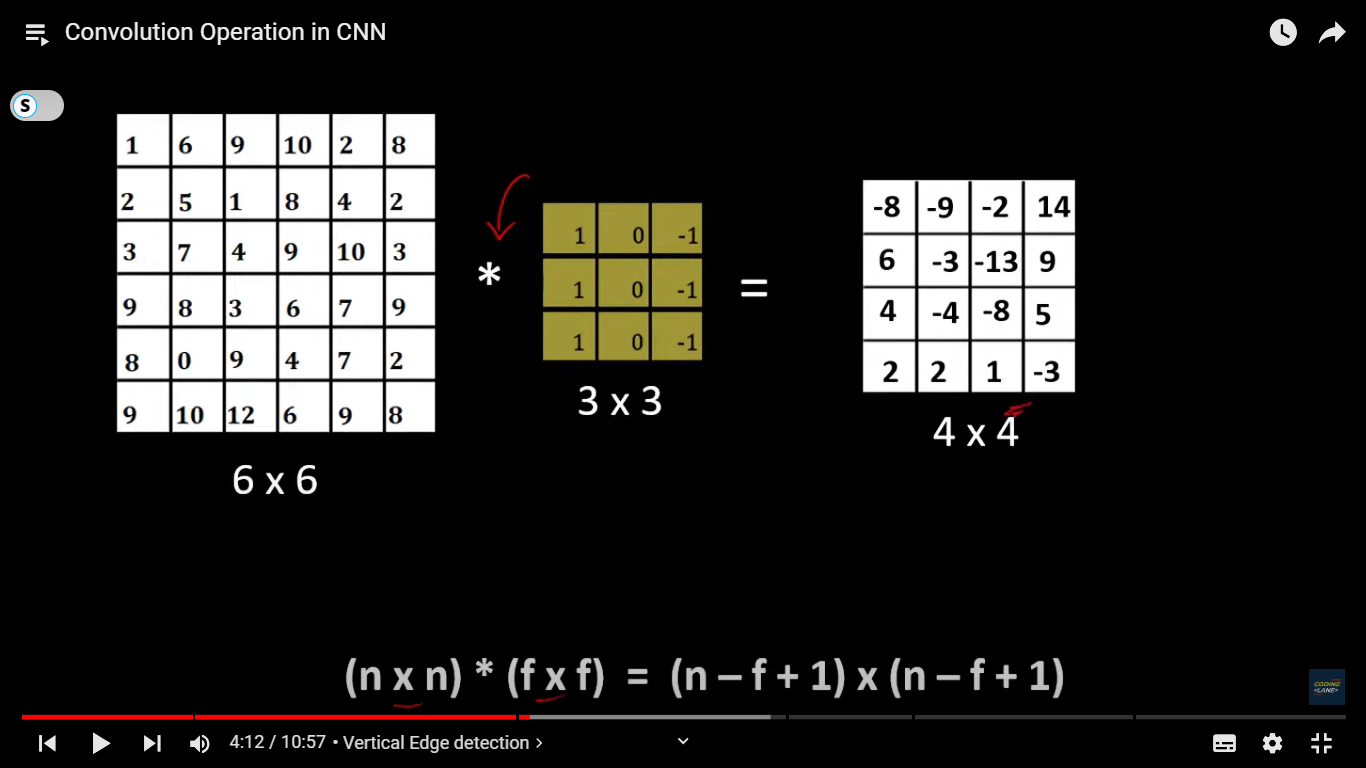
Our CPU might not be able to handle them properly or even if they did time taken to train model will be too high, more no. of parameters means overfitting also

CNN – main idea to use filters. Filters are sliding window responsible to detect its features

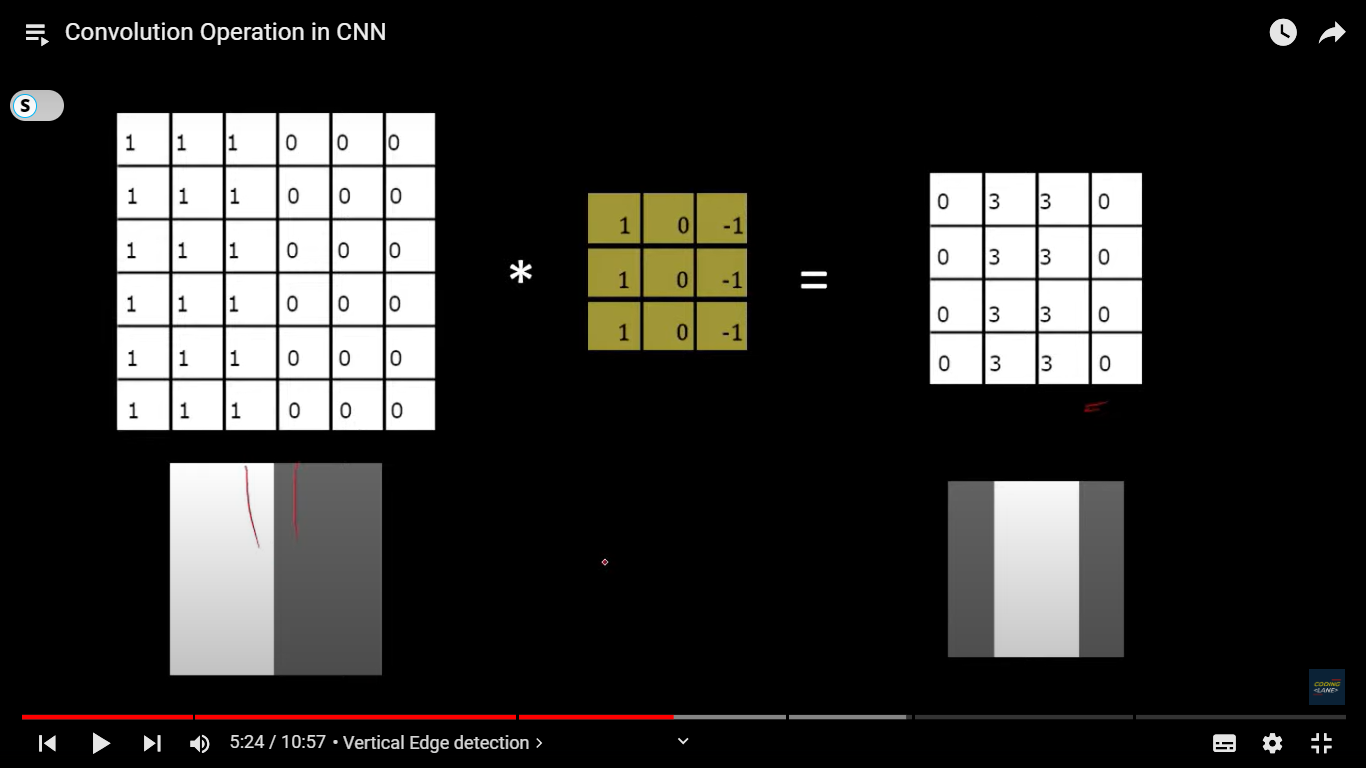
If the image is grayscale or black and white then the data will range from 0 to 255, where 0 is black

Convultional operation





convolution operation is responsible for identifying the edges and the features from the images









Same

Here it is 1 as its gray scale or else it is 3 – RGB

Padding

when we convolve a 6x6 image with a 3x3 filter we get the output as 4x4 so the size of the final image get reduced by some amount and if we are using many many such layers in a convolutional neural network then it is possible that the final size of the image get reduced so much that we might lose the valuable info  
uneven exposure of pixels.  Ther are some with more exposure to filter while some don’t

we just add layers of zeros so both our problem get solved

There are 2 types of convolution

1. Valid convolutional – no padding
2. Same conovolution – nxn is our image and fxf is feature

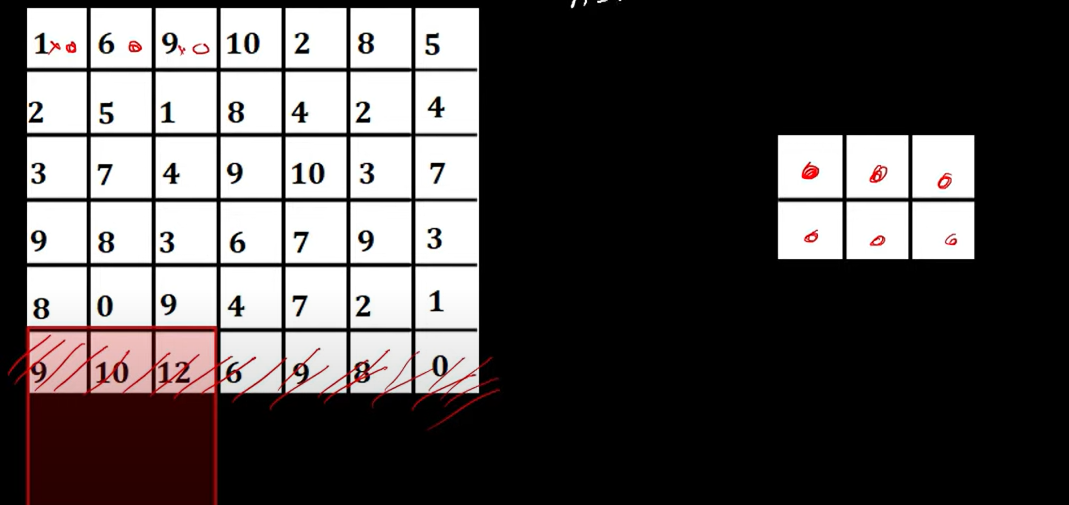
Our new input image is

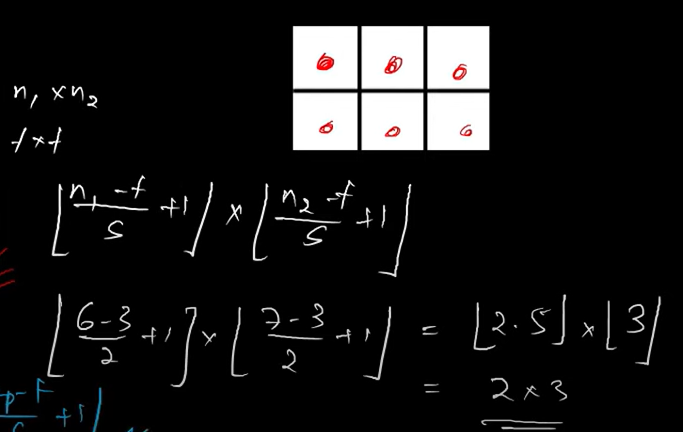
N’ = n + 2p where p Is padding

P = (f-1)/2 f is usually odd if f is even then we need to understand uneven padding



Stride - by how many pixel we move our sliding filter window





Discard if we don’t have enough pixel values

Final image becomes = floor value of (n-f)/s + 1

Max-pooling -

If fix filter and stride the we go on iterating and then we just take the max value from the feature window and keep moving

Reduce the size of image, reducing computation cost without losing info rather enhancing them

Generally applied after convolutional layer, not always necessary after convolutional layer as It reduces size if we use it too much then it might loose valuable info. So only se when convolutional layer is too big

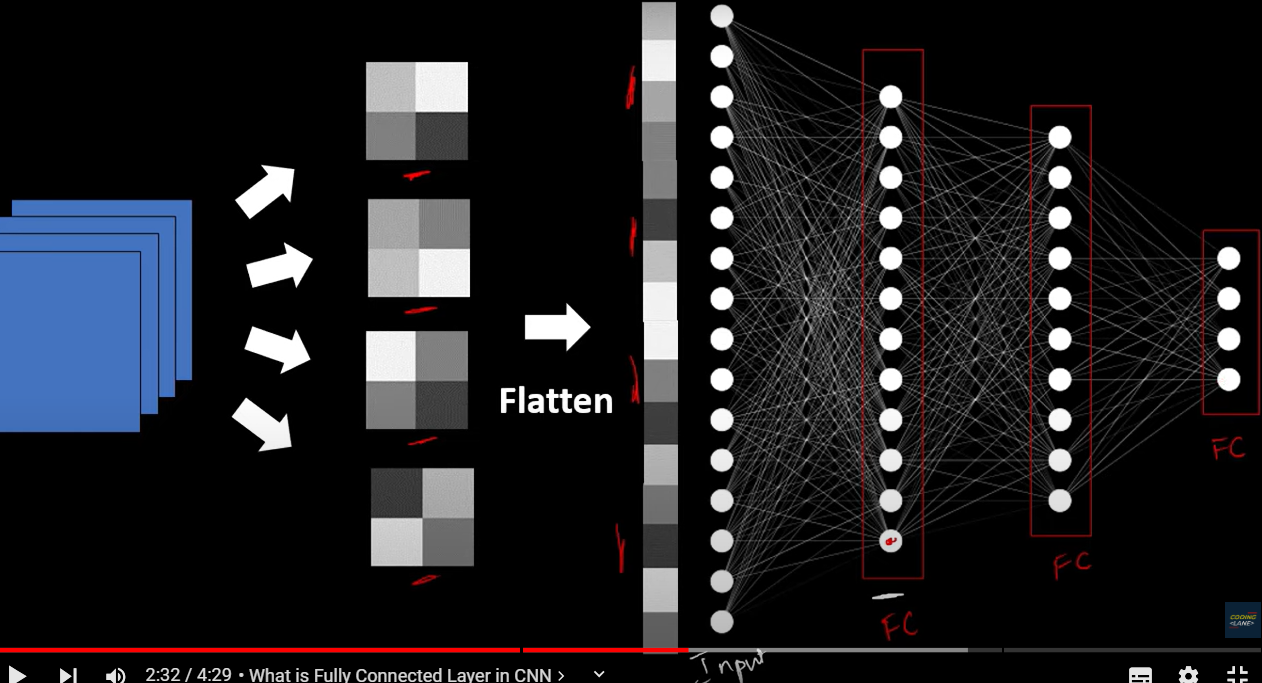
Avg Pooling – takes the average value

Fully connected layer –

Used to classfy features

A dense network of neurons and connection between 2 neurons

Before we pass images to fully connected layer they are flattened using flatten layer



Connection between neurons is also called as weights. And so fully connected layers are also used for learning and associating a feature to classification categeory

Softmax – for calssificatino into many categeory output layer = no. of classifying categeory

Sigmoid – binary classification only 1 output feature

Conlutional layer consist of = applying convolutional operation and giving it to passign to non linear activation function while passing some bias

Traditionaly in cnn – layer of convolution and maxpooling is considered as 1 layre

But before we predict we need to train the model to learn and so then we will calculate cost function based on output of the last fc layer to get amount of error we are getting

For Binary - binary cross entropy function

For mulit-class – categeorical entropy function

Then we back propogate to minimize the cost function eg – SGD, gradient descent