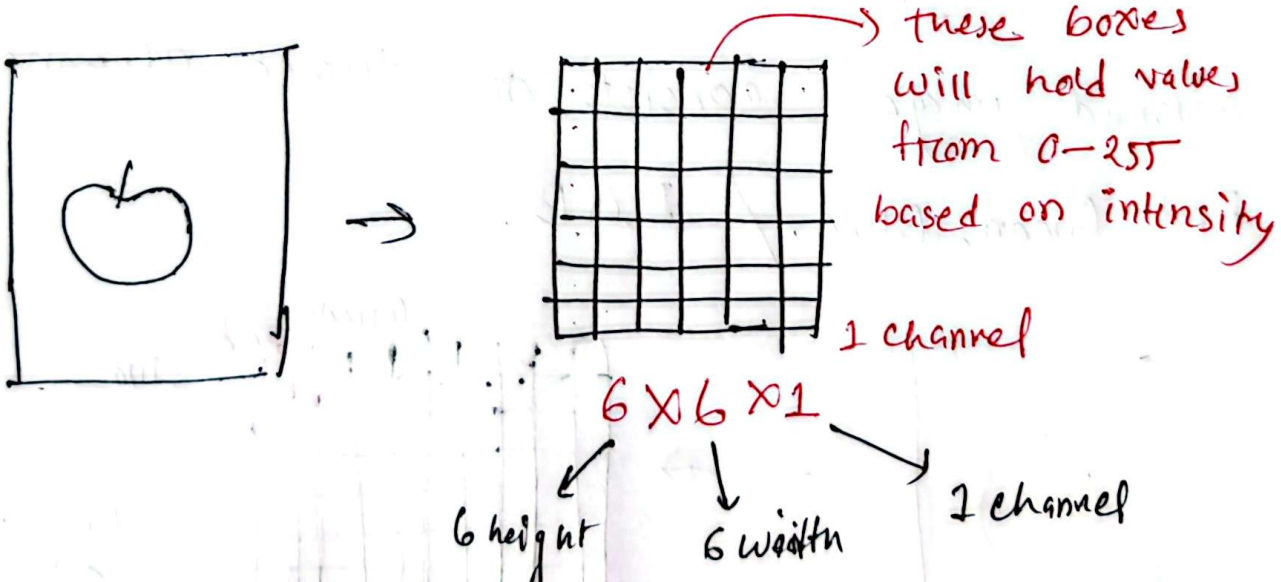
CNN

Convolutional Neural Network handles Image type datasets. Usual Artificial Neural Network may get complex due to the large number of features an image can have. So, before giving an image ~~in~~ to an ANN, we apply convolution layer on the input images to extract important patterns from the pictures.

Image:

Grey-scale Image →

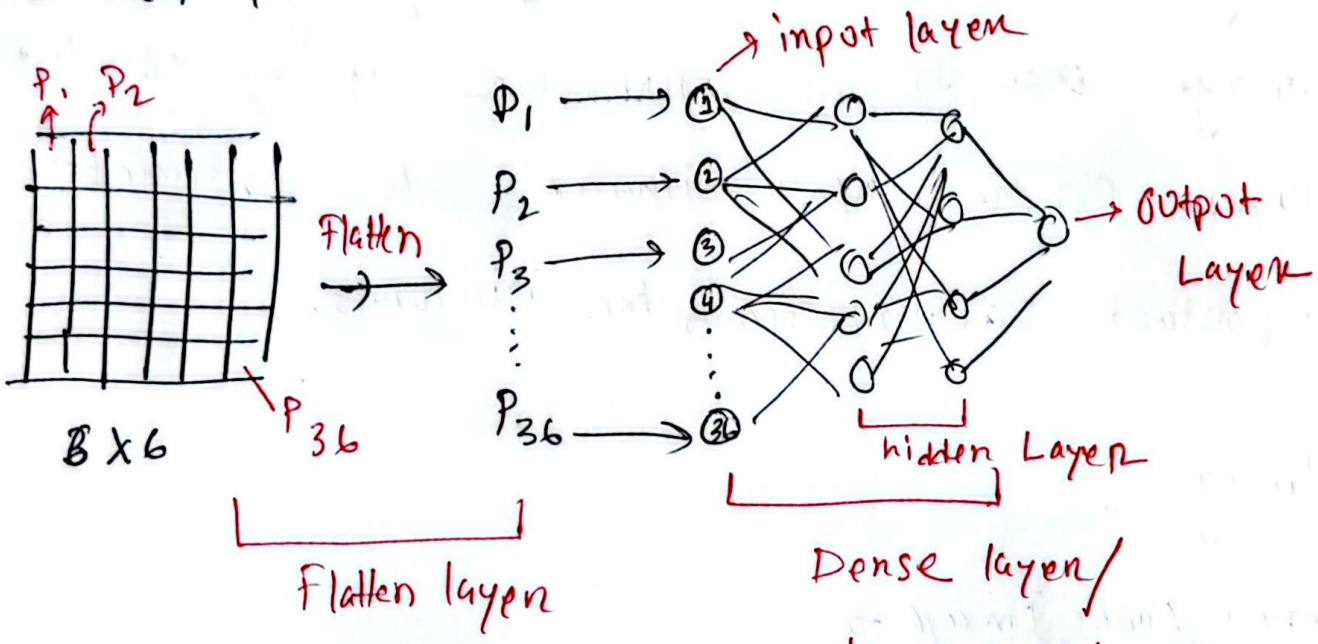
Black and white image has only one channel →



This image has 6*6 pixels = 36 pixels.

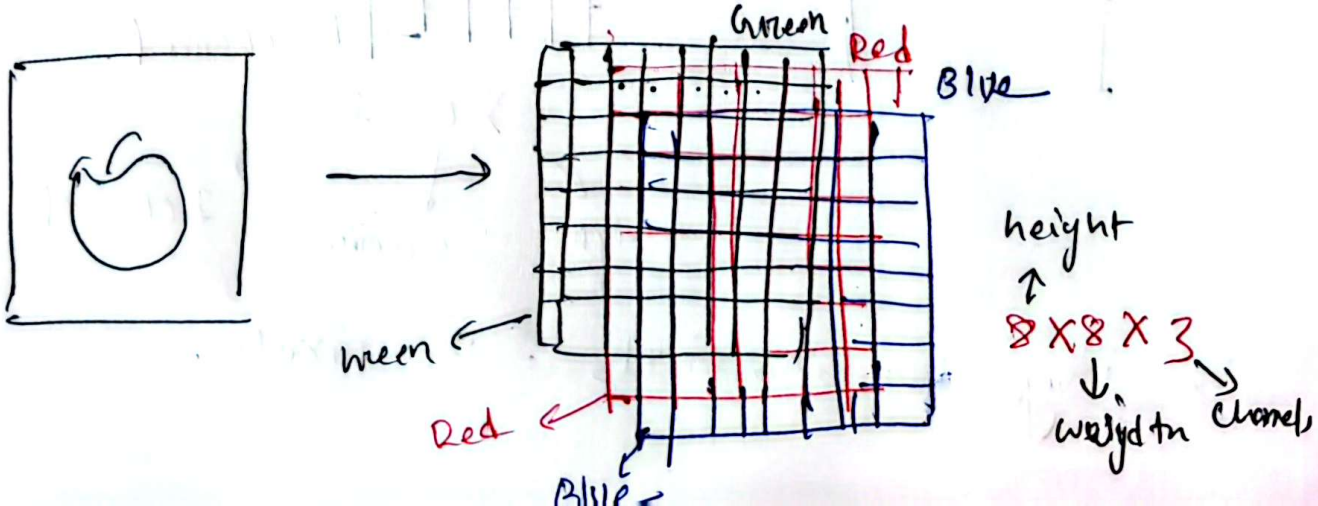
If we want to input this image inside an ANN architecture, the Neural Network will have 36 neurons in the input layer.

So, we will Flaten the image from 2D to 1D and then pass the pixel values inside NN.



Colored Image →

Colored images consists of three channels → Red, Green, Blue. / RGB.



In case of three channel imges, the number of pixel is usually same for each channel.

Each channel has = 8 X 8 = 64 pixels.

there are total 3 channels,

so, total pixel 64 X 3 = 192 pixels.

for simplicity we have taken example of small dimention image. in real the dimention can be more and which will increase the number of pixels / feature for the input layer. To handle this problem we add convolution to our image before training with Neural Network / dense layer.

Filter:

we use filters to perform convolution on image. filters are used to extract pattern from imges.

For example →

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

this filter is used to extract horizontal Edges.

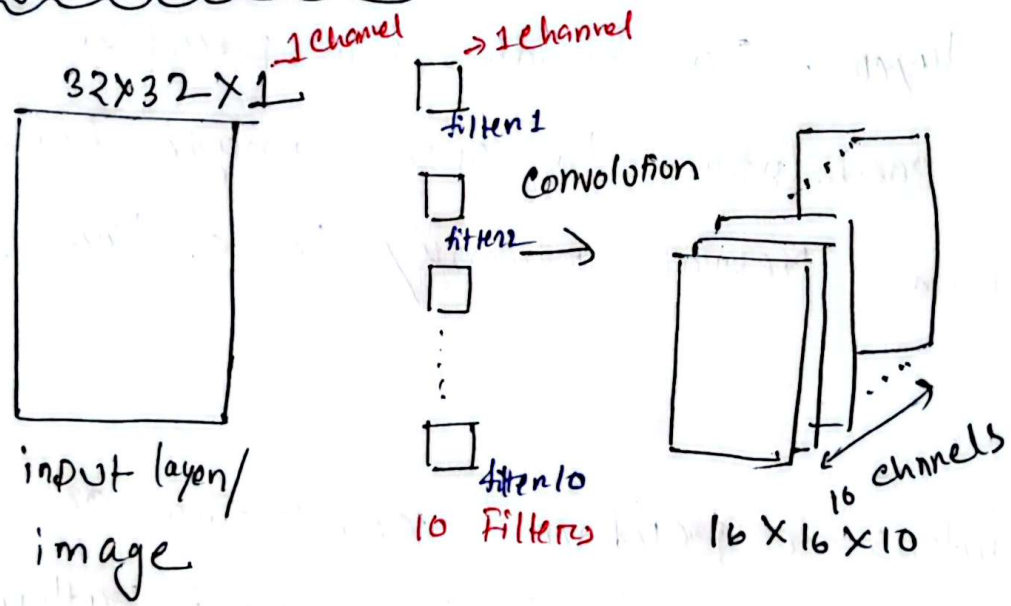
1	0	-1
1	0	-1
1	0	-1

This filter is used to extract vertical edges from an image.

Check the "Tea cup" image from the slide to understand it better.

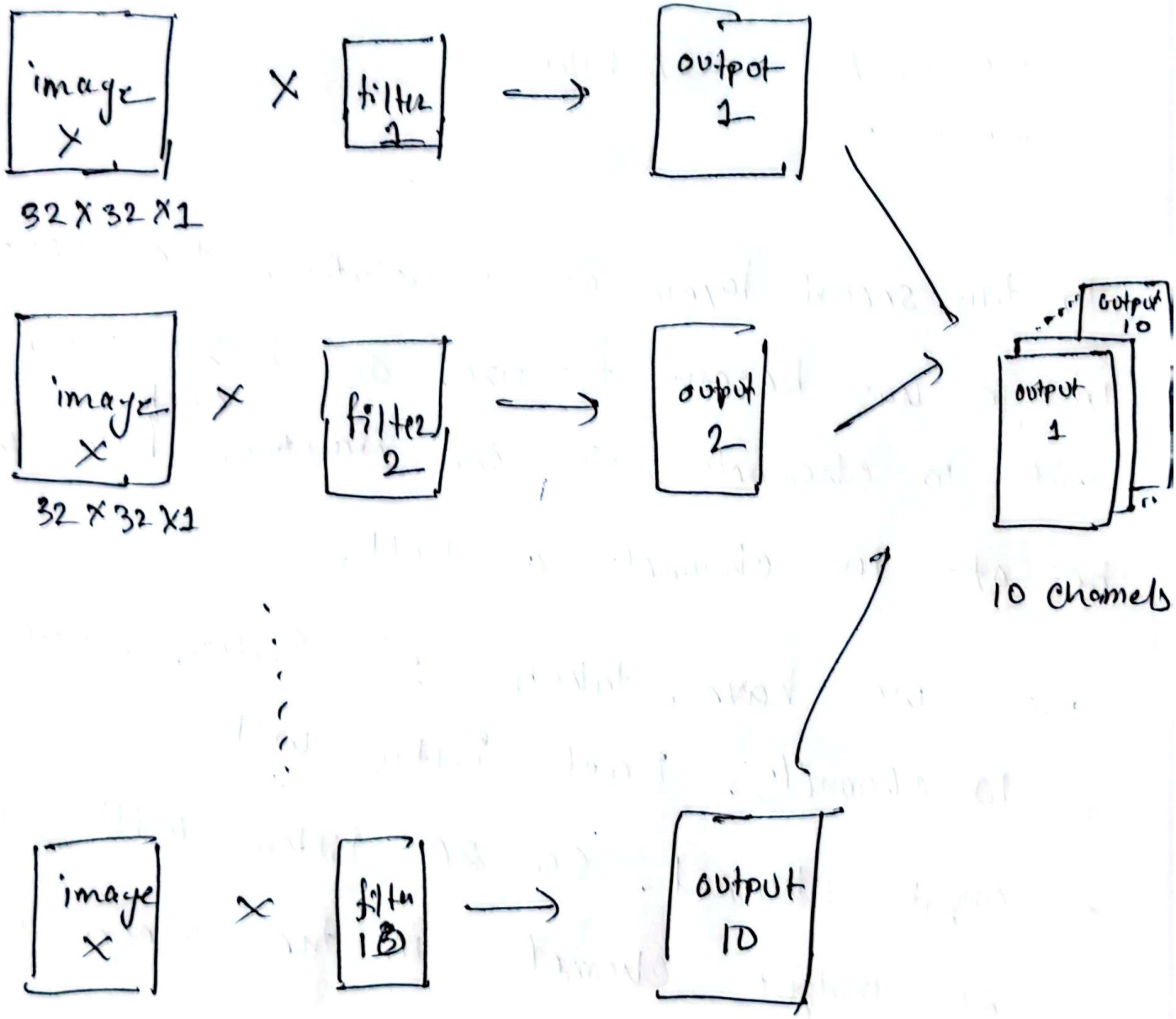
Convolution layer:

Single channel image →

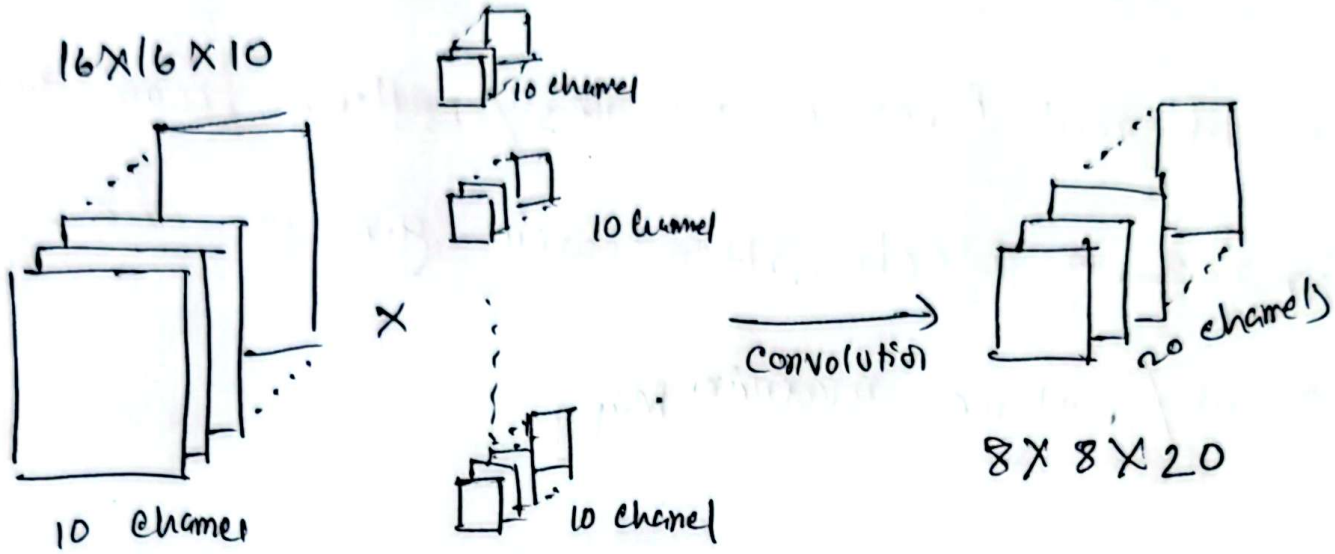


Here the image consists of 1 channel. So the filter will be of 1 channel as well. But we can use multiple feature to extract multiple different types of pattern (horizontal edge, vertical

edge etc). Here 10 filters will extract 10 different types of features / pattern from the image → Each filter will give us one output / feature map



Now on this 10 channel output, we can perform another convolution operation.



In this second layer of convolution, the input image has become the size of 16X16X10 with 10 channels. So, ~~the~~ Now ~~the~~ each filter will be of 10 channels as well.

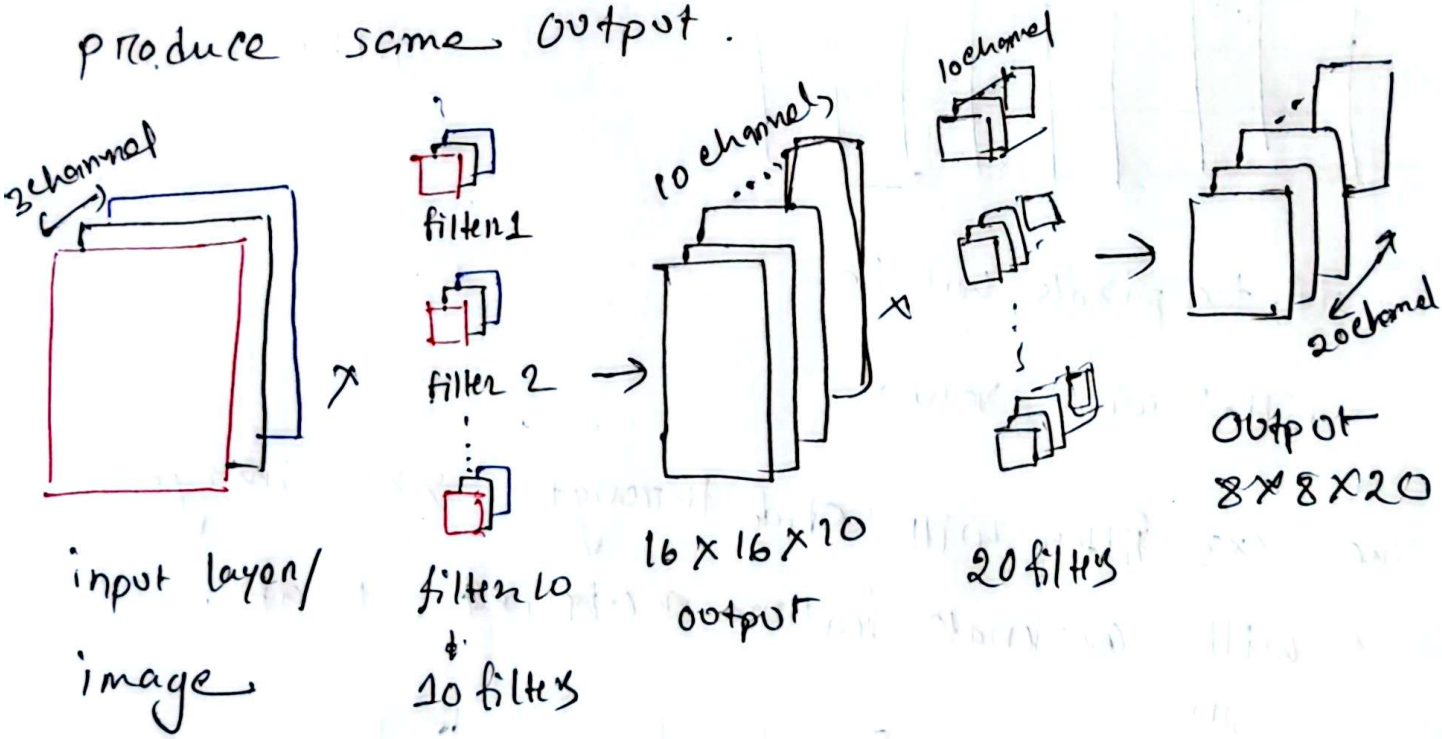
Here we have taken 20 filters, each of 10 channels. Each filter will give us 1 output channel. So 20 filter will give us 20 output channel in the dimention of 8X8X20.

We can see the feature is reducing an we can pass this to 8X8X20 size output

into a flaten layer and then pass the values inside ANN/ Dense layers for prediction.

Colored image:

The operation is same as grey scaled image, but the only difference is the input images consists of three ~~images~~ channels. So, we need 3 channels per filter for the first convolution layer. The other layer will produce same output.



Here ~~per~~ 1 filter produces 1 chanel / feature map ~~matrix~~ for the output as well.

Convolution Operation:

Let us.

2 channel:

Let's consider an image of 8×8×1 and filter of 3×3 size.

Image

0.6	0.2	0.6	-0.6	0.2	-0.6
0.1	-0.2	-0.3	-0.1	0.2	0.3
-0.5	-0.1	-0.3	0.5	0.1	0.3

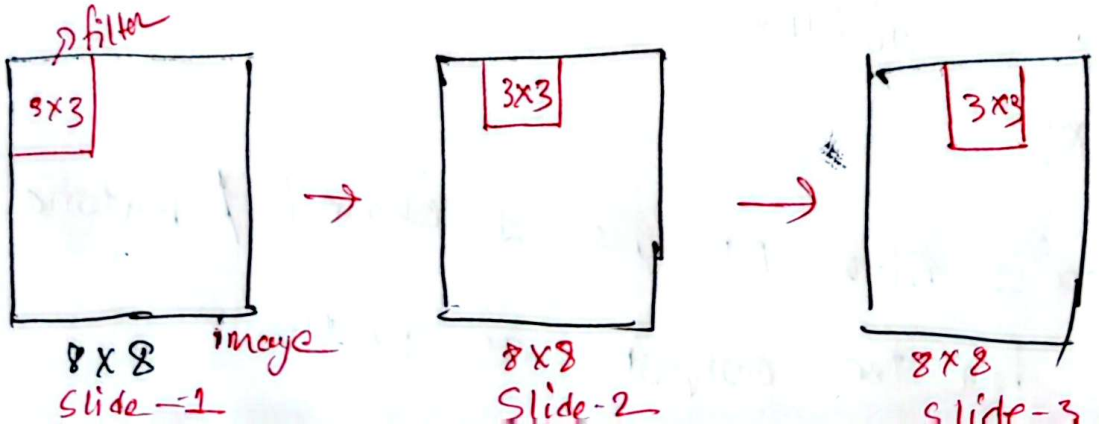
Filter

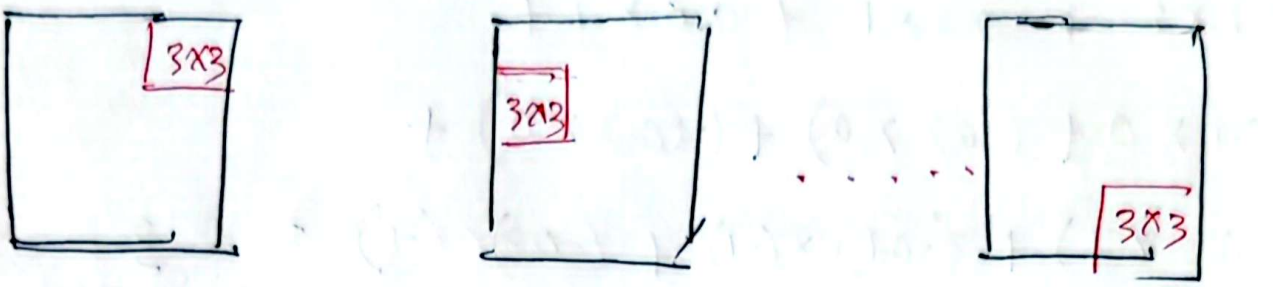
1	1	1
0	0	0
-1	-1	-1

3 × 3 × 1 filter

All the pixels will be filled with values

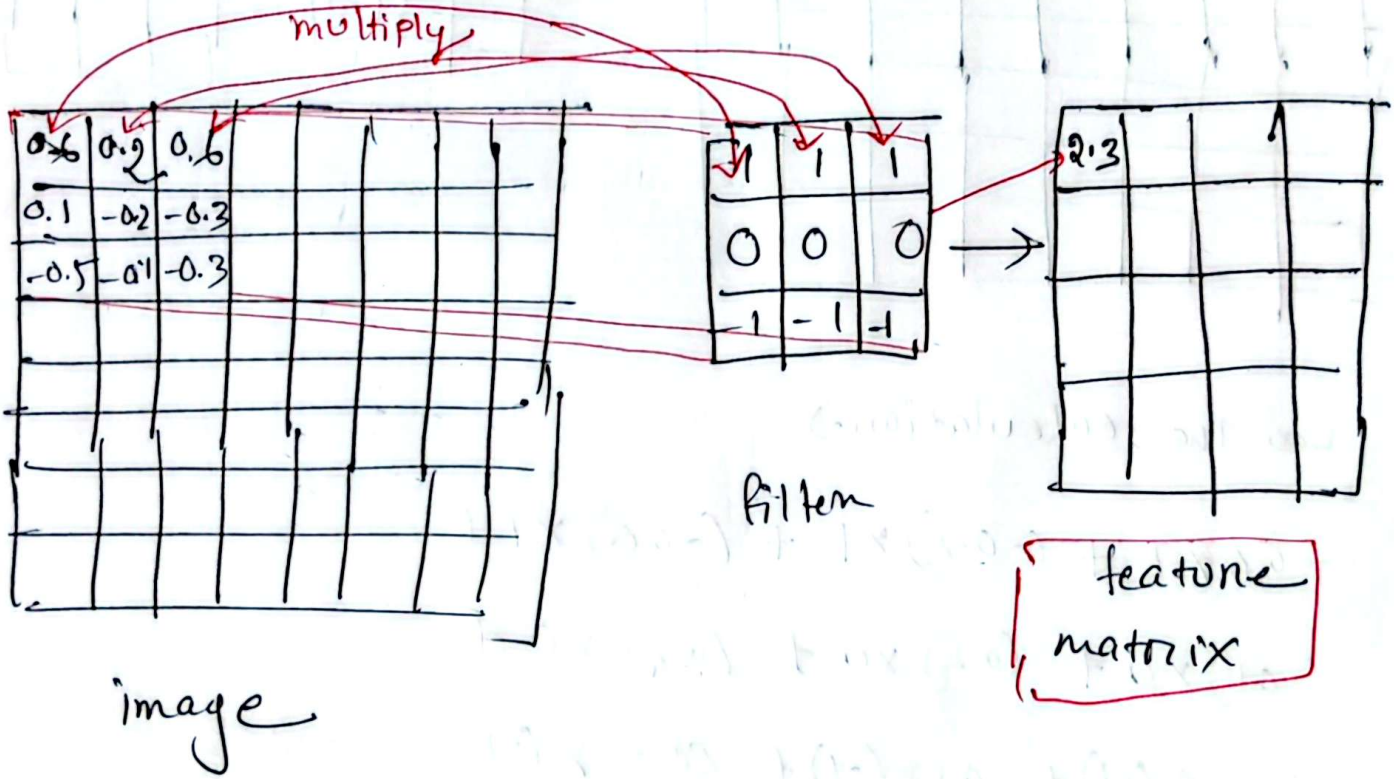
The 3×3 filter will slide through 8×8 image and will generate feature map.





we can fix the value of Stride (How many steps the filter skips to slide though the image) and the feature maps will look different based on the value of Stride.

Let us calculate the first Slide →



here we multiply each pixel with its corresponding positions filter value and add all the multiplications.

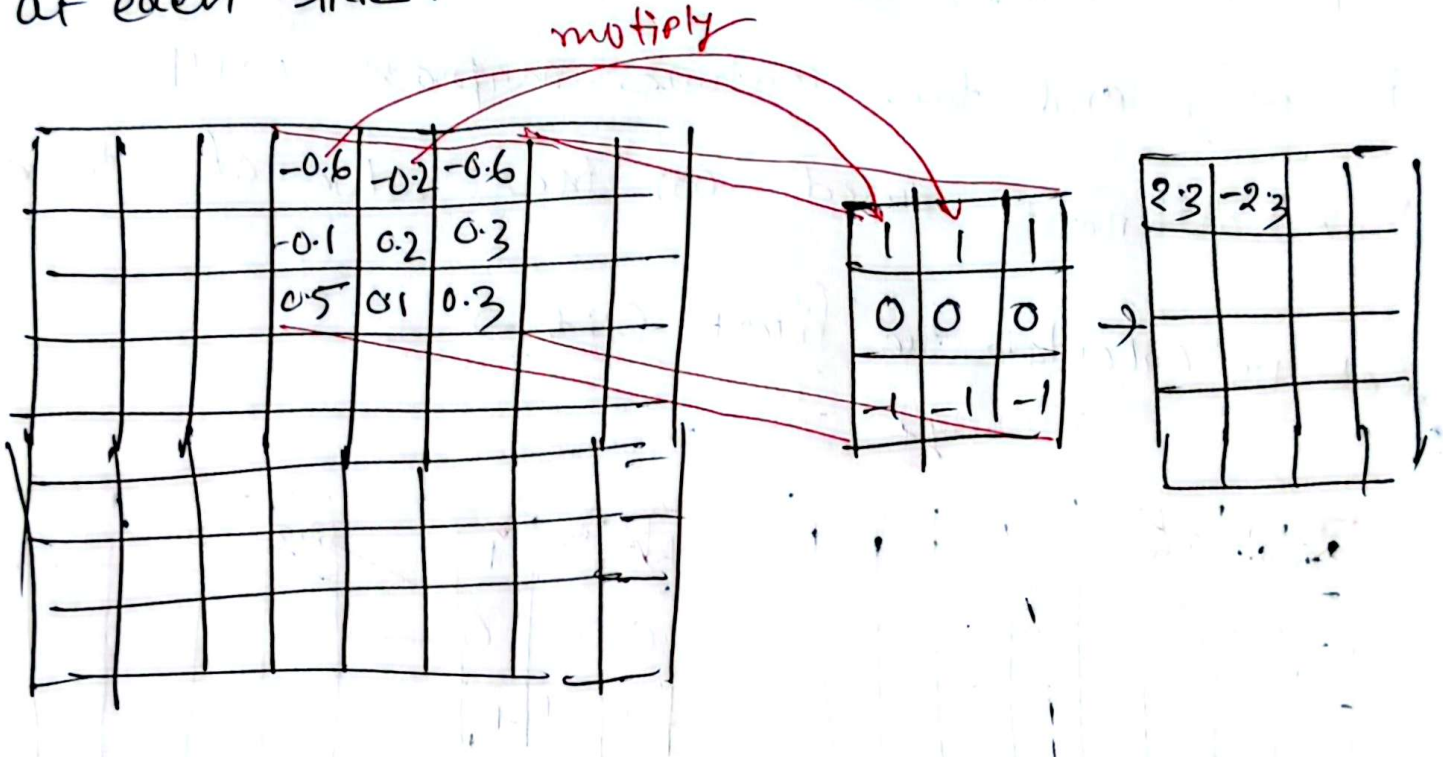
$0.6 \times 1 + 0.2 \times 1 + 0.6 \times 1 +$

$0.1 \times 0 + (-0.2 \times 0) + (-0.3 \times 0) +$

$-0.5 \times (-1) + (-0.1) \times (-1) + (-0.3) \times (-1) = 2.3$

second slide!

Let us consider we jump/skip three pixels at each slide.



see the calculation →

$-0.6 \times 1 + (-0.2) \times 1 + (-0.6) \times 1 +$

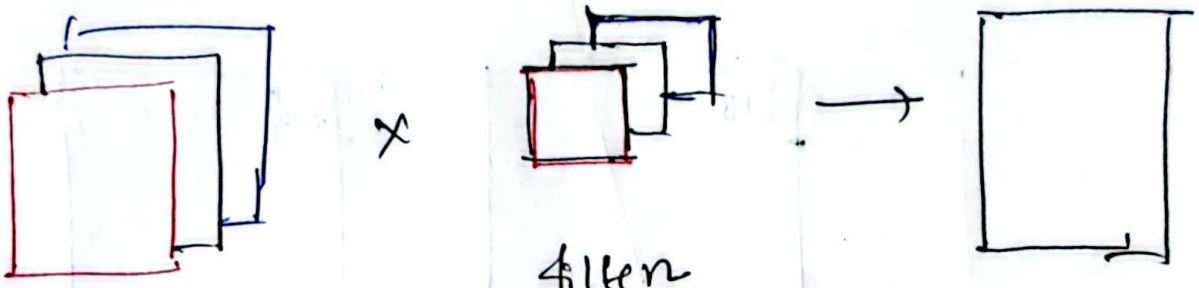
$(-0.1) \times 0 + (0.2) \times 0 + (0.3) \times 0 +$

$0.5 \times (-1) + 0.1 \times (-1) + (0.3) \times (-1) = -2.3$

this how we slide through the entire image and generate the feature map for one filter.

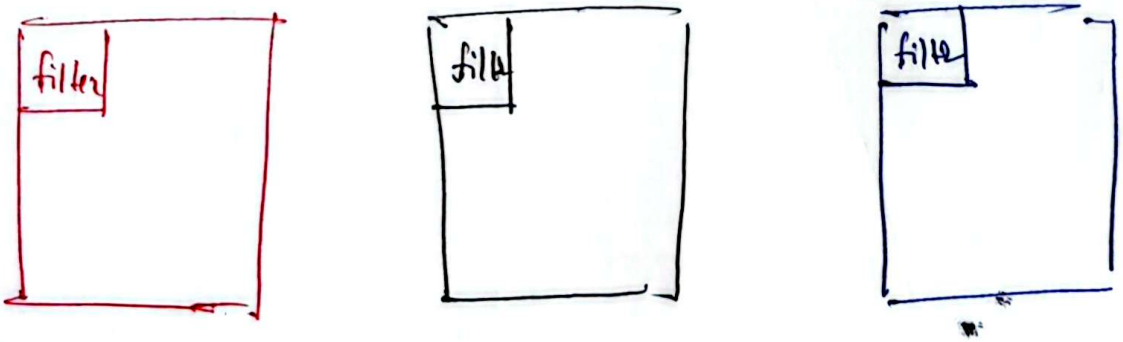
Three channel:

We know for colored image (3 channel) we use three channel filter.



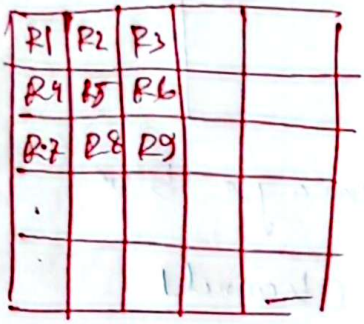
Here for 1 filter we will get 1 feature matrix as well.

Let us see the first slide for colored image →



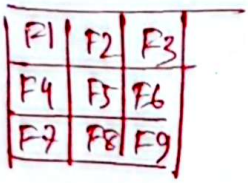
each channel in image has its own channel in the filter. We will multiply the value like before

and sum all the multiplication up to generate ONE number for the first slide.



Red channel image

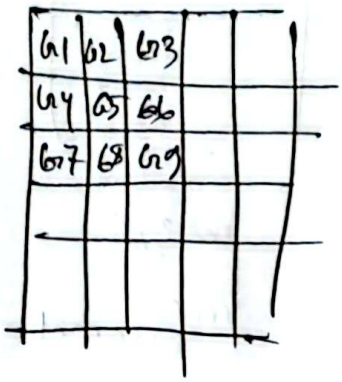
×



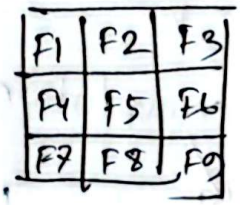
Red channel filter

⊛ convolution ↓

R1F1 + R2F2 + R3F3
+ R4F4 + R5F5 + R6F6
+ R7F7 + R8F8 + R9F9



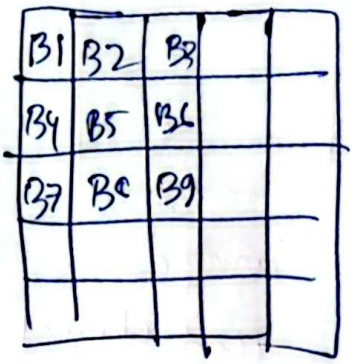
Green channel Image



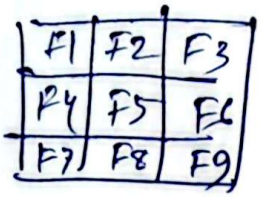
Green channel filter

↓

G1F1 + G2F2 + ... + G9F9



Blue channel Image



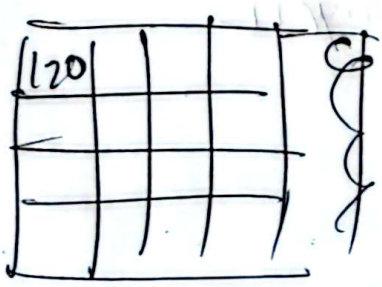
Blue channel filter

↓

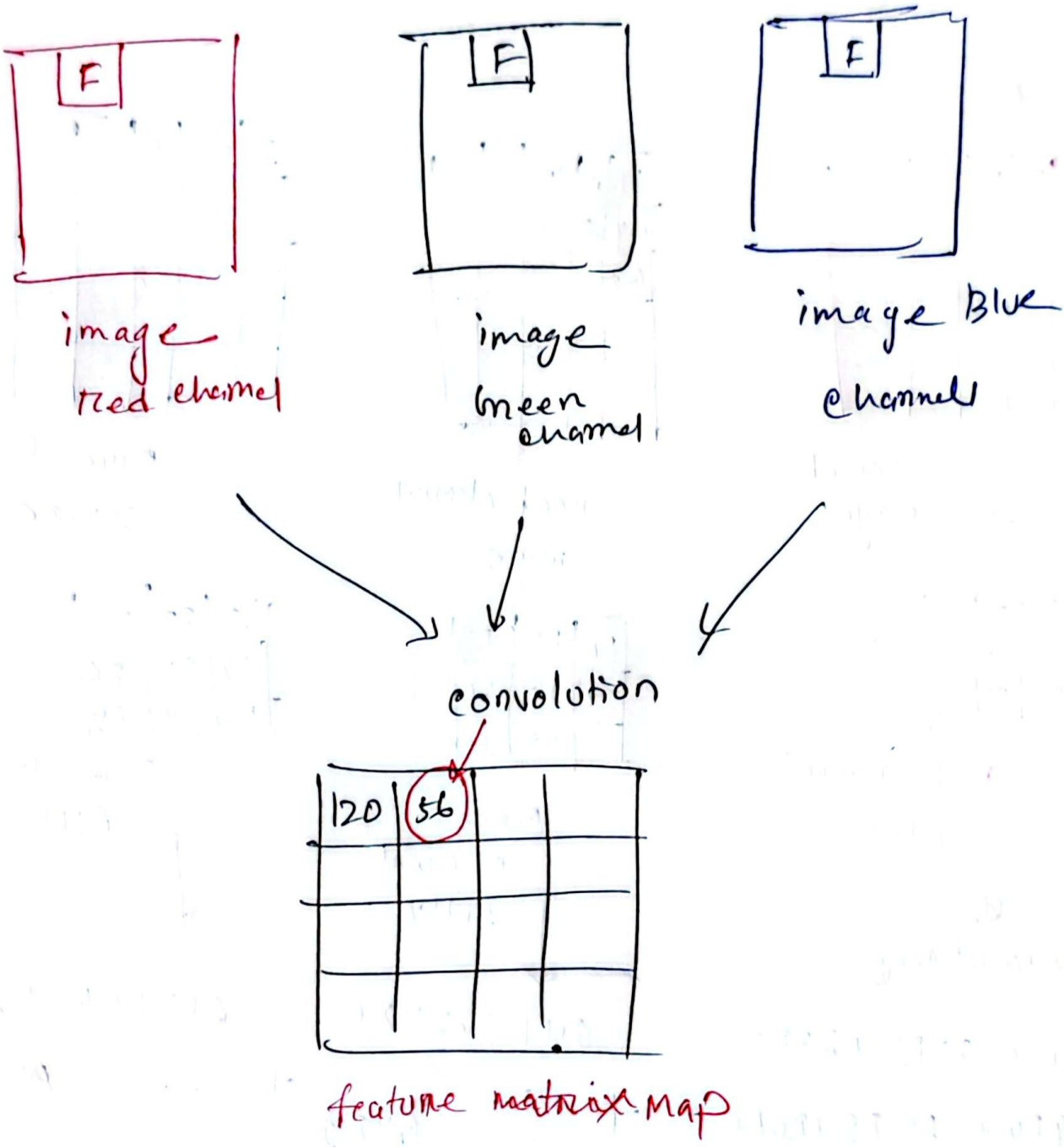
B1F1 + B2F2 + ... + B9F9

Let us assume the summation = 120.

So the feature map will look like →



Second Slide →

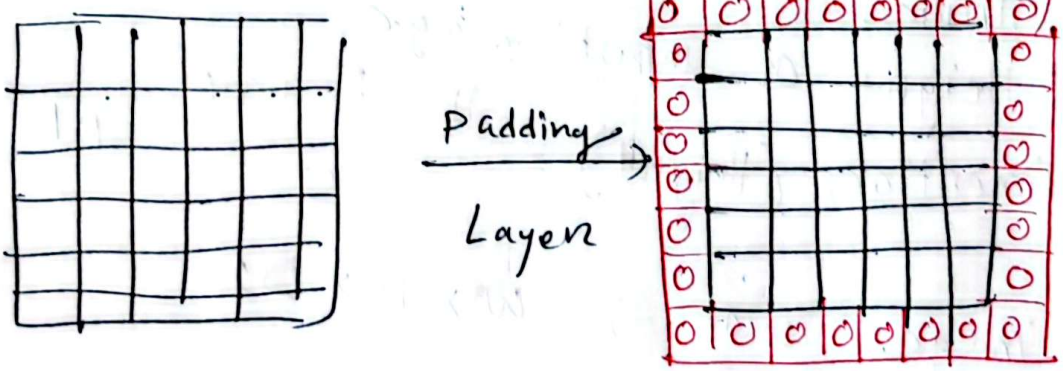


Now we need to know how to calate the dimention of this featured matrix. Before that we will learn about padding layer.

~~convolution operation~~

Padding Layer:

In general the feature map generated from the ~~feature~~ convolution operation has size less than the original input. In order to keep the size equal, the input image / feature map is padded with zeros around the input / feature map. It is called **Zero Padding**.



Actual Image / Feature map

1 Zero padded layer.

We can add multiple layer of padding. Sometimes the outer most row / column of the Actual image / feature map gets copied into the padding for more

accuracy. The padding layer can be added before convolution, but it is not mandatory.

Calculating the dimention of feature map:

The generated feature map's dimention can vary based on the input image size, filter size, Stride and padding.

$$\text{Output dimention (width)} = \frac{W - F + 2P}{S} + 1$$

W = Width of input image

F = Width/height of filter

P = Number of zero padding

S = Stride

H = Height of input image.

$$\text{output dimention (Height)} = \frac{H - F + 2P}{S} + 1$$

If input imge is of → $W_1 \times H_1 \times D_1$

Filter → $F \times F \times D_2$

Number of filters ⇒ K

the output/feature map dimention = $W_2 \times H_2 \times K$

where $W_2 = \frac{W_1 - F \times 2P}{S} + 1$ and $H_2 = \frac{H_1 - F + 2P}{S} + 1$

Convolution operation as Matrix ~~Operation~~ Multiplication

Let us consider an ~~image~~ filter of $11 \times 11 \times 3$ and ~~filter of~~ image of $227 \times 227 \times 3$. If $P = 0$ and $S = 4$,

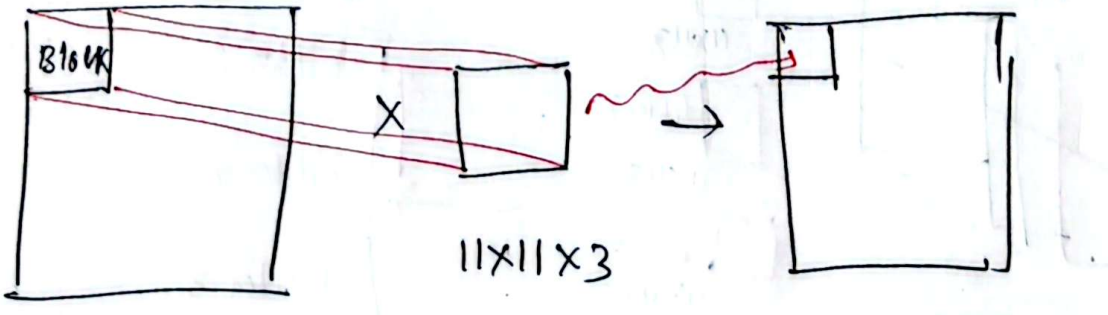
the dimention of featone map $= \frac{W - F + 2P}{S} + 1$

$= \frac{227 - 11 + 2 \times 0}{4} + 1$

$= 55$

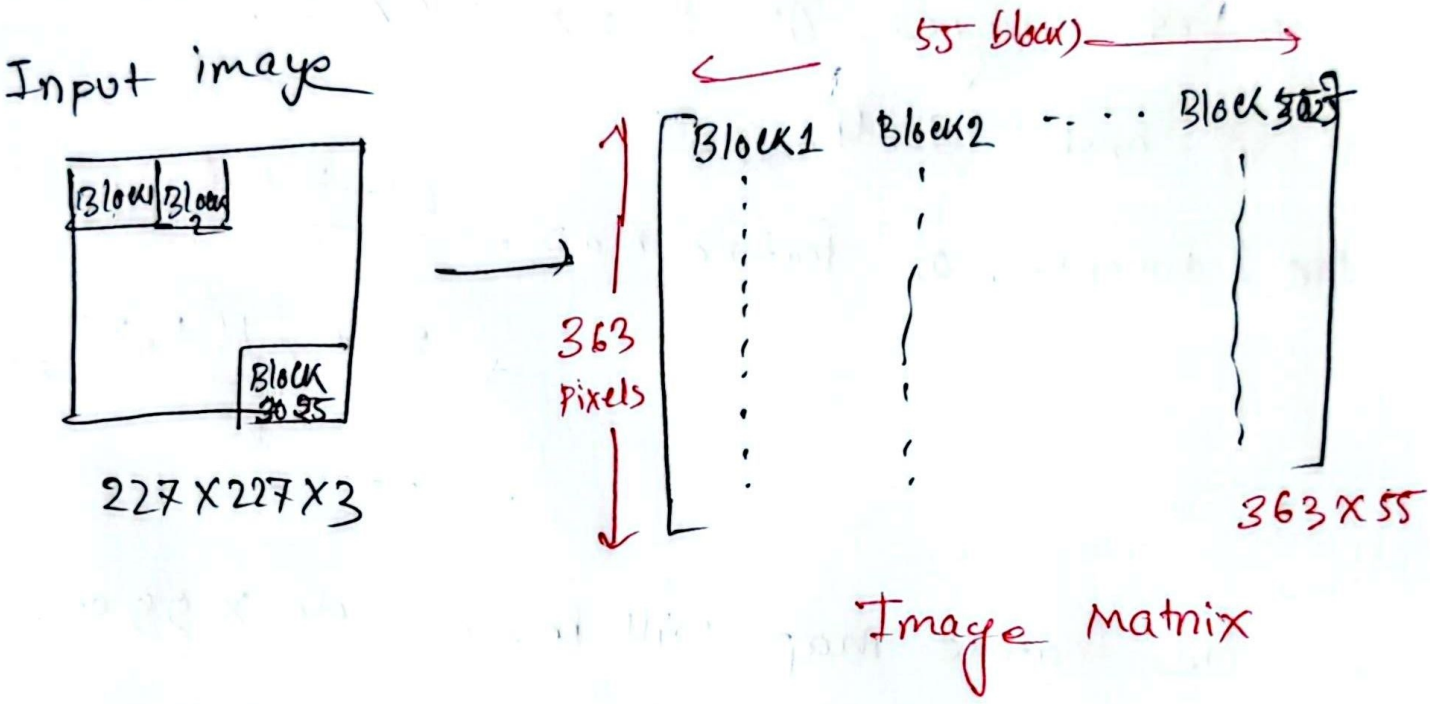
so, the featone map will be of $55 \times 55 \times 1$.

so, we are basically dividing the image into $55 \times 55 = 3025$ blocks using the $11 \times 11 \times 3$ filter.

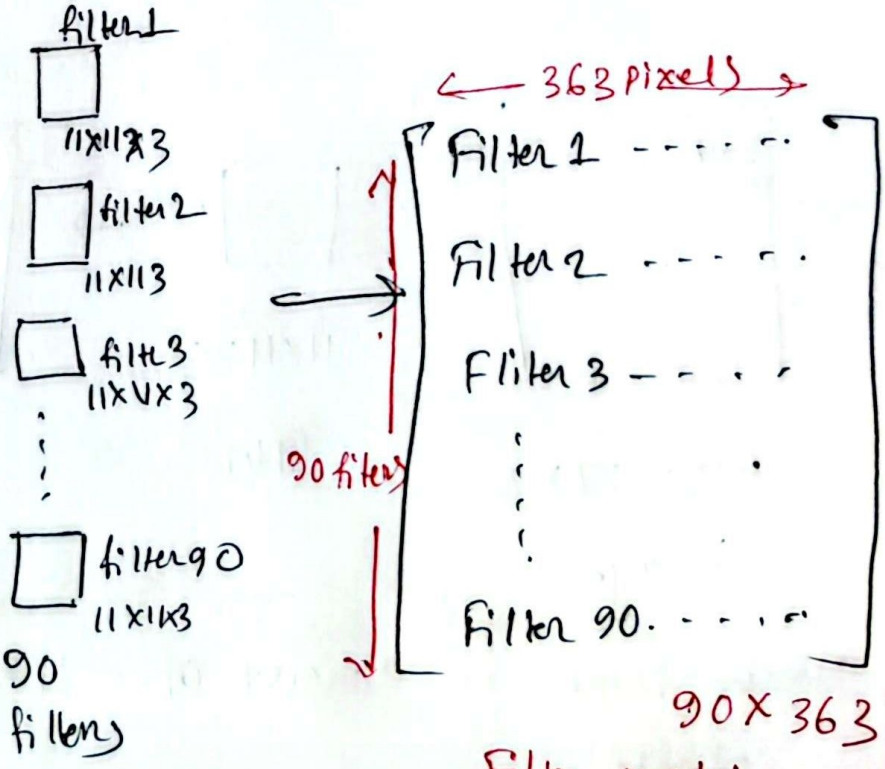


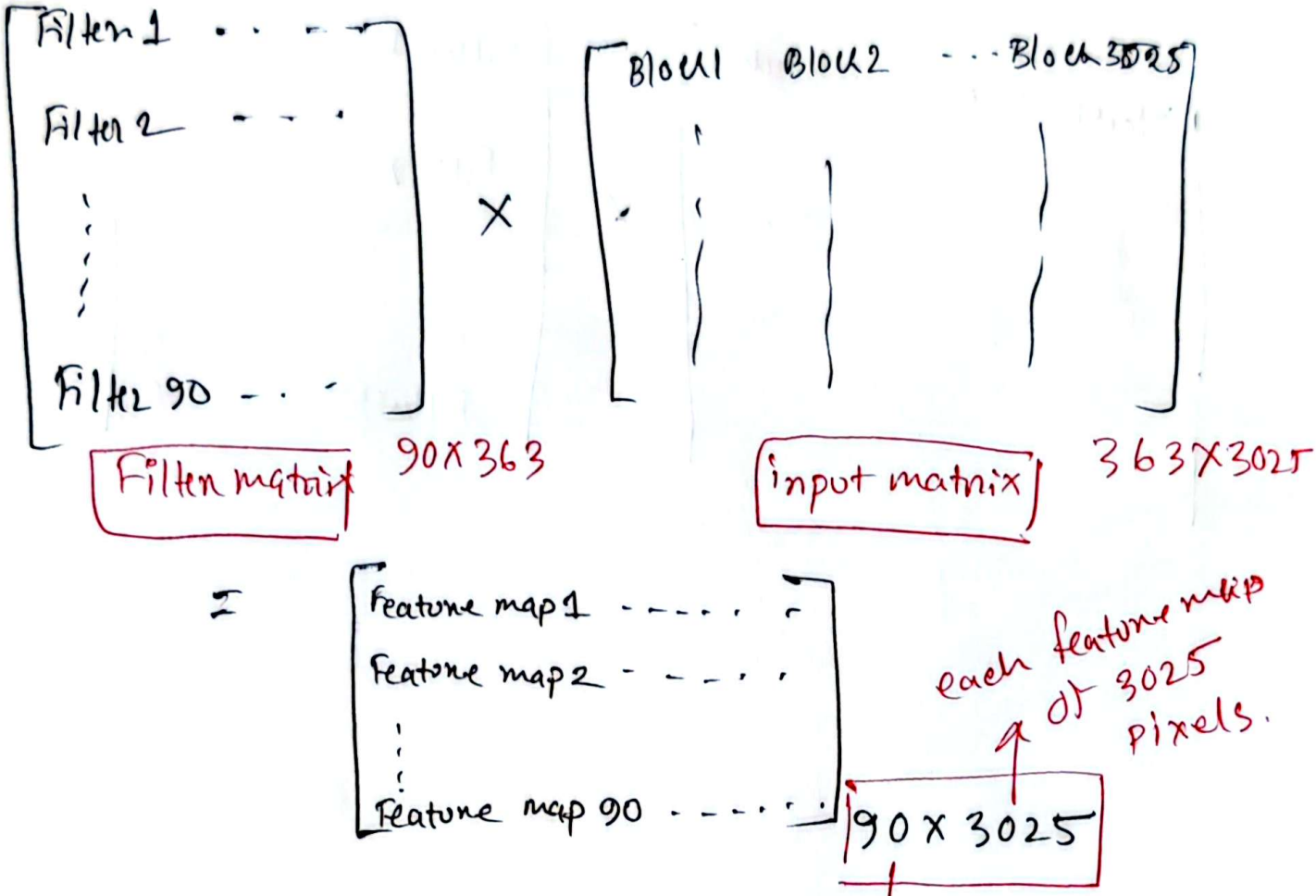
Each block is consist of $11 \times 11 \times 3$ pixels $= 363$ pixels.

We can represent each image as 3025 blocks where each block has 363 pixels.



If there are total 90 filters use in the Convolution layer, the filter matrix will be off





$\therefore$ output feature map matrix = Filter Matrix x input matrix

$= (90 \times 363) \times (363 \times 3025)$

$= 90 \times 3025$

$= 90 \times 55 \times 55$

55 × 55 — feature map dimension

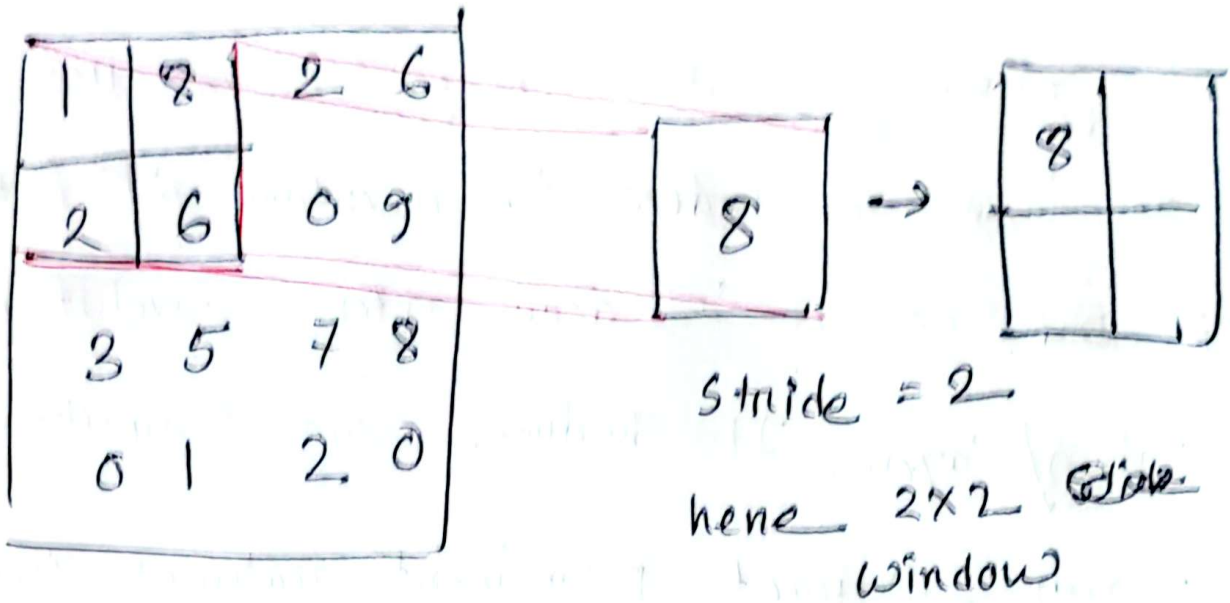
90 — 90 channels.

Pooling Layer:

After generating feature maps, we can use pooling layer to reduce the dimention of feature map. So it is done after convolution operation/ layer. It reduces the computational complexity, extract prominent features and reduces dimention.

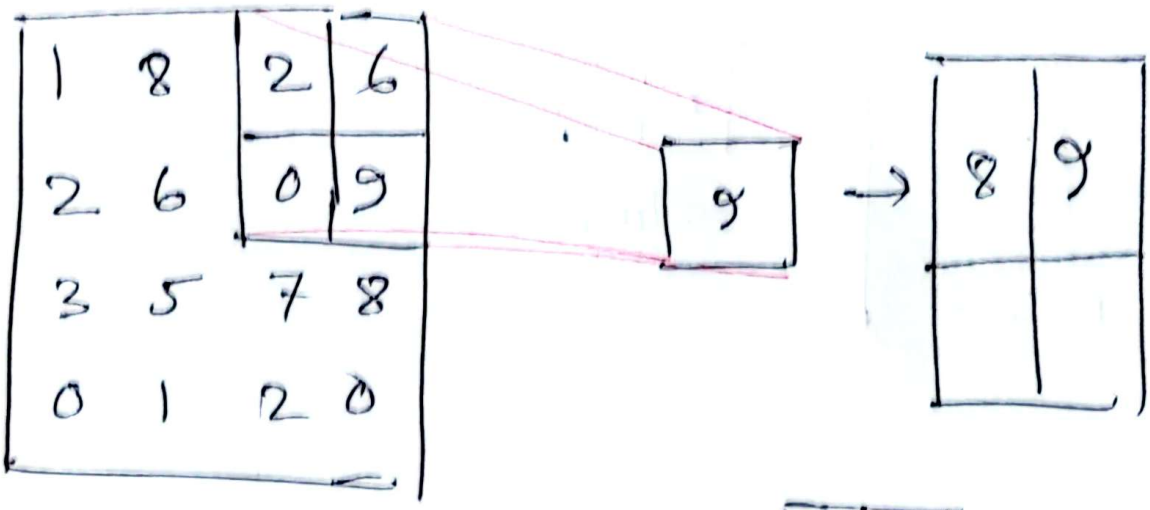
We can do two types of pooling:

Max pooling:



The 2×2 window extracts the maximum value from the feature matrix. here among 1, 8, 2, 6, → 8 is extracted.

If Stride = 2 → second slide is →



Similarly we will get → Other values

8	9
5	8

Average pooling:

Instead of pooling the max value, it take the average of all values inside the window.

Normalization:

Normalization scales the input images value into a specific range. Please read the slide to understand the types of Normalization.