Assignment 1 Report

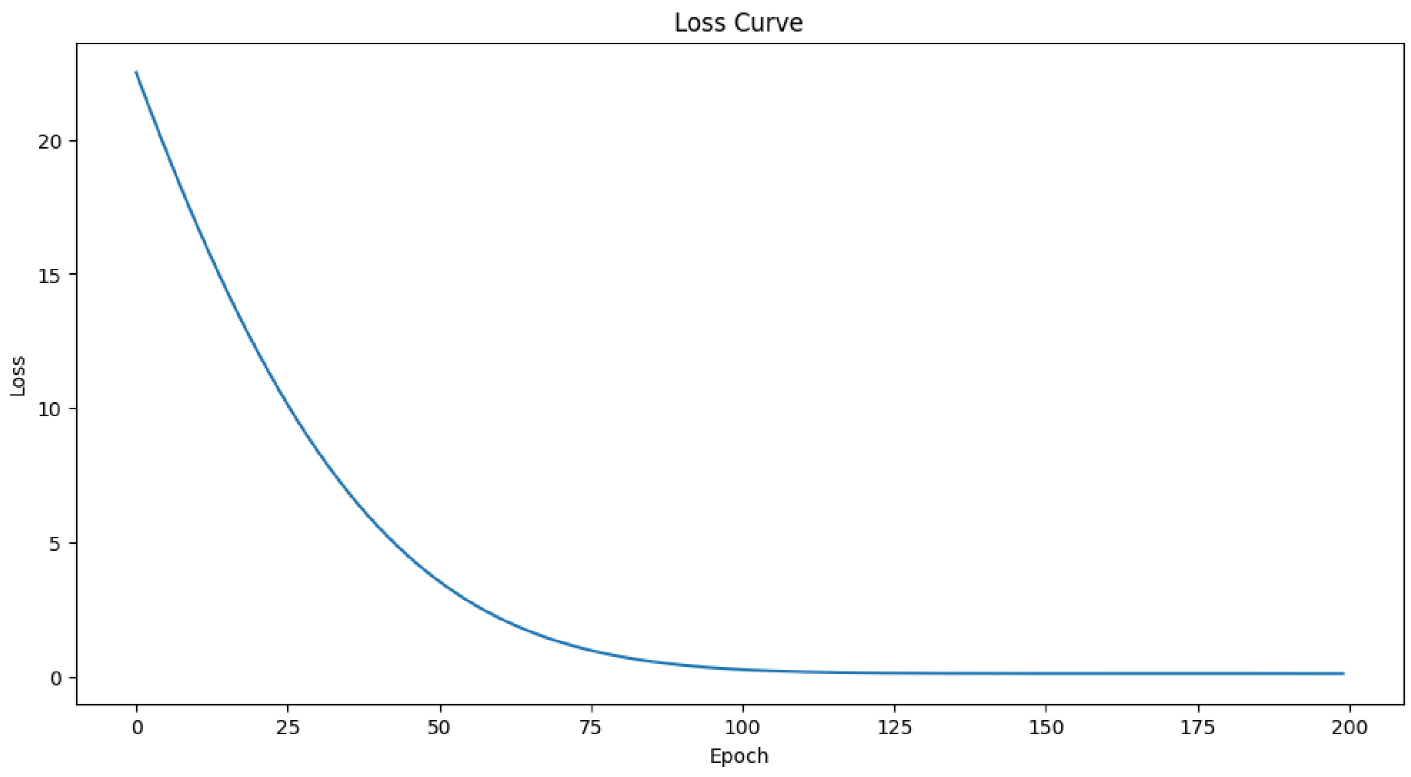
DSAI (AI60006)

Name - Debanjan Saha

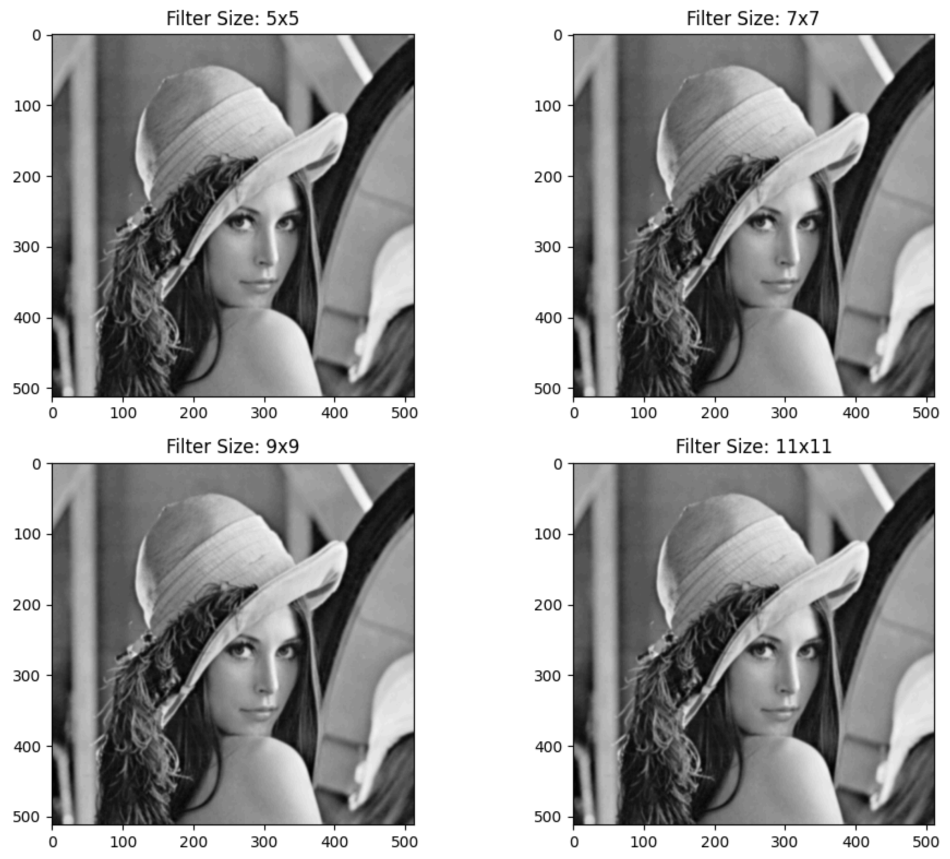
Roll - 19CS30014

Click here to access notebook

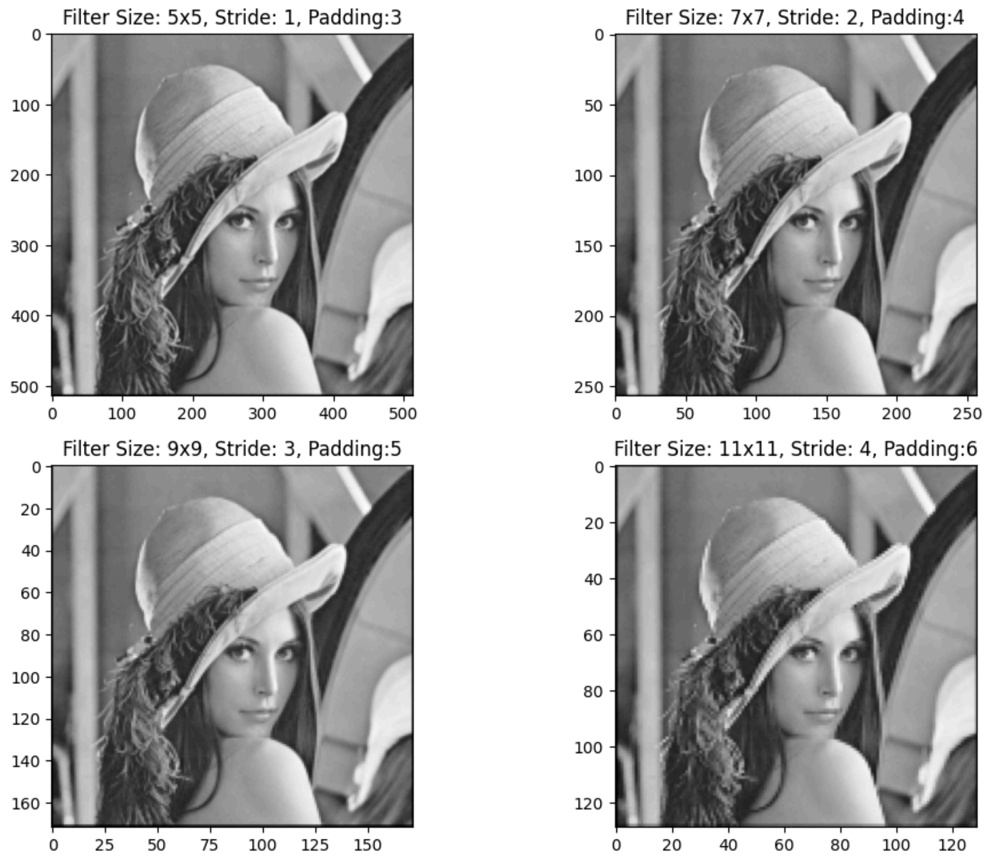
>> Question 2: Plot of Loss Curve



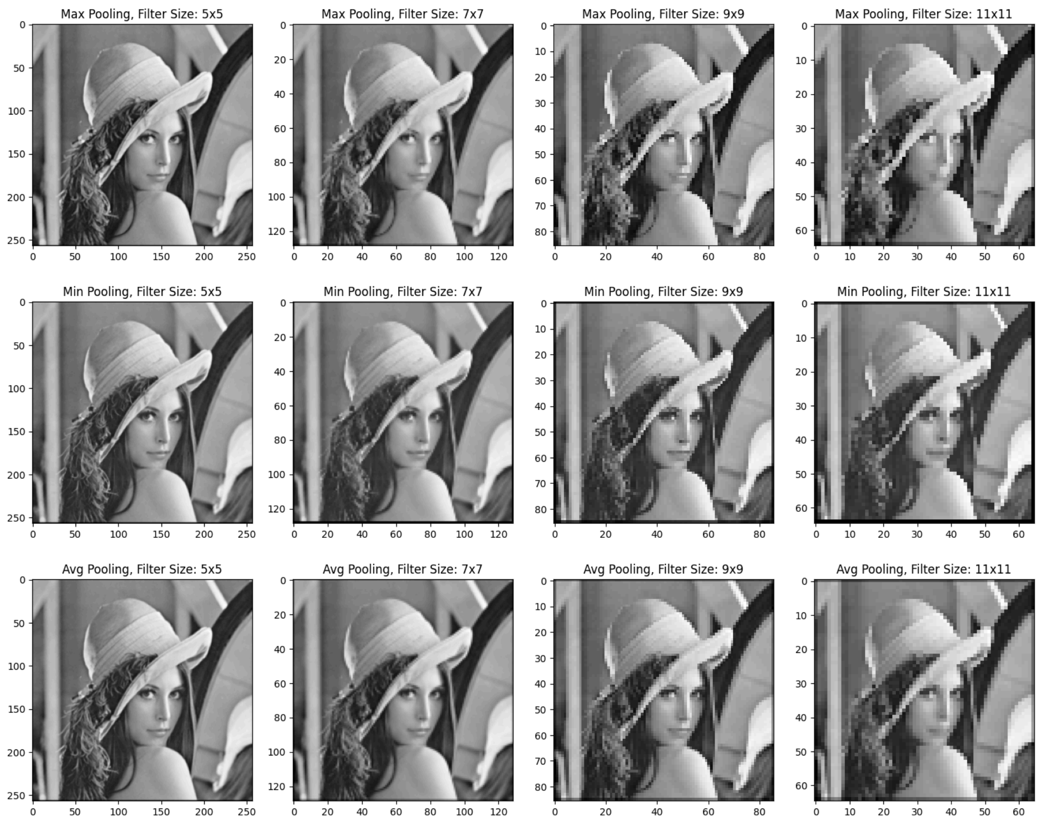
>> Question 3:



>> Question 4:


Filter Size: 5x5, Stride: 1, Padding:3
Filter Size: 7x7, Stride: 2, Padding:4
Filter Size: 9x9, Stride: 3, Padding:5
Filter Size: 11x11, Stride: 4, Padding:6

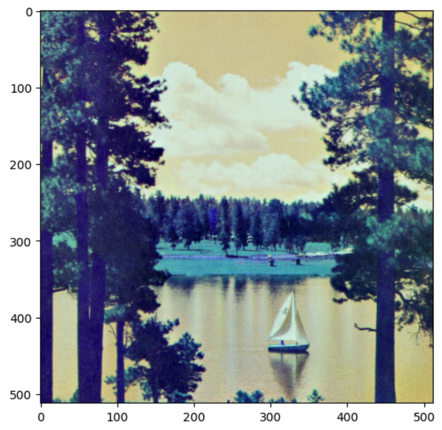
>> Question 5:


Max Pooling, Filter Size: 5x5
Max Pooling, Filter Size: 7x7
Max Pooling, Filter Size: 9x9
Max Pooling, Filter Size: 11x11
Min Pooling, Filter Size: 5x5
Min Pooling, Filter Size: 7x7
Min Pooling, Filter Size: 9x9
Min Pooling, Filter Size: 11x11
Avg Pooling, Filter Size: 5x5
Avg Pooling, Filter Size: 7x7
Avg Pooling, Filter Size: 9x9
Avg Pooling, Filter Size: 11x11

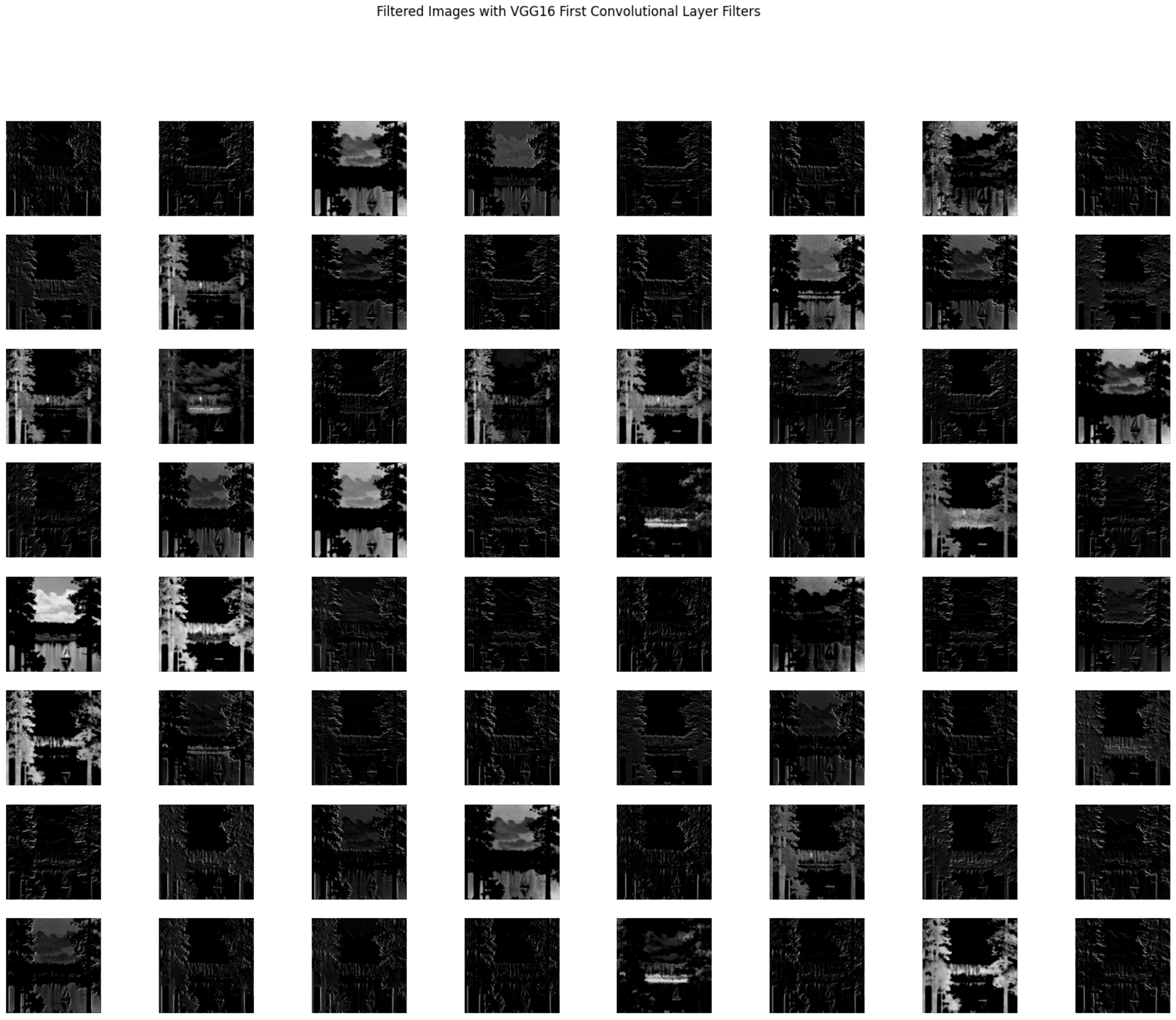
>> Question 6:

I used VGG16 model to apply the filters of the **first block's first conv layer (block1_conv1).** The visualisations of the filters are present in attached notebook.

Original Image:



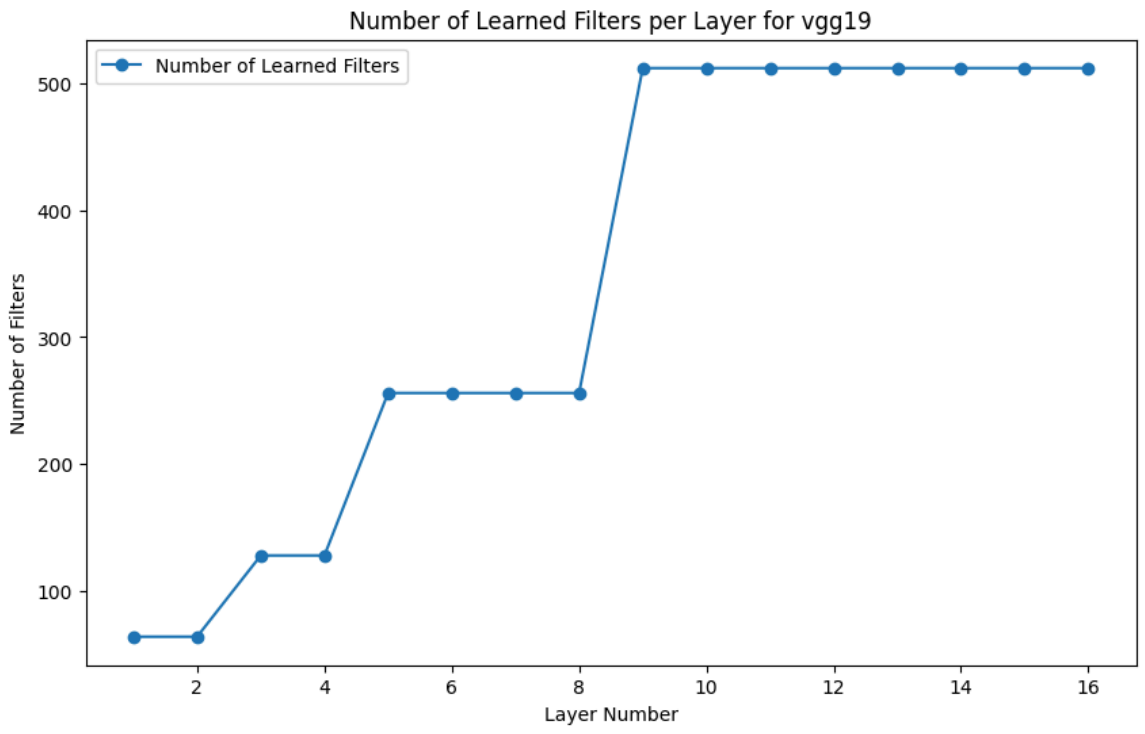
Visualisation of feature maps after applying filters on the input image:

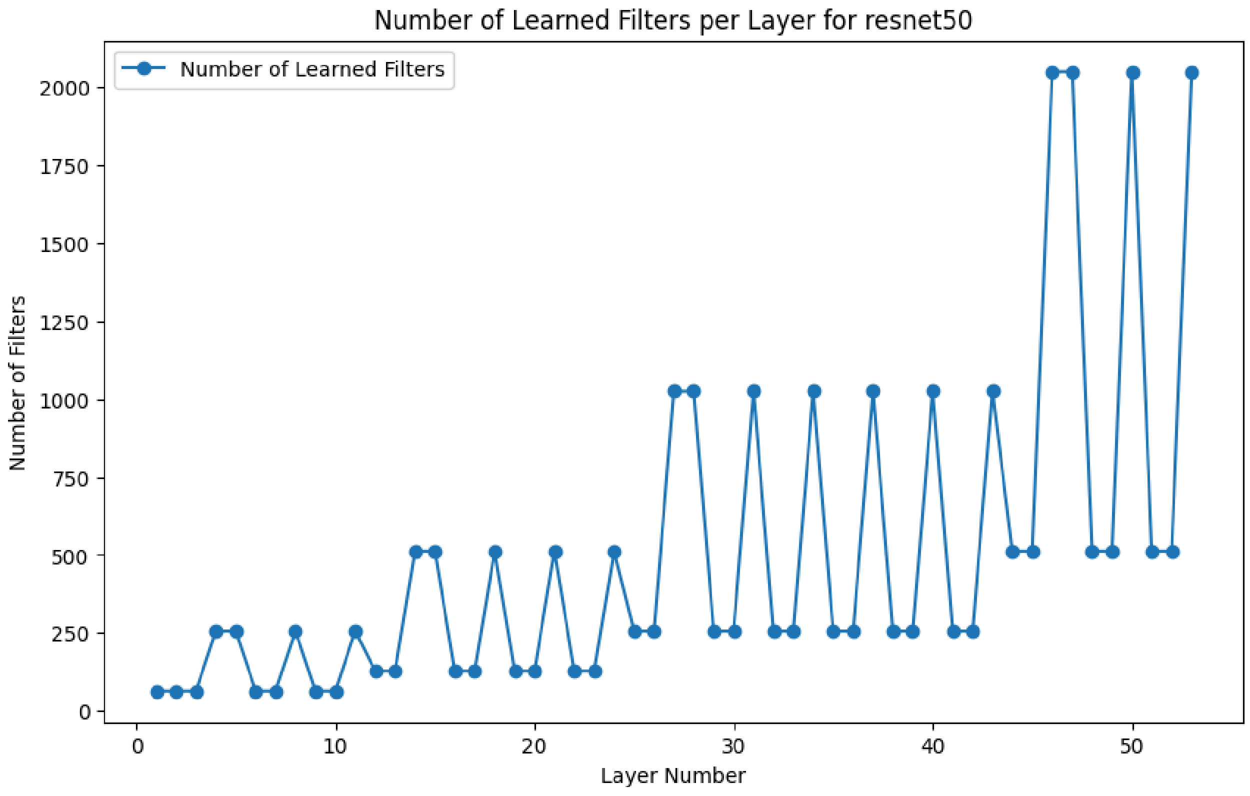

Filtered Images with VGG16 First Convolutional Layer Filters

>> Question 7:

Visualisations of the filters and the feature maps of pre-trained CNN models(vgg19, resnet50) can be seen in the attached notebook.

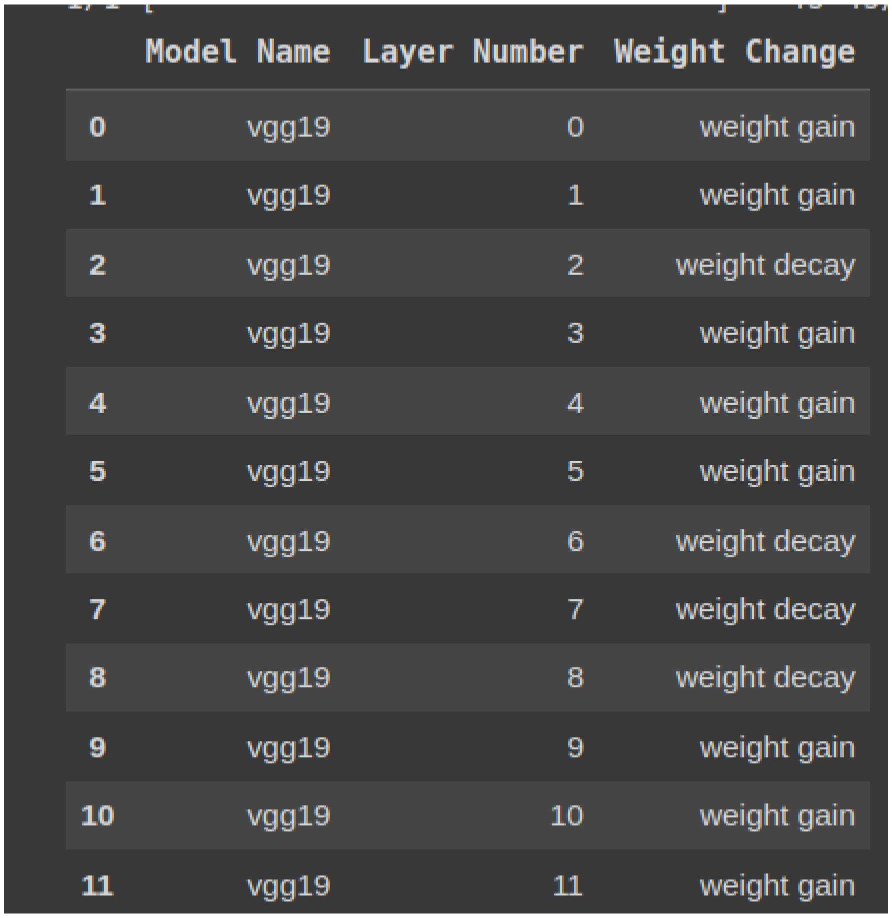
Plot of the count of layerwise filters for 2 models are shown as follows:





>> Question 8

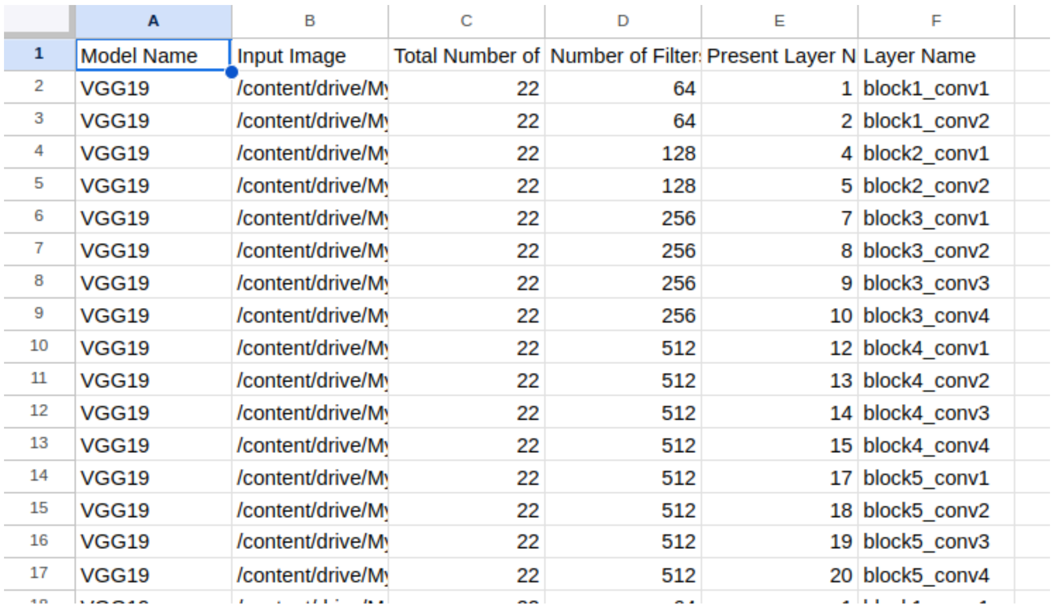
Table has been attached in a file named ‘table_data.csv’. The weight changes are demonstrated in the attached ipynb notebook. For example, for a retrained vgg19 model, the weight changes are shown as follows:



	Model Name	Layer Number	Weight Change
0	vgg19	0	weight gain
1	vgg19	1	weight gain
2	vgg19	2	weight decay
3	vgg19	3	weight gain
4	vgg19	4	weight gain
5	vgg19	5	weight gain
6	vgg19	6	weight decay
7	vgg19	7	weight decay
8	vgg19	8	weight decay
9	vgg19	9	weight gain
10	vgg19	10	weight gain
11	vgg19	11	weight gain

>> Question 8.a

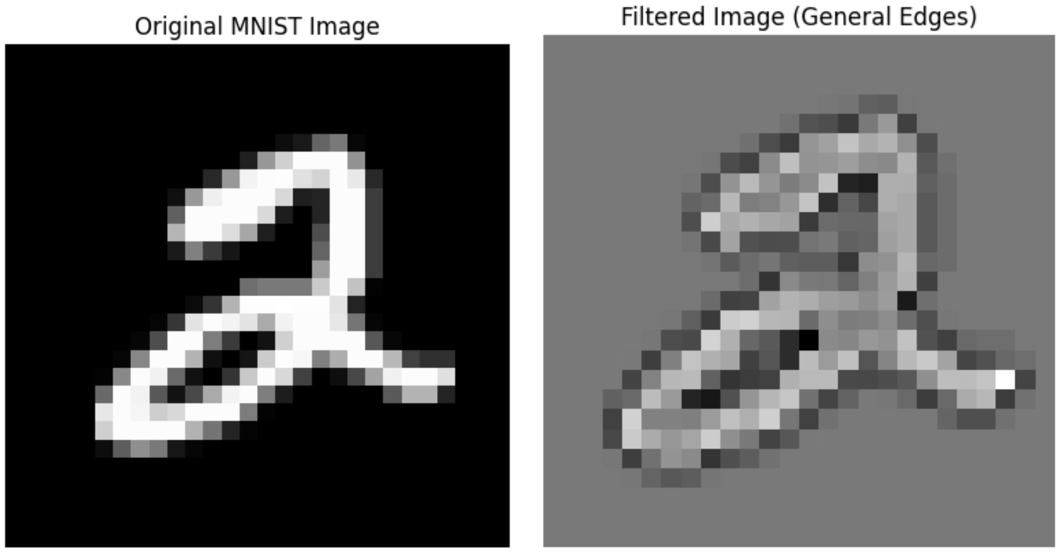
For the **VGG19** model we can see the number of filters for a few layers below as we computed during solving Question 8 first:



	A	B	C	D	E	F
1	Model Name	Input Image	Total Number of	Number of Filter	Present Layer N	Layer Name
2	VGG19	/content/drive/My	22	64	1	block1_conv1
3	VGG19	/content/drive/My	22	64	2	block1_conv2
4	VGG19	/content/drive/My	22	128	4	block2_conv1
5	VGG19	/content/drive/My	22	128	5	block2_conv2
6	VGG19	/content/drive/My	22	256	7	block3_conv1
7	VGG19	/content/drive/My	22	256	8	block3_conv2
8	VGG19	/content/drive/My	22	256	9	block3_conv3
9	VGG19	/content/drive/My	22	256	10	block3_conv4
10	VGG19	/content/drive/My	22	512	12	block4_conv1
11	VGG19	/content/drive/My	22	512	13	block4_conv2
12	VGG19	/content/drive/My	22	512	14	block4_conv3
13	VGG19	/content/drive/My	22	512	15	block4_conv4
14	VGG19	/content/drive/My	22	512	17	block5_conv1
15	VGG19	/content/drive/My	22	512	18	block5_conv2
16	VGG19	/content/drive/My	22	512	19	block5_conv3
17	VGG19	/content/drive/My	22	512	20	block5_conv4

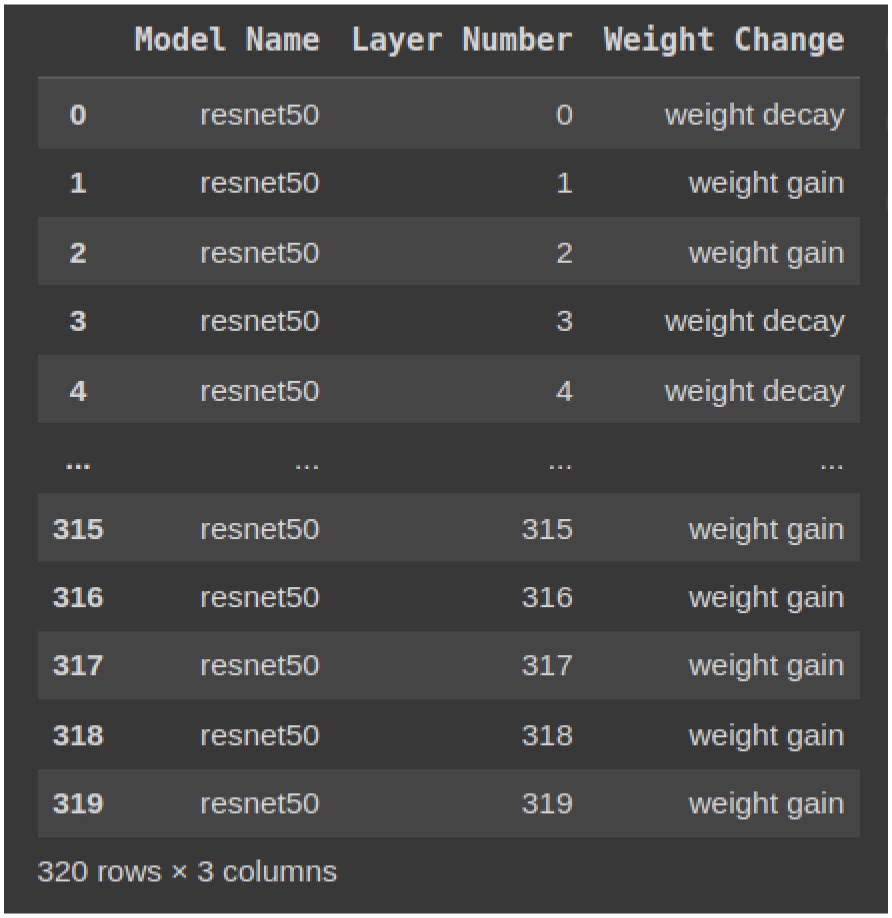
So, for Layer 1-2, 64 filters will get applied, similarly for Layer 3-4, 128 filters will get applied to the image. In total, all conv_2D layers contain **5504** filters. Its calculation can be found in the notebook. **<u>So a total of 5504 filters get applied to an image/input</u>**

<u>>> Question 8.b:</u>
When we apply a General Edge detector filter to an MNIST image, we get the following:



<u>>> Question 8.c and 8.d:</u>
To determine the count of changed filters during batch processing of MNIST images/ single image through a model, we can compare the filter weights before and after processing. This involves computing the absolute difference between the weights of each filter at each layer. Filters with a non-zero difference indicate a change in weight. By counting the number of filters with non-zero differences, we can determine the count of changed filters.

A side-by-side comparison of filters layer-wise involves visually inspecting the filter weights before and after processing. This can be done by plotting the filters as images or displaying them in a table format. Highlighting the modifications resulting from applied multipliers can be achieved by visually comparing the filters. By examining these modifications, we may gain insights into how the applied multipliers have influenced the learning process at each layer. For example, the changes done when retraining a **ResNet50** model using a single image can be shown as follows:



	Model Name	Layer Number	Weight Change
0	resnet50	0	weight decay
1	resnet50	1	weight gain
2	resnet50	2	weight gain
3	resnet50	3	weight decay
4	resnet50	4	weight decay
...	...	...	...
315	resnet50	315	weight gain
316	resnet50	316	weight gain
317	resnet50	317	weight gain
318	resnet50	318	weight gain
319	resnet50	319	weight gain

320 rows × 3 columns

>> Question 8.e:

Tensor Processing Units (TPUs) are accelerators for machine learning tasks like image classification, optimised for matrix operations and parallelism. TPUs leverage massive parallelism and efficient memory access to accelerate convolution operations in convolutional neural networks (CNNs). The architecture's optimised matrix multiplication capabilities enable faster computation of convolutional filters and feature maps. TPUs' unique memory hierarchy minimises data movement and maximises memory bandwidth, enhancing efficiency during convolution operations. Learning rate and optimization algorithms may need adjustment to account for TPU-specific characteristics, optimising training performance. While filter alterations may not be visually apparent, TPUs offer significantly improved efficiency and speed in applying filters and generating feature maps compared to traditional computing platforms. In this way, TPU architecture influence computation changes and convolution operations at each layer.