Computational Neuroscience
IIT Madras

Assignment 3 : Convolutional Neural Network

Ganga Meghanath
EE15B025

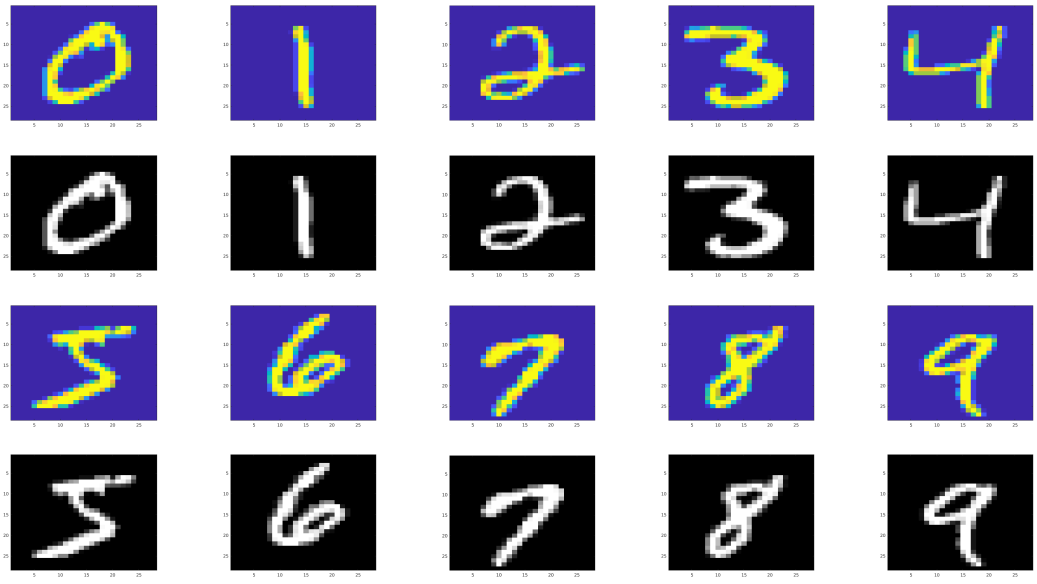
October 29, 2018

1 Aim

Understanding convolutional neural networks

2 Dataset Analysis

The dataset comprises of handwritten digits ranging from 0 to 9 as shown below (color and grayscale):



The dataset has been visualised using :

figure(1);imagesc(images(:,:,i));

figure(2);imagesc(images(:,:,i));colormap(gray);

where i stands for the i^{th} datapoint in the training set.

3 Question 1

3.1 Performance

The performance after training using the given configuration is :

No. of Epochs	No. of Filters	Filter size	No. of training data
3	20	9	10000

Training time	No. of test images	Accuracy
141.47s	10000	93.14%

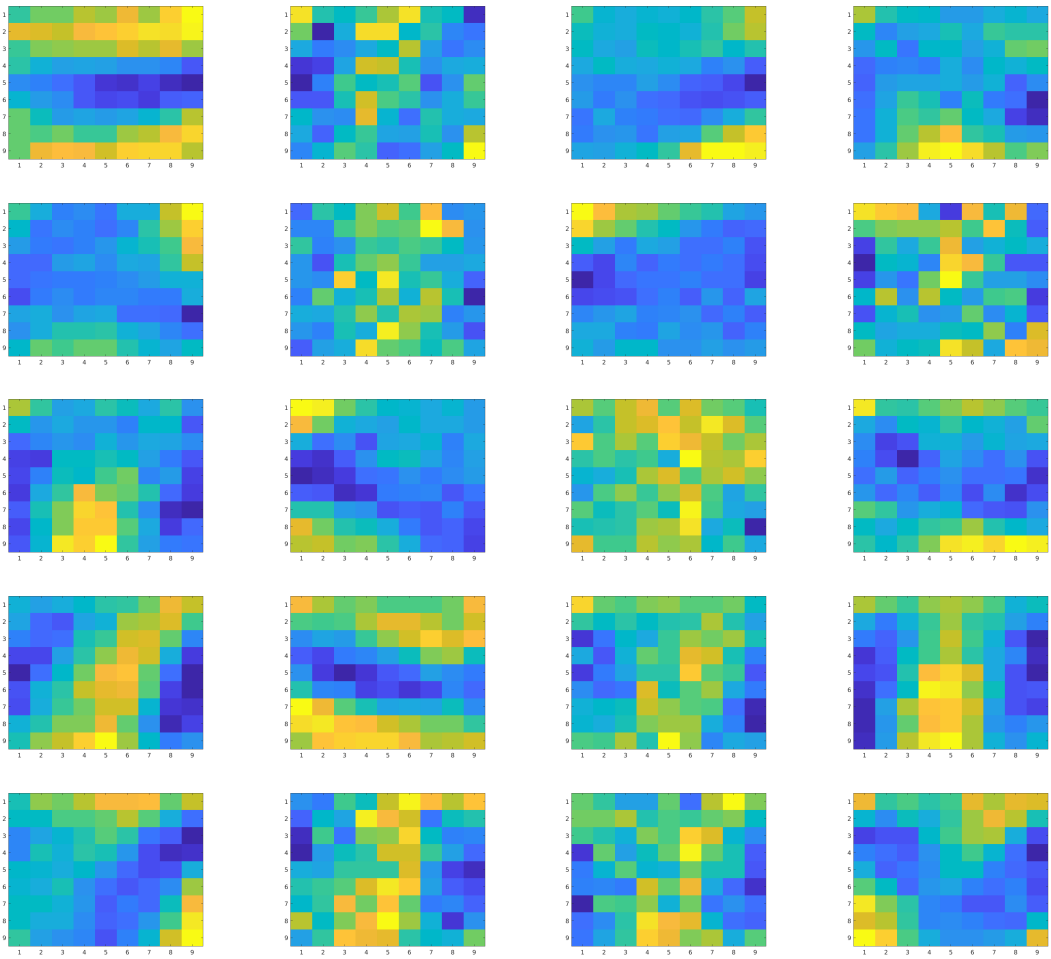
Note : The CNN_Network has been saved.

4 Question 2

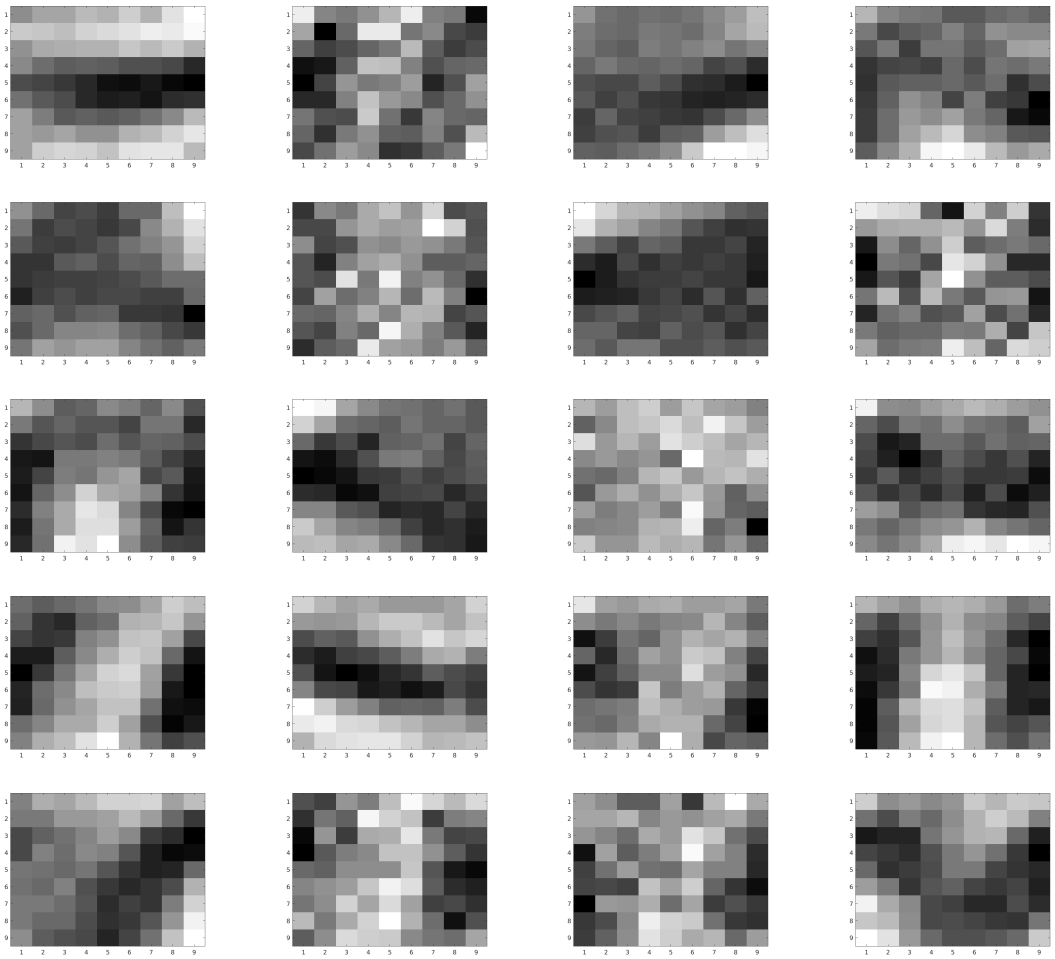
The visualisation of the weights for :

No. of Epochs	No. of Filters	Filter size	No. of training data
3	20	9	10000

has been portrayed below. Note that the edges are more easily identifyable in the grayscaled version of the filter plots.



The corresponding gray scale visualisation is given below :

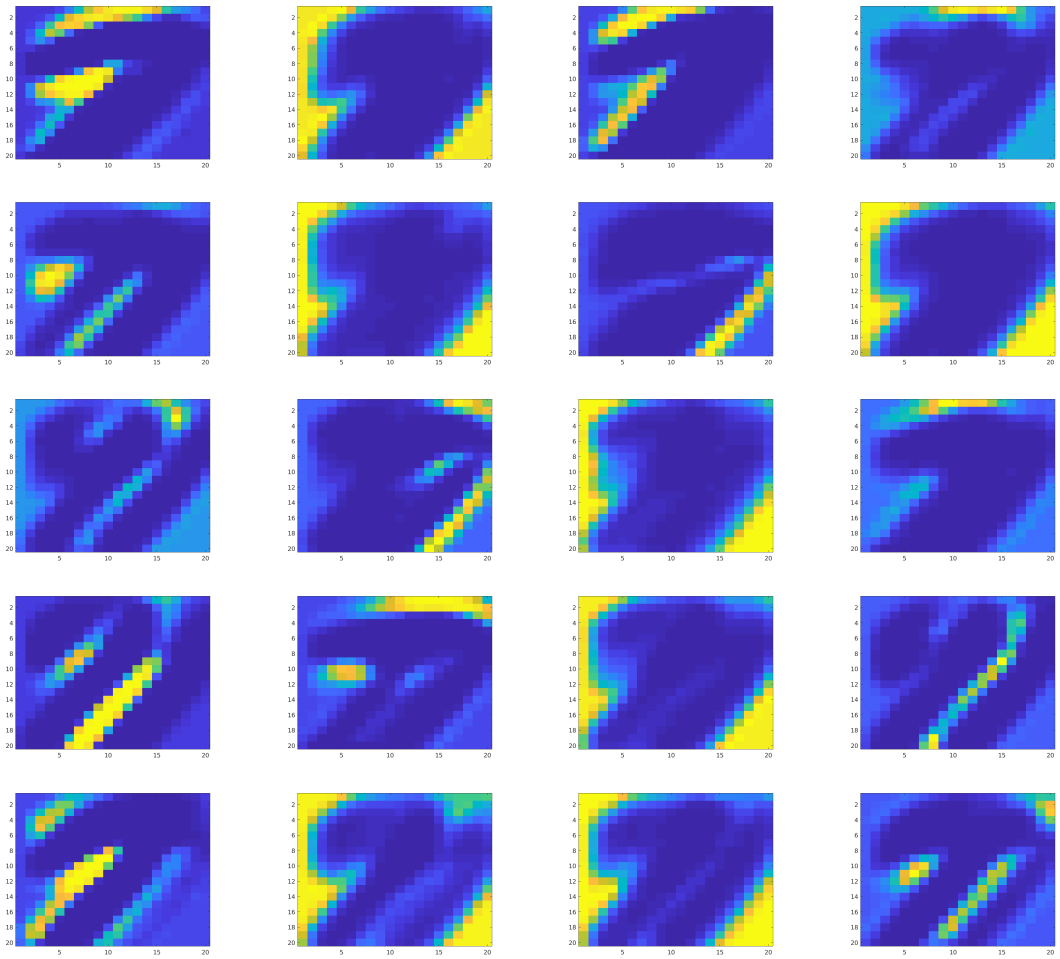


NOTE : The python file for loading ***CNN_ Network*** and plotting the filters has been named ***"visualise.m"***

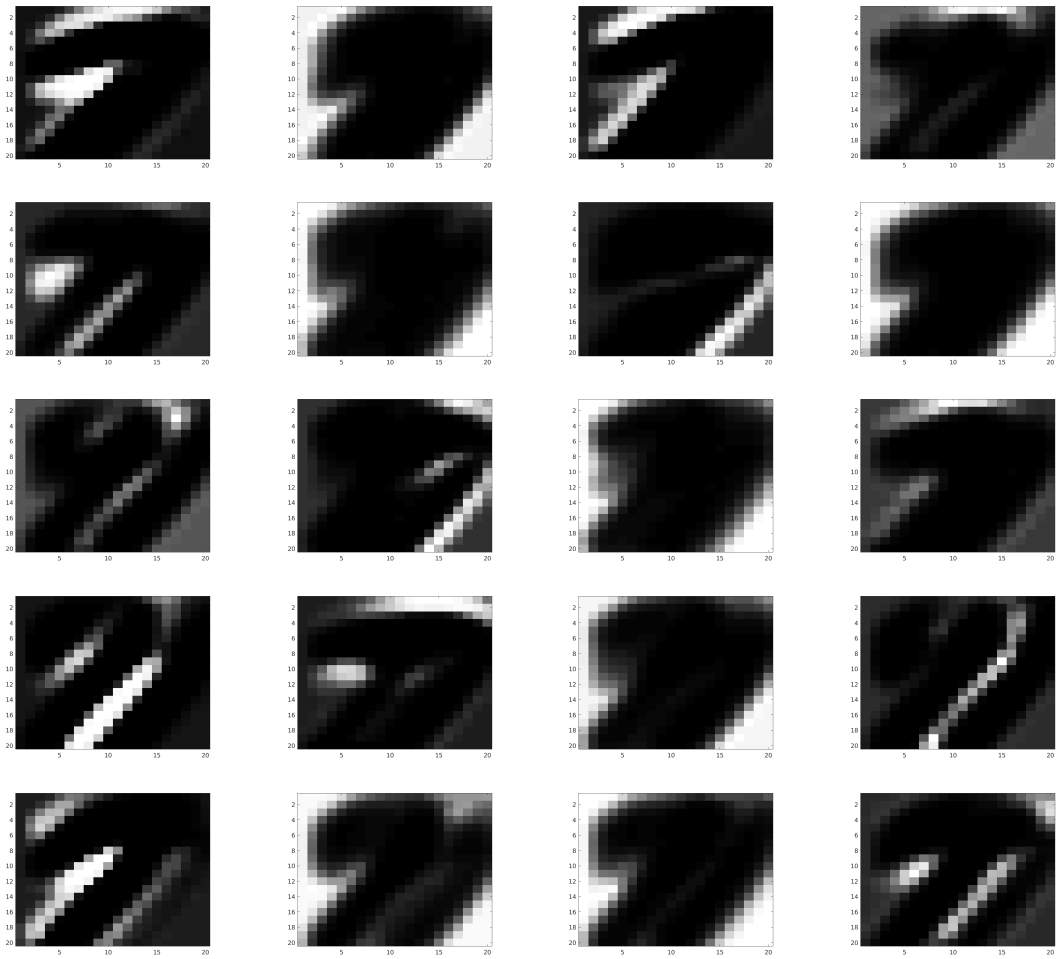
If we compare the arrangement to a matrix with 5 rows and 4 columns indexed from [1,1] to [5,4], the we can see that :

- Filter [1,1] responds to a horizontal lines
- Filter [4,4] responds to vertical lines
- Filter [1,3], [4,1] and [5,1] respond to lines that are slanting right
- Filter [3,2] and [4,2] responds to lines slanting left
- Filter [2,1], [3,4] and [5,4] may respond to curvy edges

The activations are given by :



The corresponding gray scale visualisation is given below :



NOTE : The python file for loading ***CNN_ Network*** and plotting the activations of the filters has been named ***"visualise.m"***

If we compare the arrangement to a matrix with 5 rows and 4 columns indexed from [1,1] to [5,4], the we can see that activations [2,1], [2,3], [4,1], [4,4], [5,1] and [5,4] mainly seems to respond to lines.

Notice that some of the activations look like curvy strokes, Eg : [4,2]

Some of them seems to respond to corner like strokes, Eg : [3,2], [1,1]

Mainly, it can be seen that some of these activations seems to be very similar which could mean that the filters might get activated for similar inputs and hence could be redundant.

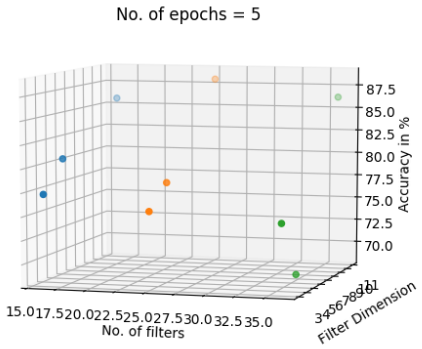
5 Question 3

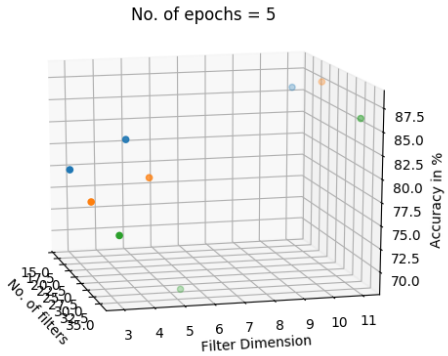
While doing Hyperparameter tuning using 3 different values each for *No. of Epochs*, *No. of Filters* and *Filter Dimension* by training using initial 1000 training images, we obtain the following results :

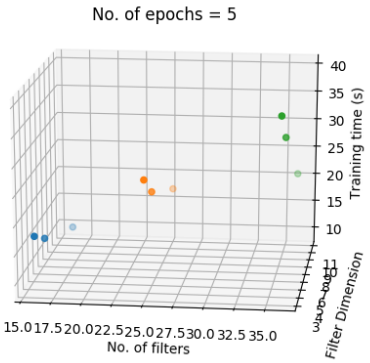
No. of Epochs	**No. of Filters**	**Filter Dimension**	**Accuracy**	**Training time**
5	16	3	77.29	17.95s
5	16	5	80.45	15.02s
5	16	11	85.63	9.41s
5	25	3	75.85	28.09s
5	25	5	78.22	23.73
5	25	11	87.97	17.19
5	36	3	75.06	39.39
5	36	5	68.95	33.63
5	36	11	86.22	20.53
10	16	3	74.57	17.6s
10	16	5	74.9	15.28s
10	16	11	88.08	9.35
10	25	3	73.57	27.13s
10	25	5	74.61	23.41s
10	25	11	87.01	14.36s
10	36	3	72.45	38.73s
10	36	5	70.22	33.56s
10	36	11	84.86	20.46s
20	16	3	74.7	17.39s
20	16	5	78.12	15.21s
20	16	11	87.62	9.34s
20	25	3	73.48	27.03s
20	25	5	73.14	23.57s
20	25	11	86.97	14.36s
20	36	3	72.7	38.78s
20	36	5	69.25	33.40s
20	36	11	83.92	20.5s

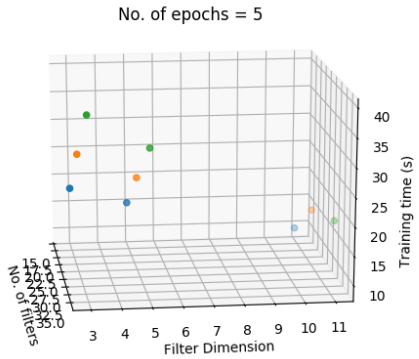
To get a better analysis : Red, Green and Blue dots correspond to Filter sizes 16, 25 and 36 respectively.

5.1 For No. of epochs = 5

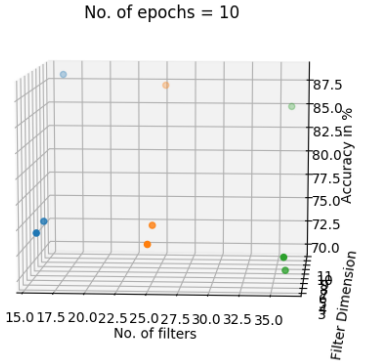


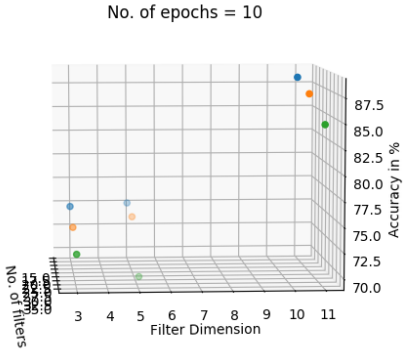


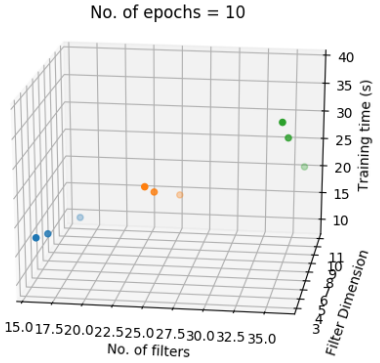


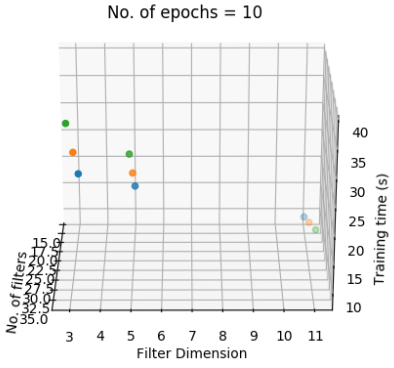


5.2 For No. of epochs = 10

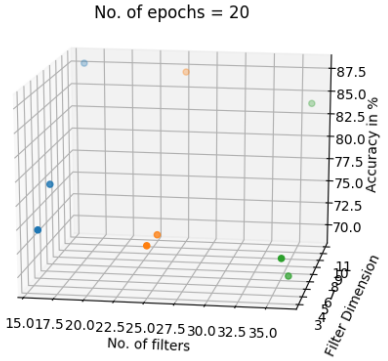


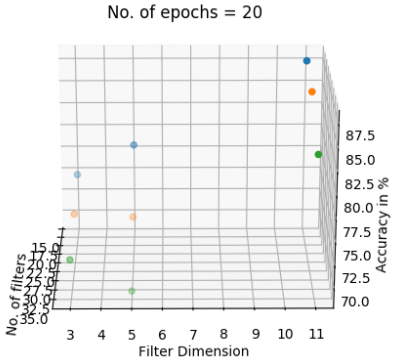


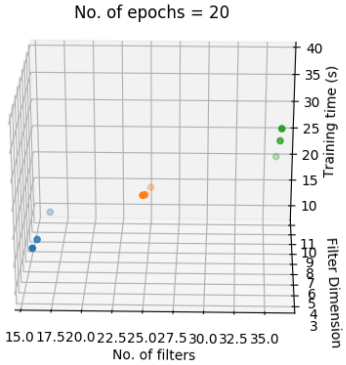


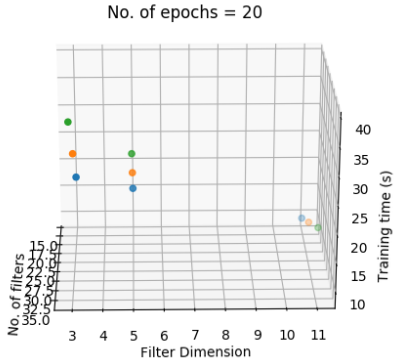


5.3 For No. of epochs = 20









5.4 Observations

As can be seen from the above plots :

- Accuracy generally decreases as the number of filters increases, given a constant no. of training epochs. This could probably be due to insufficient training of larger number of weights compared to when there are smaller no. of weights.
- Accuracy generally increases as the dimension of filters increases, given a constant no. of training epochs. This could probably be because more spatial information is captured while using a larger filter in place of a smaller filter. If this is was a deeper network, then the larger spatial dependence could have been captured in deeper layers. But since this is a shallow network with just one convolutional layer, larger filter helps capture spatial dependencies between the pixels of the image that could aid in better classification results.
- Training time generally increases as the number of filters increases, given a constant no. of training epochs. This is because the number of weights increases and hence the number of updates also increases and hence the training time increases with the amount of additional computation required (especially in the last fully connected layer).
- Training time generally decreases as the filter size increases, given constant no. of epochs and no. of filters. This is because the no. of neurons in the fully connected layer reduces due to reduction in the output image size after convolution. Since majority of neurons are present in the fully connected layer (Convolutional layer has weight sharing since the same filter is used over the entire image), the effective amount of computation required reduces and hence training time reduces.

Note : The code for plotting has been named **"plot.py"**

Note : The code for generating the results for the hyperparameter tuning has been named ***Traincnn_ loop.m*** and for malual tuning, use *Traincnn_ tuning.m*