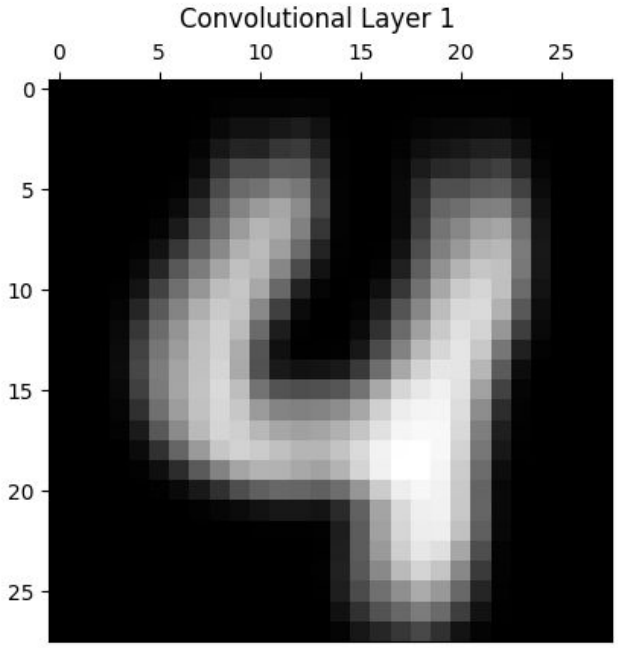
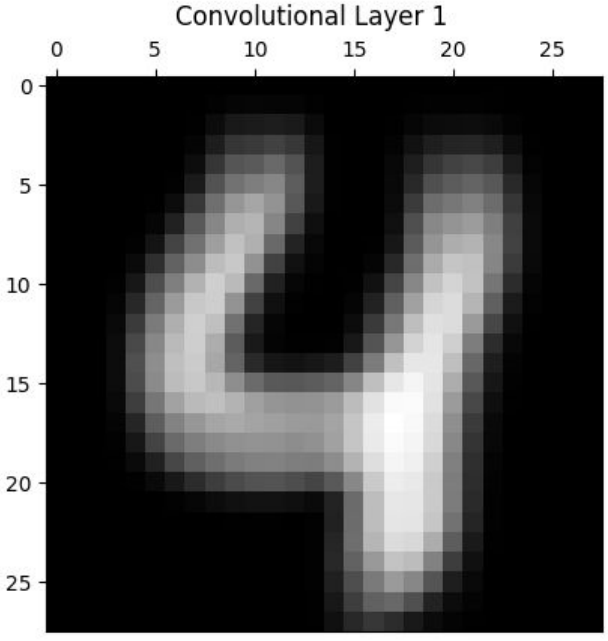
Name : **Akhilesh Soni**

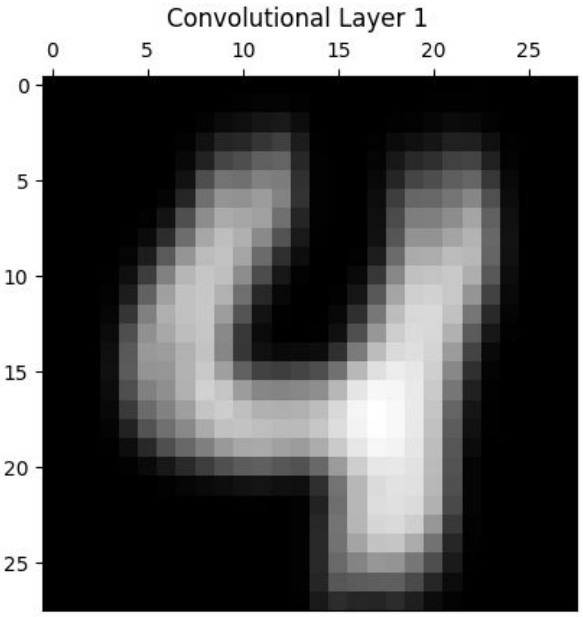
Roll No : **201530001**

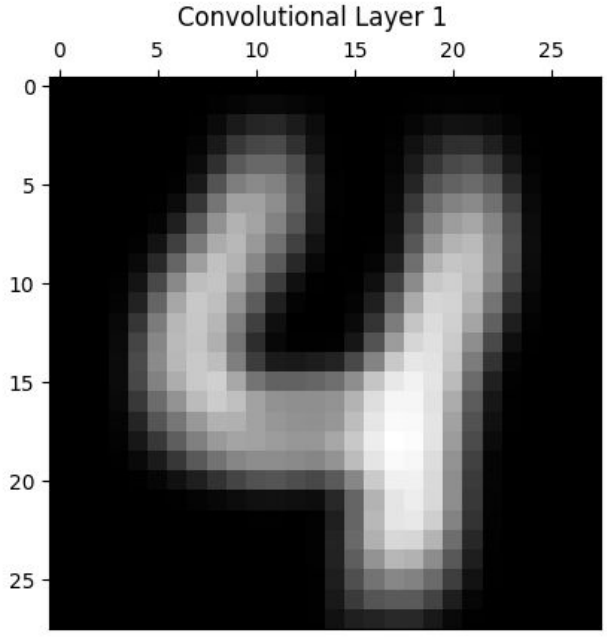
Q1.1)

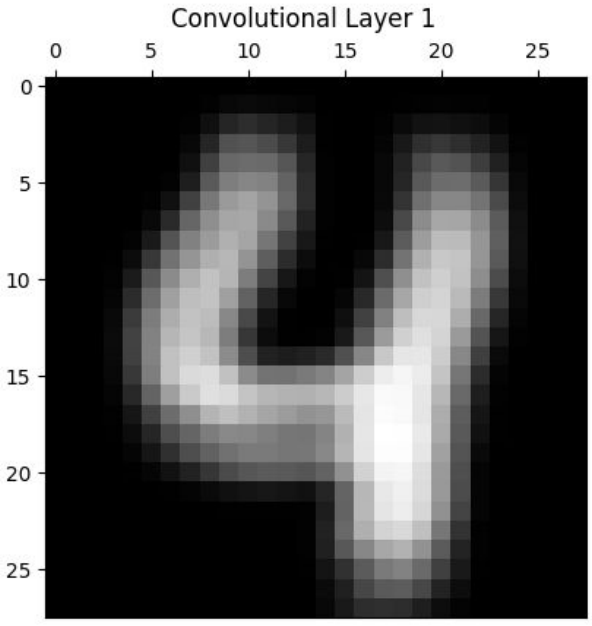
Input Image :

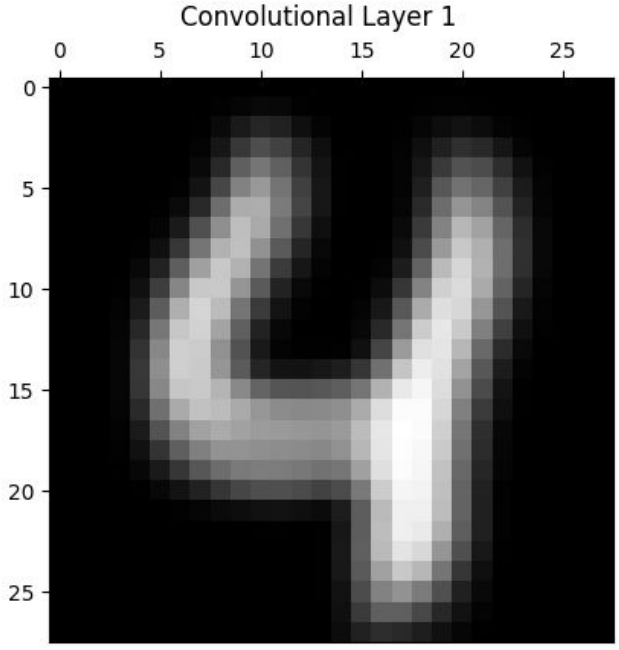

Convolutional Layer 1
0
5
10
15
20
25
0
5
10
15
20
25

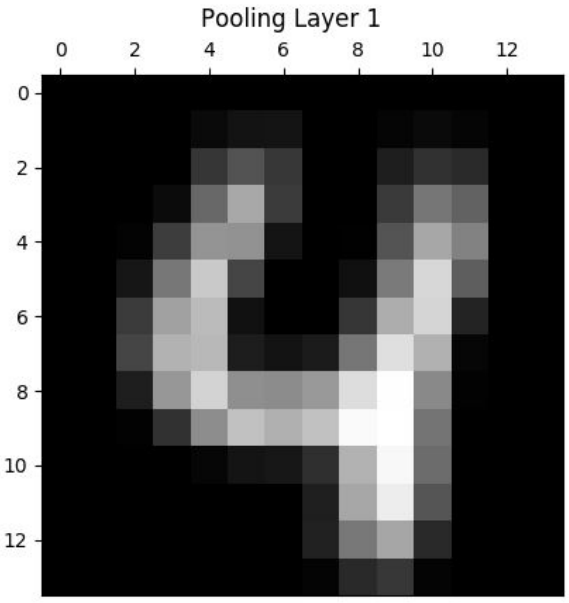

Convolutional Layer 1
0
5
10
15
20
25
0
5
10
15
20
25

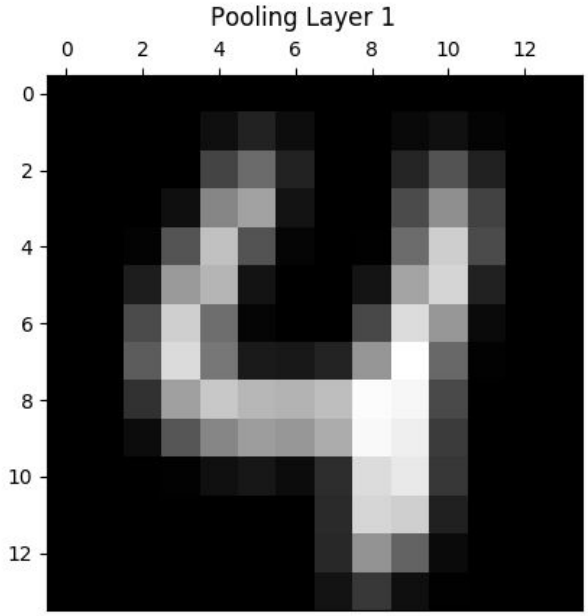

Convolutional Layer 1
0
5
10
15
20
25

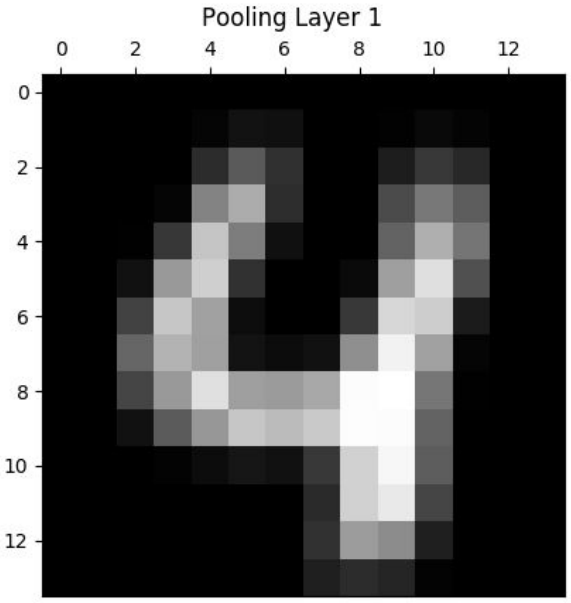

Convolutional Layer 1
0
5
10
15
20
25

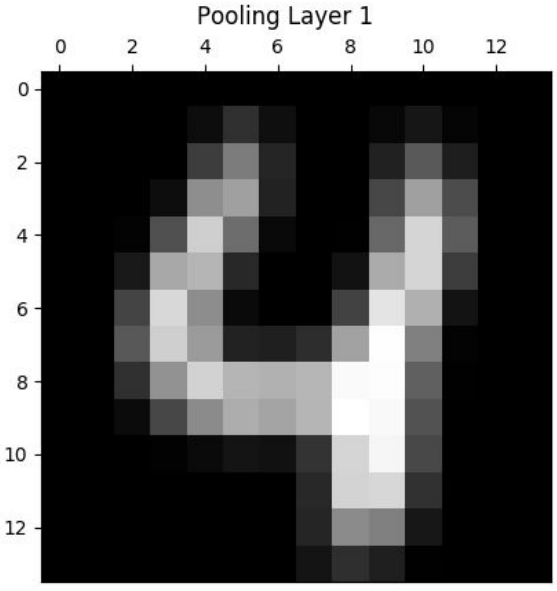

Convolutional Layer 1
0
5
10
15
20
25

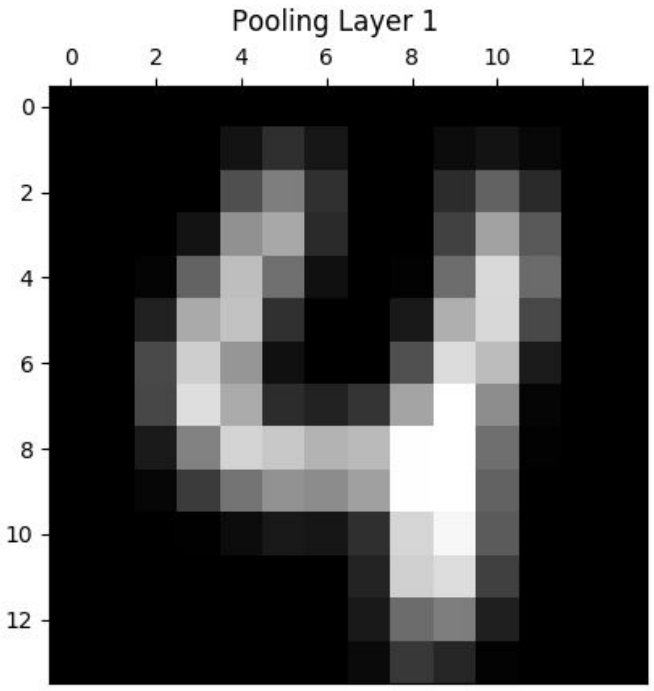

Convolutional Layer 1
0
5
10
15
20
25

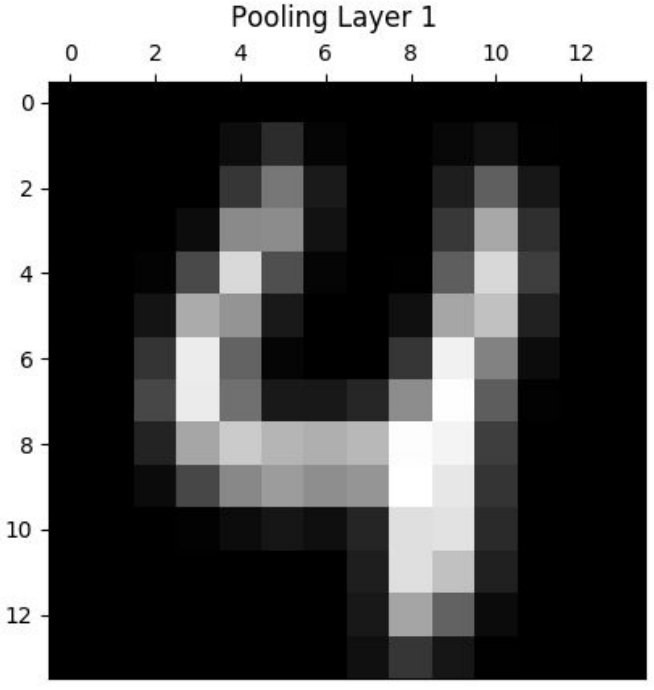

Pooling Layer 1
0
2
4
6
8
10
12
0
2
4
6
8
10
12

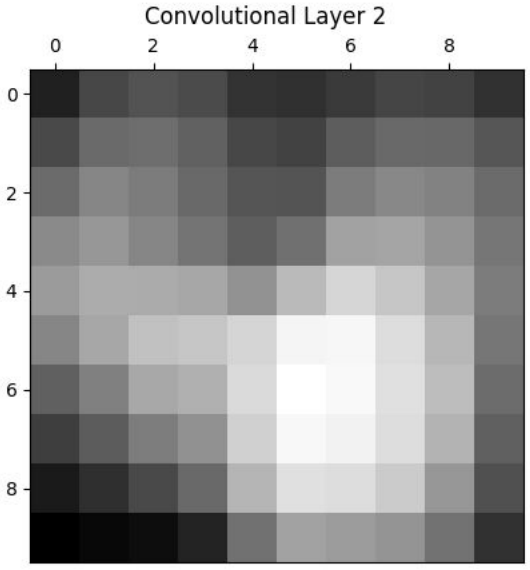

Pooling Layer 1
0
2
4
6
8
10
12
0
2
4
6
8
10
12

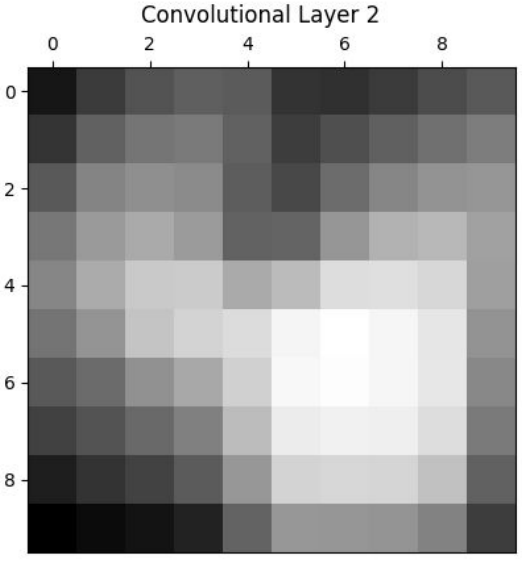

Pooling Layer 1
0
2
4
6
8
10
12

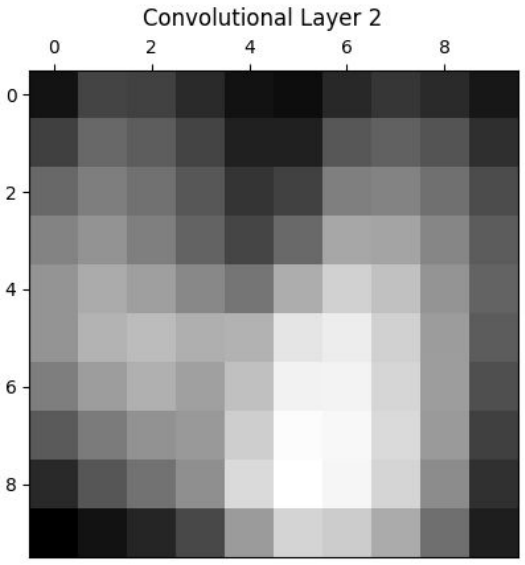

Pooling Layer 1
0
2
4
6
8
10
12

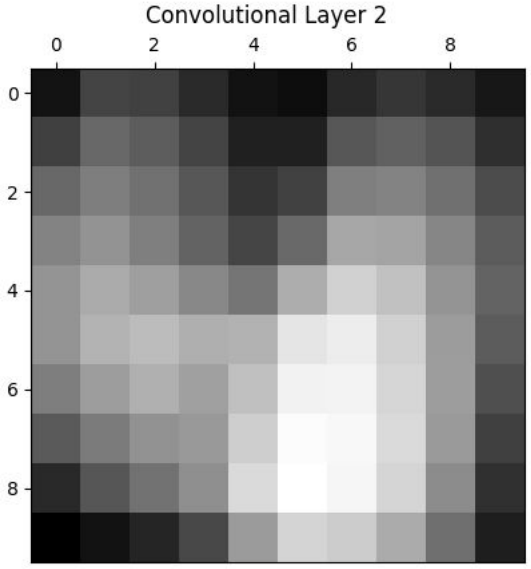

Pooling Layer 1
0
2
4
6
8
10
12

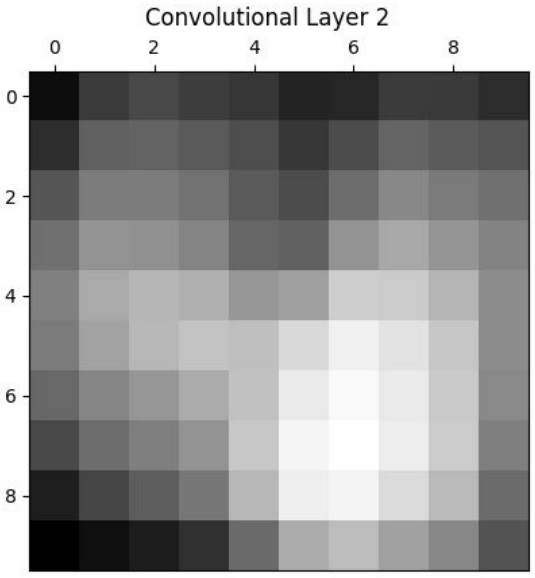

Pooling Layer 1
0
2
4
6
8
10
12

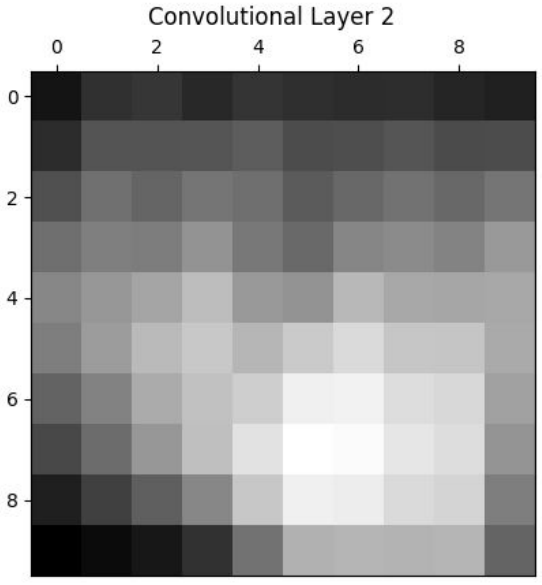

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

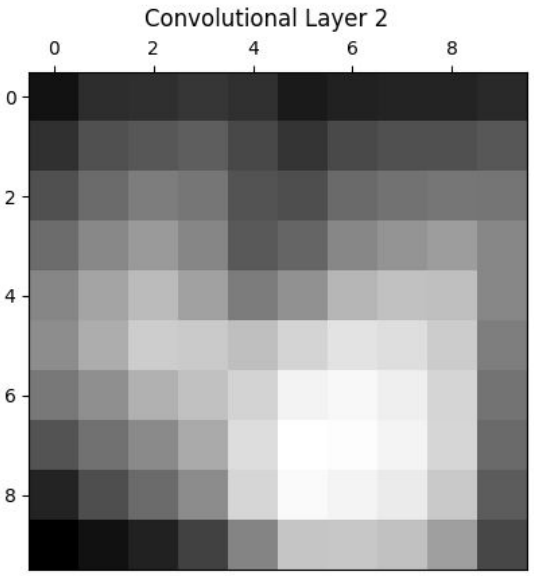

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

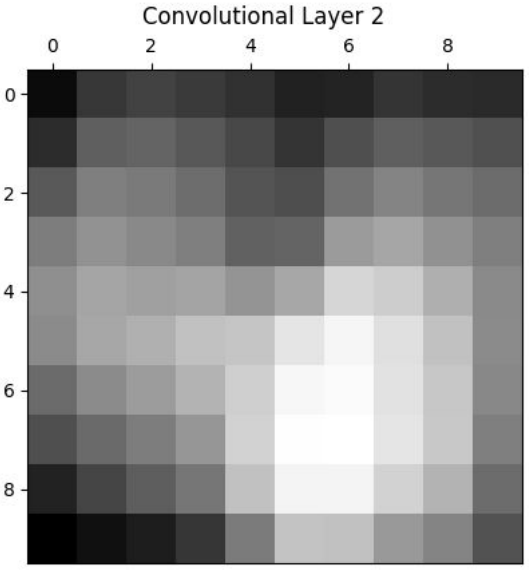

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

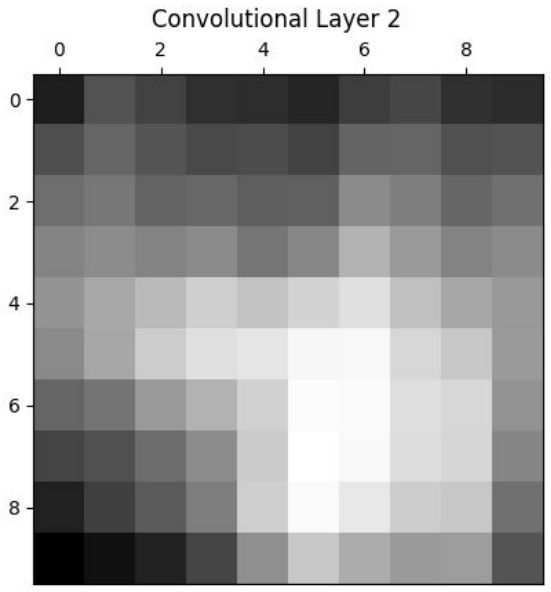

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

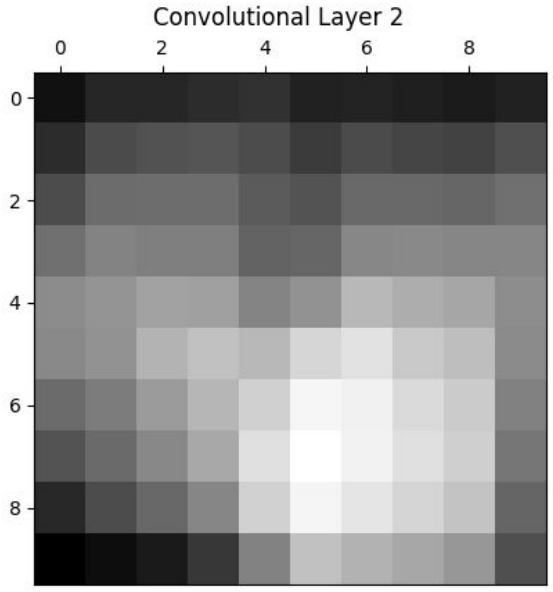

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

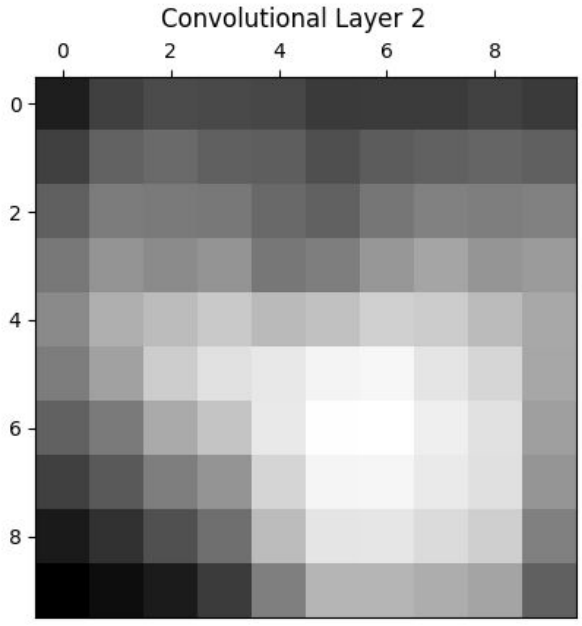

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

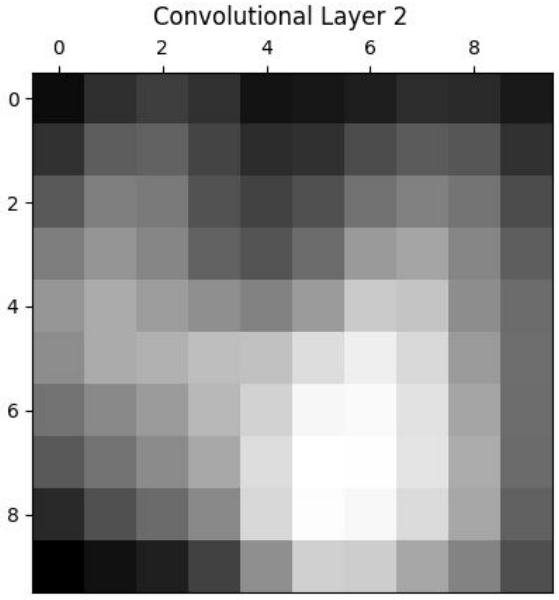

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

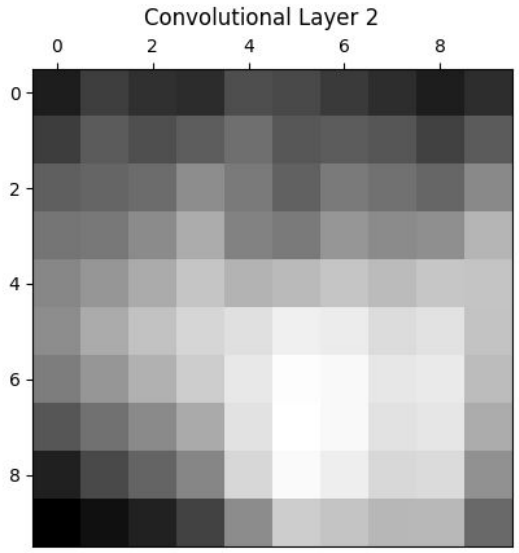

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

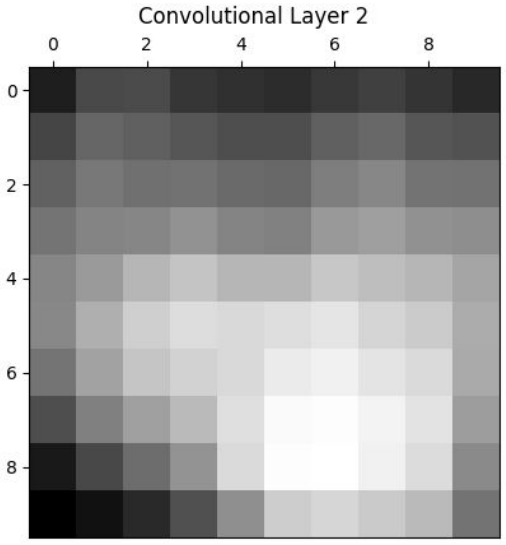

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

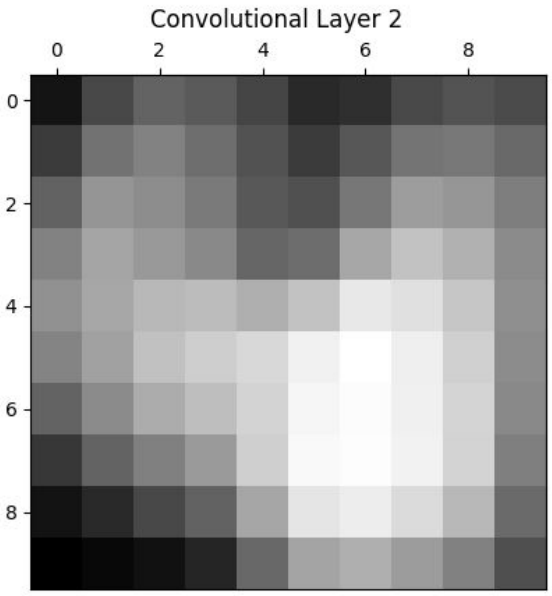

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

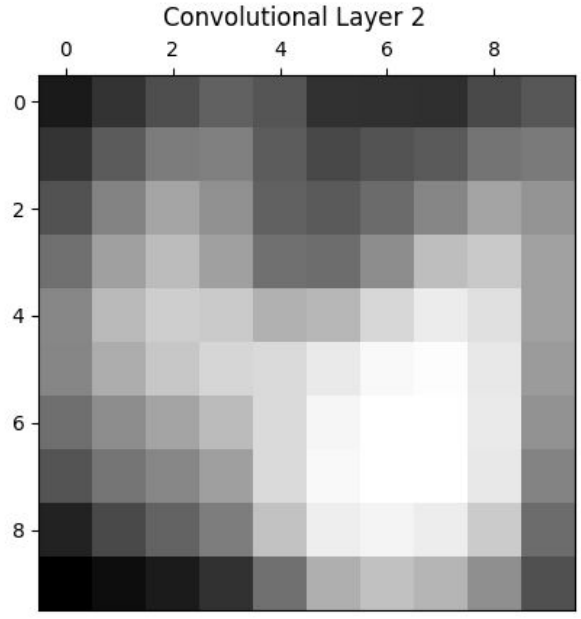

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

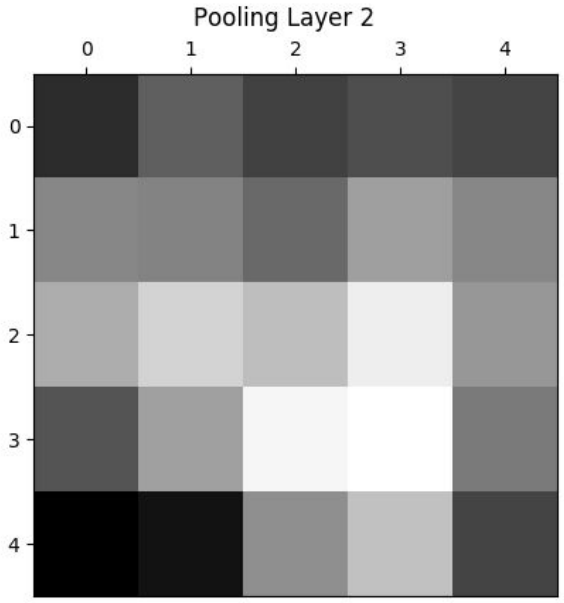

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

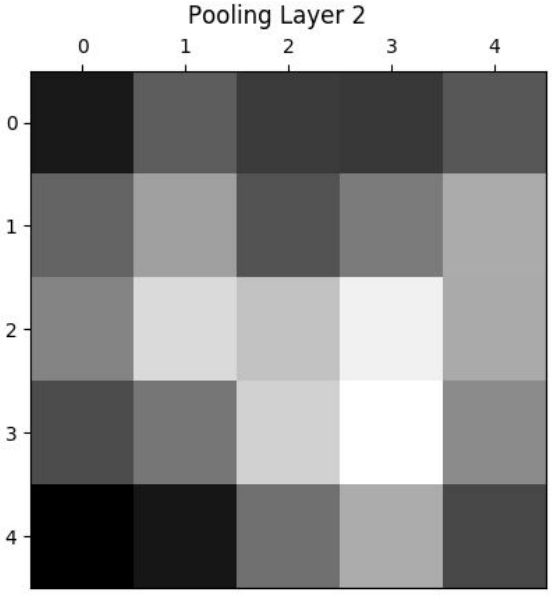

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

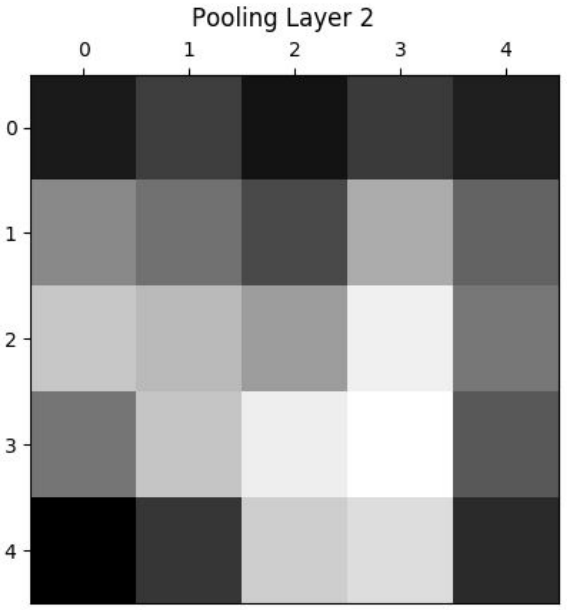

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

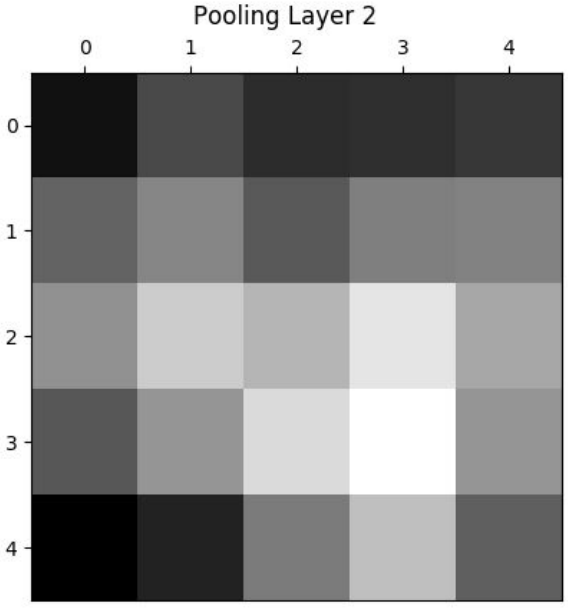

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

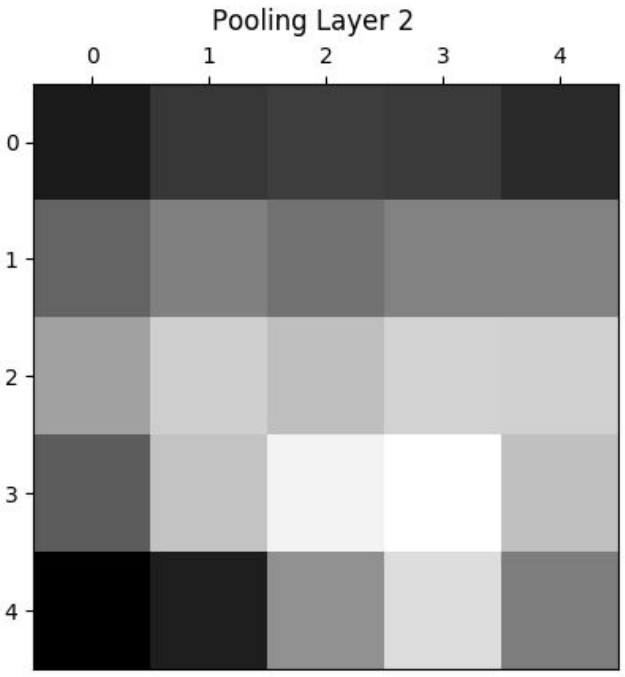

Convolutional Layer 2
0
2
4
6
8
0
2
4
6
8

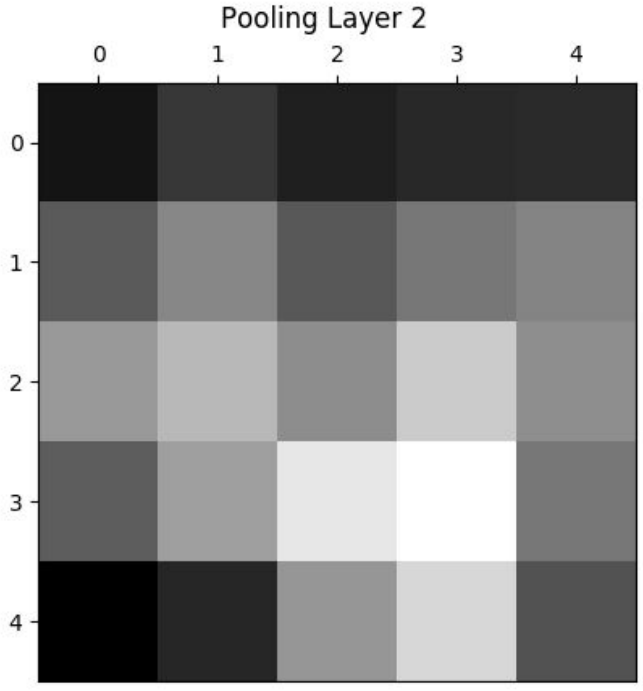

Pooling Layer 2
0
1
2
3
4
0
1
2
3
4

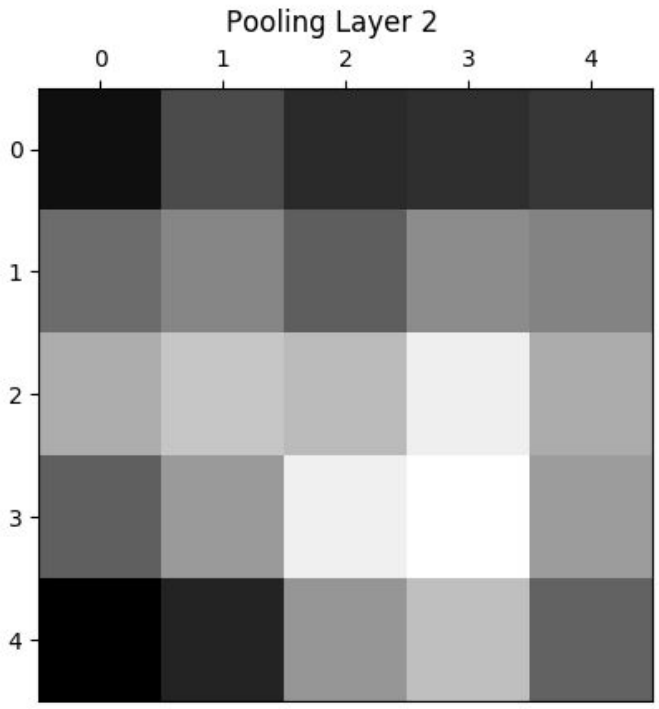

Pooling Layer 2
0
1
2
3
4
0
1
2
3
4

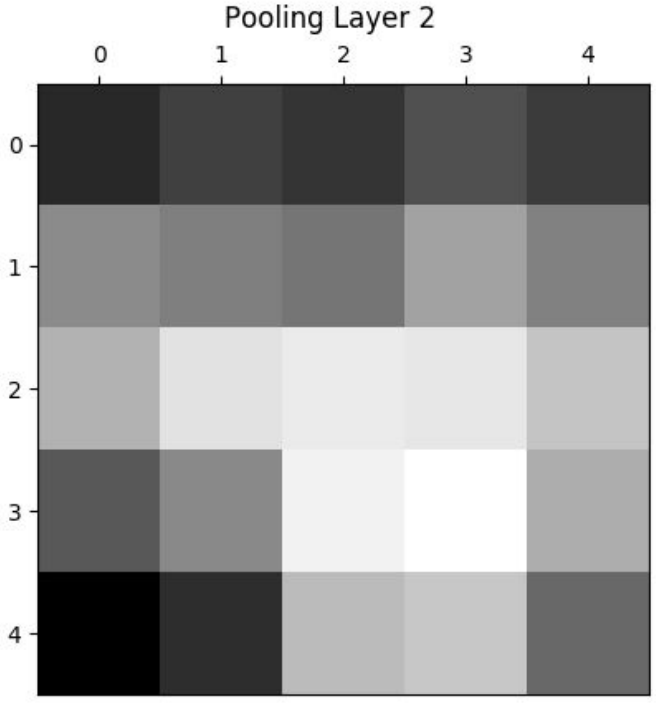

Pooling Layer 2
0
1
2
3
4
0
1
2
3
4

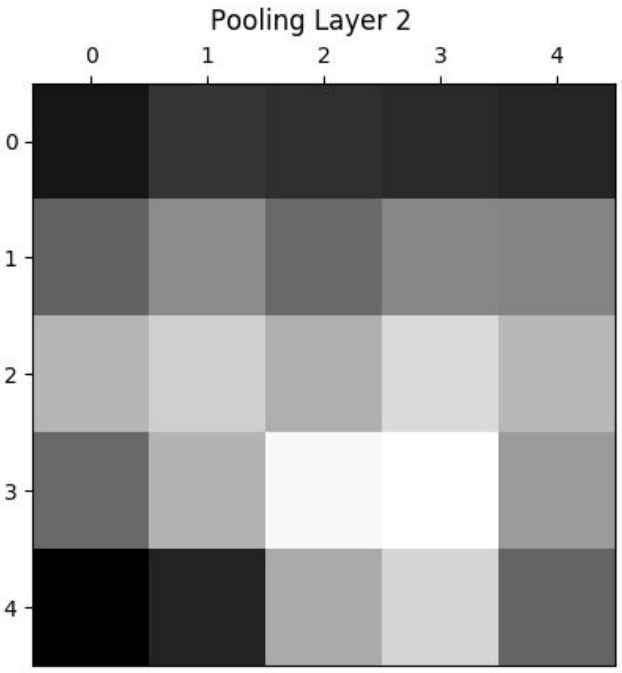

Pooling Layer 2
0
1
2
3
4
0
1
2
3
4

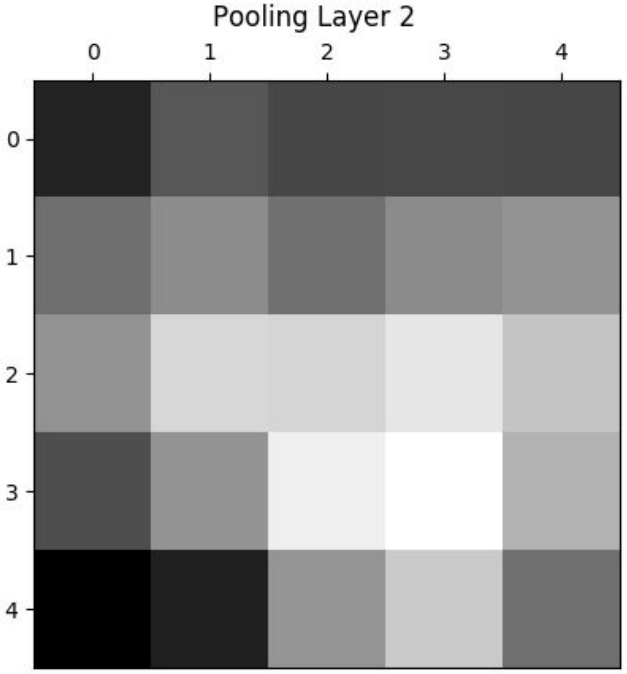

Pooling Layer 2
0
1
2
3
4
0
1
2
3
4

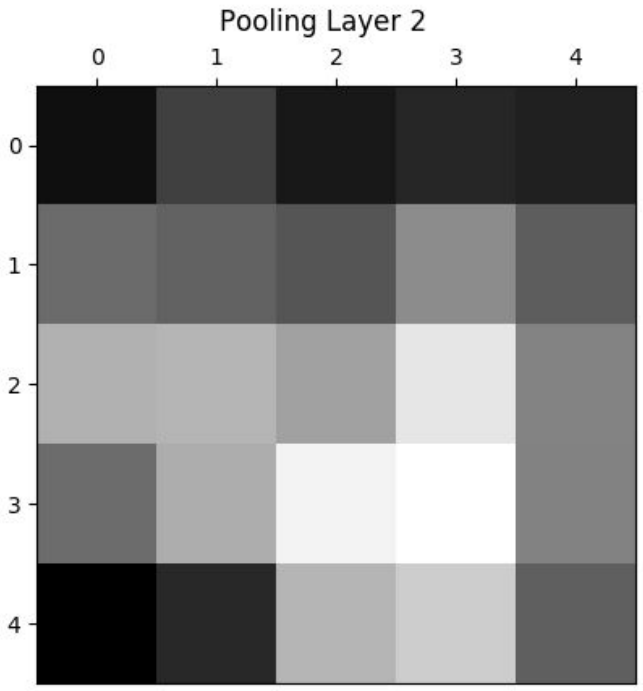

Pooling Layer 2
0
1
2
3
4
0
1
2
3
4

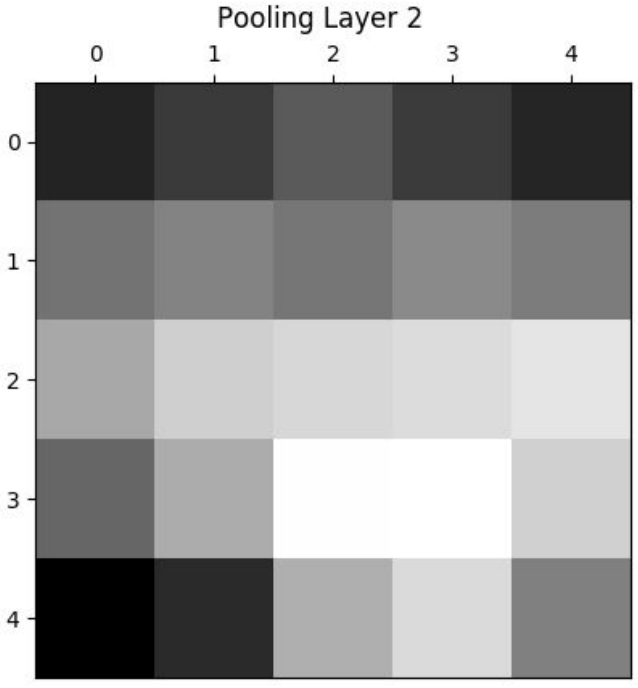

Pooling Layer 2
0
1
2
3
4
0
1
2
3
4

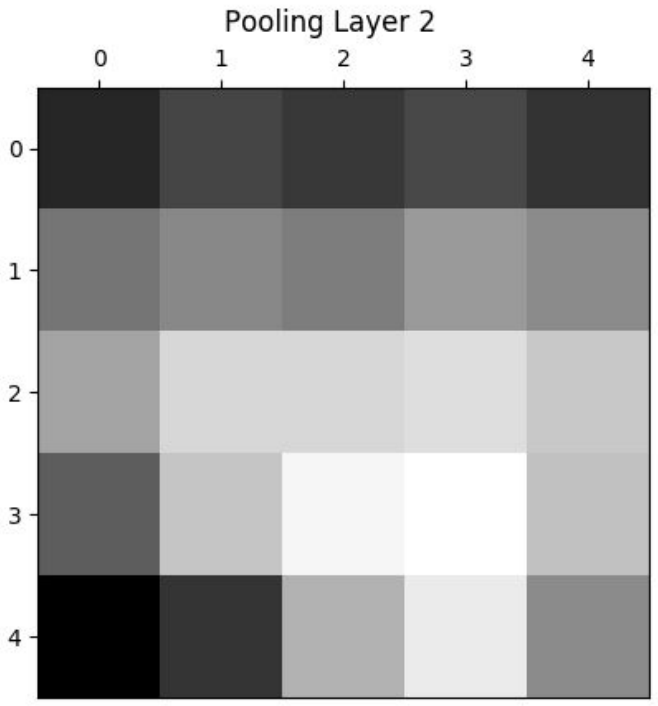

Pooling Layer 2
0
1
2
3
4
0
1
2
3
4

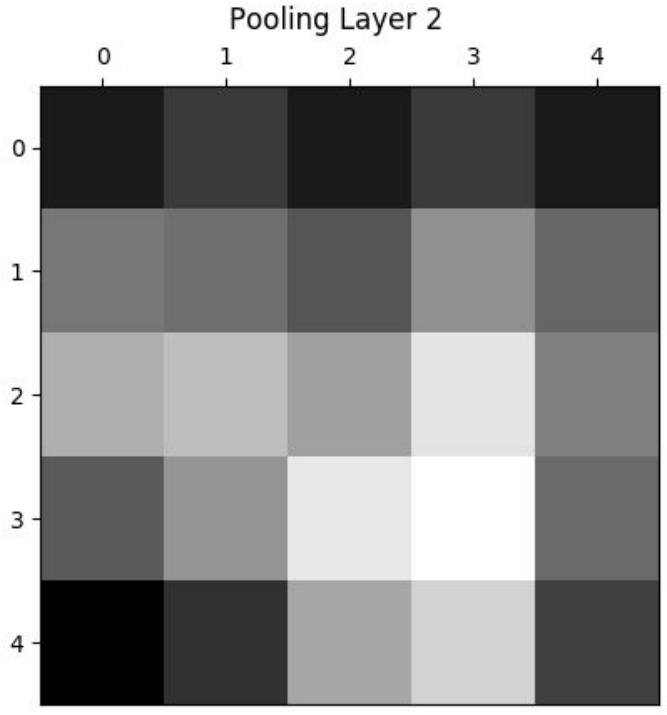

Pooling Layer 2
0
1
2
3
4
0
1
2
3
4

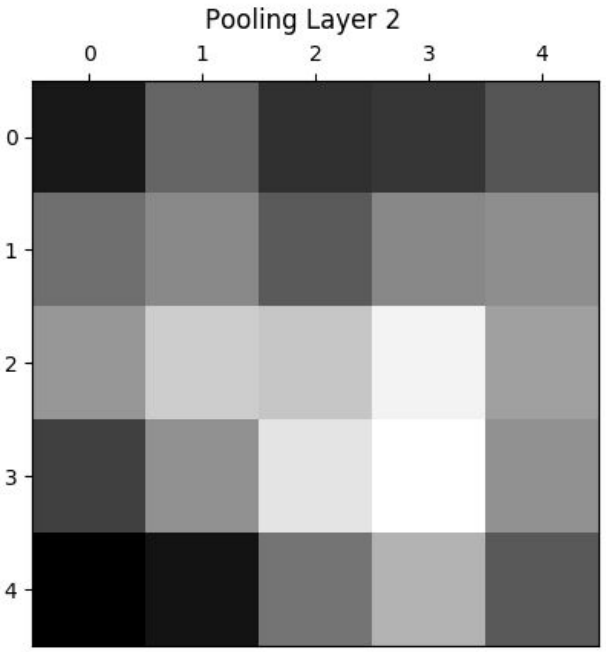

Pooling Layer 2
0
1
2
3
4
0
1
2
3
4

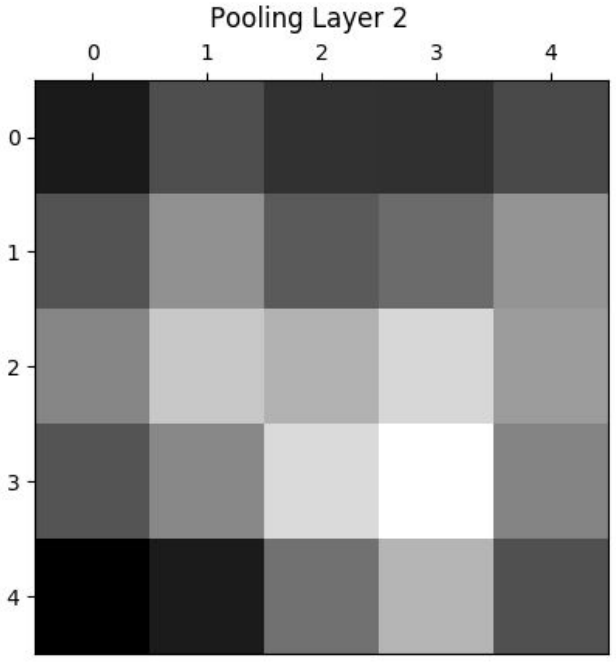

Pooling Layer 2
0
1
2
3
4
0
1
2
3
4


Pooling Layer 2
0
1
2
3
4
0
1
2
3
4


Pooling Layer 2
0
1
2
3
4
0
1
2
3
4


Pooling Layer 2
0
1
2
3
4
0
1
2
3
4


Pooling Layer 2
0
1
2
3
4
0
1
2
3
4


Pooling Layer 2
0
1
2
3
4
0
1
2
3
4

Outputs :

Q1.2)

1) **No. of parameters =** (w*h*3+1)*K
2) 0 parameters
3) **Parameters :**

   Convolutional Layer 1 : 456

   Pooling Layer 1 : 0

   Convolutional Layer 2 : 2416

   Pooling Layer 2 : 0

   Fully Connected Layer 1 : 48120

   Fully Connected Layer 2 : 10164

   Fully Connected Layer 3 : 850

   According to given Lenet Architecture , total parameters are :

   Convolutional Layer : 2872

   Pooling Layer : 0

   Fully Connected Layer : 59134

   Fully Connected Layer contains most number of parameters.

4) **Memory Consumption :** (Assumption: Each neuron contains 1 byte)

   Initial Convolutional Layer : 28*28*6/1000 = 4.704 Kb

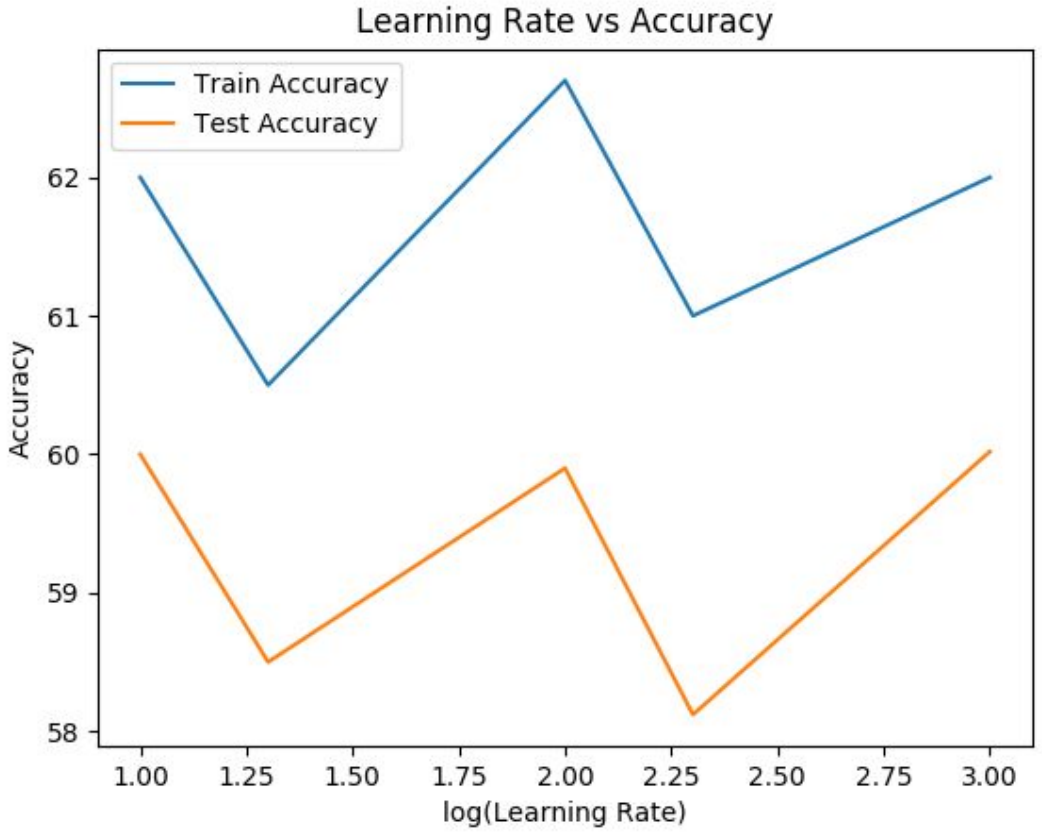
   Fully Connected Layers : (84)/1000 = .084Kb

   Initial Convolutional Layer consumes more memory .

5)

Learning Rate	**Train Accuracy(%)**	**Test Accuracy(%)**
0.001	62.00	60.02
0.005	61.00	58.12
0.01	62.70	59.9
0.05	60.50	58.50
0.1	62.00	60.00

Learning Rate	Train Loss	Test Loss
0.001	1.10	1.136
0.005	1.11	1.1775
0.01	1.08	1.143
0.05	1.125	1.1681
0.1	1.09	1.129



Explanation : When we increase the learning rate, ideally loss should decrease faster but after a certain value of learning rate , there should be an increase in loss due to overshooting.

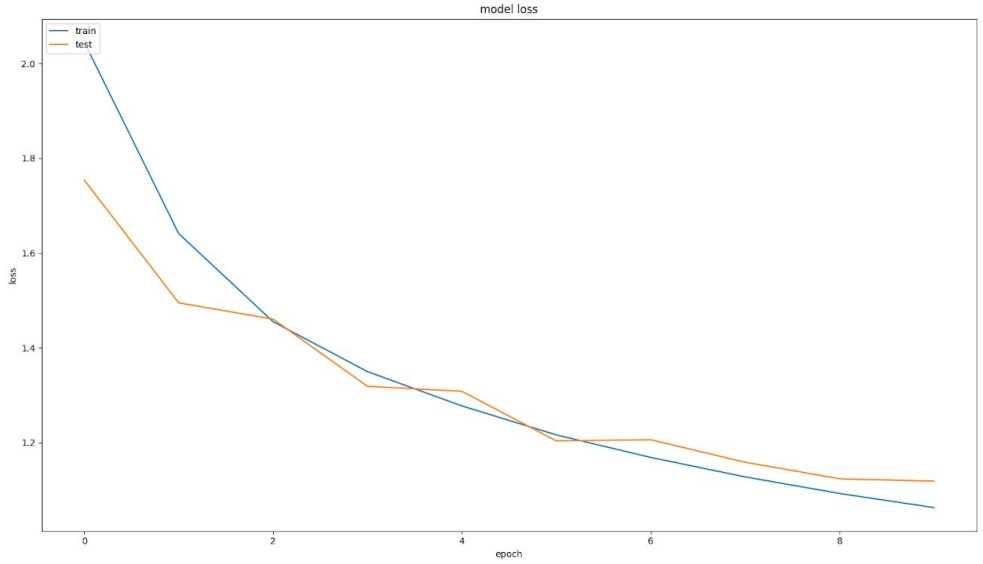
We get the same observations experimentally , as there are multiple local minima's.

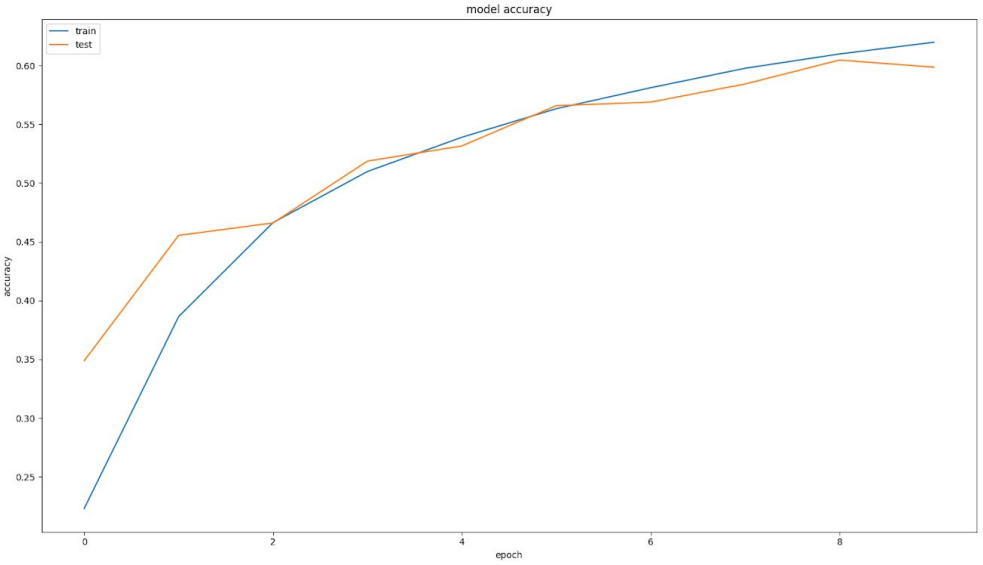
6)

Batch Size	**Train Accuracy(%)**	**Test Accuracy(%)**
1	62.50	59.85
30	63.50	60.28
50	60.00	60.47
65	58.10	54.79
80	58.00	54.67
100	57.50	56.25

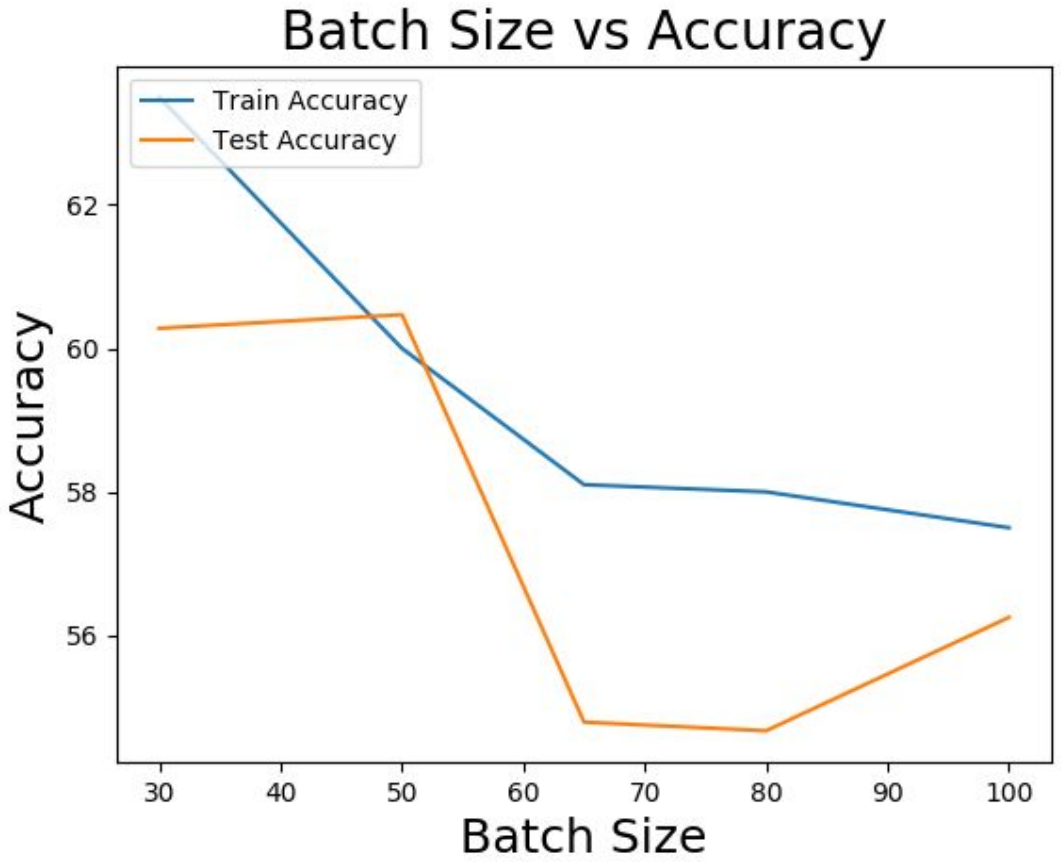
Batch Size	**Train Loss**	**Test Loss**
1	1.10	1.118
30	1.03	1.139
50	1.13	1.16
65	1.17	1.28
80	1.20	1.26
100	1.21	1.239

For Batch Size = 1





From the loss curve , we can observe that test loss curve is little fluctuating.

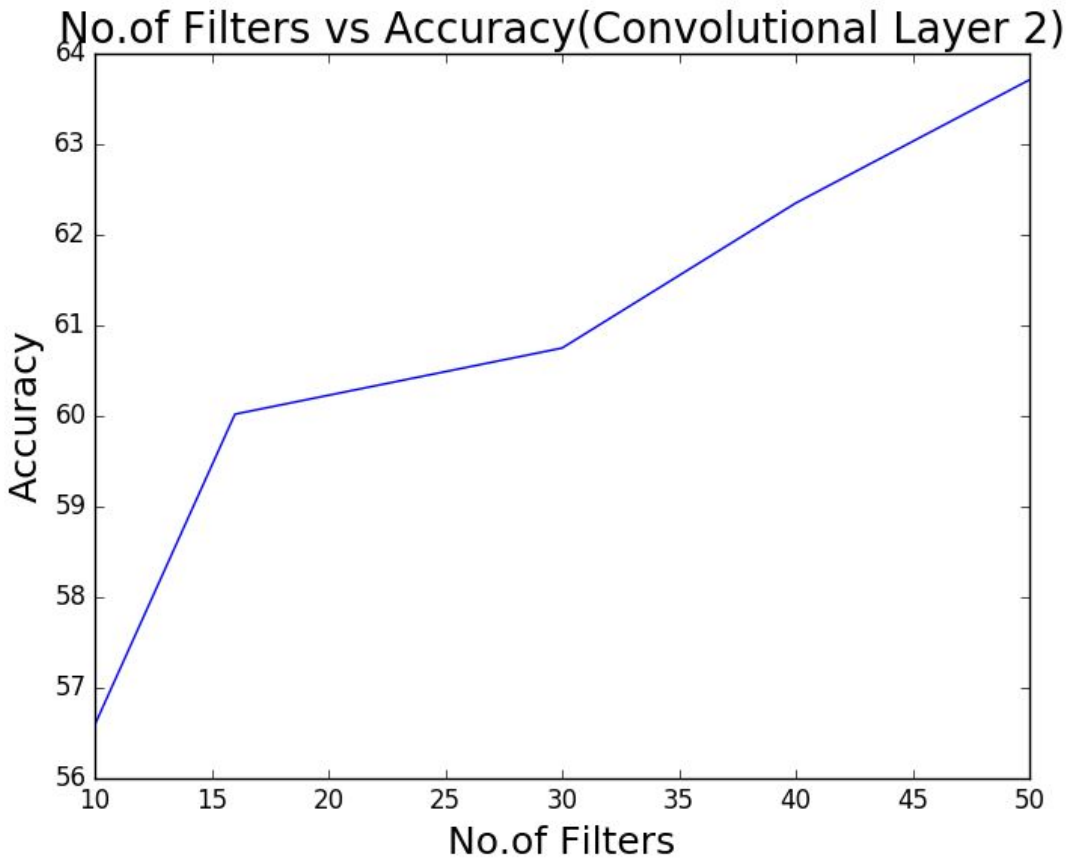


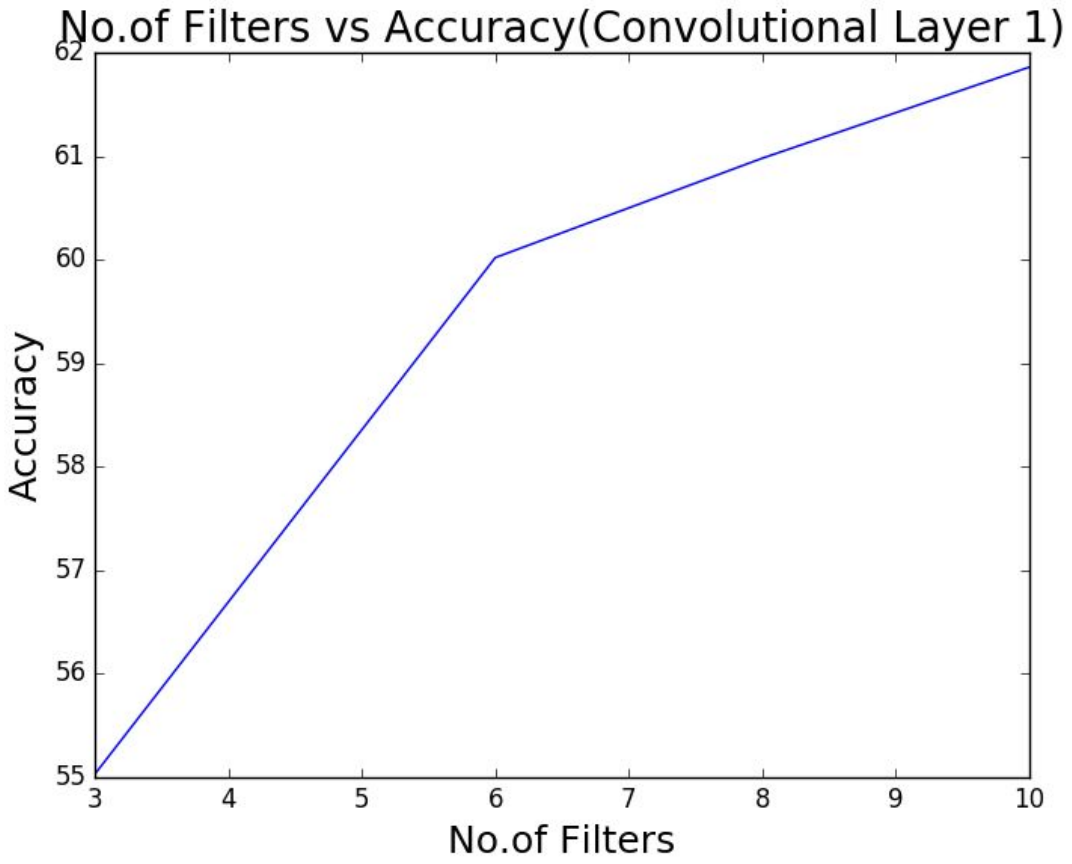
7) For Convolution Layer 2 :

No. of Filters	Test Accuracy(%)	Test Loss
10	56.59	1.221
16	60.02	1.136
30	60.75	1.126
40	62.35	1.102
50	63.71	1.056

For Convolution Layer 1 :

No. of Filters	Test Accuracy(%)	Test Loss
3	55.03	1.056
6	60.02	1.136
8	60.97	1.110
10	61.86	1.07





Explanation : As we increase the number of filters in layers , there is an increase in the number of features due to which accuracy increases as CNN is able to fit the data more accurately.

8)Observations:

1) **Remove Layer 2** : **Test Accuracy :** 32.5 %

2) If we add a layer in middle containing 9 filters of size (5x5) and do not pool after 1st layer , then :

Observations :

conv2d_1 (Conv2D)	(None, 28, 28, 6)	456
activation_1 (Activation)	(None, 28, 28, 6)	0

conv2d_2 (Conv2D) (None, 24, 24, 9) 1359

activation_2 (Activation) (None, 24, 24, 9) 0

max_pooling2d_1 (MaxPooling2 (None, 12, 12, 9) 0

conv2d_3 (Conv2D) (None, 8, 8, 16) 3616

activation_3 (Activation) (None, 8, 8, 16) 0

max_pooling2d_2 (MaxPooling2 (None, 4, 4, 16) 0

flatten_1 (Flatten) (None, 256) 0

dense_1 (Dense) (None, 120) 30840

activation_4 (Activation) (None, 120) 0

dense_2 (Dense) (None, 84) 10164

activation_5 (Activation) (None, 84) 0

dense_3 (Dense) (None, 10) 850

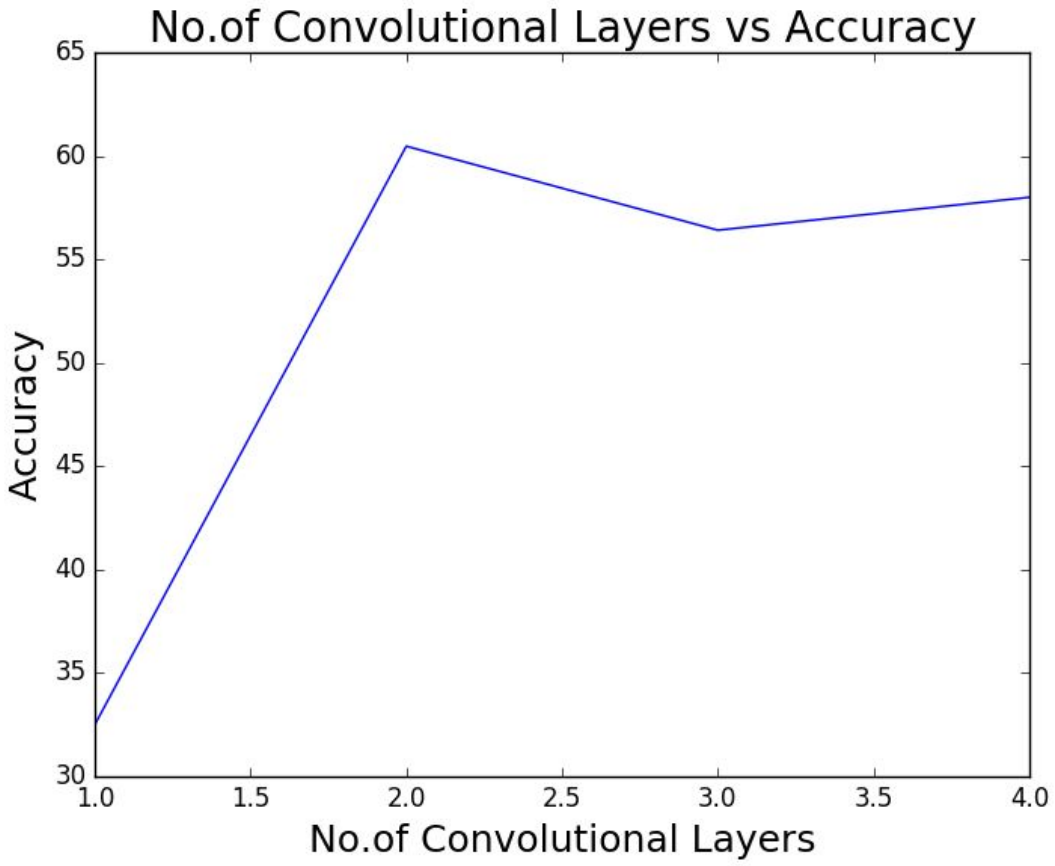
activation_6 (Activation) (None, 10) 0

Test score: 1.2273285781860352

Test accuracy: 0.5641

Observations:

No. of Layers	**Accuracy(%)**
1	32.5%
2	60.47%
3	56.41%
4	58.92%



Explanation : As we increase the number of convolutional layers , we can identify more complex features which should ideally result in better classification but there is a chance of overfitting , hence performance might decrease.

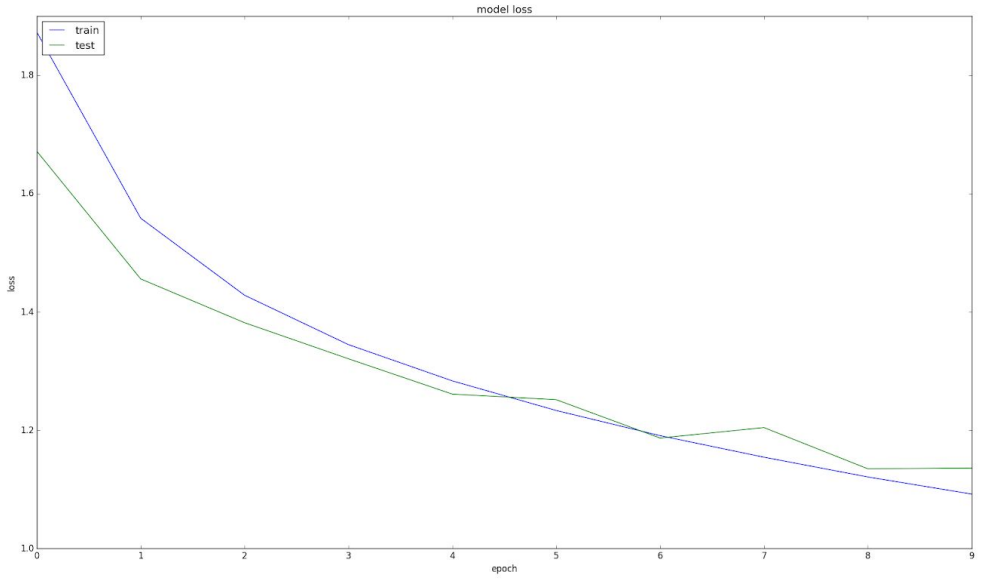
9) Activation Function :

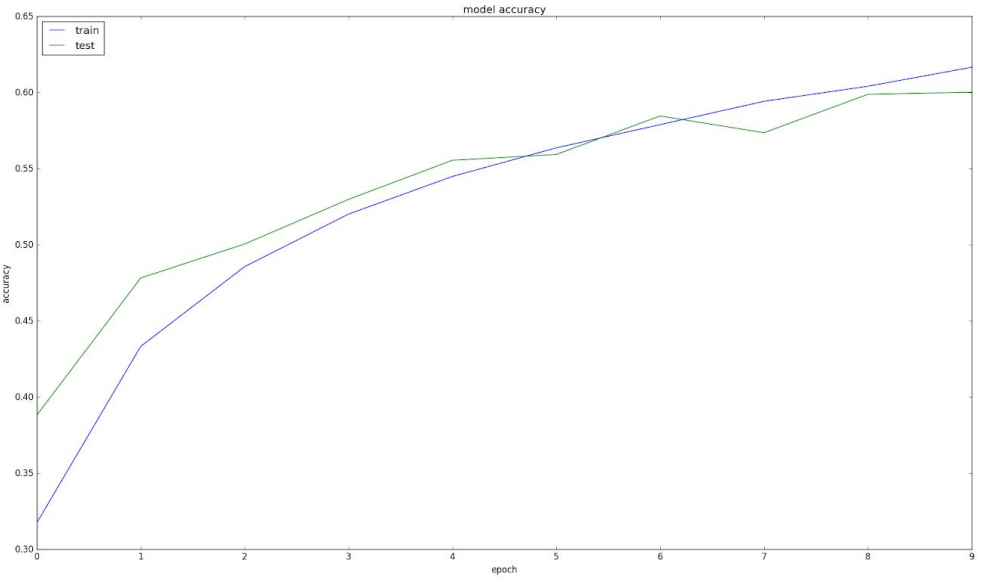
1)**Relu : Test Accuracy =** 60.02 %

Activation Function :

f(x) = x , x>0

0 , x<0

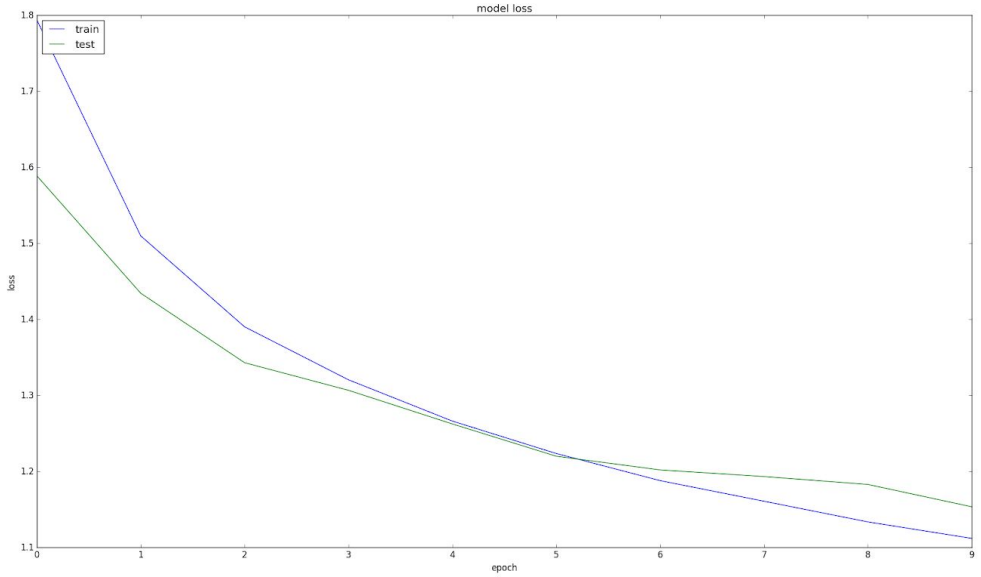


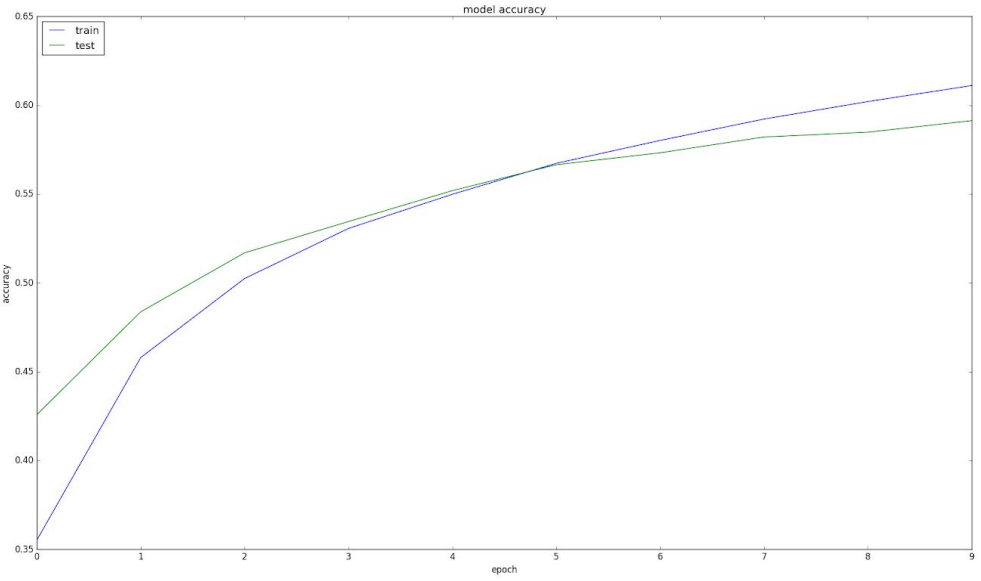


Tanh : Test Accuracy = 59.13

Activation Function :

$$f(x) = (e^{x} - e^{-x}) / (e^{x} + e^{-x})$$

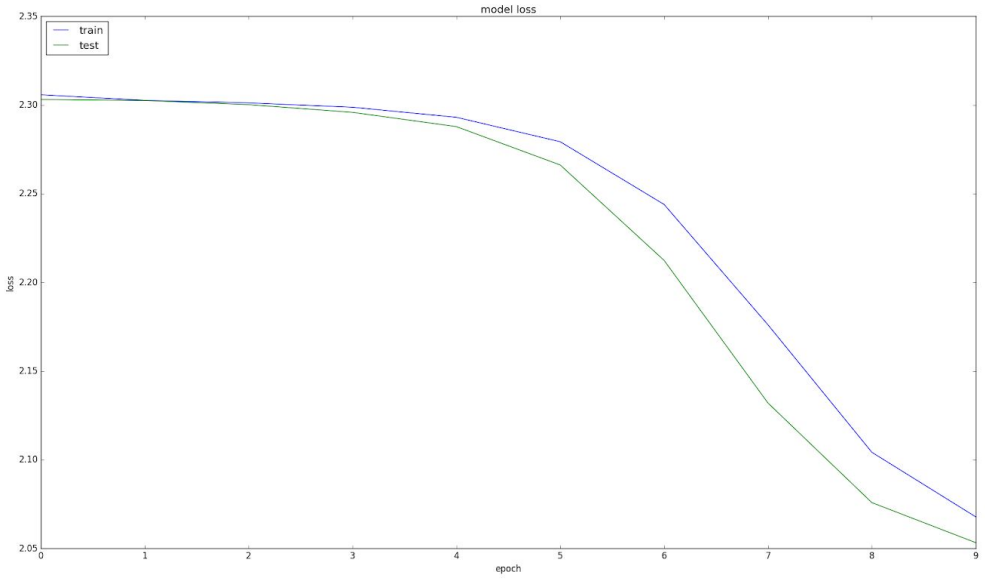


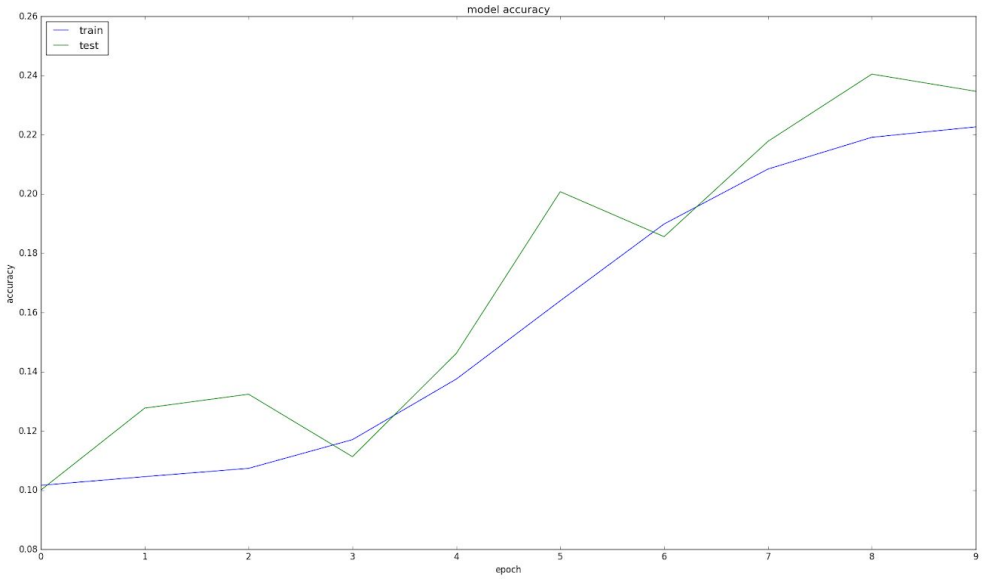


Sigmoid : Test Accuracy = 23.47 %

Activation Function :

f(x) = $1/(1+e^{-x})$





We observe that relu activation function performs best among the three and sigmoid activation function performs quite poor.

10) Skewing, Spherical Transform, Ripple Transform ,Scaling

Q2)

a)For Dermatology Dataset

Using Tanh Activation :

Hidden Neurons: 5

Learning Rate : 0.001

Threshold Param : 0.01

5 fold Cross Validation Accuracy : 79.58

Using sigmoid Activation :

5 fold Cross Validation Accuracy : 82.50

Using Tanh Activation :

Hidden Neurons: 10

Learning Rate : 0.001

Threshold Param : 0.01

5 fold Cross Validation Accuracy : 90.0

Using sigmoid Activation :

5 fold Cross Validation Accuracy : 91.50

b)For Pendigit dataset :

Using Tanh Activation :

Hidden Neurons: 5

Learning Rate : 0.001

Threshold Param : 0.01

5 fold Cross Validation Accuracy : 95.73

Using sigmoid Activation :

5 fold Cross Validation Accuracy : 96.91

Using Tanh Activation :

Hidden Neurons: 10

Learning Rate : 0.001

Threshold Param : 0.01

5 fold Cross Validation Accuracy : 96.42

Using sigmoid Activation :

5 fold Cross Validation Accuracy : 97.73