SMAI Assignment 2
201501020

Q1: [0.5 point]

What are the number of parameters in convolution layers with K filters each of size 3*w*h.

- Ans : number of weights in 1 filter = (3 *w* h) + 1.
- Total number of weights = number of filters * number of weights
  - = K * (3 * w* h + 1)
    = **3 * k * w * h + k**

Q2: [0.5 points]

What are the number of parameters in a max pooling operation?

- **Ans : 0**
- If pooling matrix dimensions are included then its 3 (width and height and stride)

Q3: [0.5 point]

Which of the operations contain most number of parameters? (a) conv (b) pool (c) Fully connected layer (FC) (d) Relu

- **Ans: (c) fully connected layer** n^2 *p (n*n is size of image and p is output layer size)

1. Convolutional Layer 1 : 456
2. Pooling Layer 1 : 0
3. Convolutional Layer 2 : 2416
4. Pooling Layer 2 : 0
5. Fully Connected Layer 1 : 48120
6. Fully Connected Layer 2 : 10164
7. Fully Connected Layer 3 : 850

   According to given Lenet Architecture , total parameters are :

   Convolutional Layer : 2872

   Pooling Layer : 0

   Fully Connected Layer : 59134

Q4: [0.5 point]

Which operation consume most amount of memory? (a) initial convolution layers (b) fully connected layers at the end

- **Ans: (a) Initial Convolutional Layer**

As during memory, we consider parameters used + output, thus

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

Fully Connected Layers : ((84)*10)/1000 = .85Kb

Initial Convolutional Layer consumes more memory .

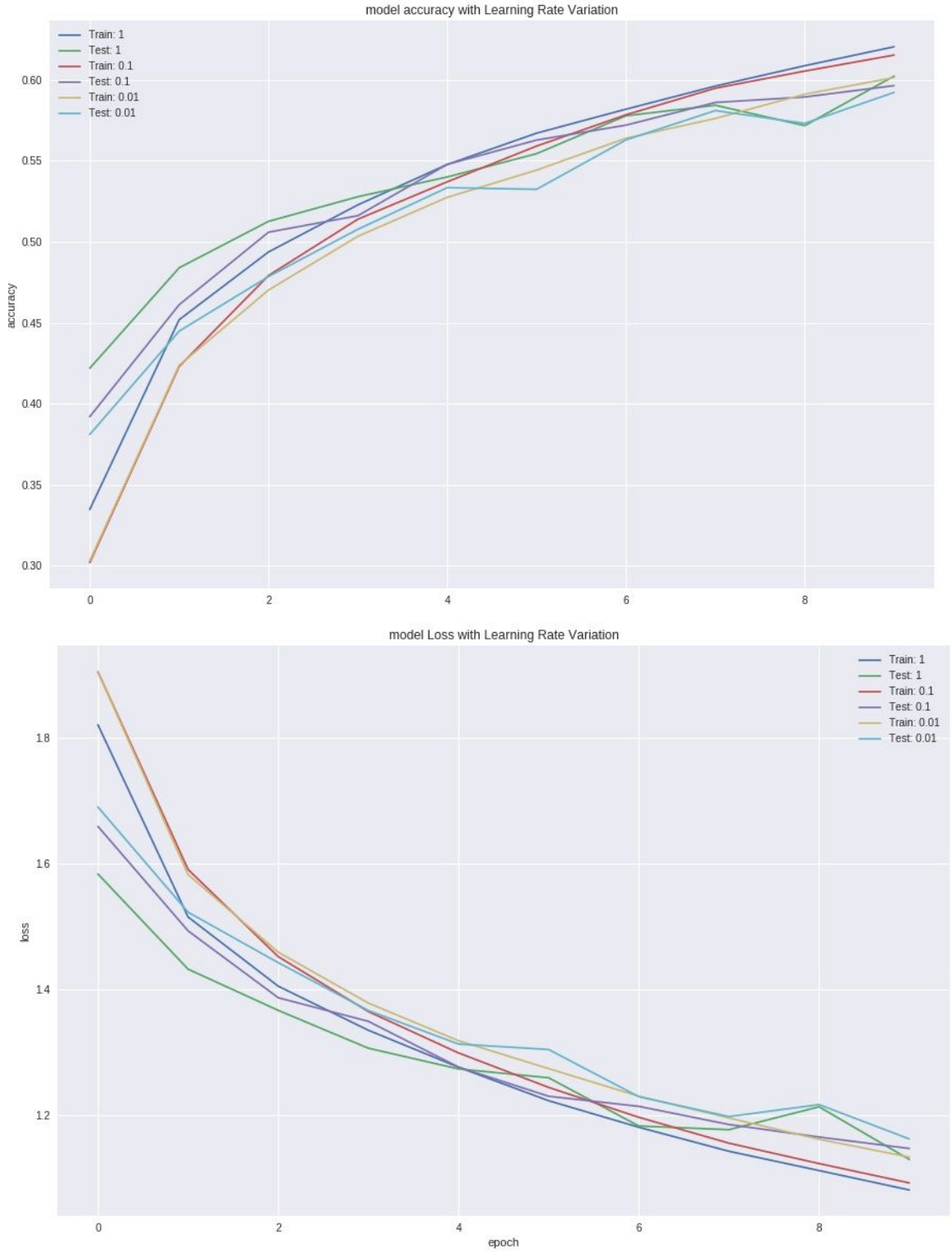
Q5: [2 points]

Experiment with **learning rate** (learningRate) and notice the behaviour of the learning process. Plot your observations in a graph with brief explanation. Take the values on a log scale. Vary only one parameter at a time.

Ans:

Learning Rate	**Train Accuracy(%)**	**Test Accuracy(%)**
1	62	60.02
0.1	61.5	59.5
0.01	60.2	59.2
0.001	60.02	60
0.0001	60	59

Learning Rate	**Train Loss**	**Test Loss**
1	0.4	0.5
0.1	0.45	0.6
0.01	0.5	0.5
0.001	0.6	0.7
0.0001	0.7	0.8



when we increase the learning rate, the rate of learning across epoch that is learning curve are more steeper. But after lot of epochs, curves get to graduate towards a each other. There is increase in training accuracy and

decrease in loss. But in testing there is little fluctuation may be due to overfitting the data as learning rate increased. Also due to overshooting, we can see there are local minimas in the above curve.

Q6: [2 points]

Currently, the **batch-size** is 50. Notice the training loss curve if batch size is changed to 1. Is it smooth or fluctuating? Show the effect of batch-size on the learning curves in a plot. Take the values on a log scale. Vary only one parameter at a time.

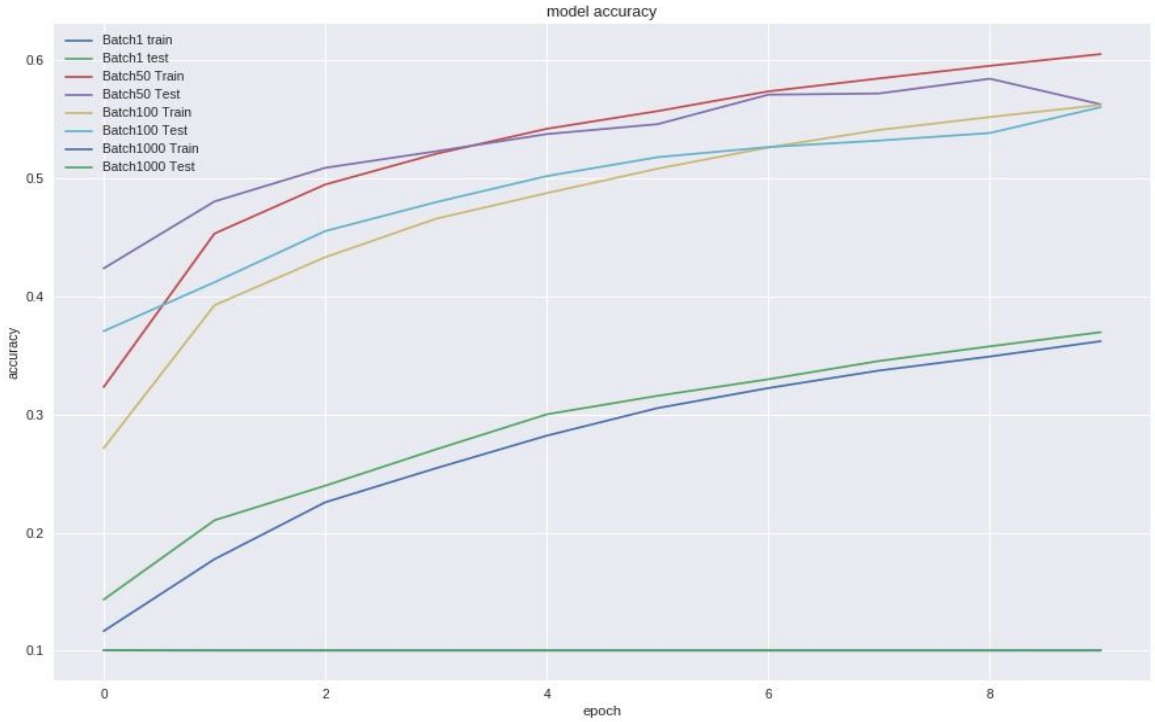
Ans:

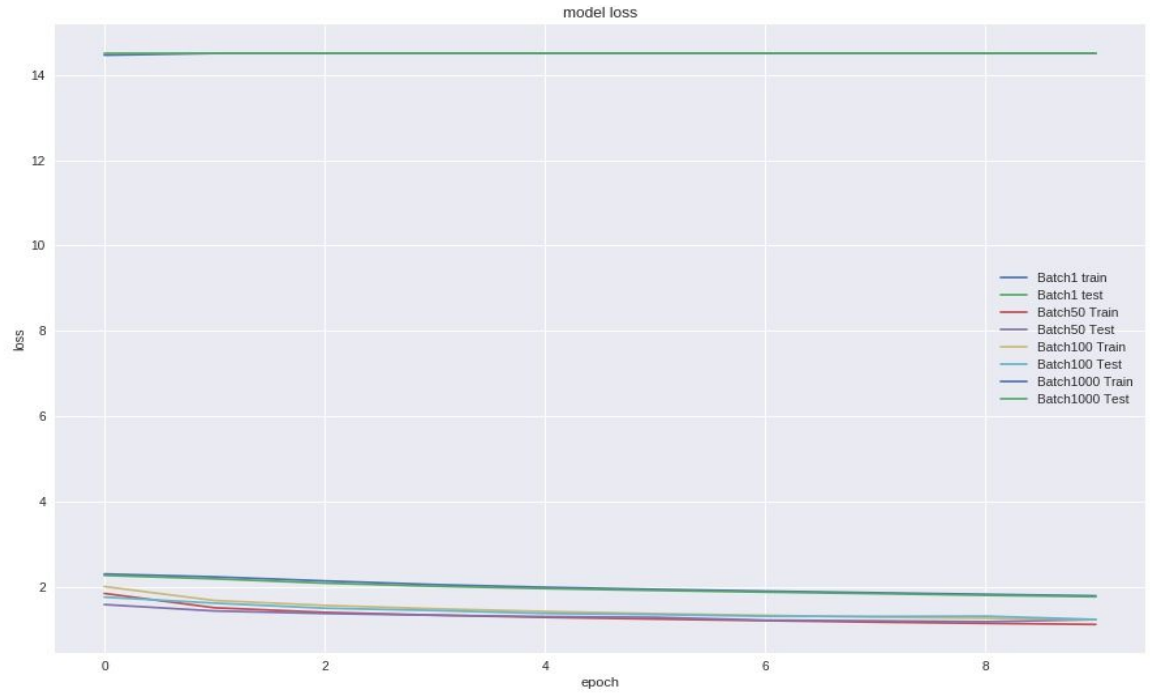
Training Loss iterations: 2.3258, 14.4501, 14.5954, 14.5739, 14.3398, 14.6304

It is fluctuating.

Batch Size	Train Accuracy(%)	Test Accuracy(%)
1	62.5	59.85
50	60	59.47
100	55.1	54.79
1000	35.1	36.1

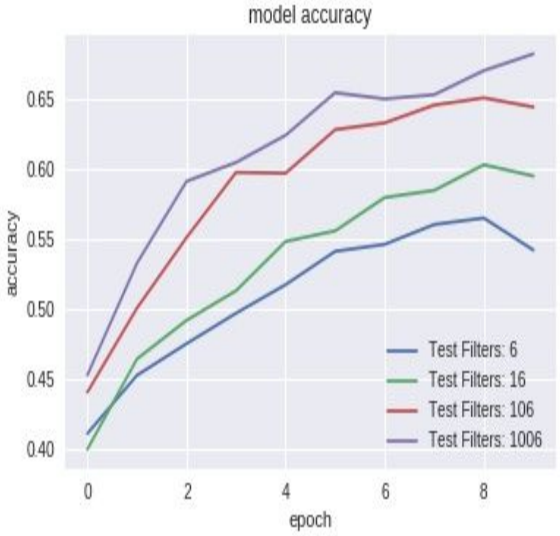
Batch Size	Train Loss	Test Loss
1	1.1	1.12
50	1.13	1.16
100	1.21	1.26
1000	1.95	2

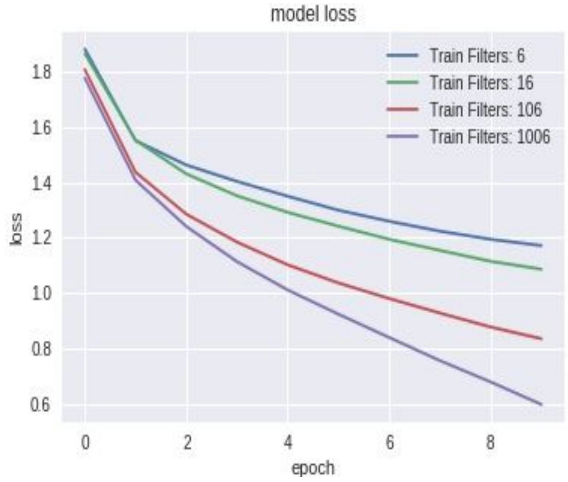

model accuracy
Batch1 train
Batch1 test
Batch50 Train
Batch50 Test
Batch100 Train
Batch100 Test
Batch1000 Train
Batch1000 Test
accuracy
epoch

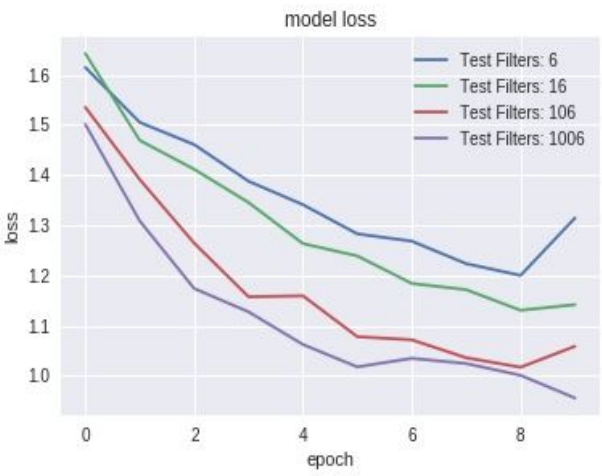

model loss
loss
epoch
Batch1 train
Batch1 test
Batch50 Train
Batch50 Test
Batch100 Train
Batch100 Test
Batch1000 Train
Batch1000 Test

Q7: [2 points]

Increase the **number of convolution filters** and experiment. Present your observations using plots and brief explanations. Take the values on a log scale. Vary only one parameter at a ti







for convolutional Layer 2:

No. of Filters	**Train Accuracy(%)**	**Train Loss**
6	59.59	1.21
16	61.02	1.136
106	71.75	0.6
1006	79.35	0.6

No. of Filters	**Test Accuracy(%)**	**Test Loss**

6	55.59	1.31
16	60.02	1.15
106	64.75	1.05
1006	67.35	0.95

for convolutional Layer 1:

No. of Filters	**Test Accuracy(%)**	**Test Loss**
3	55.03	1.056
6	60.02	1.136
8	60.97	1.11
10	61.86	1.07

Explanation : Increase in filters led to increase in testing and training accuracy. As we are taking more features out of the image, it lead to better accuracy and less loss. but more features always led to overfit the data which is shown in increase of testing accuracy vs training accuracy.

Q8: [2 points]

What do you observe if you increase the **number of layers** (depth of the network) ? Present your observations using plots and brief explanations.

Ans:

Observations:

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

2) If we add a layer in middle containing 6 filters of size (2x2), then :

Observations :

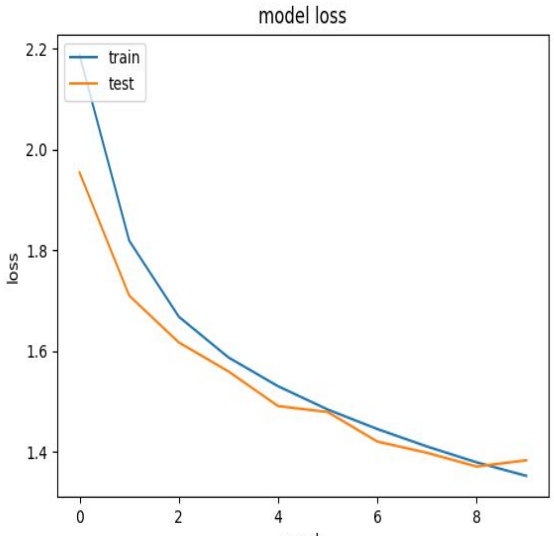
Layer (type)	Output Shape	Param #
conv2d_36 (Conv2D)	(None, 28, 28, 6)	456
activation_78 (Activation)	(None, 28, 28, 6)	0
max_pooling2d_36 (MaxPooling	(None, 14, 14, 6)	0
conv2d_37 (Conv2D)	(None, 13, 13, 6)	150
activation_79 (Activation)	(None, 13, 13, 6)	0

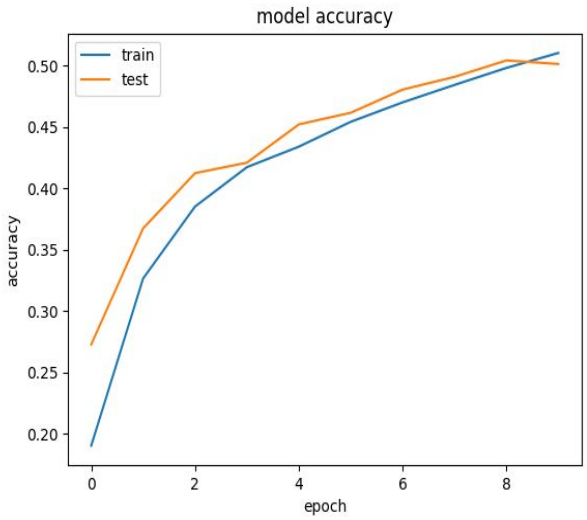
max_pooling2d_37 (MaxPooling	(None, 6, 6, 6)	0	
conv2d_38 (Conv2D)	(None, 2, 2, 16)	2416	
activation_80 (Activation)	(None, 2, 2, 16)	0	
max_pooling2d_38 (MaxPooling	(None, 1, 1, 16)	0	
flatten_15 (Flatten)	(None, 16)	0	
dense_43 (Dense)	(None, 120)	2040	
activation_81 (Activation)	(None, 120)	0	
dense_44 (Dense)	(None, 84)	10164	
activation_82 (Activation)	(None, 84)	0	
dense_45 (Dense)	(None, 10)	850	
activation_83 (Activation)	(None, 10)	0	

==========

Test score: 1.2273285781860352

Test accuracy: 0.5041





Observations:

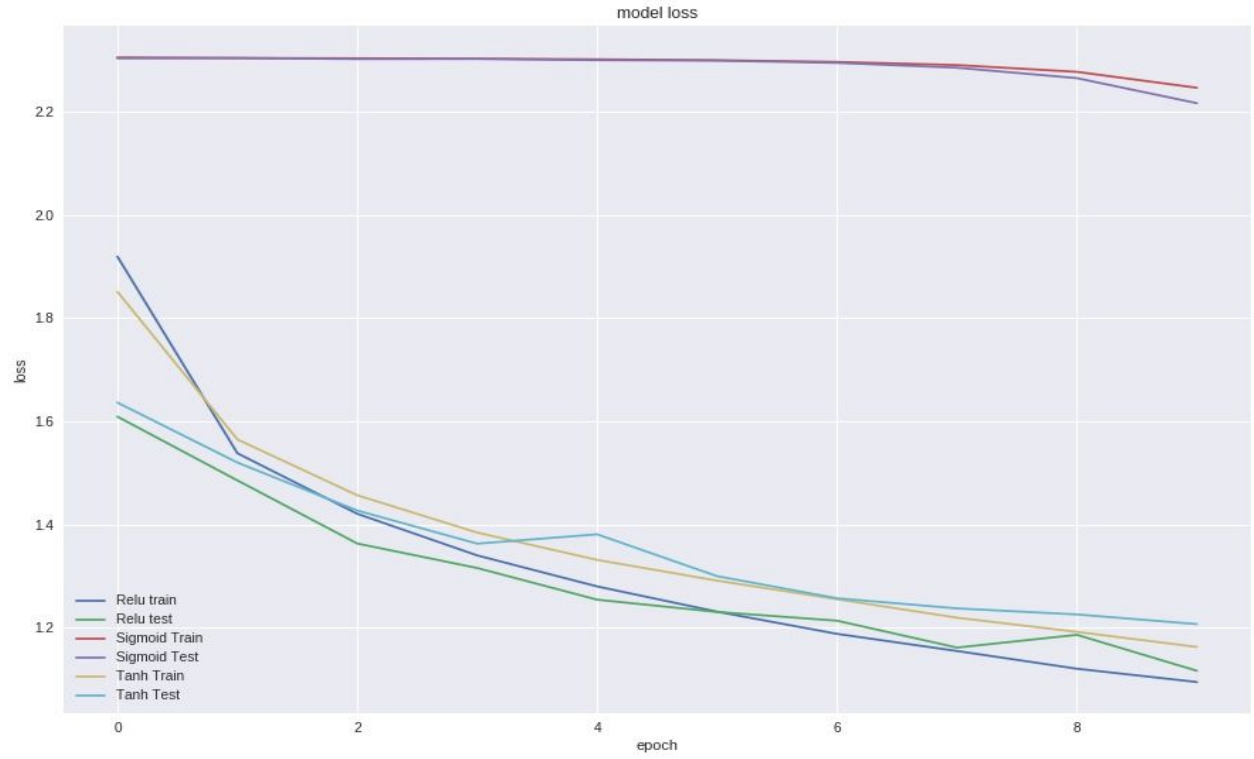
No. of Layers	Accuracy(%)
1	32.50%
2	60.47%
3	50.41%

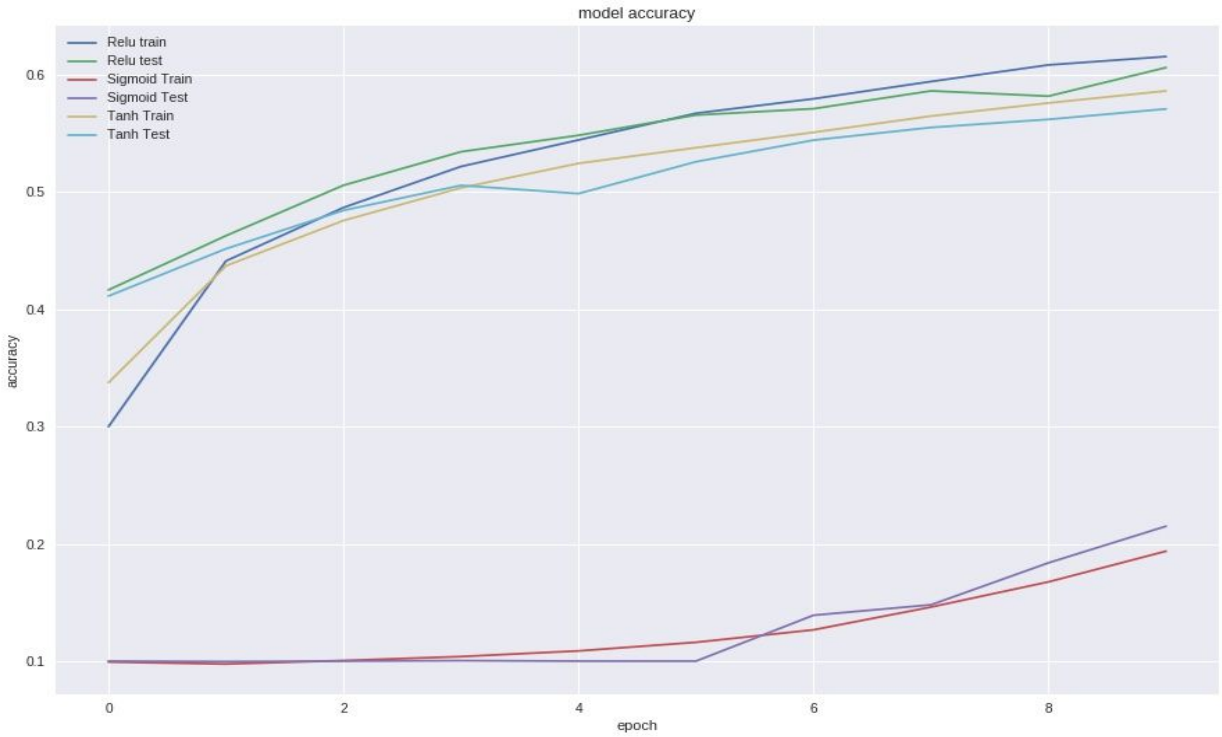
Explanation : As we increase the number of convolutional layers , we can identify more complex features which should ideally result in better classification but there is a chance of overfitting , hence performance will decrease. Without a large training set, an increasingly large network is likely to overfit and in turn reduce accuracy on the test data.

Q9: [2 points]

What do you observe if you increase the **activation functions** (tanh, relu, sigmoid) ? Present your observations using plots and brief explanations.

Ans:


model loss
loss
epoch
Relu train
Relu test
Sigmoid Train
Sigmoid Test
Tanh Train
Tanh Test
2.2
2.0
1.8
1.6
1.4
1.2
0
2
4
6
8


model accuracy
accuracy
epoch
Relu train
Relu test
Sigmoid Train
Sigmoid Test
Tanh Train
Tanh Test
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8

Activation	Train Accuracy	Test Accuracy	Train Loss	Test Loss
RELU	61.5	60.5	1.15	1.1
TANH	58.5	56.5	1.175	1.2
SIGMOID	20.1	22.02	2.1	2.05

Explanation:

We observe that relu activation function performs best among the three and sigmoid activation function performs quite poor. THIS IS due to rapidness of relu function in respect to other activation functions. gradient of relu is more than these thus updates neural network more rapid and converges it to better accuracy easily.

Q10: [1 points]

CNN training requires lot of training data. In the absence of large training data, a common practice is to use synthetic data using operations such as flipping, scaling, etc. Can you think of any other two operations techniques that can help to increase the training set? Demonstrate these effects with sufficient explanation.

a) **Transformation and Duplication: I**mage flipping,random cropping,random scaling,centre zooming
b) increasing/decreasing brightness,sharpness
c) Skewing: changing the positions
d) Ripple Transform: It distorts the image locally using a sinus function.
e) Introducing slight noises into the image pixel values.

2)

1st Part:

Activation function: I have used Relu Activation function and tanh activation function.

- Relu is simple and efficient.
- Relu and tanh are easy to implement as relu is just 2 lines of code and tanh already exist in numpy.
- Relu helps in dropping some neurons at a time and helps in not overfitting data.
- Tanh is used as its graph is over both positive and negative side.

Stopping Condition: loss>delta and delta_loss>0.01* delta and step<20000, as loss is very loss or decrease in loss is small during iterations, so it is better to stop here, as accuracy will not increase that much in some time.

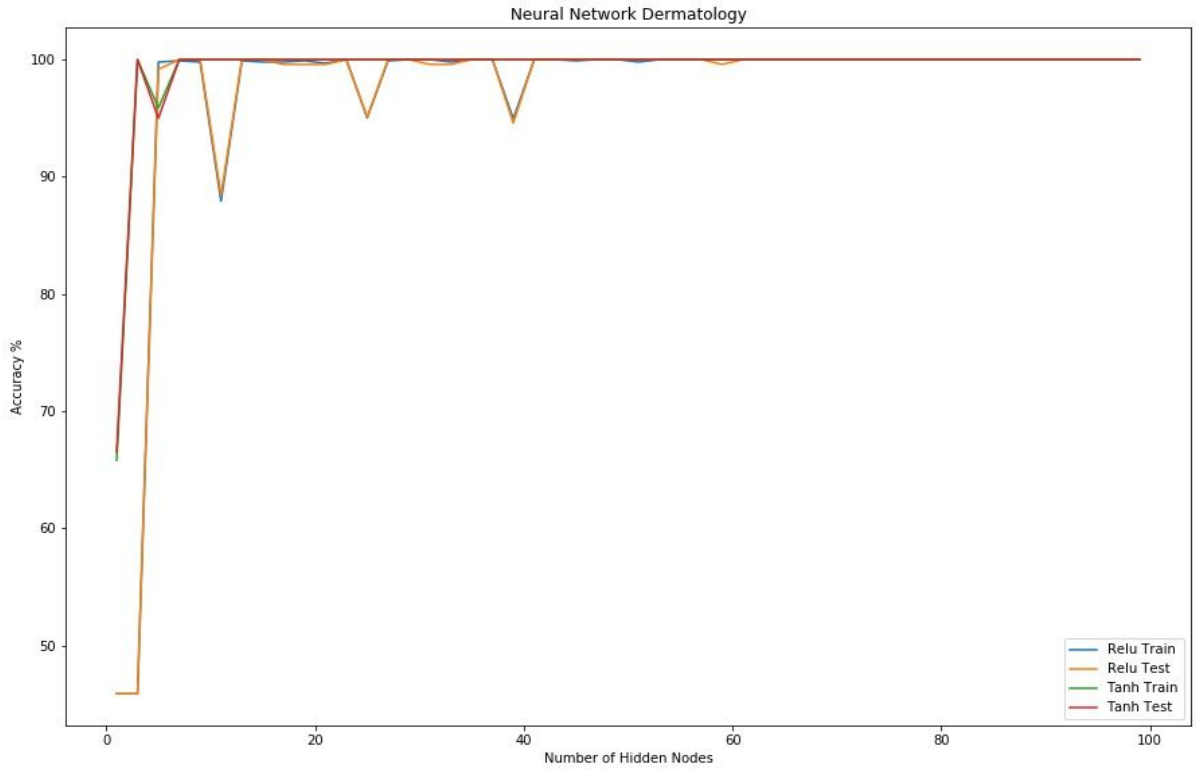
Experiments:

using pendigits

Learning Rate	Loss	Training	Testing
1	1.3857441600890648	25.506867233485938	25.437981779957955
0.1	1.3861948526051362	25.506867233485938	25.437981779957955
0.01	0.20919640732606978	96.0758665794637	94.74421864050456
0.001	0.19535817863602975	97.02419882275997	95.30483531885073

Final Model:
Dermatology Relu or tanh [learning rate: 0.01, hidden node: 10](100, 100)
pendigits Relu[learning rate: 0.001. hidden node: 10](98.17799579537491, 98.17799579537491)

a) **Dermatology: Learning Constant: 0.05 and Regularization: 0.01**

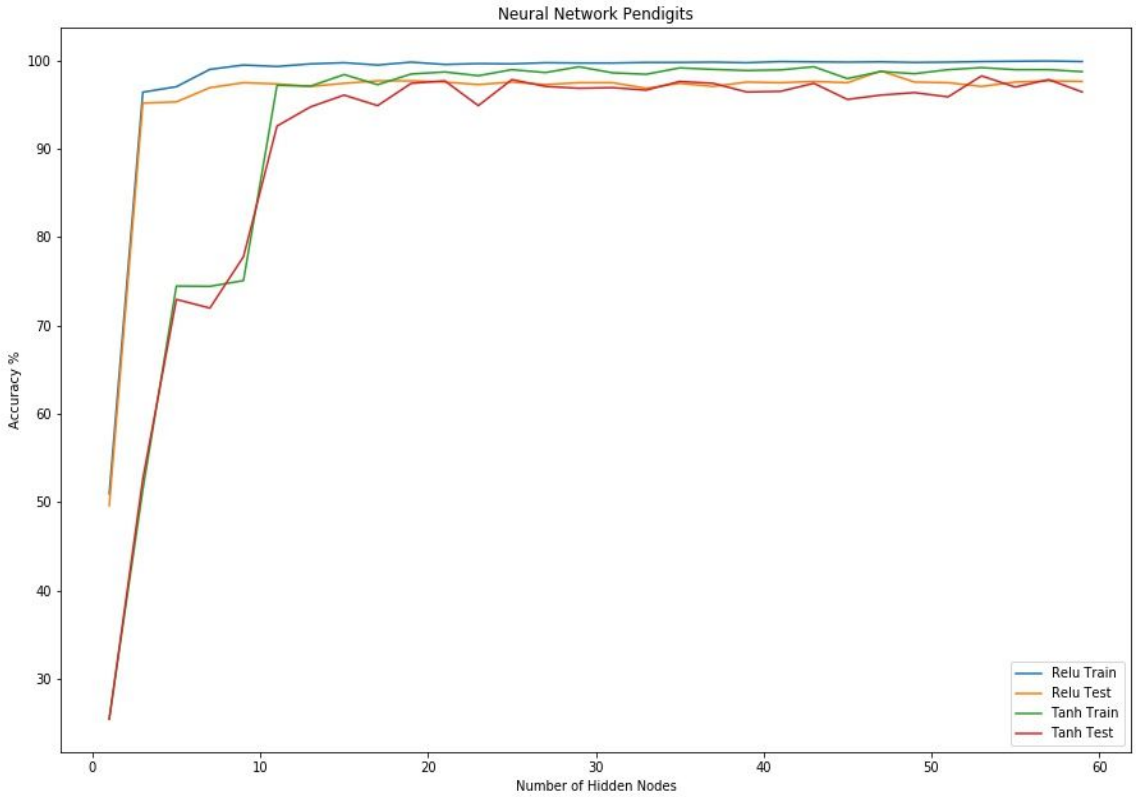


Nodes	Relu Test	Relu Train	Tanh Test	Tanh Train

1	45.87585034	45.86827627	66.52210884	65.79616473
3	45.87585034	45.86827627	100	100
5	99.17517007	99.79274611	95	95.87628866
7	100	99.89690722	100	100
9	100	99.79381443	100	100
11	88.41836735	87.91357299	100	100
13	100	99.89690722	100	100
15	100	99.79381443	100	100
17	99.58333333	99.79381443	100	100
19	99.58333333	99.89690722	100	100
21	99.58333333	99.69072165	100	100
23	100	100	100	100
25	95.10204082	95.02590674	100	100
27	100	99.89690722	100	100
29	100	100	100	100
31	99.59183673	100	100	100
33	99.58333333	99.79328027	100	100
35	100	100	100	100
37	100	100	100	100
39	94.58333333	94.94845361	100	100
41	100	100	100	100
43	100	100	100	100
45	100	99.89690722	100	100
47	100	100	100	100
49	100	100	100	100
51	100	99.79381443	100	100
53	100	100	100	100
55	100	100	100	100
57	100	100	100	100
59	99.59183673	100	100	100
61	100	100	100	100
63	100	100	100	100
65	100	100	100	100
67	100	100	100	100

69	100	100	100	100
71	100	100	100	100
73	100	100	100	100
75	100	100	100	100
77	100	100	100	100
79	100	100	100	100
81	100	100	100	100
83	100	100	100	100
85	100	100	100	100
87	100	100	100	100
89	100	100	100	100
91	100	100	100	100
93	100	100	100	100
95	100	100	100	100
97	100	100	100	100
99	100	100	100	100

b) Pendigits: Learning Constant: 0.001 and Regularization: 0.01



Nodes	Signum Test	Signum Train	Tanh Test	Tanh Train
1	49.61457603	50.94833224	25.43798178	25.50686723
3	95.16468115	96.4028777	52.62789068	51.40614781
5	95.30483532	97.02419882	72.95024527	74.46043165
7	96.91660827	98.98626553	71.96916608	74.42773054
9	97.47722495	99.47678221	77.78556412	75.08175278
11	97.33707078	99.31327665	92.57182901	97.18770438
13	97.05676244	99.60758666	94.74421864	97.08960105
15	97.40714786	99.73839111	96.07568325	98.39764552
17	97.6874562	99.47678221	94.88437281	97.25310661
19	97.6874562	99.80379333	97.40714786	98.46304774
21	97.54730203	99.54218443	97.6874562	98.69195553
23	97.26699369	99.64028777	94.88437281	98.26684107
25	97.54730203	99.60758666	97.82761037	98.95356442
27	97.26699369	99.73839111	97.05676244	98.6265533

29	97.47722495	99.70568999	96.84653118	99.28057554
31	97.47722495	99.70568999	96.91660827	98.59385219
33	96.84653118	99.77109222	96.63629993	98.43034663
35	97.40714786	99.77109222	97.61737912	99.14977109
37	97.05676244	99.80379333	97.40714786	98.98626553
39	97.54730203	99.73839111	96.42606868	98.85546109
41	97.47722495	99.86919555	96.49614576	98.92086331
43	97.61737912	99.83649444	97.40714786	99.28057554
45	97.47722495	99.80379333	95.58514366	97.93982995
47	98.80868956	99.83649444	96.07568325	98.74362328
49	97.54730203	99.77109222	96.35599159	98.49574886
51	97.47722495	99.80379333	95.87105816	98.95029431
53	97.05676244	99.86919555	98.24807288	99.1824722
55	97.54730203	99.90189666	96.98668535	98.95356442
57	97.6874562	99.93459778	97.82761037	98.95356442
59	97.61737912	99.86919555	96.42606868	98.72465664