Machine Learning

Assignment 1

Mohit Sharma
201505508

Q1. Can you make the confusion matrix for the above network.

Ans: Confusion matrix:

```
[[ 969    1    0    2    0    1    1    0    5    1]
 [   0 1116    2    0    0    2    2    1   12    0]
 [   2    0 1005    3    1    0    1    3   16    1]
 [   0    0    3  989    0    7    0    3    6    2]
 [   2    0    5    0  928    2    5    3    6   31]
 [   1    0    0    7    0  876    1    0    6    1]
 [   5    2    0    1    2   19  923    0    6    0]
 [   0    0    8    1    0    0    0 1011    6    2]
 [   0    0    1    3    0    2    0    2  964    2]
 [   2    2    0    4    4    1    0    3    5  988]]
```

Code:

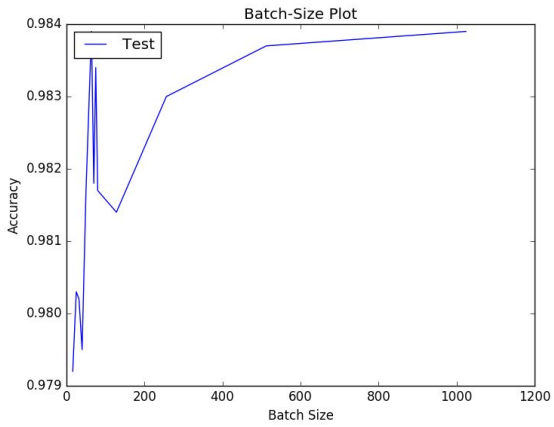
```
from sklearn.metrics import confusion_matrix

Y_pred = model.predict_classes(X_test, batch_size=batch_size)

Y_testClass = []

for i in range(len(Y_test)):

        maxP = -1

        digit = 0

        for j in range(len(Y_test[i])):

        if(maxP<Y_test[i][j]):

        maxP = Y_test[i][j]

        digit = j

        Y_testClass.append(digit)

print

print("Confusion matrix is:")

print(confusion_matrix(Y_testClass, Y_pred, labels = [0,1,2,3,4,5,6,7,8,9]))
```

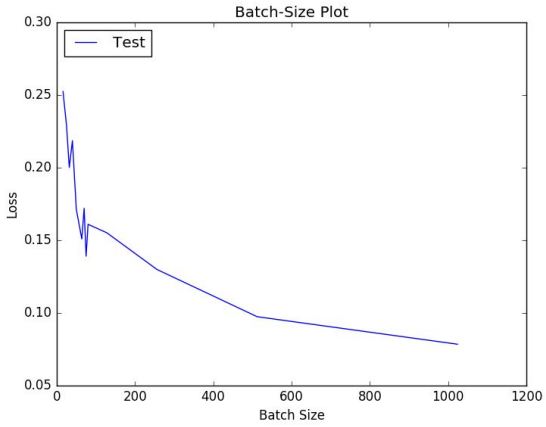
Q2. Try out the following variations and see how it affects loss and accuracy. Present your observations with plots and suitable explanation:

- Batch size
- Size of hidden layer
- choice of activation function ('relU','sigmoid','tanh')

Ans:

1. Batch Size

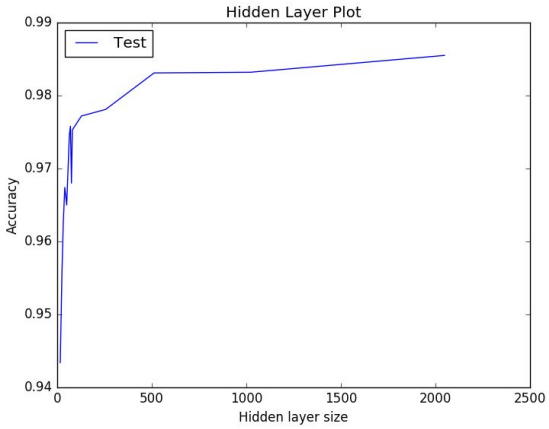


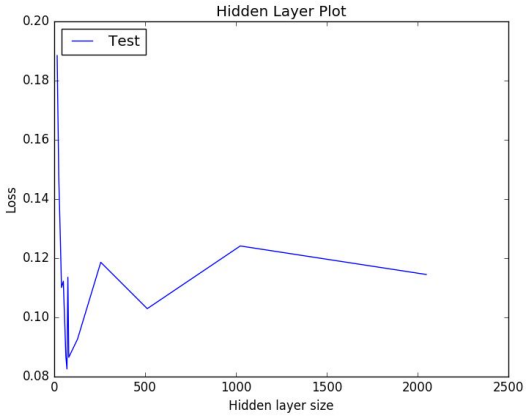


Observations:

1. Accuracy fluctuates a lot with smaller batch size because of less information being provided in single iteration. We can also say that network is getting less samples to see in one iteration if batch size is small. So number of epochs required for a smaller batch size will be more in order to have a good accuracy. That means network will take more time to stabilize.
2. With smaller batch size an epoch takes more time to complete than that of larger batch size. As number of back propagations are more in a single epoch if batch size is smaller.
3. When batch size is large network stabilizes in less no. of epochs. That is why there is not much change in accuracy for large batch sizes.
4. When batch size is too high then it'll be difficult for network to learn as there are chances that different data points will neutralize the loss done by each other.

2. Size of Hidden Layer



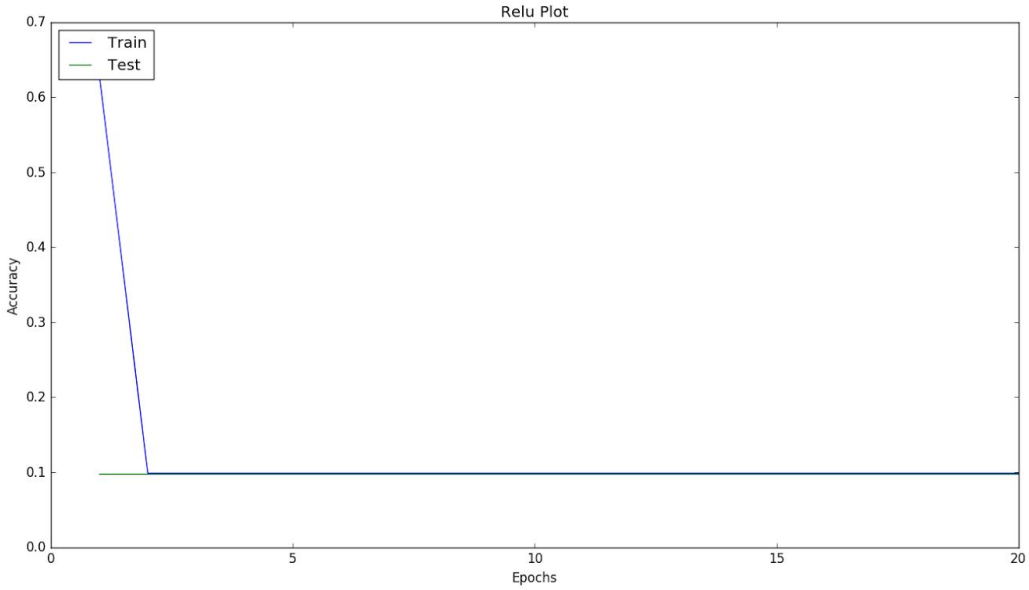


Observations:

1. We can see that accuracy increases with number of nodes as there are more weights which can learn more information.
2. But after a threshold the test accuracy will start dropping as too many parameters in network cause over fitting.
3. More time will be taken as the number of nodes are increased.

3. choice of activation function ('relu','sigmoid','tanh')

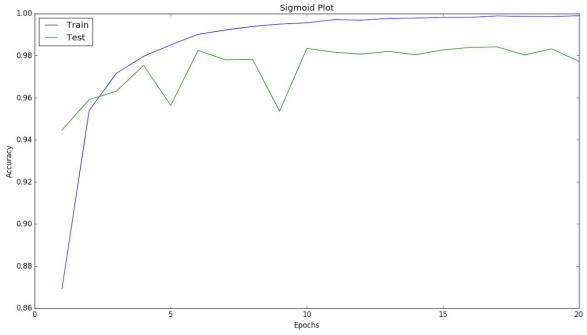
A. relu

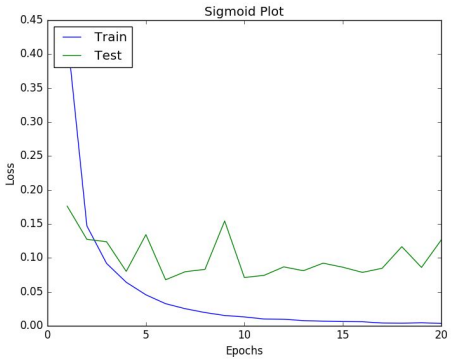


Observations:

1. Relu didn't work well in this case before of gradients exploding. Having relu as activation function will make the gradients go infinitely large as there is no function like sigmoid to bound. Loss values in every epoch were all NaNs, so couldn't be plotted.

B. Sigmoid

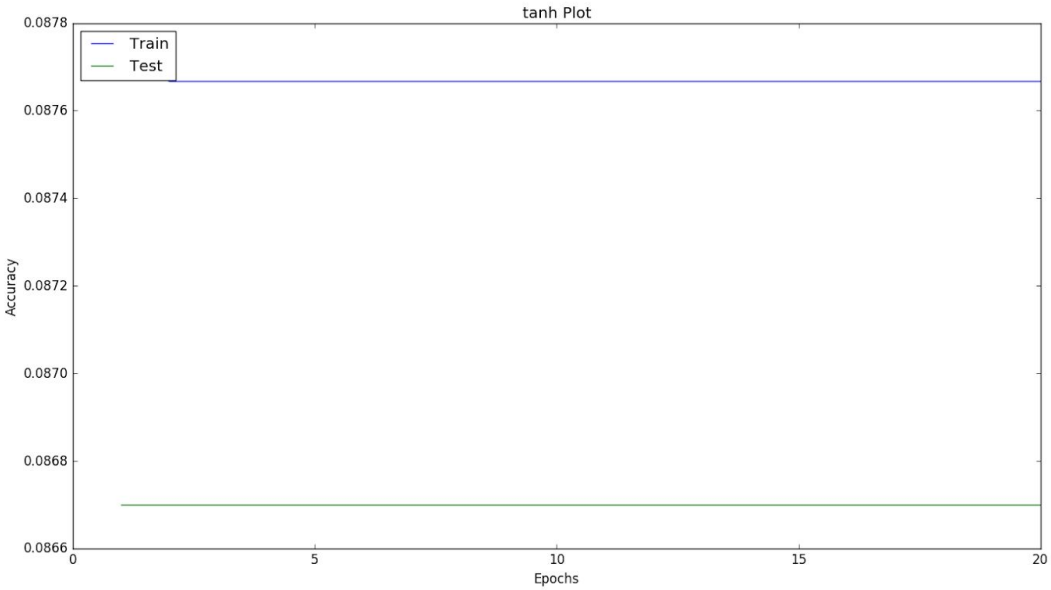


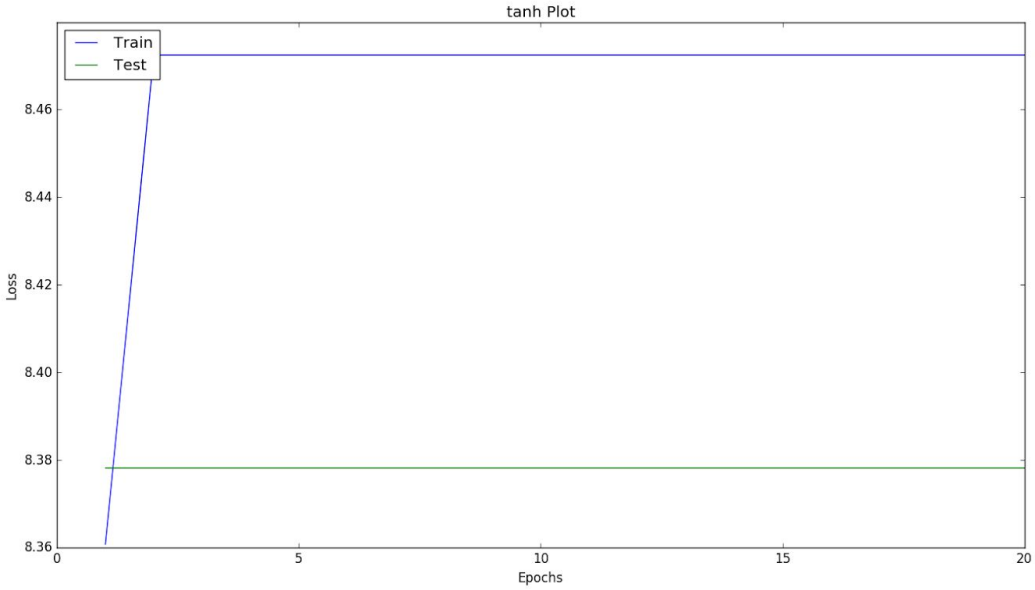


Observations:

1. Sigmoid has done a good job here. We can see accuracy increasing after every epoch.

C. tanh



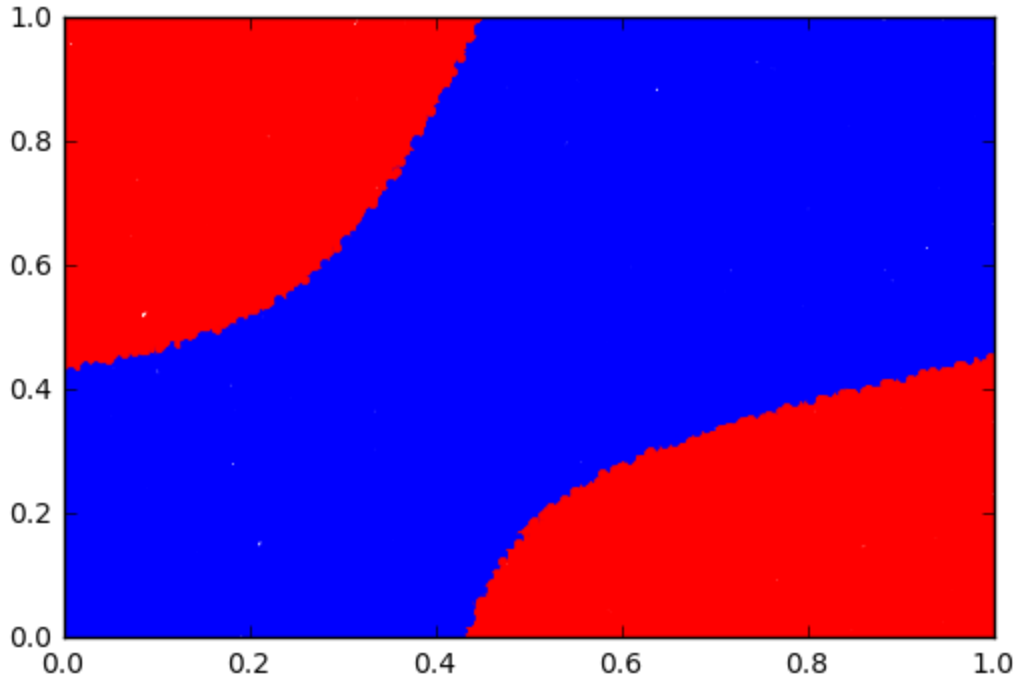


Observations:

1. No significant increase in learning with tanh.

Q3. Try classification on XOR data using similar MLP.

Ans: Plot for XOR



Q4. What are the number of parameters in convolution layers with K filters each of size 3*w*h.

Ans: Parameters with each filter = 3*w*h + 1 (1 is for bias).

For K filters = K*(3*w*h + 1)

Q5. What are the number of parameters in a max pooling operation?

Ans: No parameters are required for pooling operation. It is an operation for downsampling which involves only the data of previous layer. For example in Max Pooling we select maximum element of sub blocks. No parameter is required for that.

Q6. Which of the operations contain most number of parameters? (a) conv (b) pool (c) Fully connected layer (FC) (d) Relu

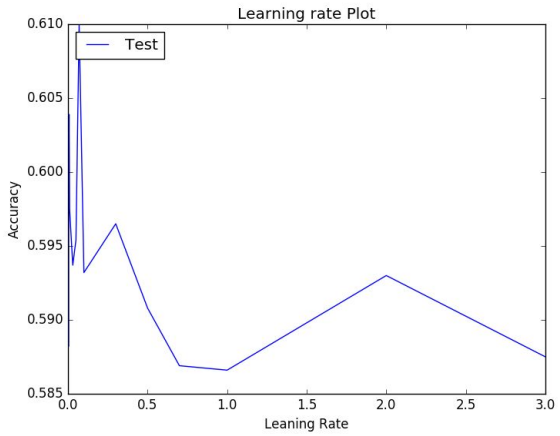
Ans: Fully connected layer. As all nodes of one FC layer are connected to next with different weights. As no weight sharing happens is there in FC layer.

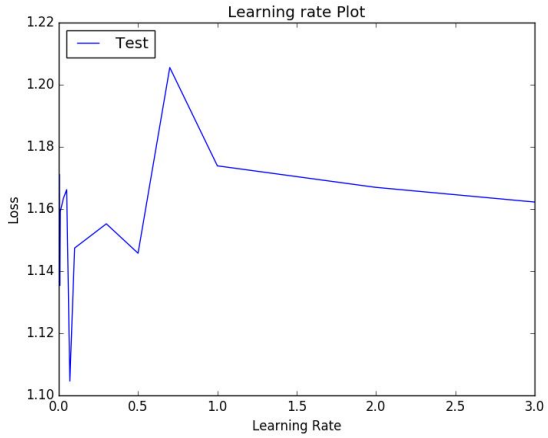
Q7. Which operation consume most amount of memory? (a) initial convolution layers (b) fully connected layers at the end

Ans: Initial convolutional layers consume the most amount of memory as we have more nodes/neurons in initial layers. In later layers number of neurons decreases because of down sampling operations done by pooling layers.

Q8. Experiment with learning rate (learningRate) and notice the behaviour of the learning process. Plot your observations in a graph with brief explanation.

Ans:



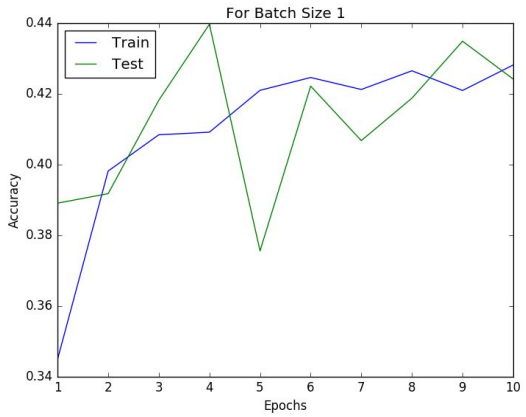


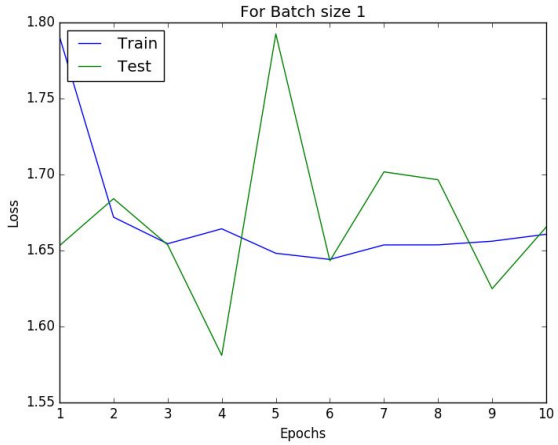
Observations:

1. Accuracy is high with smaller learning rates.
2. It increases a bit and then decreases by a large margin. Reason is that a small learning rate makes the network learn slowly and convergence time is more. As the rate is increased, the network converges faster but if it becomes too high the weights and objective function will diverge. A high learning rate will change the weights with a large margin in every update. It results in weights oscillating around some value but not able to converge.

Q9. 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.

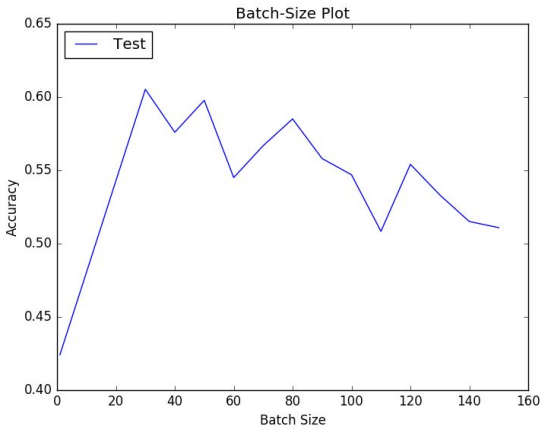
Ans:





Observations:

1. Accuracy fluctuates a lot with each epoch. As a single sample can't provide enough information in a single epoch. When next sample comes, all weights will change again making it difficult for network to get stable.
2. It'll require large number of epochs to learn.

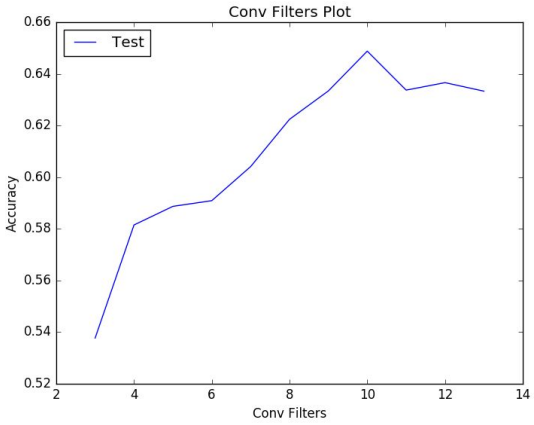


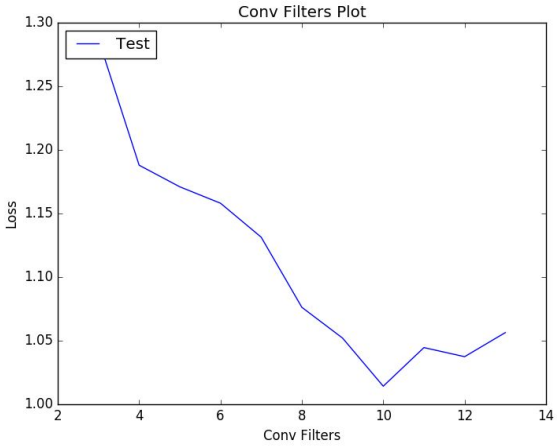
Observations:

1. Similar to the plot seen in Question 1 for batch-size variation. But here we can observe the drop in learning after some threshold as discussed in Question 1.

Q10. Increase the number of convolution filters and experiment. Present your observations using plots and brief explanations.

Ans:



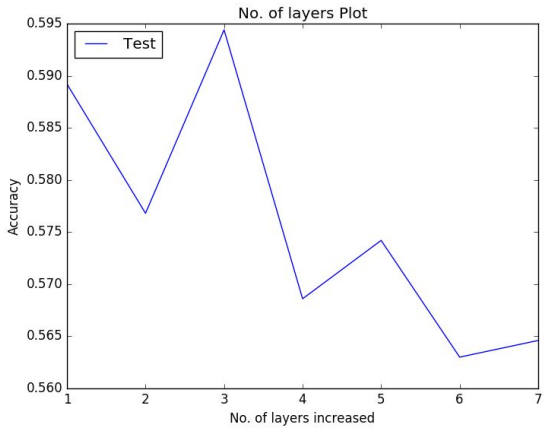


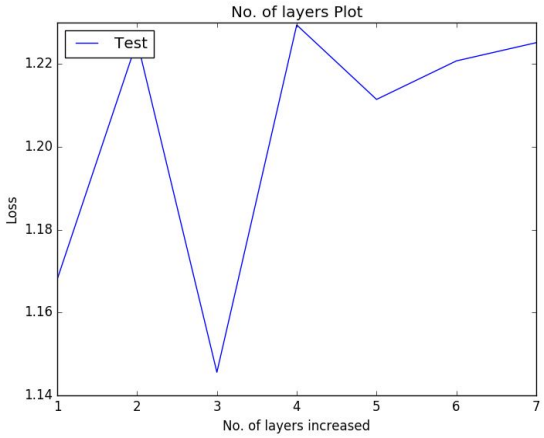
Observations:

1. Accuracy increases with number of convolution filters as more filters can learn more about the local features. We can think of it as different filters learning different local features. As we increase the filters, local features can be learnt well. It increases the overall accuracy of network.
2. But after some threshold it starts decreasing as in that case we have more weights to be learnt which require more number of iteration. Also too many parameters start causing over fitting.

Q11. What do you observe if you increase the number of layers (depth of the network) ? Present your observations using plots and brief explanations.

Ans:





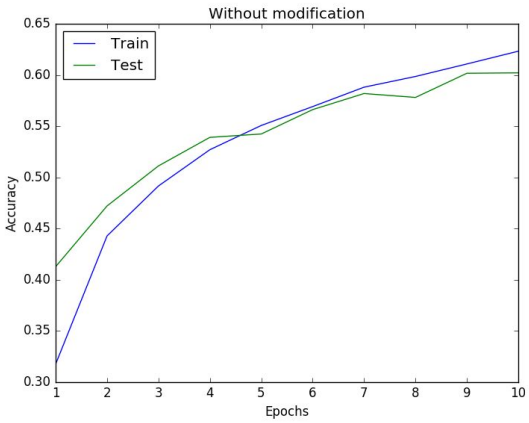
Observations:

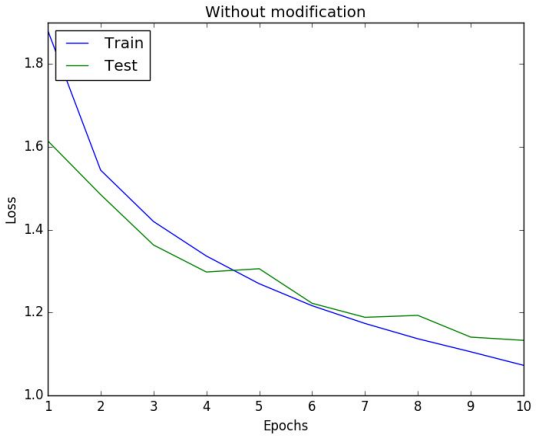
1. The graphs represent the accuracy achieved by increasing 1 FC layer every time. More number of layers start causing over fitting which result in a large drop in accuracy for test data. Also as mentioned in previous points more parameters mean more epochs are required to learn.

Q12. 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? Plot the graphs which demonstrate these effects with sufficient explanation.

Ans:

1. Without adding any synthetic data

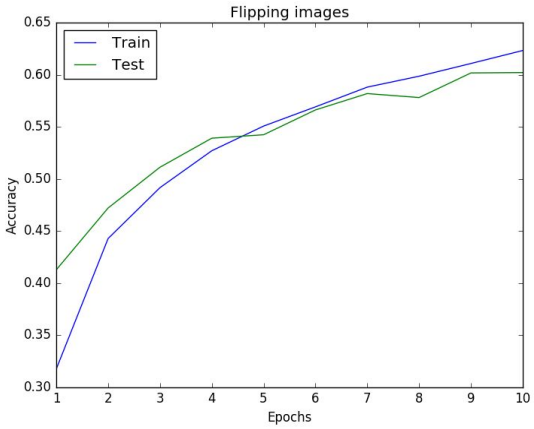


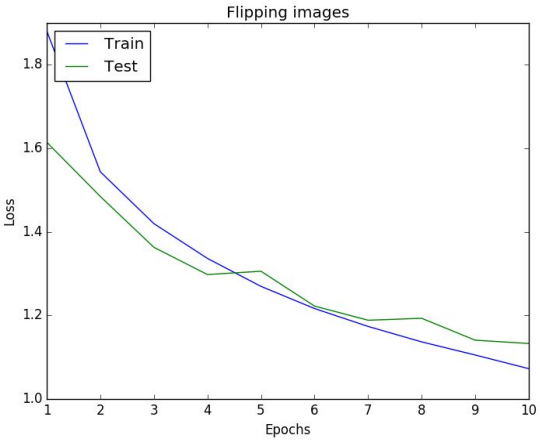


Test loss: 1.17719040623

Test accuracy: 0.5796

2. After adding Flipped images

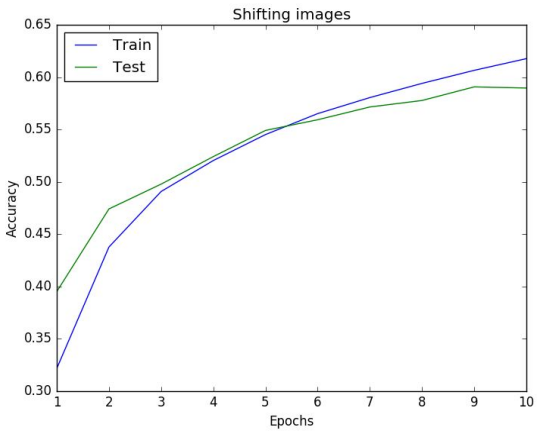


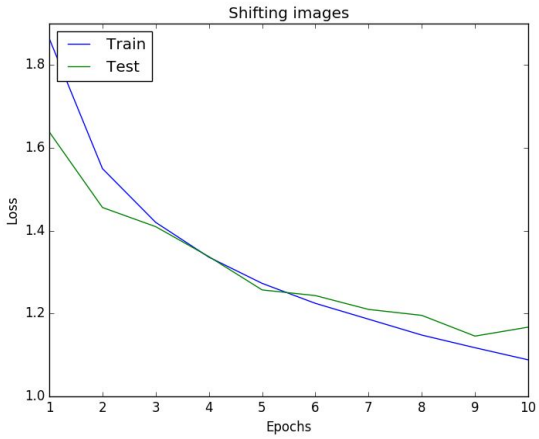


Test loss: 1.13290137653

Test accuracy: 0.6021

3. After adding images with random shifts

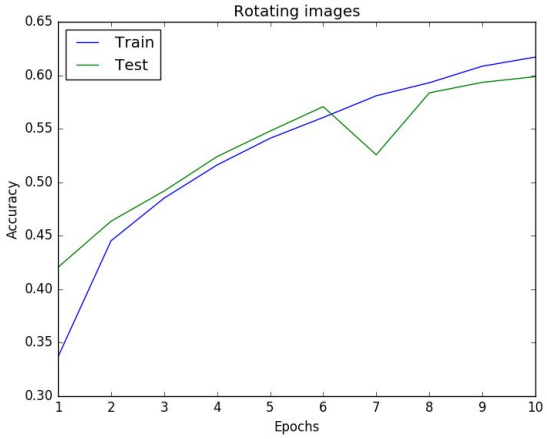


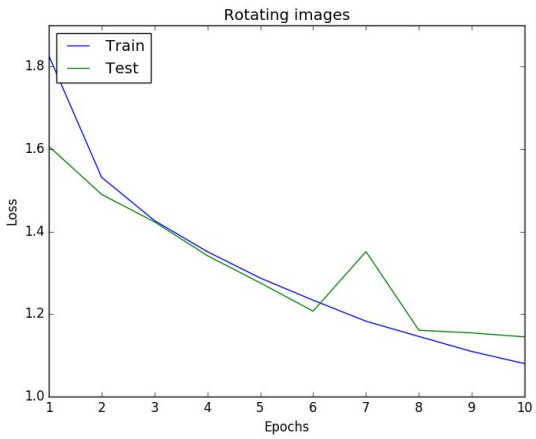


Test loss: 1.16719040623

Test accuracy: 0.5896

4. After adding images with rotations in range (0-90)

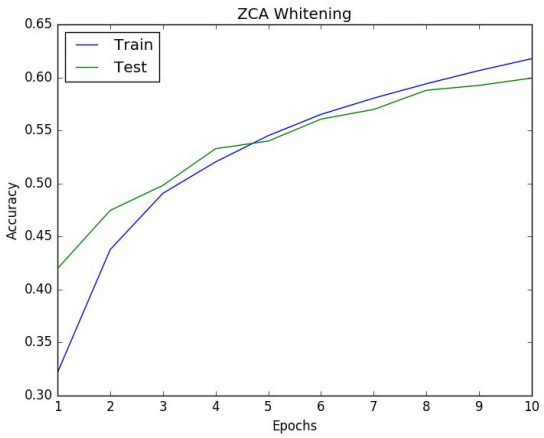


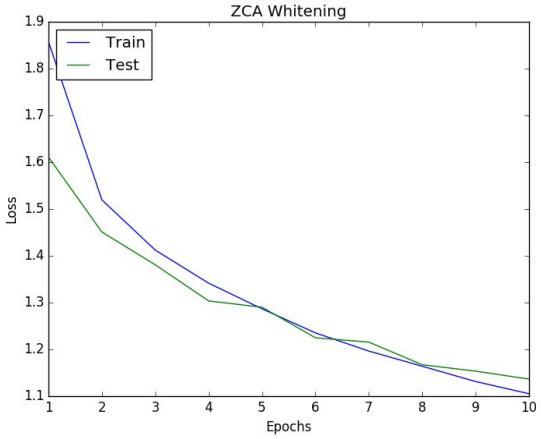


Test loss: 1.14484213276

Test accuracy: 0.5987

5. After adding images with ZCA whitening applied

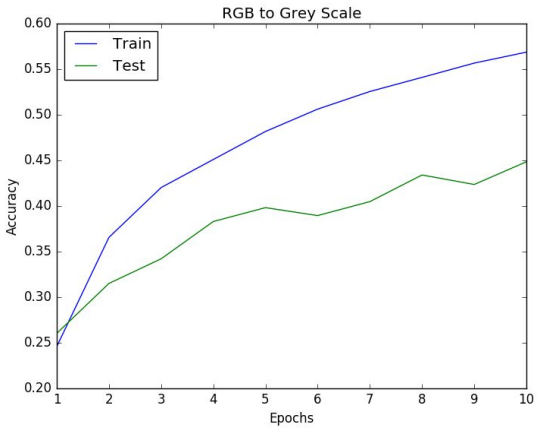


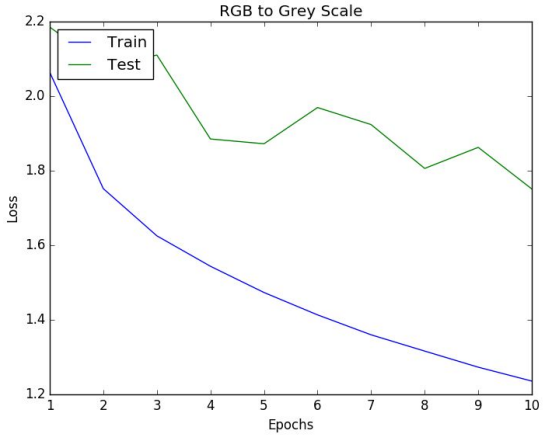


Test loss: 1.13662183189

Test accuracy: 0.5994

6. After adding images with RGB to grayscale conversion





Test loss: 1.75113763161

Test accuracy: 0.4485

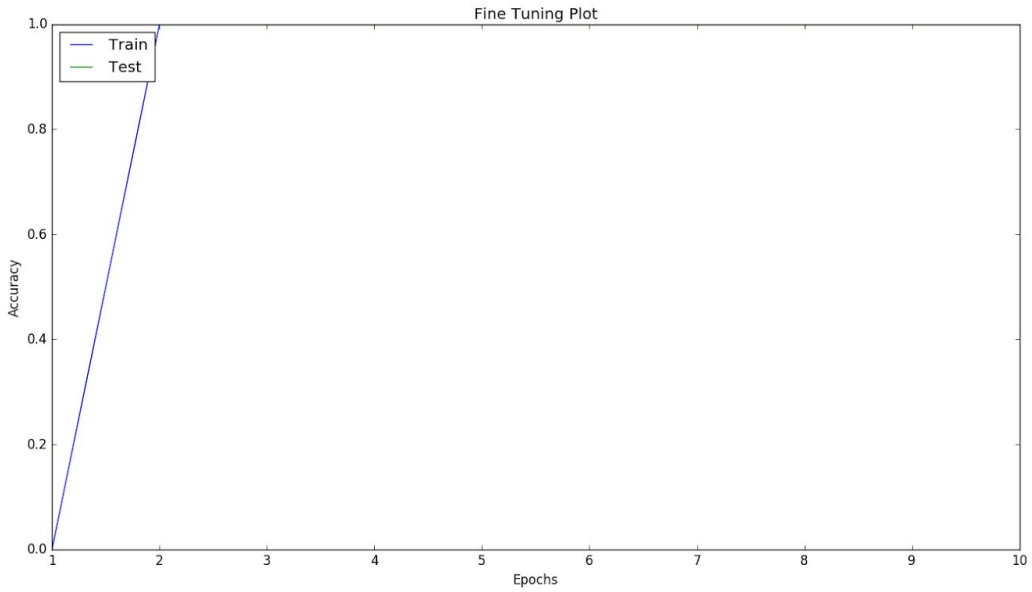
Observations:

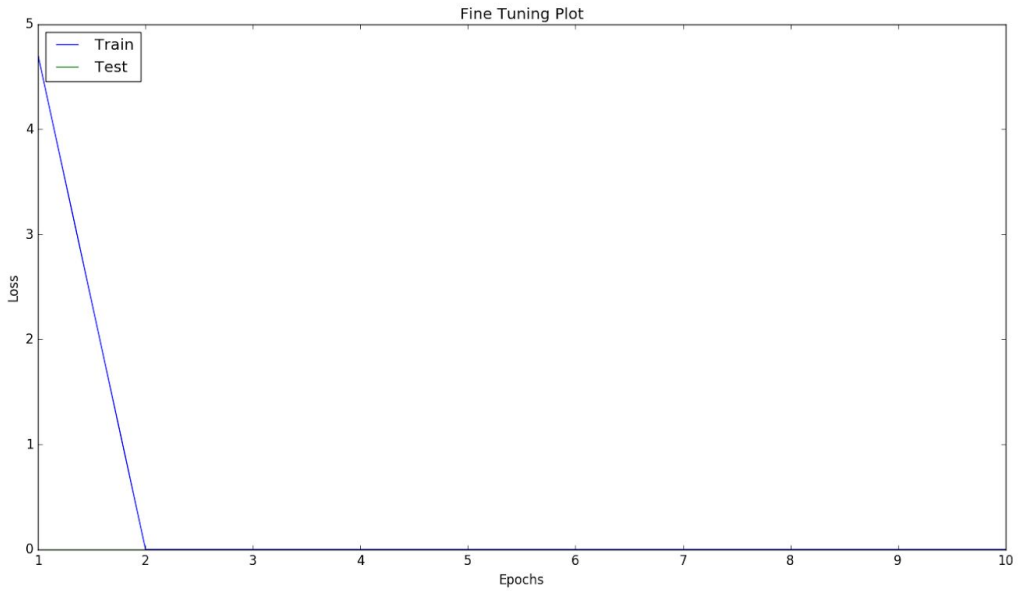
1. In all the cases there was not much difference in accuracy. Also the original data (without adding any synthetic data) had same accuracy.

2. Shifting is also a nice way of generating synthetic data as it might happen that the object of interest is located in some position while it can be learnt better at some other position. So we can generate some data using random shifting.

Q13. Choose a suitable pre-trained network and fine-tune it to a new dataset. Report the accuracy, learning curve plots and numbers along with explanation.

Ans: vgg 16 network's weights are used as pre trained data. The network is a deep netowk with 16 layers (13 Conv + 3 FC). Weight file (vgg16_weights.h5) for the trained model is available easily. Input to this model is 3*224*224. This network is then tuned for pubfig dataset (Public Figures Face Dataset). Pubfig dataset contains 32*32 RGB images. So the images in pubfig were resized by adding zero padding. Tested the network with fewer images (1000 training + 100 test) only otherwise it takes very long time to train (because of too many layers with large number of nodes). The network jumped to accuracy 1 just after the 1st epoch as all the layers except the last were already trained. It seems that there wasn't much change required in them.




Fine Tuning Plot
Train
Test
Loss
Epochs
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
10