Programming Assignment 4

Q2) Test accuracy: 0.9909

Q3) Input===>Conv===>Conv===>Pooling ===>Fully Connected ===>Softmax

First conv layer: Filter size= 3*3 , number of filters=32

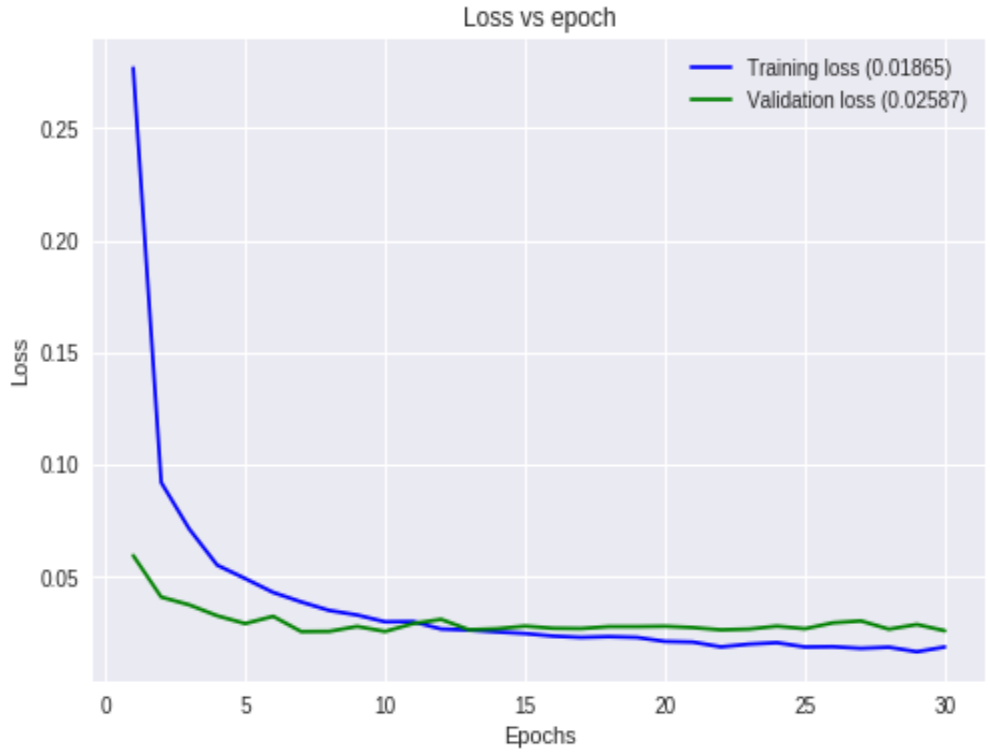
second conv layer: Filter size= 3*3 , number of filters=64

first pooling layer: filter size = 2*2

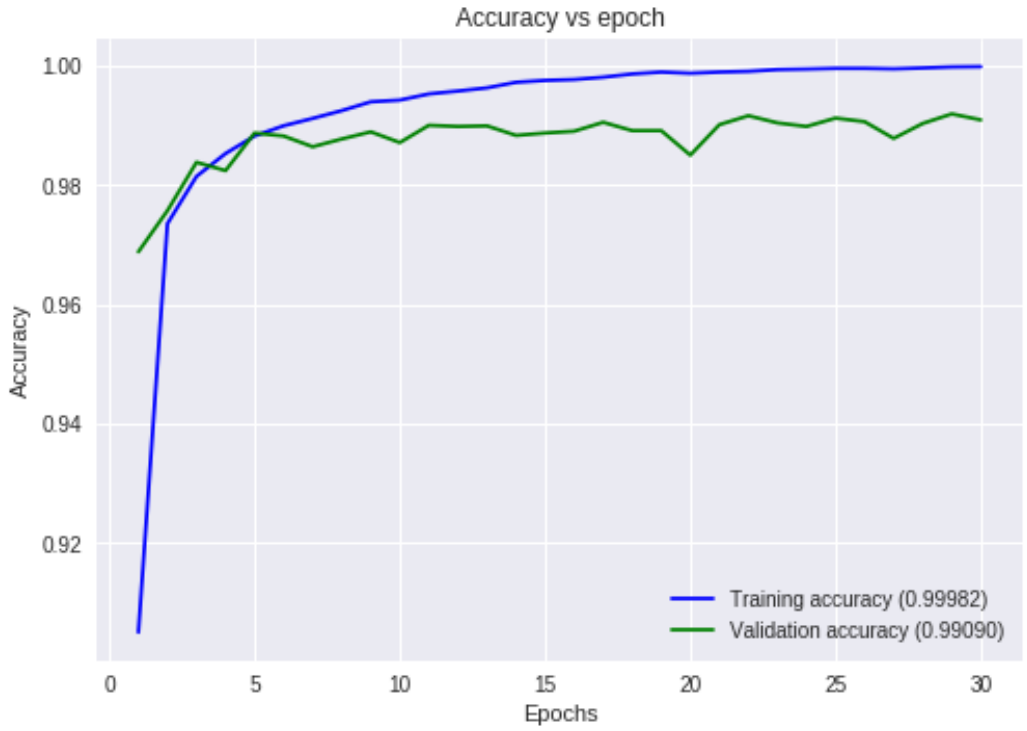
softmax output size: 10

Q4) Epochs: 30

Q5) Training Loss and Test loss vs epochs



Q6) LeNet 5

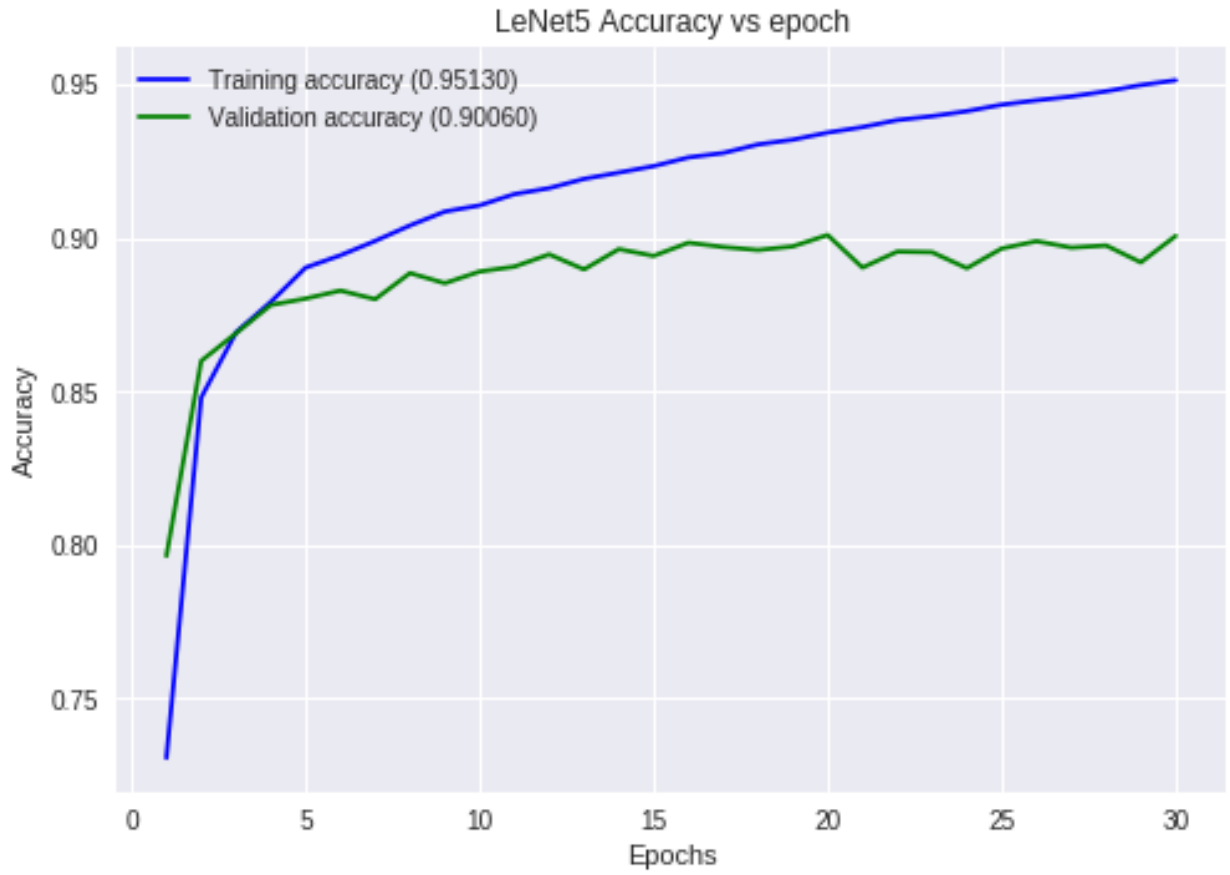


Q7) Fashion MNIST dataset:

LeNet5 model

Average training time per epoch : 5 seconds

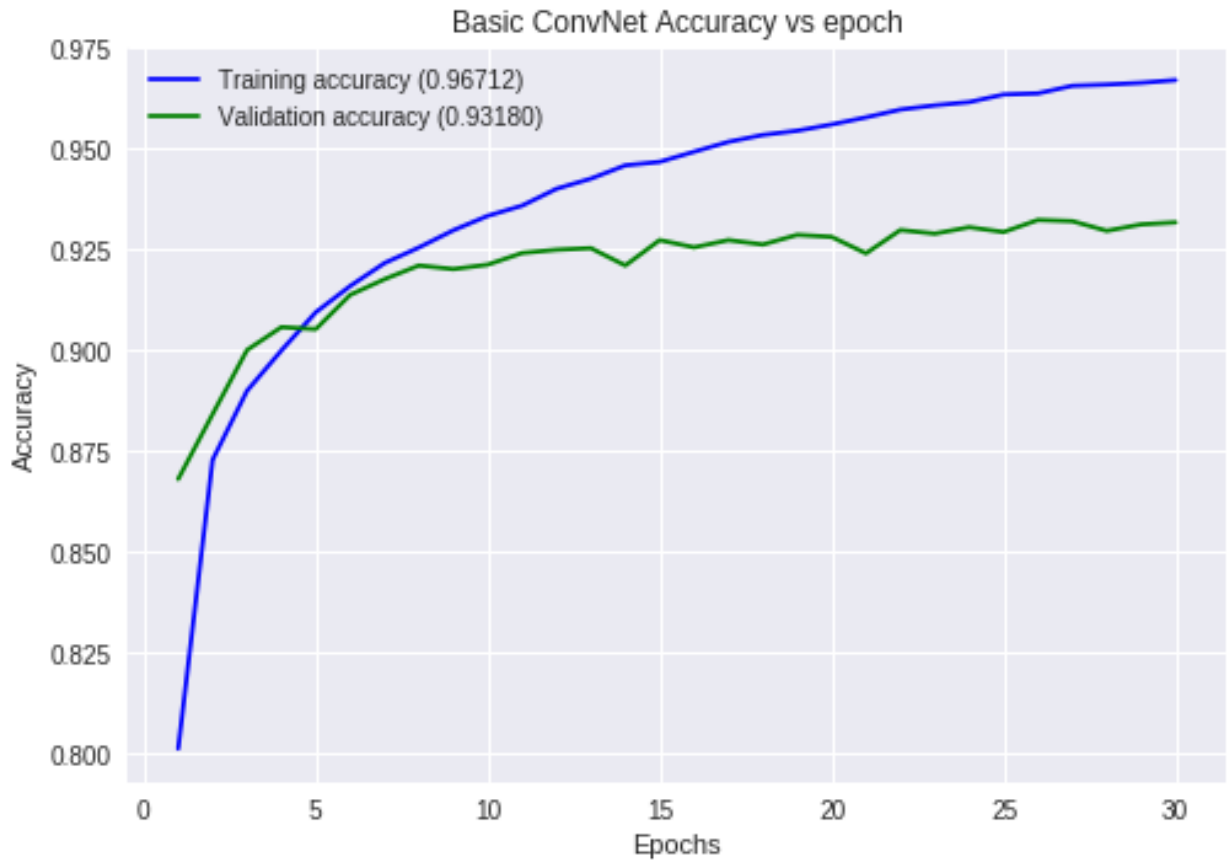
Test accuracy : 0.9006



Basic convnet model

Average training time per epoch : 9 seconds

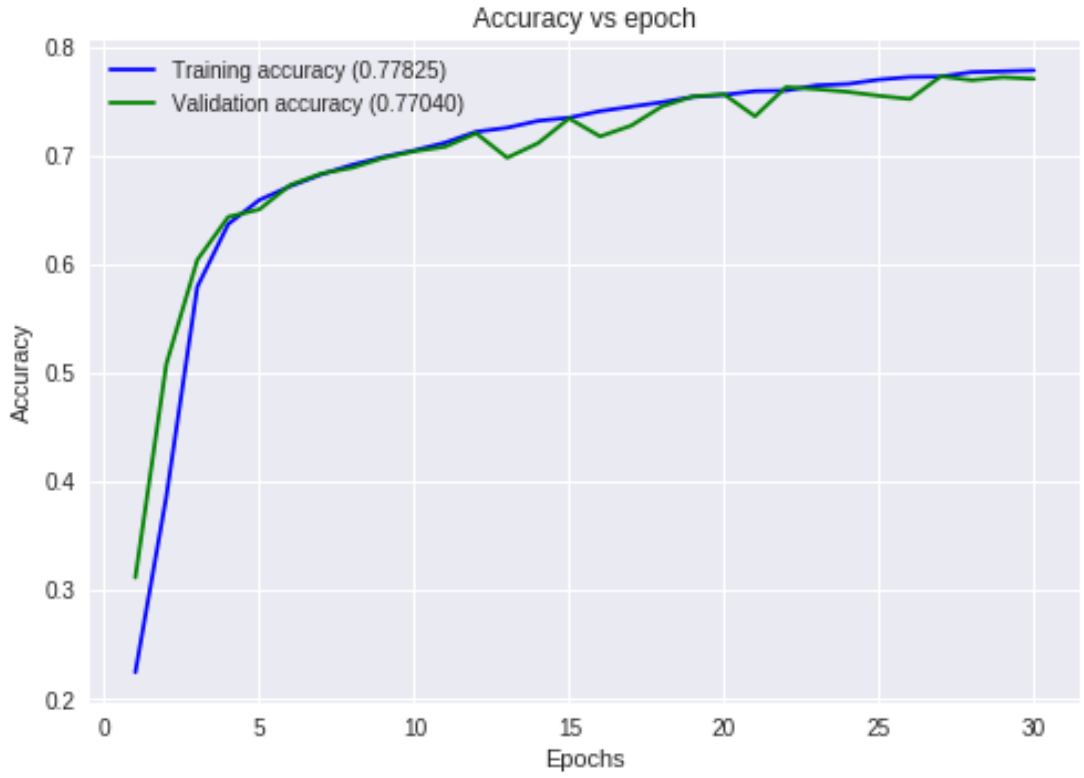
Test accuracy : 0.9294

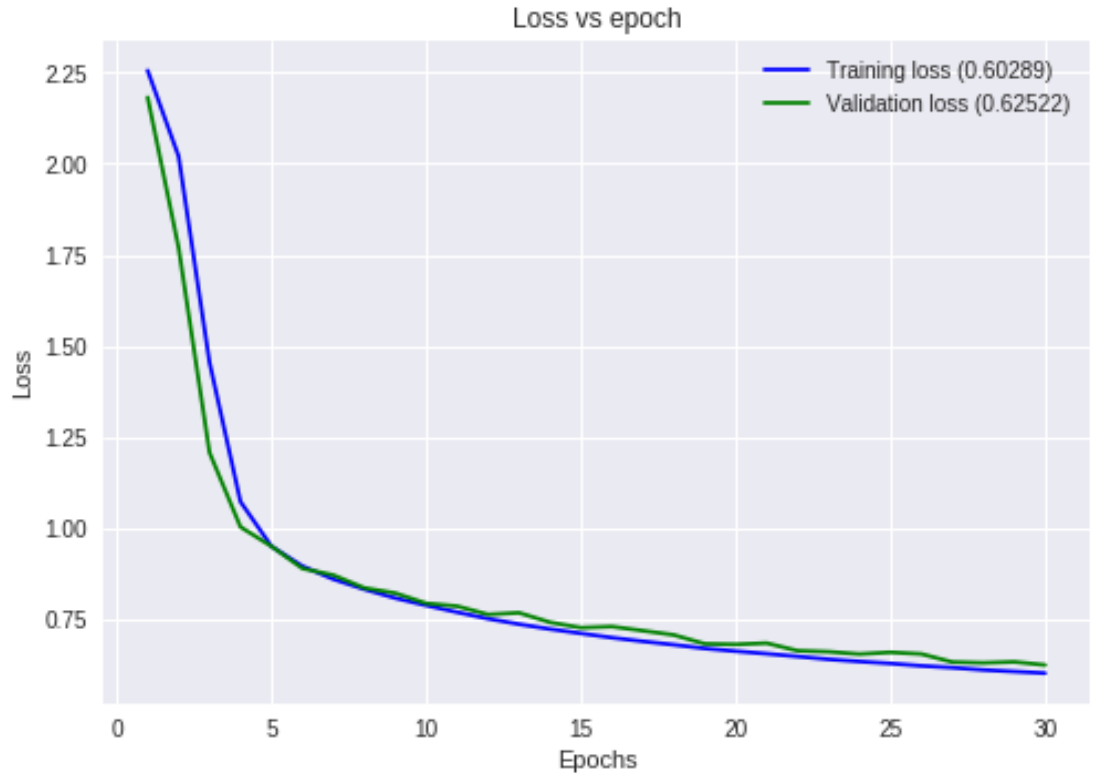


Did using a deeper net help in this case to obtain a higher accuracy?

Ans: No

Q8) Using SGD optimizer

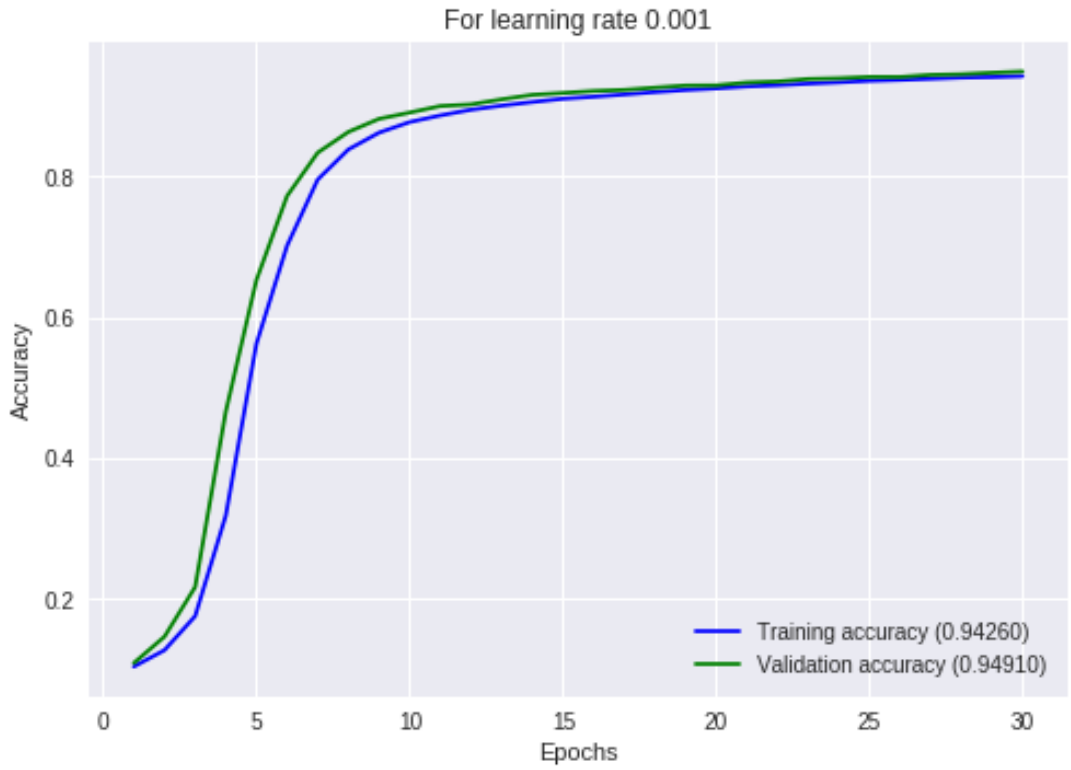




Q9) learning rates = 0.001, 0.01,0.1, 0.0005,0.0001

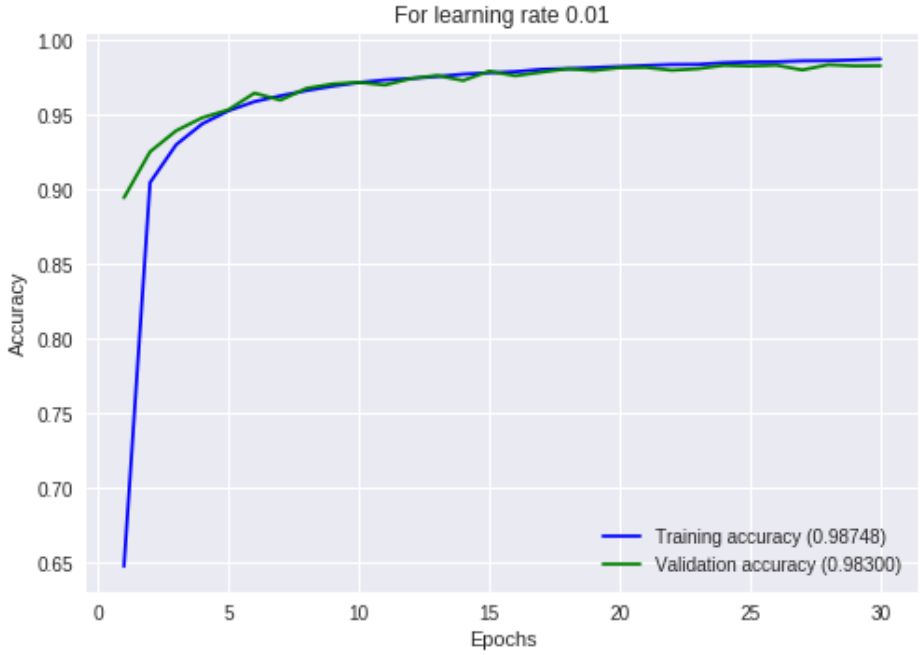
for learning rate : 0.001

Test accuracy : 0.9491



For learning rate : 0.01

Tets accuracy : 0.983

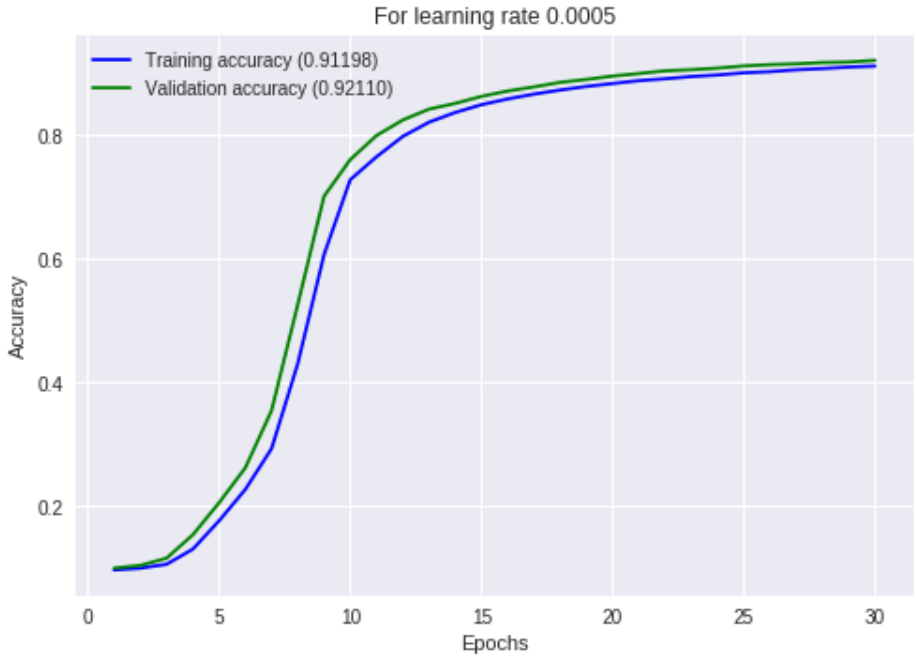


For learning rate : 0.1

Test accuracy : 0.9902

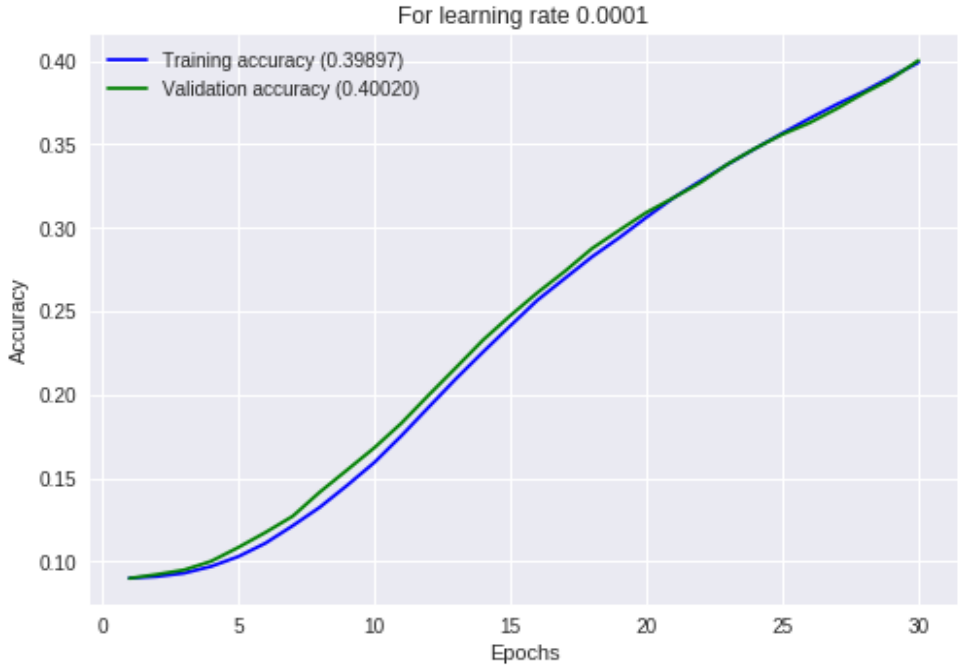
For learning rate : 0.0005

Test accuracy : 0.9211



For learning rate : 0.0001

Test accuracy : 0.4002



Learning rate 0.1 gave the best result out of all the chosen learning rates.

Q10) Using AdaDelta lead to better accuracy than stochastic gradient descent(SGD) in my case.

learning rate 0.001 gave 0.9491 accuracy.

learning rate 0.01 gave 0.983 accuracy.

learning rate 0.1 gave 0.9902 accuracy.

learning rate 0.0005 gave 0.9211 accuracy.

learning rate 0.0001 gave 0.4002 accuracy.

Changing learning rate from 0.001 to 0.01 lead to increase in accuracy. This indicates that model's weights are getting updated faster and the model achieves better accuracy in same number of iterations.

Changing learning rate from 0.01 to 0.1 lead to increase in accuracy.

Changing learning rate from 0.1 to 0.0005 lead to decrease in accuracy.

Changing learning rate from 0.0005 to 0.0001 lead to huge drop in accuracy. The model needs to trained for very large number of iterations to achieve good accuracy.

If the learning rate is low, the optimization will take a lot of time because steps towards the minimum of the loss function are very small. So, it takes long time to converge.

If the learning rate is high, then training may not converge or even diverge. Weight changes can be so big that the optimizer overshoots the minimum and makes the cost worse.