Machine Learning Coursework 1

Question 1:

1.A:

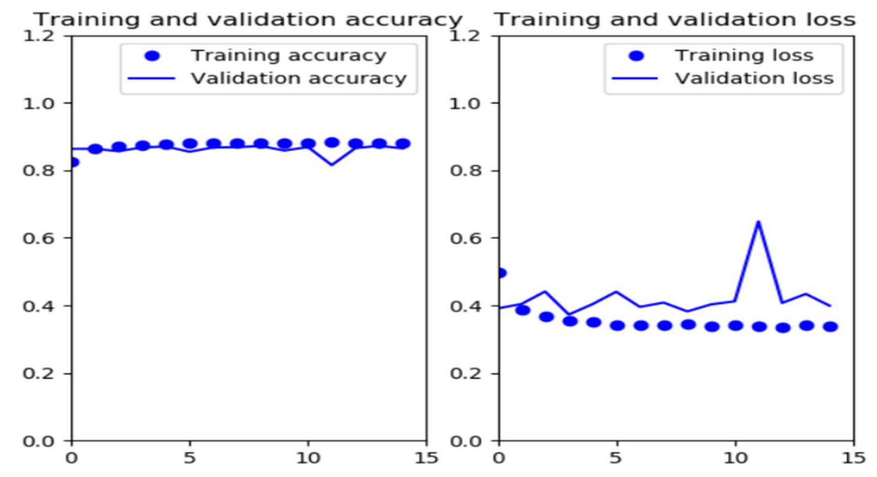


Figure.1.

Graph on left displays Training and validation accuracy, while there an increase in Training accuracy after initial epochs there is a slight dip in validation accuracy after 12th iteration. Graph on right shows Training loss keeps decreasing and attains minimal loss after 13 epochs but in case of validation loss, its value fluctuates after two or three epochs since the number of correct data classifications randomly fluctuates. After 15 epochs a significant difference can be seen between validation and training losses thus stating that both the losses don't converge. Since the validation loss is greater than training loss and difference between validation and training loss is considerable, the network is overfitting.

1.B:

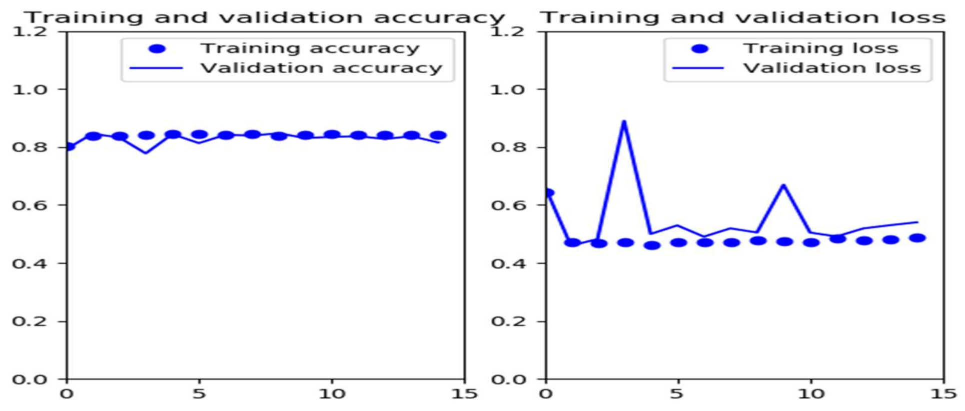


Figure.2: Higher Learning rate = 0.07

Learning rates is an important parameter which states how a model should change for identified error once the weights are updated. For first experiment a Learning rate equal to **0.07** is considered which is on a higher scale. Training accuracy is slightly lower than validation accuracy after 15 epochs. However, due to higher learning rate the model overshoots resulting into fluctuations in validation loss values. The model is thus inefficient to attain good minima.

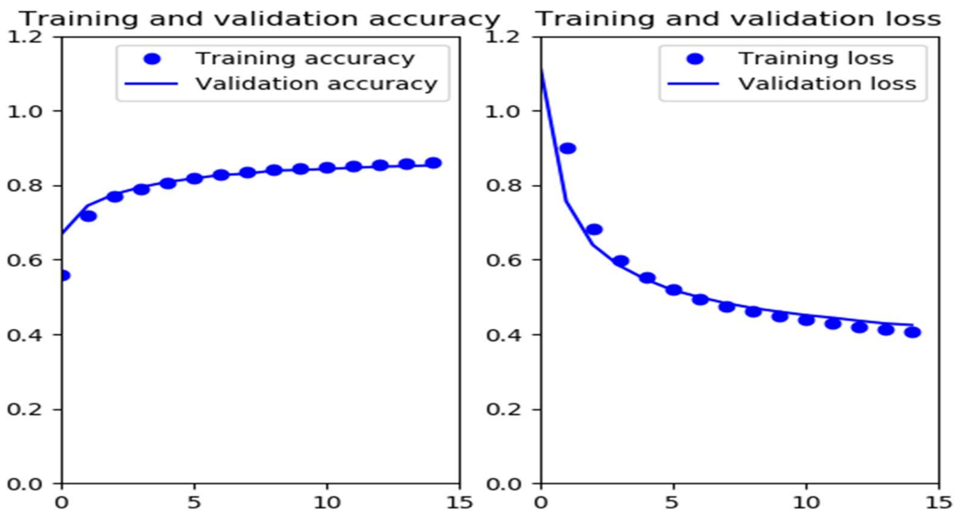


Figure.3: Lowest Learning rate = 0.0001

For second experiment we have considered a low value of Learning rate which is **0.0001**. The training accuracy and validation accuracy is good. Since the learning rate is small the required number of epochs to attain local minima is large. Thus after 15 epochs the validation loss value obtained is around 0.4. If we increase the epochs, we can find the local minimum.

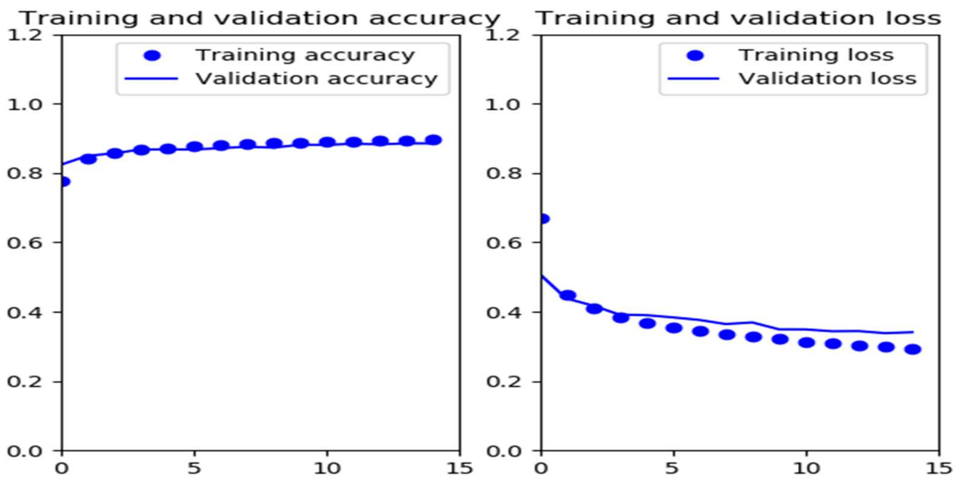


Figure.4: Appropriate Learning rate = 0.001

For the most appropriate case, we have considered learning rate to be **0.001**. As shown is above figure the training and validation accuracy is decent. Also, the validation loss is decreasing with respect to epoch. The difference between the training and validation loss is less compared to first case and the local the validation loss will attain local minima much faster compared to second situation. As the learning rate is suitable, the chance of overfitting is also reduced.

Question 2:

2.A:

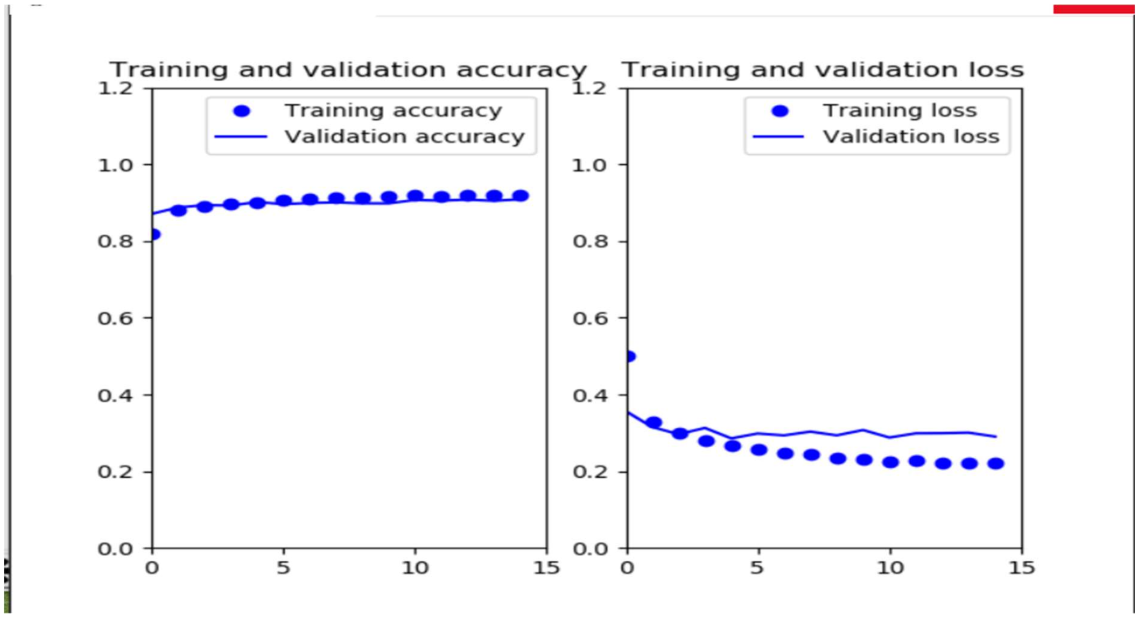
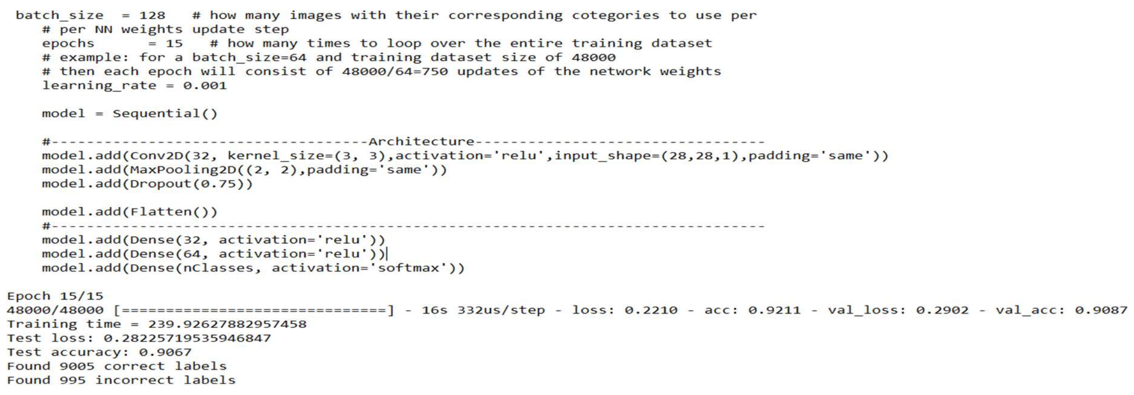


Figure.5: Graph – Design 1

As shown in above graph, training loss of the designed architecture is less than 0.28. In predefined architecture, the training loss was around 0.4 while there were remarkable variations in the validation loss as well. Thus, in newly designed architecture, we have increased the number of filters in Conv2D so that the model learns from an ideal number of filters. Kernel size is a 3 * 3 matrix. Further the max pooling layer of size 2 * 2 is used to decrease the dimensionality. Since in the original architecture overfitting was an issue, a dropout layer with probability of 0.75 is introduced to reduce the effect. Next, we have added a Flatten layer to convert multi-dimensional matrix to single dimensional matrix. Fully connected layer has two dense layers with each having node 32 and 64 respectively and uses a relu function. The output layer has 10 classes and uses 'SoftMax' to define the highest probability for the class. The number of epochs is 15. The learning rate of previous designs tends to overshoot thus we have considered 0.001 to be the appropriate learning rate value. As a result of which we have achieved training loss around 0.22. Respective architecture and results can be seen in below figure.



```
 batch_size  = 128   # how many images with their corresponding cotegories to use per
    # per NN weights update step
    epochs      = 15   # how many times to loop over the entire training dataset
    # example: for a batch_size=64 and training dataset size of 48000
    # then each epoch will consist of 48000/64=750 updates of the network weights
    learning_rate = 0.001

    model = Sequential()

    #------------------------------------Architecture--------------------------------
    model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=(28,28,1),padding='same'))
    model.add(MaxPooling2D((2, 2),padding='same'))
    model.add(Dropout(0.75))

    model.add(Flatten())
    #-------------------------------------------------------------------------------
    model.add(Dense(32, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(nClasses, activation='softmax'))

Epoch 15/15
48000/48000 [==============================] - 16s 332us/step - loss: 0.2210 - acc: 0.9211 - val_loss: 0.2902 - val_acc: 0.9087
Training time = 239.92627882957458
Test loss: 0.28225719535946847
Test accuracy: 0.9067
Found 9005 correct labels
Found 995 incorrect labels
```

Figure.6: Architecture – Design 1

2.B:

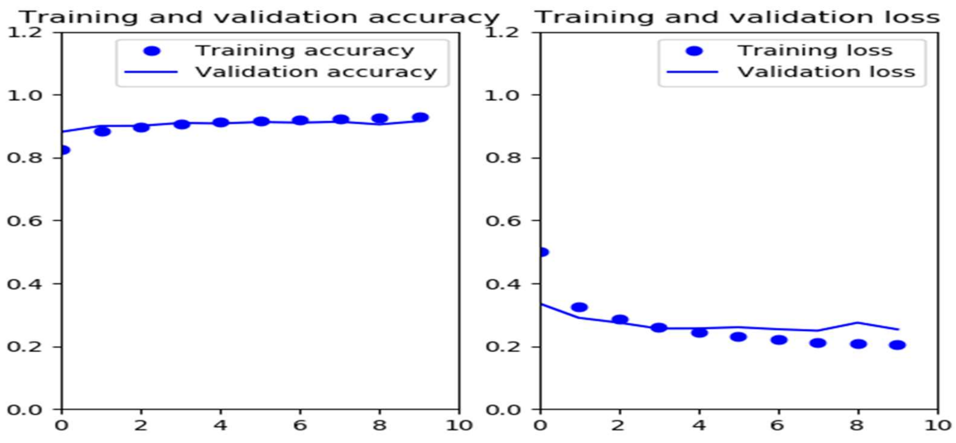
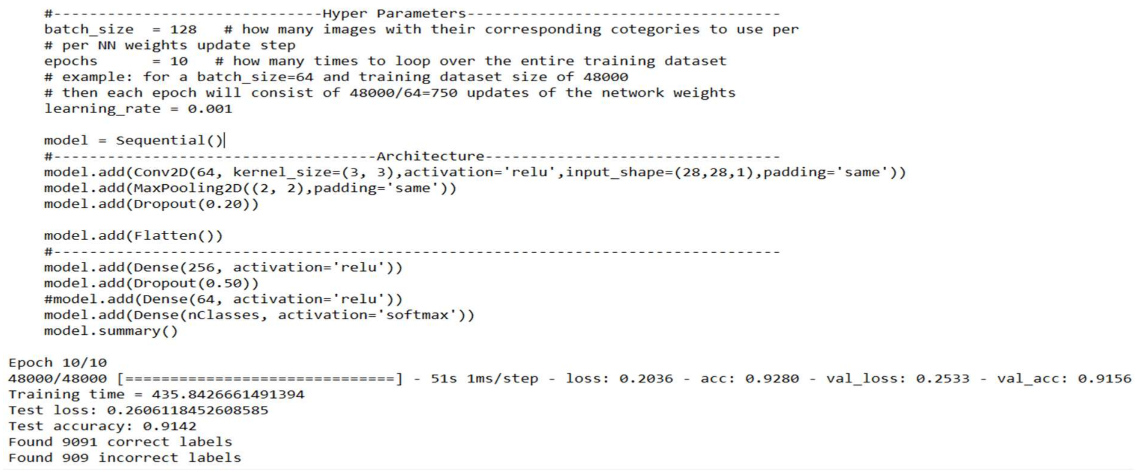


Figure.7: Graph - Design 2

As displayed in above graph, validation loss for the designed architecture is less than 0.27. Unlike the previous design, this architecture has a single Conv2D layer with 64 filters. The kernel size is same which is 3 * 3 matrix. A max pooling layer with dimension 2 * 2 matrix is used to decrease the size of convolutional layer. Since we have increased the number of layers in convolutional layer, dropout rate of 0.75 was resulting into underfitting of the designed model. Hence a drop out layer with appropriate value of .20 is selected considering the model won't overfit as well as doesn't underfit. Next, we have added a Flatten layer to convert multi-dimensional matrix to single dimension array. In order to increase the accuracy, we have added dense layer with 256 filters with a relu function. With such high amount of neuron's in dense layer, there was high chance the model would overfit. Thus, to avoid the situation we have added a drop out layer with a higher probability of 0.50. The output layer has 10 classes and uses 'SoftMax' to define the highest probability for the class. We executed the above architecture with 10 epochs, but we can observe slight increase in validation loss after 8th epochs before it reaches 0.25 after 10 epochs. Learning rate is same as the previous architecture i.e. 0.001. Once the model is trained, it provides with a validation loss around

0.25 and validation accuracy of 92%. Respective architecture and results can be seen in below figure.



```
    #------------------------------Hyper Parameters-----------------------------------
    batch_size  = 128   # how many images with their corresponding cotegories to use per
    # per NN weights update step
    epochs      = 10   # how many times to loop over the entire training dataset
    # example: for a batch_size=64 and training dataset size of 48000
    # then each epoch will consist of 48000/64=750 updates of the network weights
    learning_rate = 0.001

    model = Sequential()
    #------------------------------------Architecture---------------------------------
    model.add(Conv2D(64, kernel_size=(3, 3),activation='relu',input_shape=(28,28,1),padding='same'))
    model.add(MaxPooling2D((2, 2),padding='same'))
    model.add(Dropout(0.20))

    model.add(Flatten())
    #---------------------------------------------------------------------------------
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.50))
    #model.add(Dense(64, activation='relu'))
    model.add(Dense(nClasses, activation='softmax'))
    model.summary()

Epoch 10/10
48000/48000 [==============================] - 51s 1ms/step - loss: 0.2036 - acc: 0.9280 - val_loss: 0.2533 - val_acc: 0.9156
Training time = 435.8426661491394
Test loss: 0.2606118452608585
Test accuracy: 0.9142
Found 9091 correct labels
Found 909 incorrect labels
```

Figure.8: Architecture - Design 2

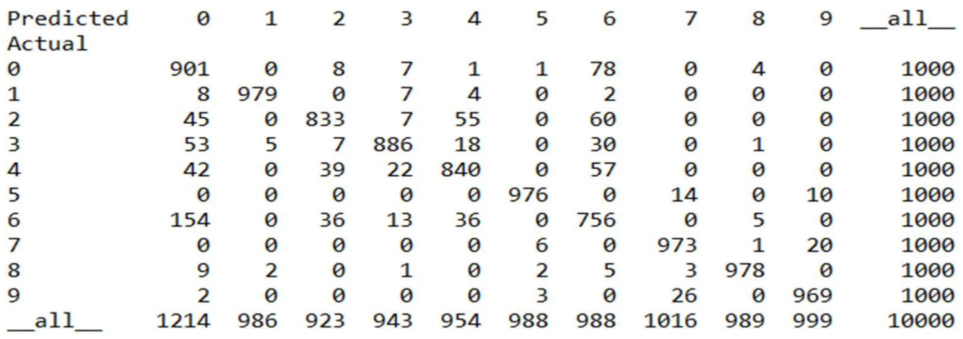
2.C:

	Test Loss	Test accuracy	Correct labels	Incorrect labels
Architecture 1	0.28	0.90	9005	995
Architecture 2	0.26	0.92	9091	909

Architecture 1 uses a single convolutional layer with 32 filter size which results in training loss of 0.22 and validation loss of 0.29. The test loss is 0.28 which is satisfactory, the results for test accuracy is great which is around 90 %. It identifies 9005 test labels correctly as compared to 995 incorrect test labels.

Architecture 2 is using a convolutional layer with 64 filters. As a result, compared to first architecture training loss and validation loss is reduced to 0.20 and 0.25 respectively. Further, test loss is decreased to 0.26 and test accuracy is increased to 92%. We can observe a growth in correct label classification and reduction in incorrect label classification. After comparing both the architecture’s, we can conclude architecture 2 is more efficient.

2.D.



```
Predicted     0    1    2    3    4    5    6     7    8    9  __all__
Actual
0           901    0    8    7    1    1   78     0    4    0     1000
1             8  979    0    7    4    0    2     0    0    0     1000
2            45    0  833    7   55    0   60     0    0    0     1000
3            53    5    7  886   18    0   30     0    1    0     1000
4            42    0   39   22  840    0   57     0    0    0     1000
5             0    0    0    0    0  976    0    14    0   10     1000
6           154    0   36   13   36    0  756     0    5    0     1000
7             0    0    0    0    0    6    0   973    1   20     1000
8             9    2    0    1    0    2    5     3  978    0     1000
9             2    0    0    0    0    3    0    26    0  969     1000
__all__    1214  986  923  943  954  988  988  1016  989  999    10000
```

Confusion Matrix

Considering the above confusion matrix, we can conclude class 1 which represents 'Trousers', is easily classified. The total correct classification is 979 while incorrect classifications counts to 21.
On the contrary, class 6 is most difficult to classify. It correctly classifies 756 'Shirts' but misclassifies around 244 objects. Most of the misclassified objects belong to class 0 which defines T-shirt/top.