Good report. No need for re-submission.
The comments I left are for you to think about.

TBMI26 – Computer Assignment Reports Deep Learning

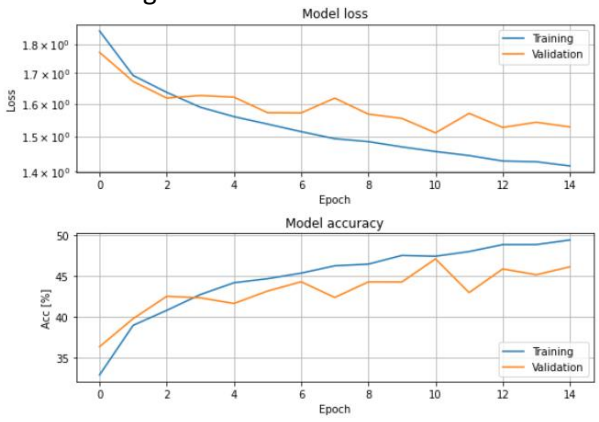
Deadline – March 14 2021

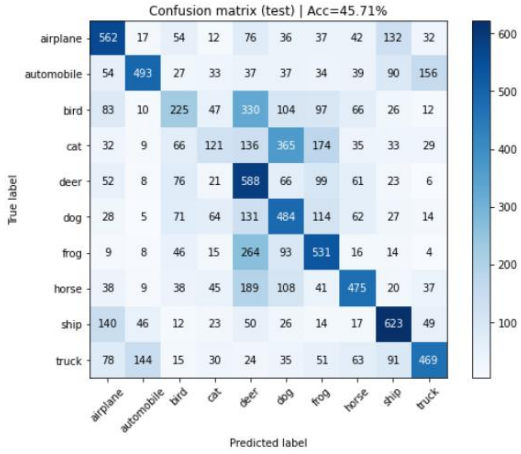
Shwetha Vandagadde Chandramouly(shwva184)
Suhani Ariga(suhar073)

In order to pass the assignment you will need to answer the following questions and upload the document to LISAM. Please upload the document in PDF format. **You will also need to upload the Jupyter notebook as an HTML-file (using the notebook menu: File -> Export Notebook As…)**. We will correct the reports continuously so feel free to send them as soon as possible. If you meet the deadline you will have the lab part of the course reported in LADOK together with the exam. If not, you'll get the lab part reported during the re-exam period.

1. **The shape of X_train and X_test has 4 values. What do each of these represent?**
   The shape of X has 4 values, 1st represents the number of observations ie color images. 2nd and 3rd represents image height and width of the image and 4th value represents the color channels (R G B).

2. **Train a Fully Connected model that achieves above 45% accuracy on the test data. Provide a short description of your model and show the evaluation image.**
   To Obtain accuracy around 45% in test data, we have used 3 dense layers with relu ac´s activation function for hidden layer and softmax as activation function for output layer.
   Number of neurons in hidden layer are as follows.
   1st : 128
   2nd : 64
   3rd : 10
   We obtained around 45.7% of test accuracy.
   Evaluation image :



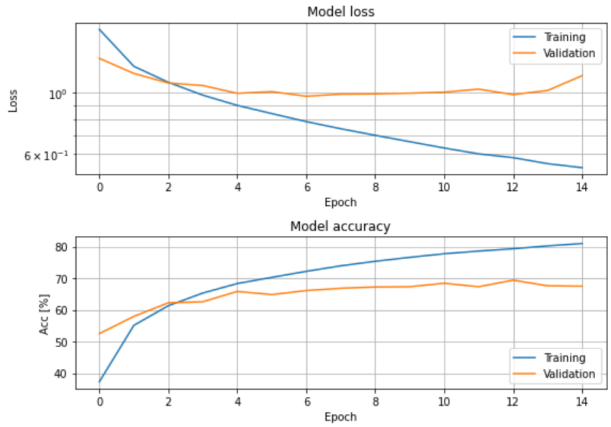


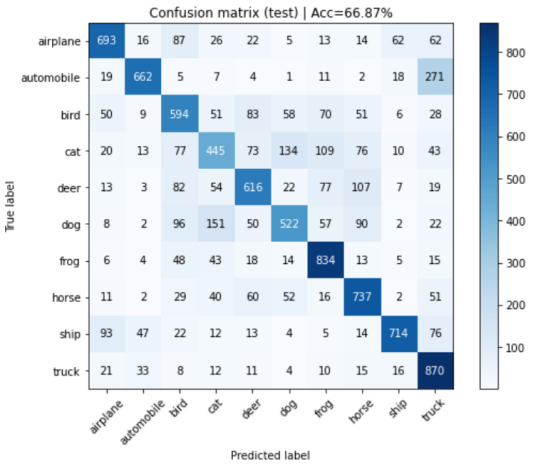
3. **Compare the model from Q2 to the one you used for the MNIST dataset in the first assignment, in terms of size and test accuracy. Why do you think this dataset is much harder to classify than the MNIST handwritten digits?**

The classification is much harder than previous MNIST handwritten digits as these are colored images , also when it comes to writing digits , the variety of ways one might write a number is not as wide as the variety of ways a picture of dog or airplane can be. Here it is harder for the model to learn these pattern , hence we will need complex model.
In MNSIT data , with just a couple of dense layers we were able to get 97% of accuracy , but with CIFAR even with 5 layers we have achieved only about 45% of accuracy of test data.

4. **Train a CNN model that achieves at least 62% test accuracy. Provide a short description of your model and show the evaluation image.**

To train CNN model we have stacked 3 blocks of the form **[convolution - activation - pooling].** This is followed by a fully connected network of two layers. In all the hidden layers including the convolution layer we have used relu as our activation function to avoid any occurrence of vanishing gradient problem. In the output layer we have used softmax as our activation function. With this model we have obtained test accuracy of 66.87% , however we can observe overfitting on the training data in this model.
Evaluation image :





5. **Compare the CNN model with the previous Fully Connected model. You should find that the CNN is much more efficient, i.e. achieves higher accuracy with fewer parameters. Explain in your own words how this is possible.**

In case of fully connected model, we had 402,250 parameters and obtained test accuracy of approximately 45%. Now in our current CNN model, we have just 73,418 parameters but have obtained accuracy of approximately 67%. This improvement in the accuracy with lesser parameters is because of the addition of the convolution layers, the image features such as locality (local features) and shift variance (independent on position) has been utilized by the convolution network but not it is not done but the naive approach of the fully connected model. We also have included pooling layers in CNN block. These are used to reduce the dimensions of the feature space. Thus, it reduces the number of parameters to learn and we do not see in this naive approach.

6. **Train the CNN-model with added Dropout layers. Describe your changes and show the evaluation image.**

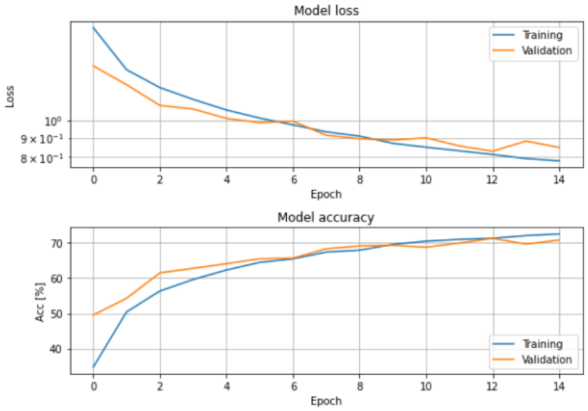
As mentioned earlier we do observe overfitting in our CNN model, adding dropout layers randomly turns off a portion of output from nodes in hidden layer this should take care of overfitting problem. For the same CNN model, two drop put layers have been added. This

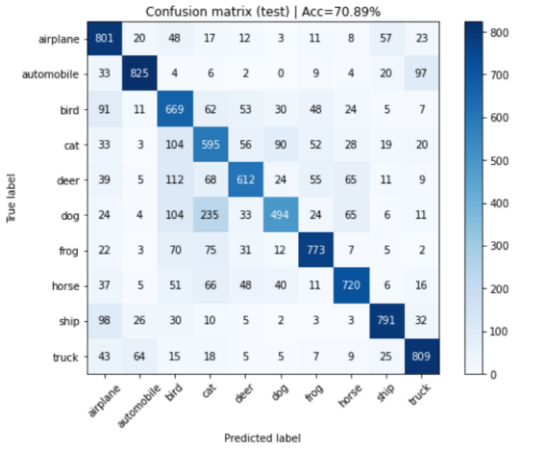
has increased our result to 70 % accuracy and more importantly here we do not see any overfitting on the training data.

```
x_in = Input(shape=X_train.shape[1:])

# === Add your code here ===
x = Conv2D(32,(3,3), activation = 'relu' , input_shape=(32, 32, 3))(x_in)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(64, (3, 3), activation='relu')(x)
x = MaxPooling2D((2, 2))(x)
x = Dropout(0.1)(x)
x = Conv2D(64, (3, 3), activation='relu')(x)
x = MaxPooling2D((2, 2))(x)
x = Dropout(0.2)(x)
x = Flatten()(x)
x = Dense(64, activation='relu')(x)
x = Dense(10, activation='softmax')(x)
# ==========================

model = Model(inputs=x_in, outputs=x)
```





7. Compare the models from Q4 and Q6 in terms of the training accuracy, validation accuracy, and test accuracy. Explain the similarities and differences (remember that the only difference between the models should be the addition of Dropout layers).
Hint: what does the dropout layer do at test time?

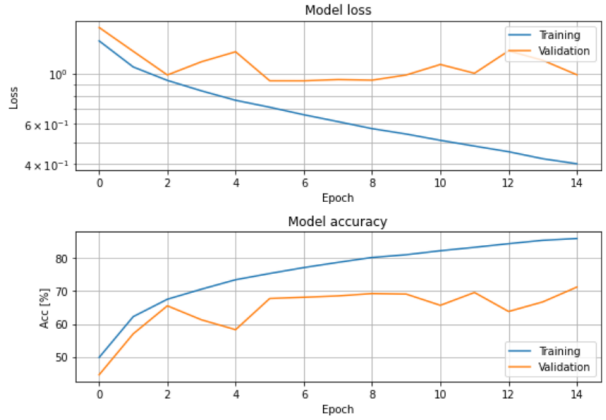
The models in Q4 and Q6 are exactly same except for the dropout layer which has been added in Q6 model.

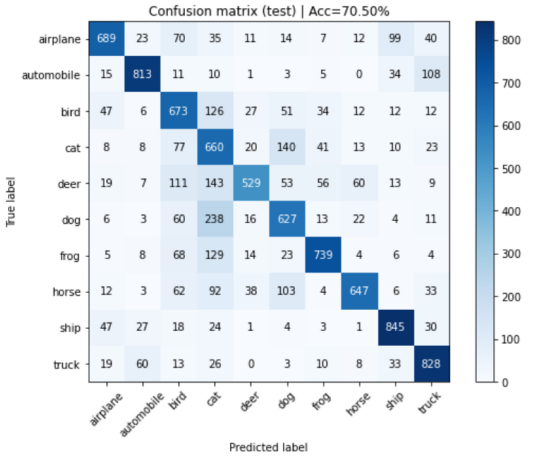
In the model mentioned in Q4 we can clearly observe overfitting on training data, this is evident in the evaluation loss graph as we can see that the training loss seems to keep reducing, but not the validation loss, also we can observe around 15% gap in training (approximately 81%) and test validation accuracy (approximately 66).

But for the same network on adding a couple of dropout layer in convolution stack, we see the improvement in the model, we get around 70 to 72 % accuracy in all data (test, train and validation), this is because adding the dropout layer randomly switches off a portion of output from its previous layer this prevents the model from relying on each parameter thus preventing overfitting.

8. Train the CNN model with added BatchNorm layers and show the evaluation image.

It is always ideal to have input centered around zero and has uniform range, but during training when we update parameters in one layer, it changes the input for the next layer , to avoid this we can perform batch normalization after each layer.

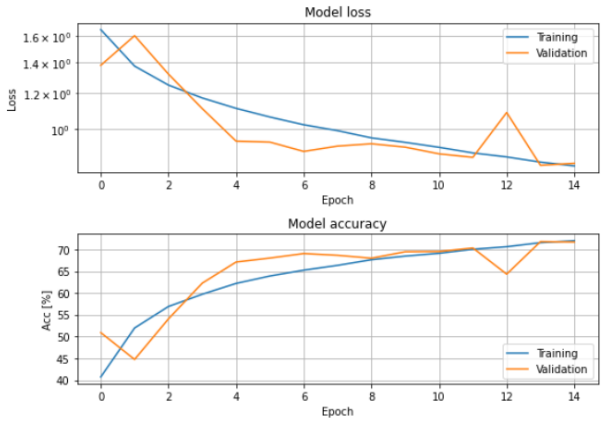


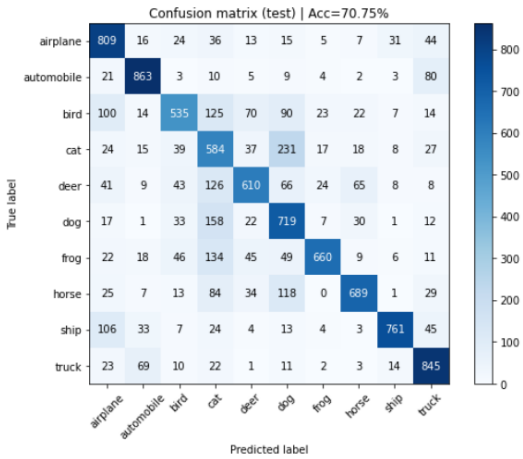


We again see the problem with overfitting here, that is because we have removed the dropout layer to see the influence of batch normalization. On adding dropout layers, we can easily take care of this issue.

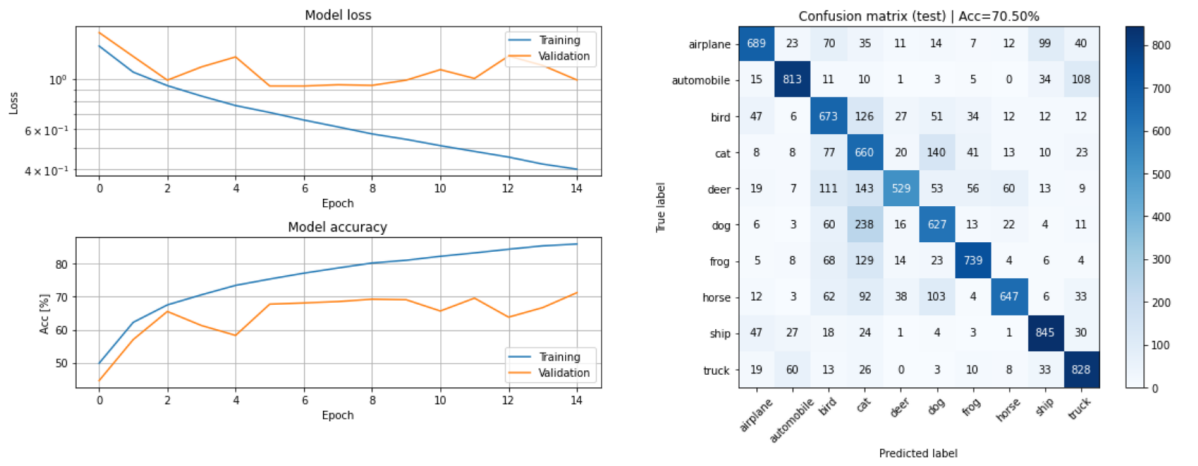
9. When using BatchNorm one must take care to select a good minibatch size. Describe what problems might arise if the wrong minibatch size is used.
You can reason about this given the description of BatchNorm in the Notebook, or you can search for the information in other sources. Do not forget to provide links to the sources if you do!

Our observation on batchsize is as follows , when we choose large number of batches , then each batch will contain small amount of data , this computes faster , but however the accuracy rate is not so great , this is because the small batches are not able to represent the data well. When we choose batch size as a small number , we seem to get better accuracy , but however if we choose a very small number , say just 5 batches in our case , it becomes computationally heavy and takes a lot of time to train.
Few of the batch sizes we have tried out , we have kept momentum as 0.75 for all these.
Batch size = 5 :

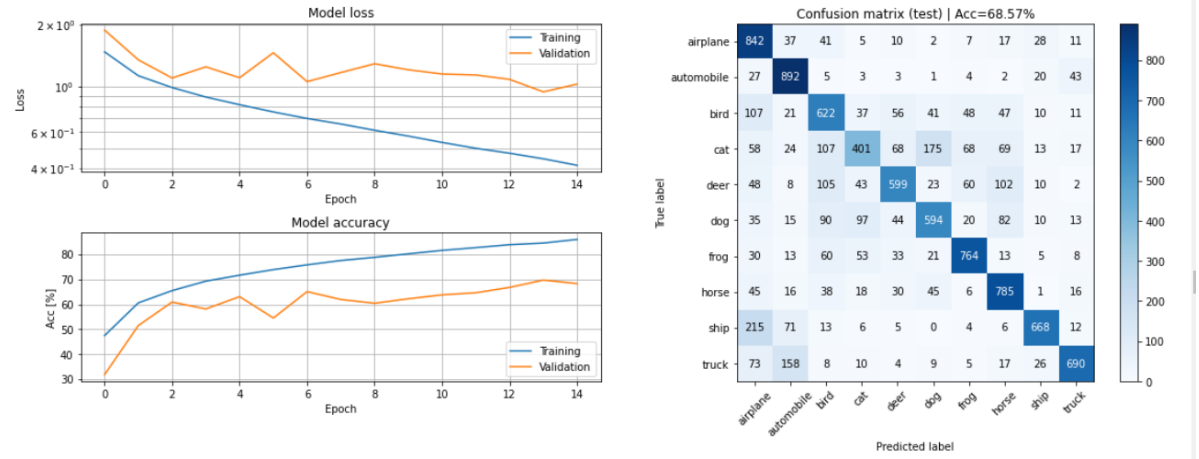




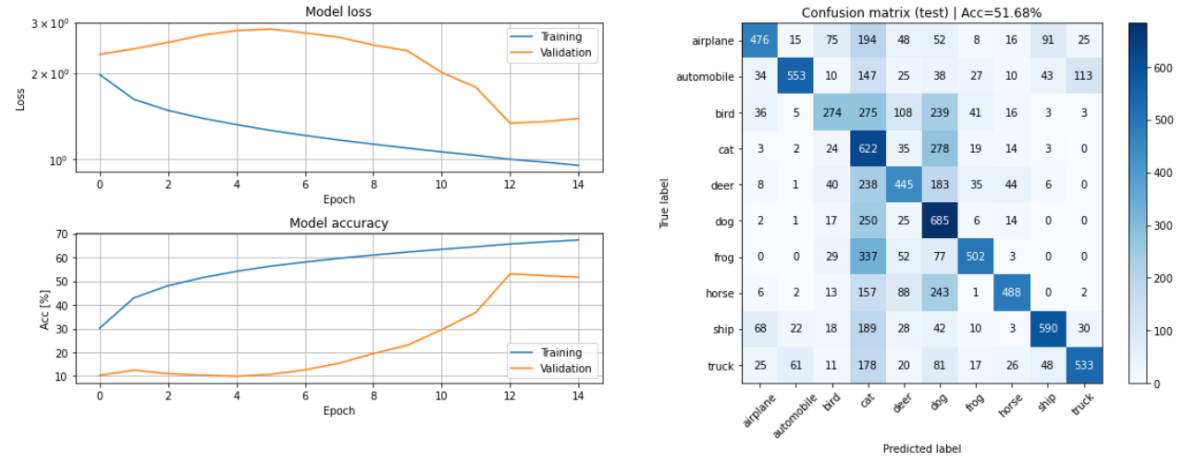
Batchsize = 32 :



batch size 100 :



Batch size : 1000



10. Design and train a model that achieves at least 75% test accuracy in at most 25 epochs. Explain your model and motivate the design choices you have made and show the evaluation image.

To achieve 75% of test accuracy , we used the combination of all the things we experimented till now. We kept the same stack of convolution blocks that we had in Q4 , to make it more efficient and to battle with overfitting problem , we added dropout layers and also batch

normalization after each convolution and dense layers to make sure inputs to all layers are centered around zero and normalized. Our evaluation image looks like this.