Deep Learning Assigment0 Report

PART II

1. Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? Provide the main statistics about the entries of the dataset (mean, std, number of missing values, etc.)

This is a dataset of the Traffic Collision in Los Angeles, in a different part of the city. It mainly describes data about the age group, descent and location of where the collision occurs. It also collects the time it occurs.

Main Statistics:

```
The mean values:
DR Number             1.612149e+08
Time Occurred         1.352427e+03
Area ID               1.107458e+01
Reporting District    1.153360e+03
Crime Code            9.970000e+02
Victim Age            4.138902e+01
Premise Code          1.024317e+02
dtype: float64


The median values:
DR Number             161213495.0
Time Occurred              1430.0
Area ID                      11.0
Reporting District         1162.0
Crime Code                  997.0
Victim Age                   38.0
Premise Code                101.0
dtype: float64
```

The mode values:

	DR Number	Time Occurred	Area ID	Reporting District	Crime Code	\
0	100100007	1800.0	12.0	645.0	997.0	
1	100100767	NaN	NaN	NaN	NaN	
2	100100831	NaN	NaN	NaN	NaN	
3	100101004	NaN	NaN	NaN	NaN	
4	100101322	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	
619942	252104066	NaN	NaN	NaN	NaN	
619943	252104067	NaN	NaN	NaN	NaN	
619944	252104070	NaN	NaN	NaN	NaN	
619945	252104071	NaN	NaN	NaN	NaN	
619946	252104076	NaN	NaN	NaN	NaN	

	Victim Age	Premise Code
0	30.0	101.0
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN
...	...	...
619942	NaN	NaN
619943	NaN	NaN
619944	NaN	NaN
619945	NaN	NaN
619946	NaN	NaN

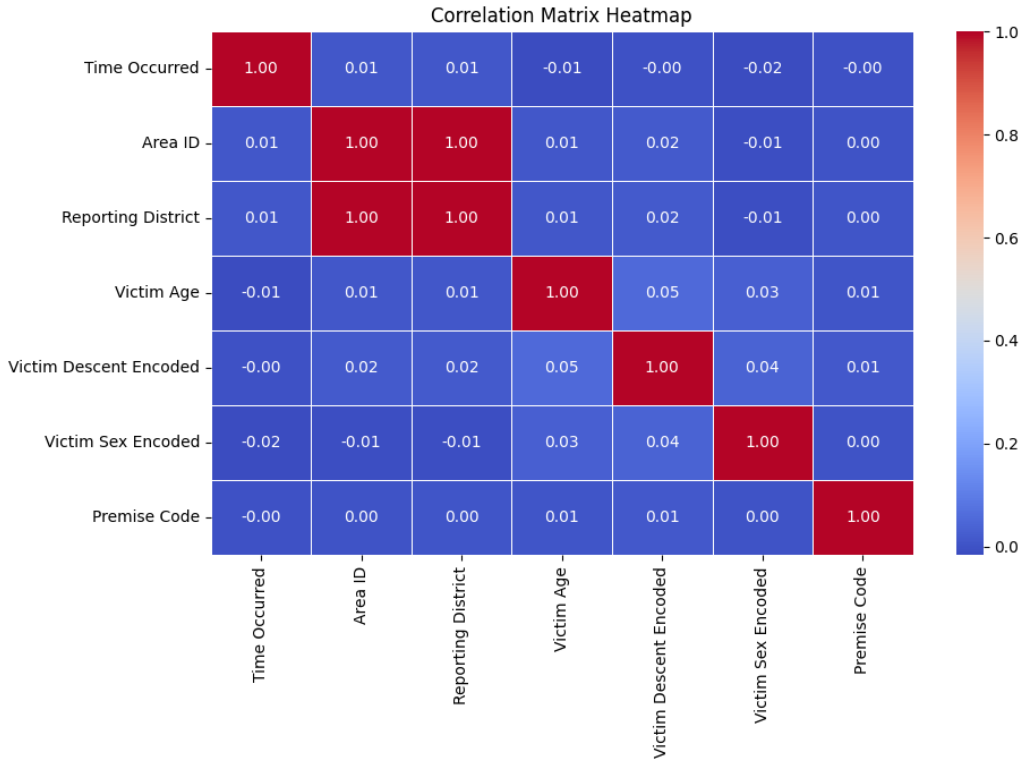
[619947 rows x 7 columns]

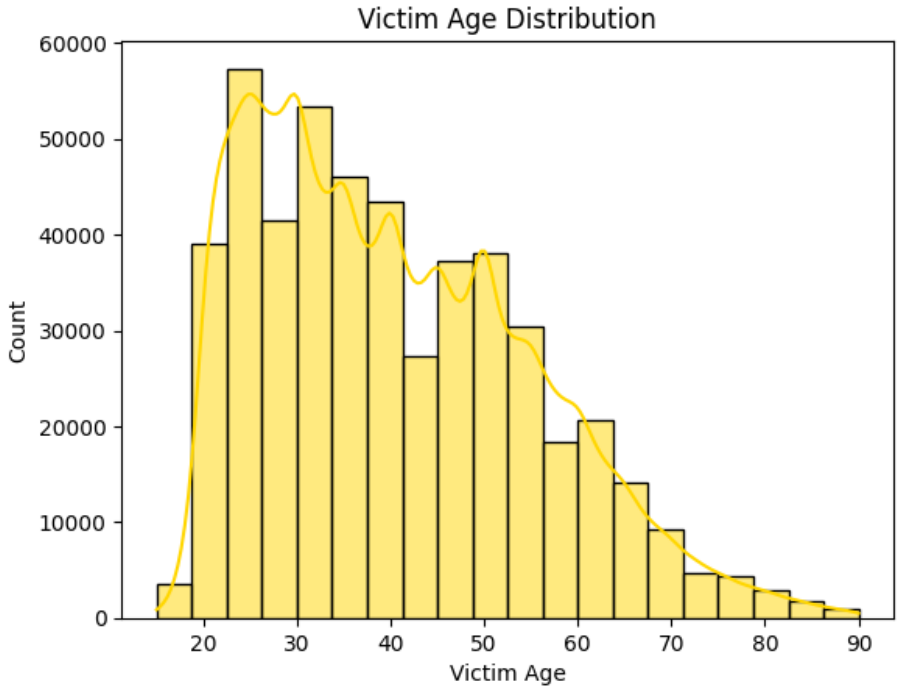
```
The standard deviation values:
DR Number             3.729467e+07
Time Occurred         6.053523e+02
Area ID               5.883969e+00
Reporting District    5.895246e+02
Crime Code            0.000000e+00
Victim Age            1.672163e+01
Premise Code          2.354118e+01
dtype: float64


The variance values:
DR Number             1.390892e+15
Time Occurred         3.664514e+05
Area ID               3.462109e+01
Reporting District    3.475393e+05
Crime Code            0.000000e+00
Victim Age            2.796129e+02
Premise Code          5.541874e+02
dtype: float64


The min values:
DR Number             100100007.0
Time Occurred                 1.0
Area ID                       1.0
Reporting District          100.0
Crime Code                  997.0
Victim Age                   10.0
Premise Code                101.0
dtype: float64
```

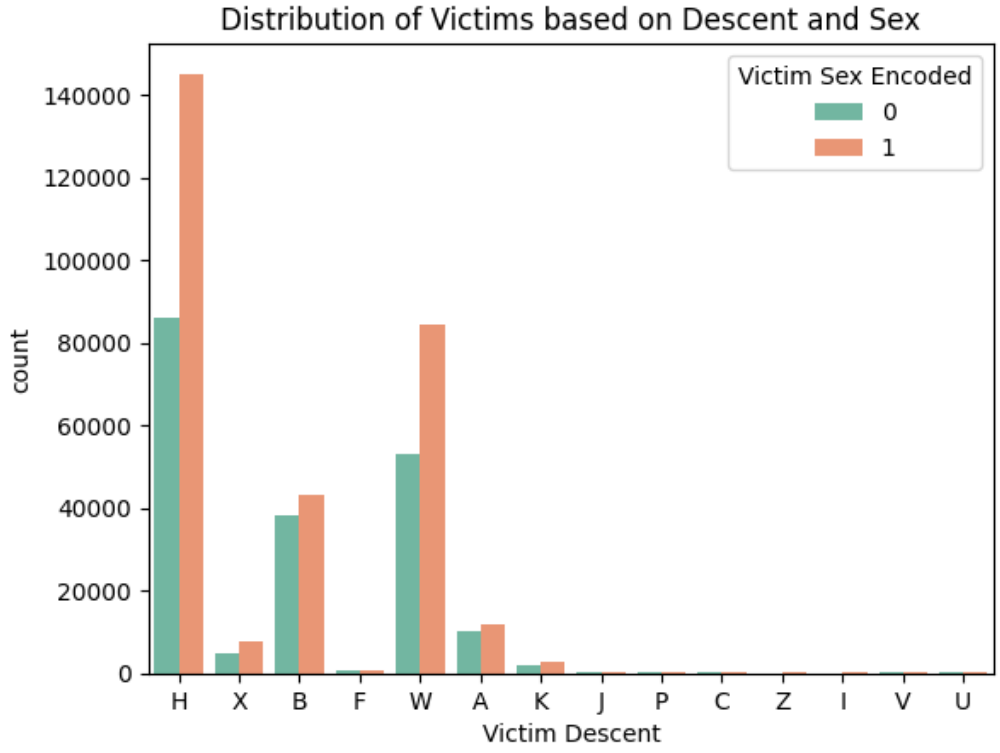
2. What kind of preprocessing techniques have you applied to this dataset?
   - Checking the outliers (the output result in the code shows that there were no outliers based on the boxplots)
   - Handling the missing values. Includes dropping duplicates which there were none, handling the NaN values using bfill() method. Also dropping a few rows as it did not signify much in the data.
   - Further cleaned some data that was not providing much info, like removing data where there were drivers in the age groups 90 above and 15 below. Since it is not very realistic.
   - Using label encoder and one-hot encoding methods to convert categorical data.

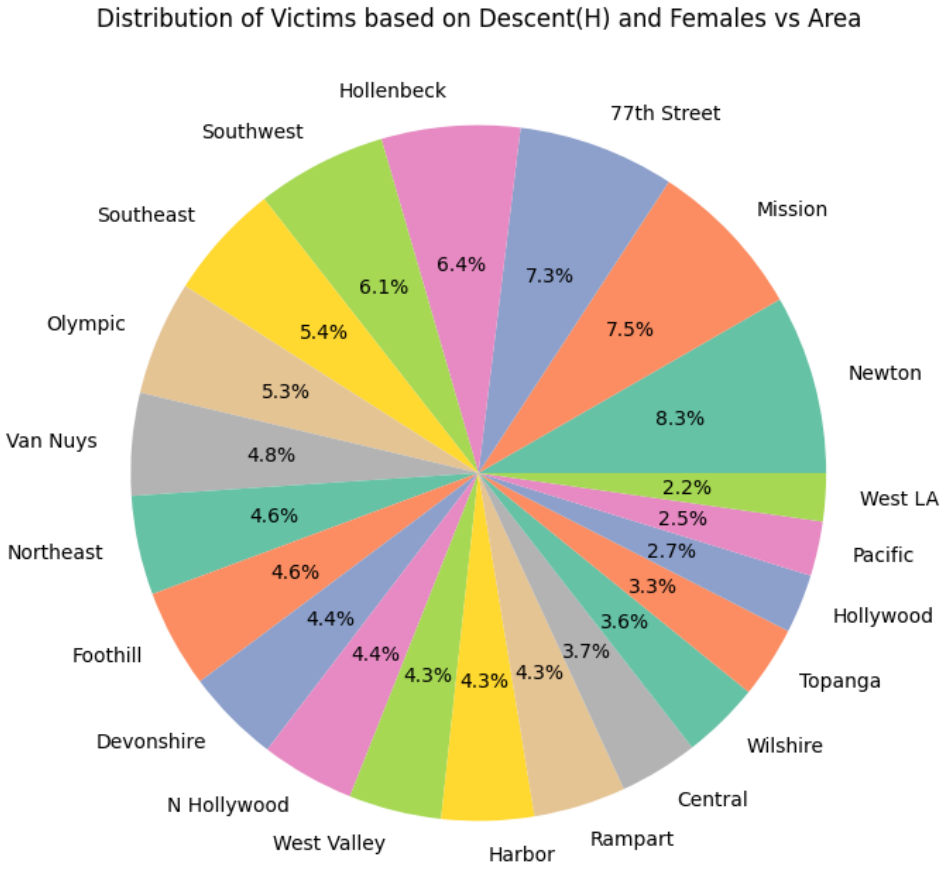
- I did not normalize since there is not a lot of features that require normalization. And further normalizing them just affects the visualization and the model training and accuracy.

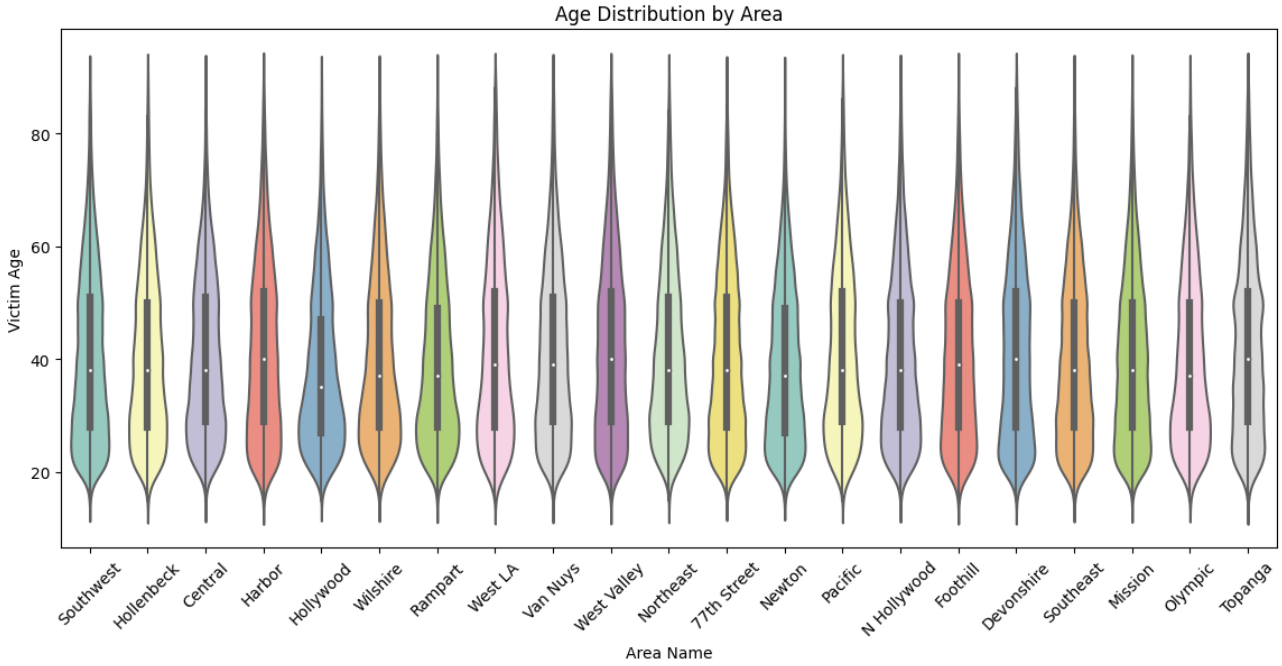
3. Provide at least 5 visualization graphs with a brief description for each graph, e.g. discuss if there are any interesting patterns or correlations.




Victim Age Distribution
Count
Victim Age
0
10000
20000
30000
40000
50000
60000
20
30
40
50
60
70
80
90


Distribution of Victims based on Descent and Sex
Victim Sex Encoded
0
1
count
Victim Descent
0
20000
40000
60000
80000
100000
120000
140000
H
X
B
F
W
A
K
J
P
C
Z
I
V
U


Distribution of Victims based on Descent(H) and Females vs Area
Hollenbeck
77th Street
Southwest
Mission
Southeast
Newton
Olympic
West LA
Van Nuys
Pacific
Northeast
Hollywood
Foothill
Topanga
Devonshire
Wilshire
N Hollywood
Central
West Valley
Rampart
Harbor
6.4%
7.3%
7.5%
6.1%
8.3%
5.4%
5.3%
4.8%
2.2%
4.6%
2.5%
2.7%
4.6%
3.3%
4.4%
3.6%
4.4%
3.7%
4.3%
4.3%
4.3%


Age Distribution by Area
Victim Age
80
60
40
20
Southwest
Hollenbeck
Central
Harbor
Hollywood
Wilshire
Rampart
West LA
Van Nuys
West Valley
Northeast
77th Street
Newton
Pacific
N Hollywood
Foothill
Devonshire
Southeast
Mission
Olympic
Topanga
Area Name

4. Provide brief details and mathematical representation of the ML methods you have used. What are the key features? What are the advantages/disadvantages?

I went with the classification task with the Victim age split into three groups: Youth, Adults and Seniors as the target and the rest of the numeric features for the model training.

In my case with the dataset I decide to use the age group as the target and other features like Victim age, Victim sex encoded, victim descent encoded, premise code, area id, time occurred, reporting district.(can be refereed to confusion matrix above) Even though we can see there is very less relationship between the features the engineering group of age will help understand what can be gather from this dataset and if further features are required.

The advantage here is that if based on the features selected with can gather information on how data can be classified then that can be used to further collect that data in relation to these aspects based on the models we use.

The disadvantage is that since there is a limited relationship between the current features lot of feature engineering might be required and even further train and error might be need when training models.

Before creating the models, I went and used SMOTE to handle the class imbalance and further improve the accuracy.

ML Model #1: Random Forest Classifier

```
Random Forest Classifier - Validation Set Accuracy:  0.7034330011074197
Random Forest Classifier - Test Set Accuracy:  0.6975513719699766

Classification Report for Validation Set:
              precision    recall  f1-score   support

      Adults       0.57      0.58      0.57     26962
     Seniors       0.57      0.55      0.56     27098
       Youth       0.96      0.99      0.97     27210

    accuracy                           0.70     81270
   macro avg       0.70      0.70      0.70     81270
weighted avg       0.70      0.70      0.70     81270


Classification Report for Test Set:
              precision    recall  f1-score   support

      Adults       0.57      0.57      0.57     27076
     Seniors       0.56      0.54      0.55     27179
       Youth       0.95      0.98      0.97     27015

    accuracy                           0.70     81270
   macro avg       0.69      0.70      0.70     81270
weighted avg       0.69      0.70      0.70     81270

Random Forest - Loss: 0.5677
Random Forest - Training Time: 290.1040 seconds
```

Based on the features selected we can see a good accuracy between validation and test phases. This shows that the model is learning and classifying well and correctly. This is a good sign that it is not overfitting. Overall, the model classifies the three groups correctly, but there can be further improvement for the adults and seniors' group. And a low value for the loss is also a sign that the model is learning and not memorizing data. The training time is slightly higher but this is because it is a huge dataset.

ML Model #2: Logistic Regression Classifer

```
Logistic Regression - Loss: 1.0662
Logistic Regression - Training Time: 858.9940 seconds
Logistic Regression - Validation Set Accuracy:  0.4141995816414421
Logistic Regression - Test Set Accuracy:  0.4150855174110988

Classification Report for Logistic Regression (Validation Set):
              precision    recall  f1-score   support

      Adults       0.38      0.32      0.35     26962
     Seniors       0.40      0.34      0.36     27098
       Youth       0.45      0.58      0.51     27210

    accuracy                           0.41     81270
   macro avg       0.41      0.41      0.41     81270
weighted avg       0.41      0.41      0.41     81270


Classification Report for Logistic Regression (Test Set):
              precision    recall  f1-score   support

      Adults       0.38      0.33      0.35     27076
     Seniors       0.40      0.33      0.36     27179
       Youth       0.45      0.59      0.51     27015

    accuracy                           0.42     81270
   macro avg       0.41      0.42      0.41     81270
weighted avg       0.41      0.42      0.41     81270
```

For this model we can see that accuracy is not great for either validation or test phases. Also, it has a comparatively high loss value. This is a sign that based on the data in this dataset the features are not as great to create a logistic regression model. One factor being that this is a huge dataset, and the model is finding it hard to learn over the data. Alos the high training time is another sign that the model is finding it hard to train and learn the data. Since the model is not able to converge this model does not provide much insight into the dataset chosen and cannot help generating info or classify new data correctly if tested with.

ML Model #3: Gradient Boosting Classifier

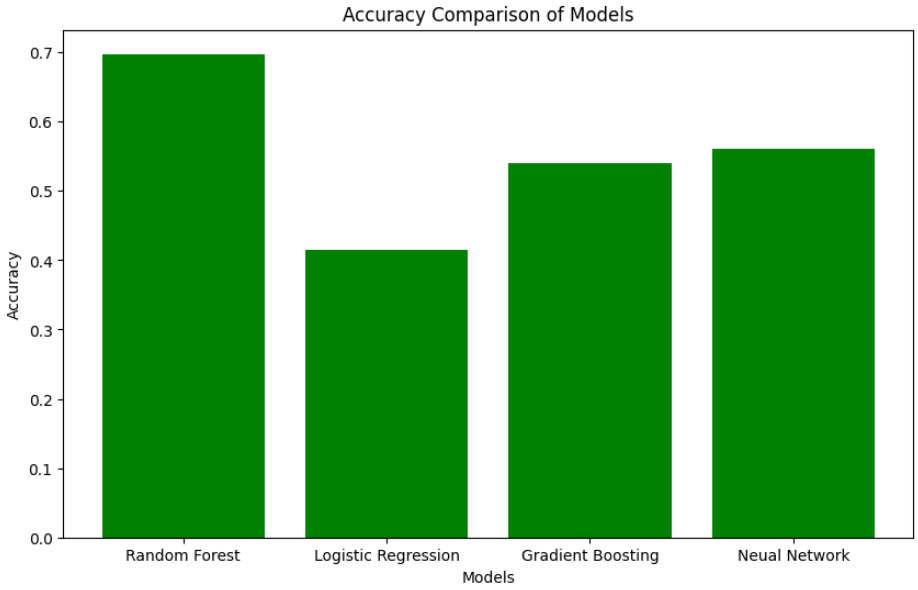
For the gradient boosting g model, the accuracy is much better than Logistic regression. However, it's still pretty low. Even though the loss value is less than logistic regression it still is not the best and shows that the model is not performing well with the features chosen. The metrics pf precision a, recall and f1_score show a balance which shows that there is a chance for the model to improve based on the improvements made either with the features or the model setup. In these metrics we can see that it classifies the youth group well and finds it harder to classify the other two groups.

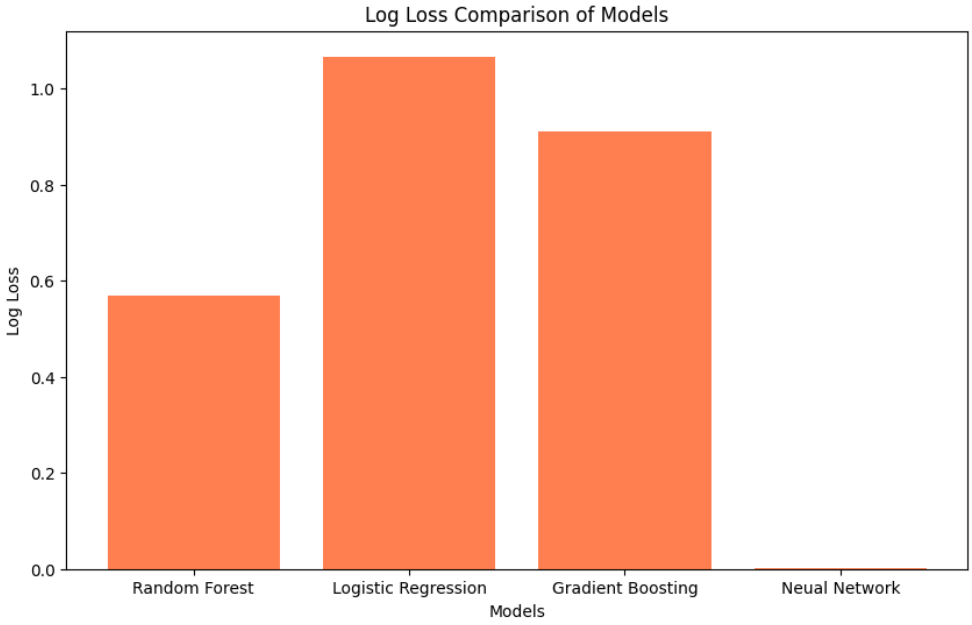
```
 Validation Accuracy: 0.5433
 Test Accuracy: 0.5403
Validation Set - Log Loss: 0.9088
Test Set - Log Loss: 0.9100

Classification Report:
              precision    recall  f1-score   support

      Adults       0.45      0.57      0.50     27076
     Seniors       0.49      0.43      0.46     27179
       Youth       0.74      0.62      0.67     27015

    accuracy                           0.54     81270
   macro avg       0.56      0.54      0.54     81270
weighted avg       0.56      0.54      0.54     81270

Gradient Boosting - Training Time: 355.7824 seconds
```

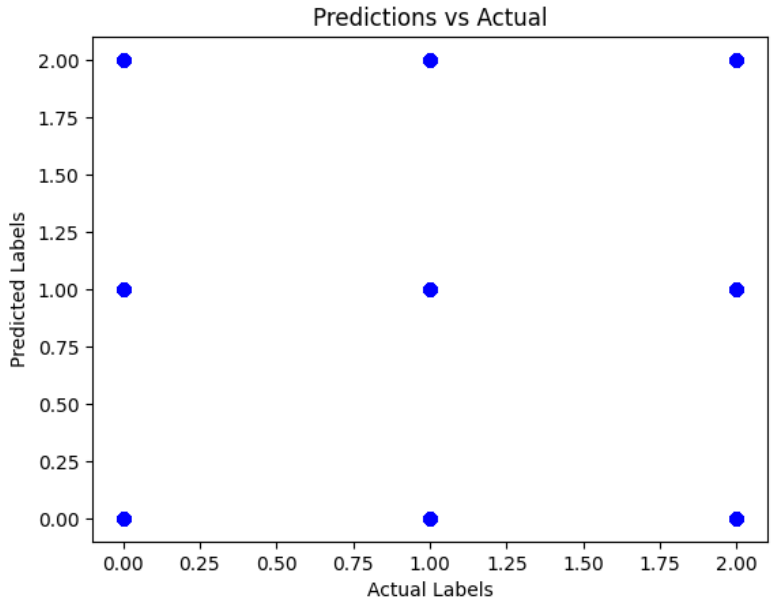
5. Provide brief details of the NN model you have used.

The NN I have is a fully connected neural network with five layers, that is three hidden layers. It takes input and maps to 64 hidden units. The second layer maps 128 units and the third layer to 256 and the last is the output layer. After that the activation function of RELU is applied. Also added dropout to help improve accuracy. The training loop trains over 20 epochs using Adam optimizer with learning rate of 0.001 and CrossEntropyLoss for loss function.

6. Provide your loss value and accuracy for all 4 methods (3 ML models & 1 NN).



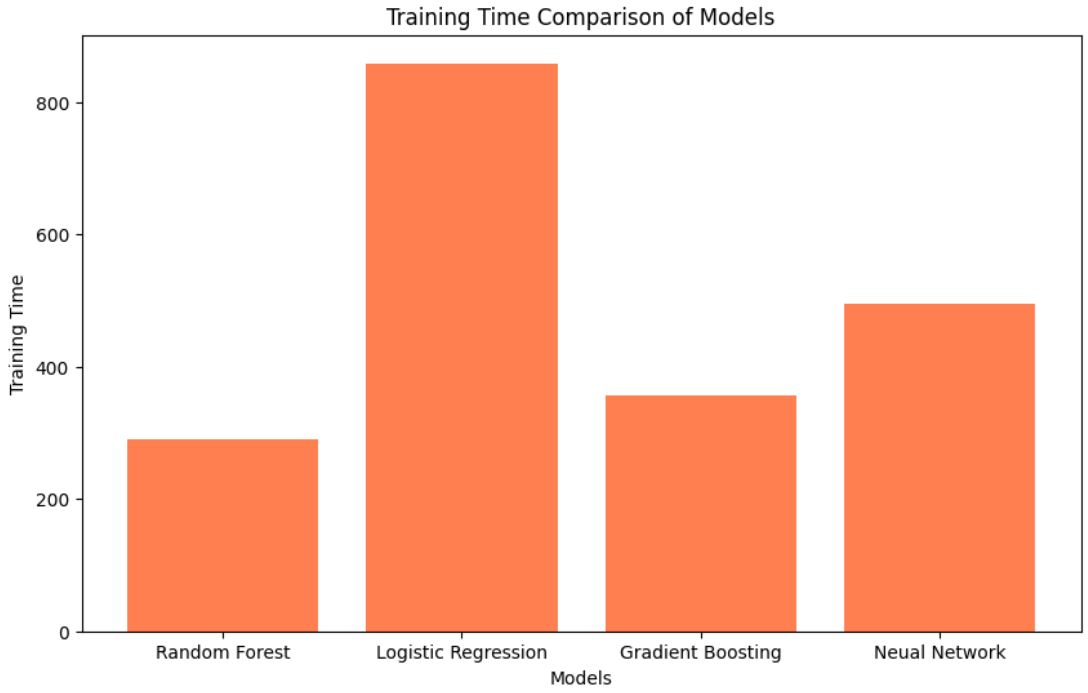




7. Show the plot comparing the predictions vs the actual test data for all methods used. Analyze the results. You can consider accuracy/time/loss as some of the metrics to compare the methods.

Above are the two graphs for the Loss and Accuracy comparisons of the models and we can see that the loss value of Neural network has the least value, even though it says that this makes the model better, in my case the accuracy has shown to be not so the case. Even though the neural network will be the best model compared to all models, in this case the features selected do not seem to give that result. Whereas the value with high accuracy is the Random Forest classifier. This shows that maybe the features selected, or engineering seem suitable for the random forest classifier. However, the accuracy of the neural network is still above 50% which shows that further tuning of the parameters can help yield between results.

Further graphs show that the training time is high for logistic regression, which makes sense as it had difficulty learning based on the features selected and that it did no converge either showing that the model was not the best for the chosen features or features engineered.



Part III

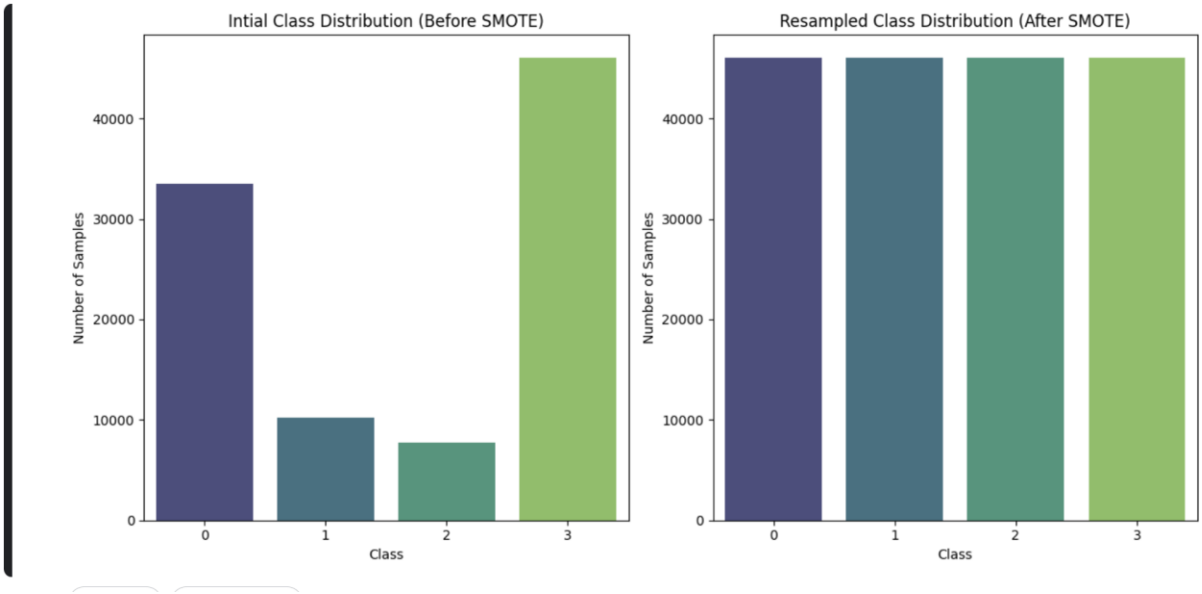
1. Provide a brief overview of your dataset (e.g. type of data, number of samples, and features. Include key statistics (e.g. mean, standard deviation, number of missing values for each feature).
   After using SMOTE to handle classes imbalance the statistics of the data is as follows:

```
Training Data Statistics:
Number of Samples: 184104
Image Dimensions: 28 x 28
Number of Classes: 4

Class Distribution in Training Set:
0    46026
1    46026
2    46026
3    46026
Name: count, dtype: int64

Pixel Statistics (Training Set):
Mean Pixel Value: -0.6714
Standard Deviation: 0.3604
Min Pixel Value: -1.0
Max Pixel Value: 1.0
```

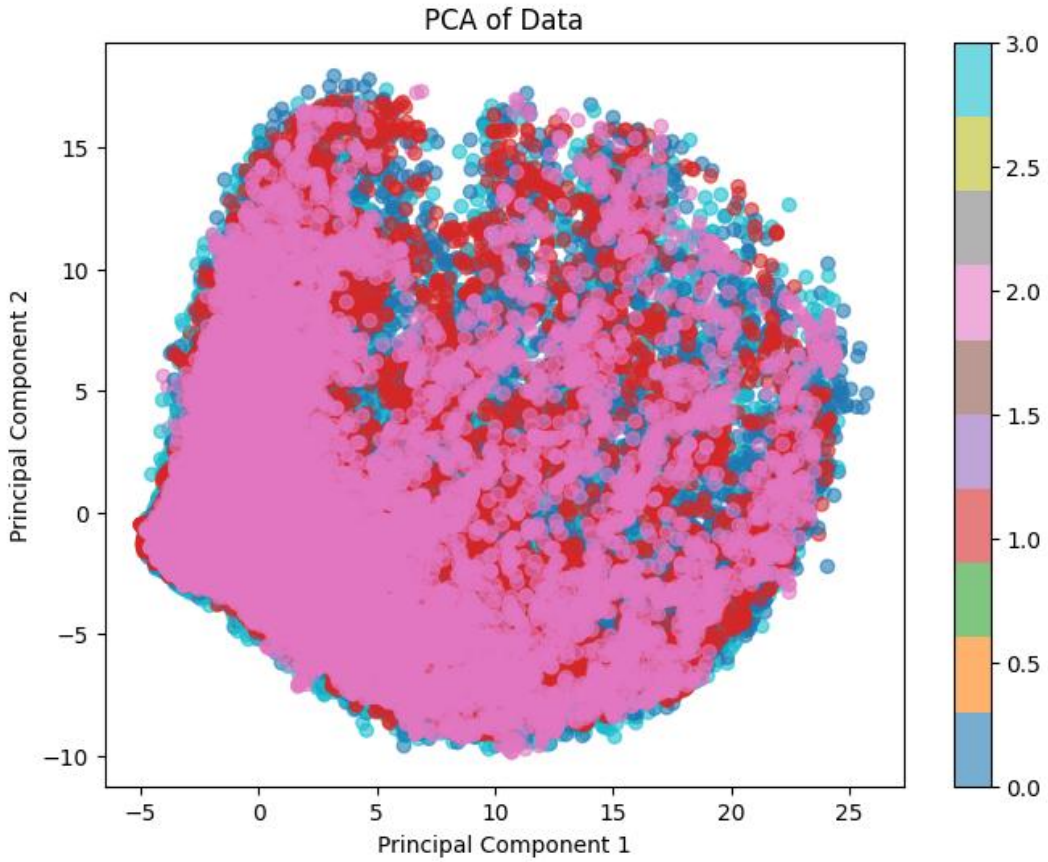
---

2. Include at least 3 graphs, such as histograms, scatter plots, or correlation matrices. Briefly describe the insights gained from these visualizations.

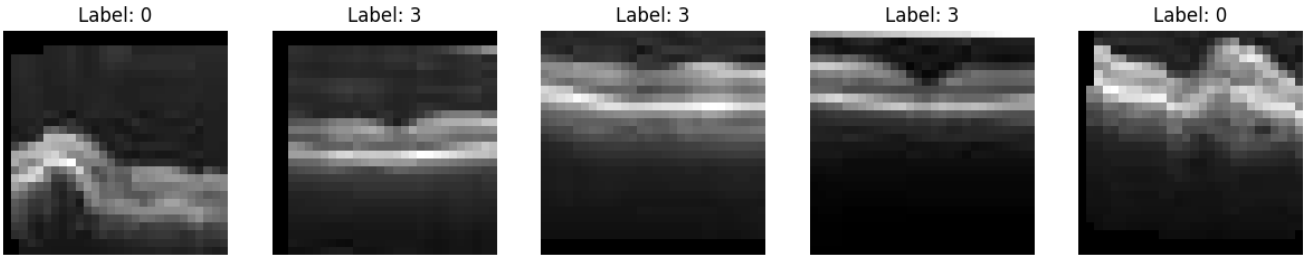
Visualization 1: This visualization shows the imbalance in the 4 classes before using SMOTE to handle the data. We can see the majority samples are part of 0 and 1 before we handle the data. Finally, we get the samples in each class equally.



Visualization 2: After the handling of the classes I went ahead and used the PCA visualization to see the classes and the distribution.



Visualization 3: For this I have displayed some of the images and the labels of the dataset.



3. Describe the NN you have defined.

For this part I have defined a CNN model. It has 3 convolutional layers, starting with 64 filters that give 64 feature maps, followed by 128 filters and last 256 filters. The model

also has two max-pooling layers of size 2. Lastly the model has two fully connected layers. The model also has the dropout layer where I have set it to a value of 0.5.

The model has the forward pass function/method that handles the input and is passed though the layers. Here I used the RELU activation function in the convolutional layers. Did not use the SoftMax activation function since the loss function of CrossEntropyLoss method handles it during the training/validation/testing phase.

4. Describe how one or more of the techniques (regularization, dropout, early stopping) you applied have impacted the model's performance.

Here I have used the Adam optimizer as part of the regularization technique. I used a value for the weight_decay = 0.00001 which helped increase the performance. This method helped prevent overfitting and helped generalize the features better and hence get good performance for both training and validation as both had comparatively similar low loss values.
Another technique that helped to make the model robust is the use of dropout. It stabilizes the training and validation accuracy hence showing that the model is learning and not overfitting.

In my base model I have not exactly used the early stopping specifically however a version of it I did by saving the best weight based on the validation loss decrease.

5. Discuss the results and provide relevant graphs:
   a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.

Here we can see that the model has learned over the training epoch. The training loss kept decreasing from .9034 to .227 and along with training accuracy increasing gradually. Similarly, the validation loss was also low, and the accuracy reached a value of 80.39%. Model validation reached the peak at epoch 4 and the values kept changing here and there a little.

The time it took to train was 247.27 seconds.

Then below with the testing data we can see we get an accuracy of 73.10%. Also, the other metrics show that even though they are not perfect, the model has made some correct predictions.

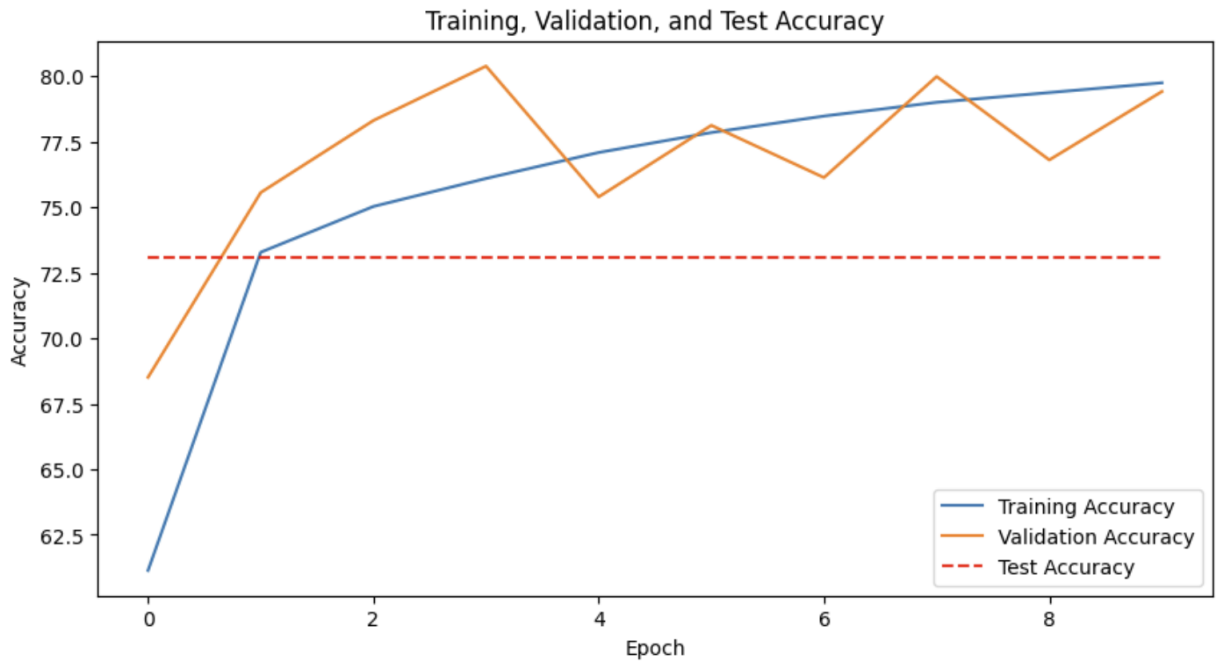
However, the testing accuracy is less than validation and that can show some generalization gap.

```
Epoch [1/10], Train Loss: 0.9034, Train Accuracy: 61.13%, Val Loss: 0.7482, Val Accuracy: 68.51%
Epoch [2/10], Train Loss: 0.6852, Train Accuracy: 73.28%, Val Loss: 0.6190, Val Accuracy: 75.56%
Epoch [3/10], Train Loss: 0.6434, Train Accuracy: 75.03%, Val Loss: 0.5616, Val Accuracy: 78.31%
Epoch [4/10], Train Loss: 0.6148, Train Accuracy: 76.10%, Val Loss: 0.5071, Val Accuracy: 80.39%
Epoch [5/10], Train Loss: 0.5925, Train Accuracy: 77.09%, Val Loss: 0.6100, Val Accuracy: 75.40%
Epoch [6/10], Train Loss: 0.5732, Train Accuracy: 77.85%, Val Loss: 0.5523, Val Accuracy: 78.13%
Epoch [7/10], Train Loss: 0.5586, Train Accuracy: 78.48%, Val Loss: 0.5922, Val Accuracy: 76.14%
Epoch [8/10], Train Loss: 0.5438, Train Accuracy: 79.01%, Val Loss: 0.5197, Val Accuracy: 79.99%
Epoch [9/10], Train Loss: 0.5348, Train Accuracy: 79.38%, Val Loss: 0.5839, Val Accuracy: 76.81%
Epoch [10/10], Train Loss: 0.5227, Train Accuracy: 79.75%, Val Loss: 0.5233, Val Accuracy: 79.42%
Best model weights saved to base_model_best_weights.pt from epoch 4
Total Training Time: 247.27 seconds
```

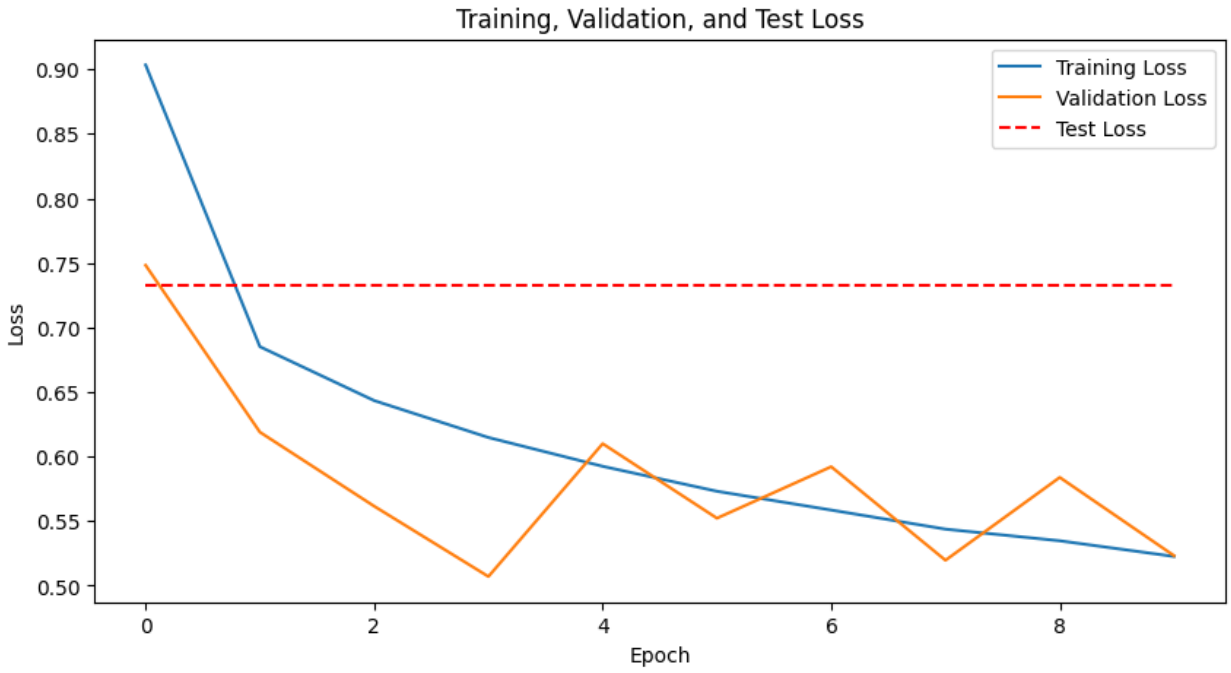
```
Predicted: [3 2 3 3 0 2 1 0 0 3]
Actual Labels: [3 2 3 3 0 2 1 0 0 3]
Test Accuracy: 73.10%
Precision: 0.7301, Recall: 0.7310, F1: 0.7231
```

b. Plot the training and validation accuracy over time (epochs).

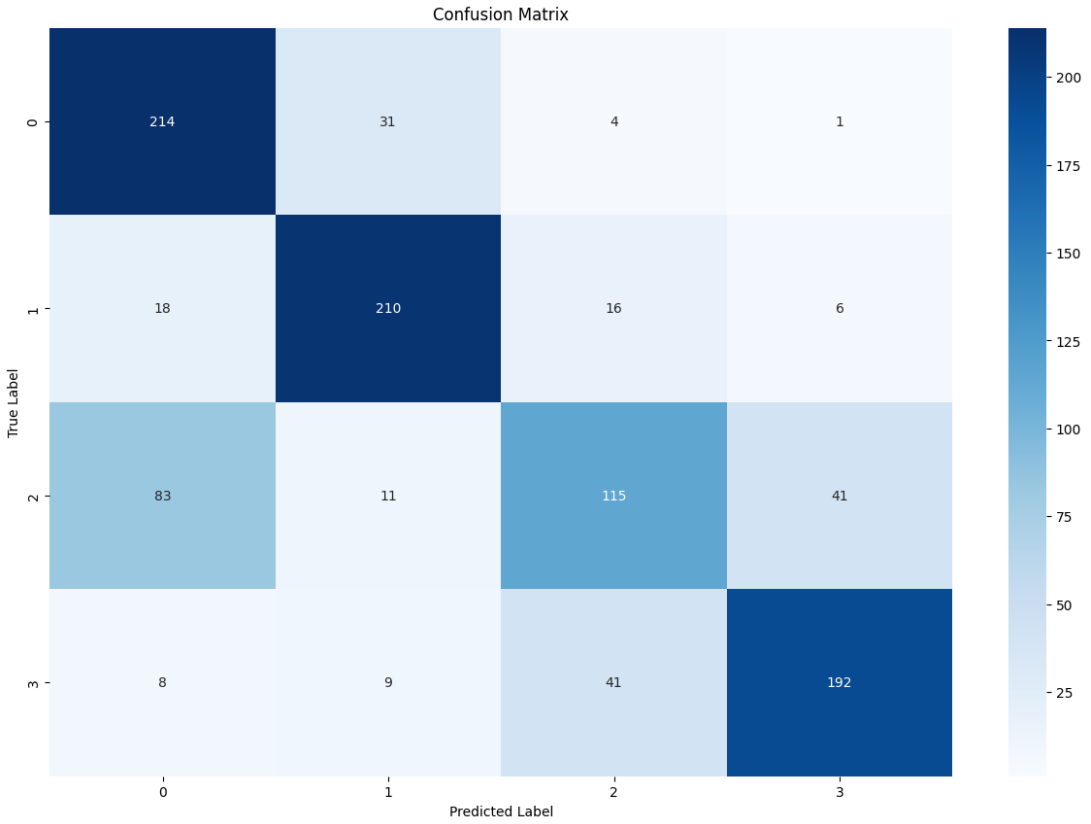
Below is the graph for accuracy between training, validation and testing data. We can see that over the epoch the value for training and validation is close in range. And as we see the accuracy is slightly lesser than the other two phases.



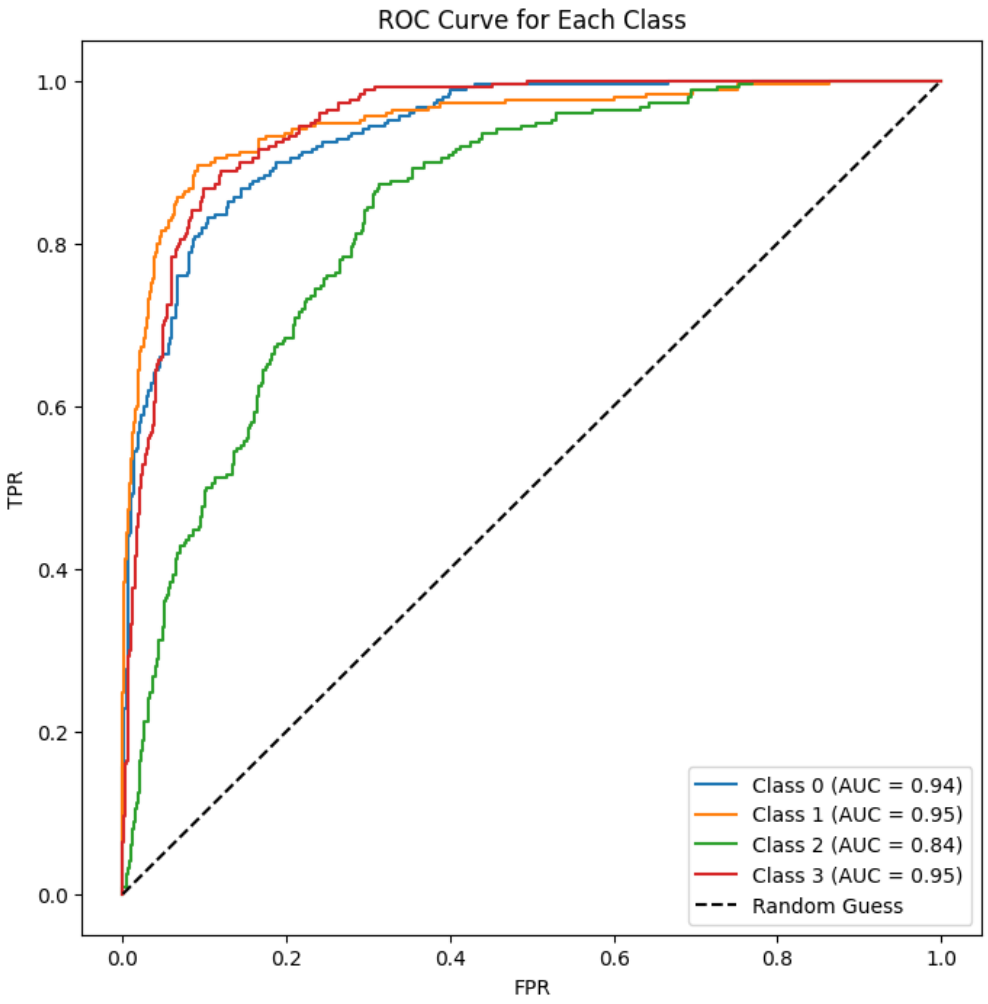
c. Plot the training and validation loss over time (epochs).



d. Generate a confusion matrix using the model's predictions on the test set.



e. Report any other evaluation metrics used to analyze the model's performance on the test set.



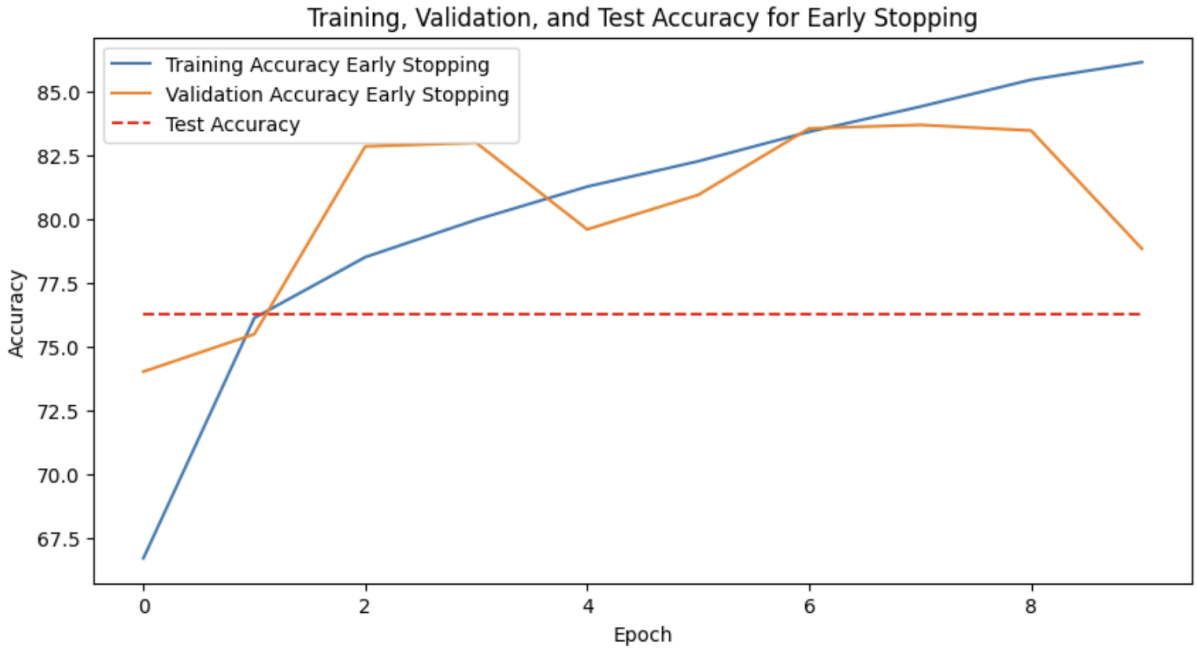
6. Discuss all the methods you used that help to improve the accuracy or the training time (Step 9).

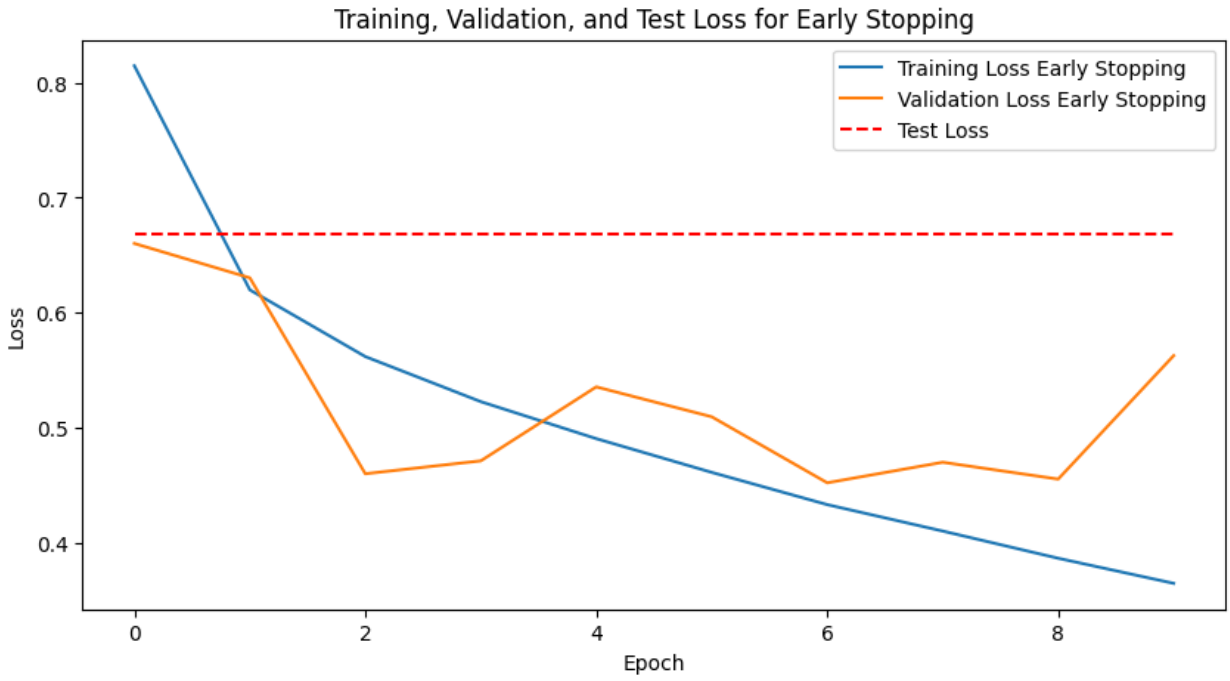
The 4 different methods I tried are:

1. Hyperparameter change #1: Here I used the base model which uses the Adam optimizer. The addition I have used is the early stopping method. I have set a counter of 0 and a patience value of 4. Along with that another change I added is to increase the batch size from 32 to 64. Attached is the metrics of the method used. Here we can see that the test accuracy for this improved model has increased compared to the base model. Also, we can see that the training time for the early stopping model has come down significantly. Along with a slight increase in precision, recall, f1_score.

```
Epoch [1/10], Train Loss: 0.8146, Train Accuracy: 66.73%, Val Loss: 0.6602, Val Accuracy: 74.03%
Epoch [2/10], Train Loss: 0.6199, Train Accuracy: 76.13%, Val Loss: 0.6303, Val Accuracy: 75.50%
Epoch [3/10], Train Loss: 0.5619, Train Accuracy: 78.51%, Val Loss: 0.4600, Val Accuracy: 82.85%
Epoch [4/10], Train Loss: 0.5227, Train Accuracy: 79.97%, Val Loss: 0.4712, Val Accuracy: 82.99%
Epoch [5/10], Train Loss: 0.4905, Train Accuracy: 81.27%, Val Loss: 0.5355, Val Accuracy: 79.60%
Epoch [6/10], Train Loss: 0.4612, Train Accuracy: 82.26%, Val Loss: 0.5095, Val Accuracy: 80.95%
Epoch [7/10], Train Loss: 0.4332, Train Accuracy: 83.42%, Val Loss: 0.4521, Val Accuracy: 83.55%
Epoch [8/10], Train Loss: 0.4102, Train Accuracy: 84.40%, Val Loss: 0.4700, Val Accuracy: 83.69%
Epoch [9/10], Train Loss: 0.3865, Train Accuracy: 85.45%, Val Loss: 0.4554, Val Accuracy: 83.47%
Epoch [10/10], Train Loss: 0.3648, Train Accuracy: 86.14%, Val Loss: 0.5627, Val Accuracy: 78.84%
Total Training Time for Early Stopping Model: 189.30 seconds
```

```
Predicted: [3 2 3 3 0 3 1 0 0 3]
Actual Labels: [3 2 3 3 0 2 1 0 0 3]
Test Accuracy Early Stopping: 76.30%
Precision: 0.7621, Recall: 0.7630, F1: 0.7573
```

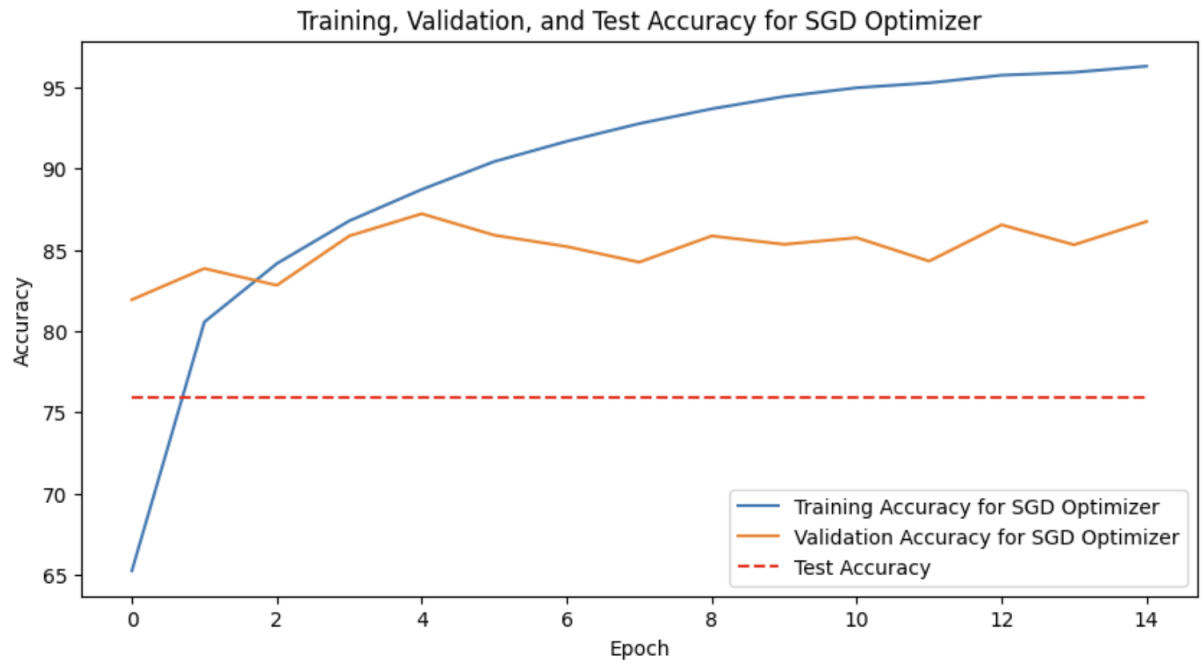


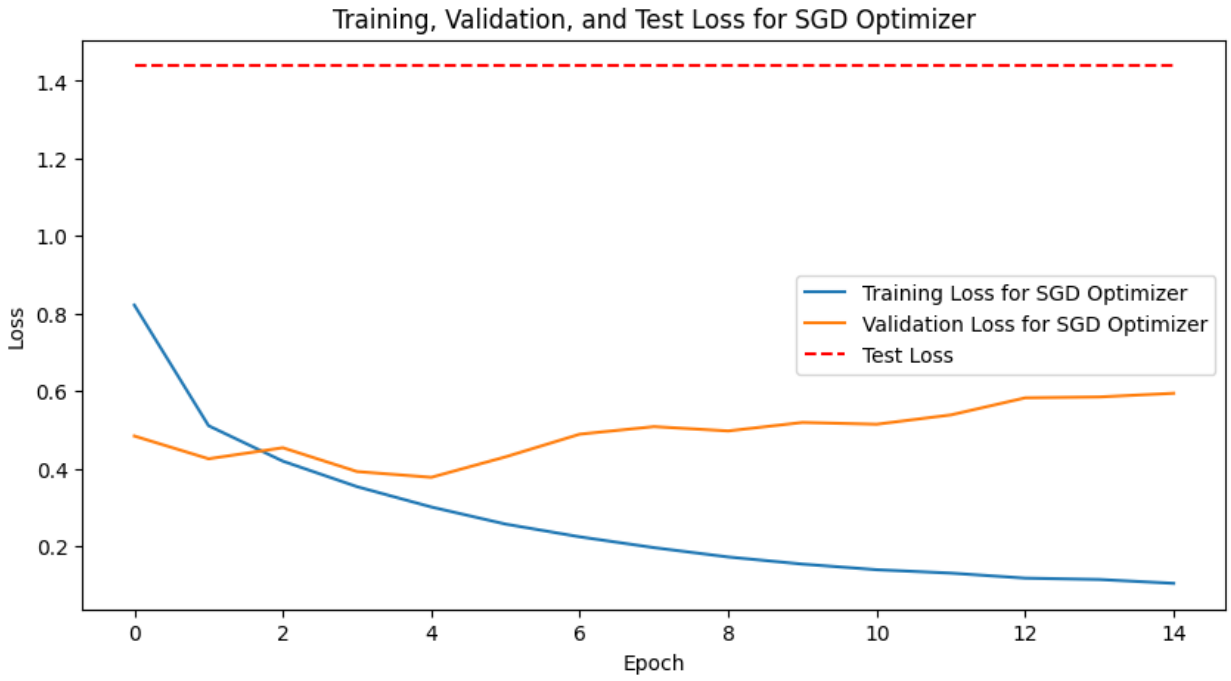


2. Hyperparameter change #2: For the second change I went with the same base model but changed the optimizer to SGD with learning rate 0.01 and momentum of 0.9. And I went back to using the data with batch size of 32 with increasing the epoch to 15. Attached the metrics. We can see that here the training time has increased, but that I increased the number of epochs. From it we can notice that there seems to be learning happening as the training loss decreases and accuracy increases. However, the variation in the loss values for validation shows signs of overfitting. In the beginning it performs well but later it starts generalizing.
   A further possible scenario that we can use to better improve the model is to combine it with early stopping so that it stops when the validation loss stops improving. This can also save time even though the epochs are more.

```
Epoch [1/15], Train Loss: 0.8216, Train Accuracy: 65.25%, Val Loss: 0.4838, Val Accuracy: 81.93%
Epoch [2/15], Train Loss: 0.5103, Train Accuracy: 80.56%, Val Loss: 0.4250, Val Accuracy: 83.85%
Epoch [3/15], Train Loss: 0.4189, Train Accuracy: 84.15%, Val Loss: 0.4535, Val Accuracy: 82.82%
Epoch [4/15], Train Loss: 0.3531, Train Accuracy: 86.78%, Val Loss: 0.3920, Val Accuracy: 85.86%
Epoch [5/15], Train Loss: 0.3006, Train Accuracy: 88.72%, Val Loss: 0.3772, Val Accuracy: 87.22%
Epoch [6/15], Train Loss: 0.2565, Train Accuracy: 90.43%, Val Loss: 0.4300, Val Accuracy: 85.90%
Epoch [7/15], Train Loss: 0.2236, Train Accuracy: 91.68%, Val Loss: 0.4885, Val Accuracy: 85.20%
Epoch [8/15], Train Loss: 0.1958, Train Accuracy: 92.77%, Val Loss: 0.5080, Val Accuracy: 84.24%
Epoch [9/15], Train Loss: 0.1716, Train Accuracy: 93.68%, Val Loss: 0.4969, Val Accuracy: 85.86%
Epoch [10/15], Train Loss: 0.1533, Train Accuracy: 94.43%, Val Loss: 0.5187, Val Accuracy: 85.34%
Epoch [11/15], Train Loss: 0.1388, Train Accuracy: 94.97%, Val Loss: 0.5142, Val Accuracy: 85.75%
Epoch [12/15], Train Loss: 0.1300, Train Accuracy: 95.27%, Val Loss: 0.5382, Val Accuracy: 84.31%
Epoch [13/15], Train Loss: 0.1171, Train Accuracy: 95.74%, Val Loss: 0.5823, Val Accuracy: 86.54%
Epoch [14/15], Train Loss: 0.1136, Train Accuracy: 95.93%, Val Loss: 0.5846, Val Accuracy: 85.31%
Epoch [15/15], Train Loss: 0.1037, Train Accuracy: 96.30%, Val Loss: 0.5941, Val Accuracy: 86.74%
Total Training Time for SGD Optimizer Loss Function Model: 358.54 seconds
```

```
Predicted: [3 0 3 3 0 3 1 0 0 2]
Actual Labels: [3 2 3 3 0 2 1 0 0 3]
Test Accuracy SGD Optimizer: 75.90%
Precision: 0.7757, Recall: 0.7590, F1: 0.7408
```

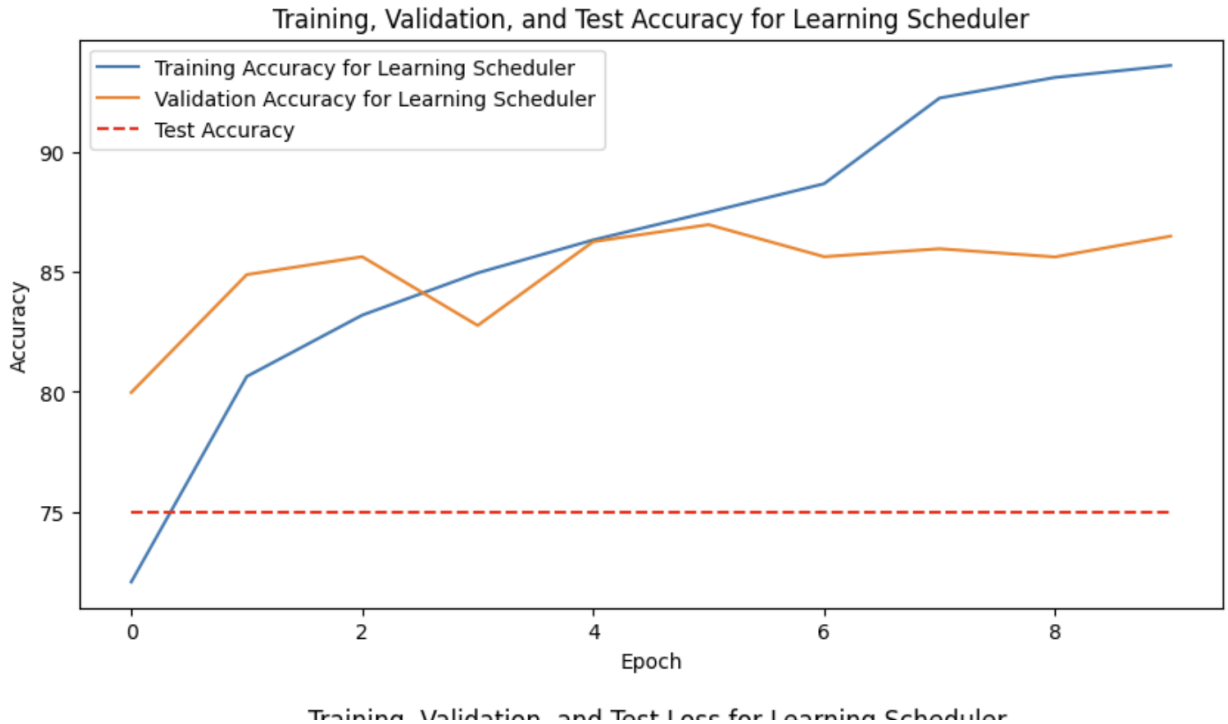




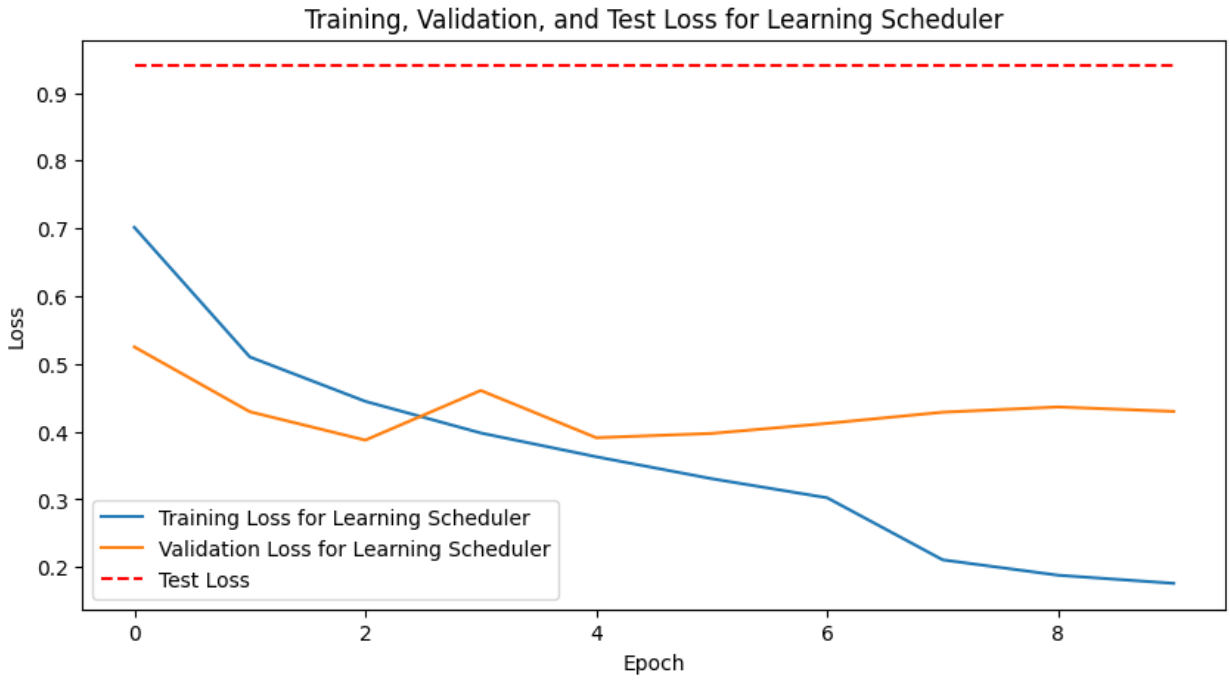
3. Hyperparameter change #3: In this using same base model, I added the learning rate scheduler method with patience value of 3 and used the scheduler ReduceLROnPlateau. Also, I added the dataset with batch size 64 for both training and validation sets and remained at 10 epochs. Based on metrics and comparing with the base model we can see training time is less and there is an increase in the accuracy of test data. And based on the data we can also see that train accuracy and validation accuracy have increased gradually, along with that the loss for both phases are also less. This shows that the model is learning effectively. Even though we do see slight fluctuation in the validation values it is still stable so we can say there is no overfitting happening in the model.
   However the testing data accuracy is less compared to validation, but this is reasonable and makes sense. So, we can say that if we try different methods or techniques along with the learning scheduler, we can improve the test accuracy.
   In total this would be the best improved model if not for the greater test accuracy of early stopping method.

```
Epoch [1/10], Train Loss: 0.7014, Train Accuracy: 72.07%, Val Loss: 0.5249, Val Accuracy: 79.98%
Epoch [2/10], Train Loss: 0.5101, Train Accuracy: 80.64%, Val Loss: 0.4293, Val Accuracy: 84.90%
Epoch [3/10], Train Loss: 0.4446, Train Accuracy: 83.21%, Val Loss: 0.3874, Val Accuracy: 85.64%
Epoch [4/10], Train Loss: 0.3979, Train Accuracy: 84.97%, Val Loss: 0.4606, Val Accuracy: 82.77%
Epoch [5/10], Train Loss: 0.3628, Train Accuracy: 86.34%, Val Loss: 0.3909, Val Accuracy: 86.27%
Epoch [6/10], Train Loss: 0.3304, Train Accuracy: 87.51%, Val Loss: 0.3973, Val Accuracy: 86.98%
Epoch [7/10], Train Loss: 0.3023, Train Accuracy: 88.69%, Val Loss: 0.4121, Val Accuracy: 85.64%
Epoch [8/10], Train Loss: 0.2106, Train Accuracy: 92.27%, Val Loss: 0.4286, Val Accuracy: 85.98%
Epoch [9/10], Train Loss: 0.1880, Train Accuracy: 93.13%, Val Loss: 0.4363, Val Accuracy: 85.64%
Epoch [10/10], Train Loss: 0.1760, Train Accuracy: 93.63%, Val Loss: 0.4298, Val Accuracy: 86.50%
Total Training Time for Learning Rate Scheduler Model: 186.74 seconds
```

```
Predicted: [3 3 3 3 0 2 1 0 0 3]
Actual Labels: [3 2 3 3 0 2 1 0 0 3]
Test Accuracy Learning Rate Scheduler: 75.00%
Precision: 0.7822, Recall: 0.7500, F1: 0.7307
```



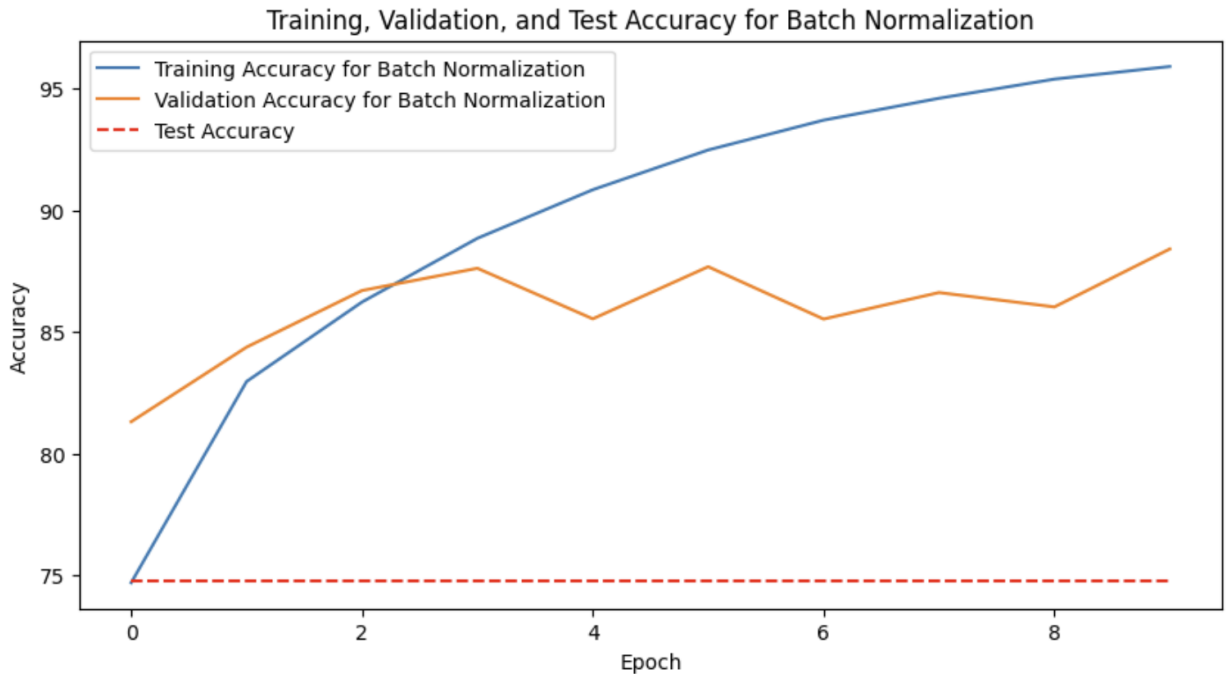
Training, Validation, and Test Loss for Learning Scheduler

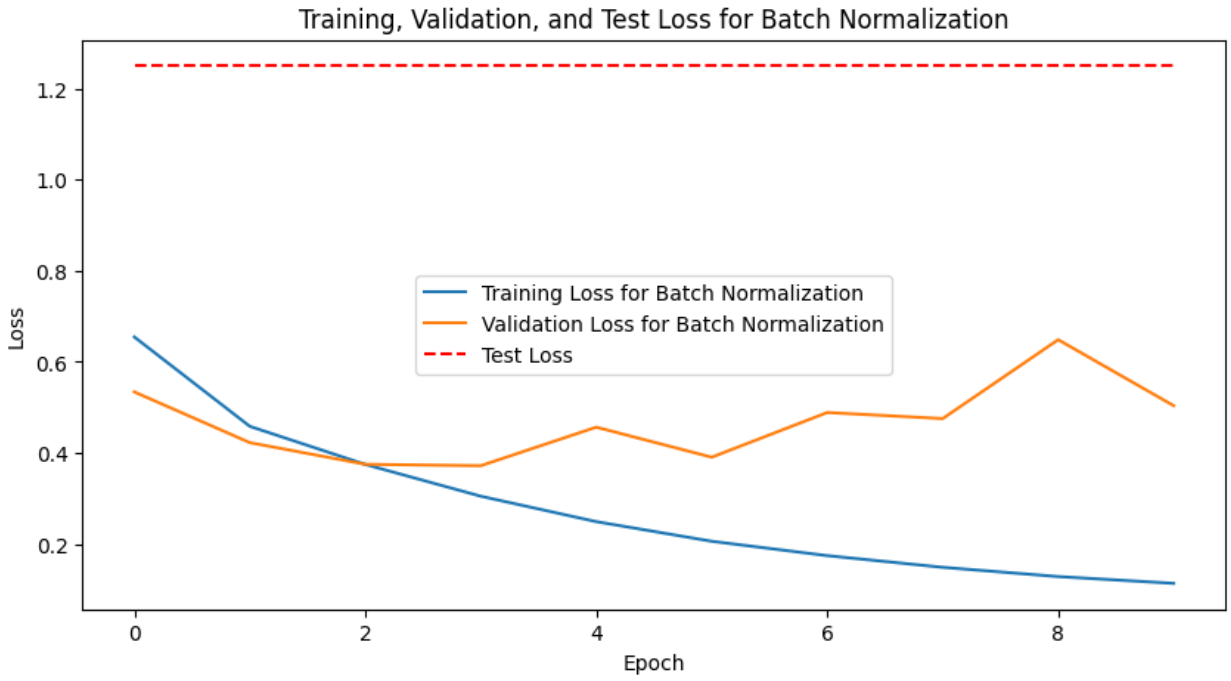


4. Hyperparameter change #4: For this I went and changed the model by adding batch normalization for each convolution layer and using the Adam optimizer. Also used the data with batch size of value 64 and with epochs 10. Attached are the metrics. The training phase metrics show that the model is learning and improving. But the fluctuations in the validation phase show that it is having difficulty learning, even though the accuracy is good there is fluctuating values of loss and could be signs of overfitting. Even though the training accuracy and testing accuracy are similar, that does not show the same when compared to validation phase data. Even though batch normalization is helping the model learn it would require further additional methods to get better accuracy for both validation and testing. Because the value of test accuracy is good yet it can be better and in range with validation only with further tunning.

```
Epoch [1/10], Train Loss: 0.6546, Train Accuracy: 74.70%, Val Loss: 0.5339, Val Accuracy: 81.31%
Epoch [2/10], Train Loss: 0.4583, Train Accuracy: 82.97%, Val Loss: 0.4225, Val Accuracy: 84.38%
Epoch [3/10], Train Loss: 0.3746, Train Accuracy: 86.23%, Val Loss: 0.3750, Val Accuracy: 86.71%
Epoch [4/10], Train Loss: 0.3048, Train Accuracy: 88.85%, Val Loss: 0.3719, Val Accuracy: 87.62%
Epoch [5/10], Train Loss: 0.2490, Train Accuracy: 90.85%, Val Loss: 0.4565, Val Accuracy: 85.54%
Epoch [6/10], Train Loss: 0.2059, Train Accuracy: 92.48%, Val Loss: 0.3905, Val Accuracy: 87.68%
Epoch [7/10], Train Loss: 0.1743, Train Accuracy: 93.71%, Val Loss: 0.4887, Val Accuracy: 85.53%
Epoch [8/10], Train Loss: 0.1487, Train Accuracy: 94.61%, Val Loss: 0.4753, Val Accuracy: 86.62%
Epoch [9/10], Train Loss: 0.1284, Train Accuracy: 95.39%, Val Loss: 0.6486, Val Accuracy: 86.03%
Epoch [10/10], Train Loss: 0.1135, Train Accuracy: 95.91%, Val Loss: 0.5037, Val Accuracy: 88.41%
Total Training Time for Batch Normalization Model: 213.20 seconds
```

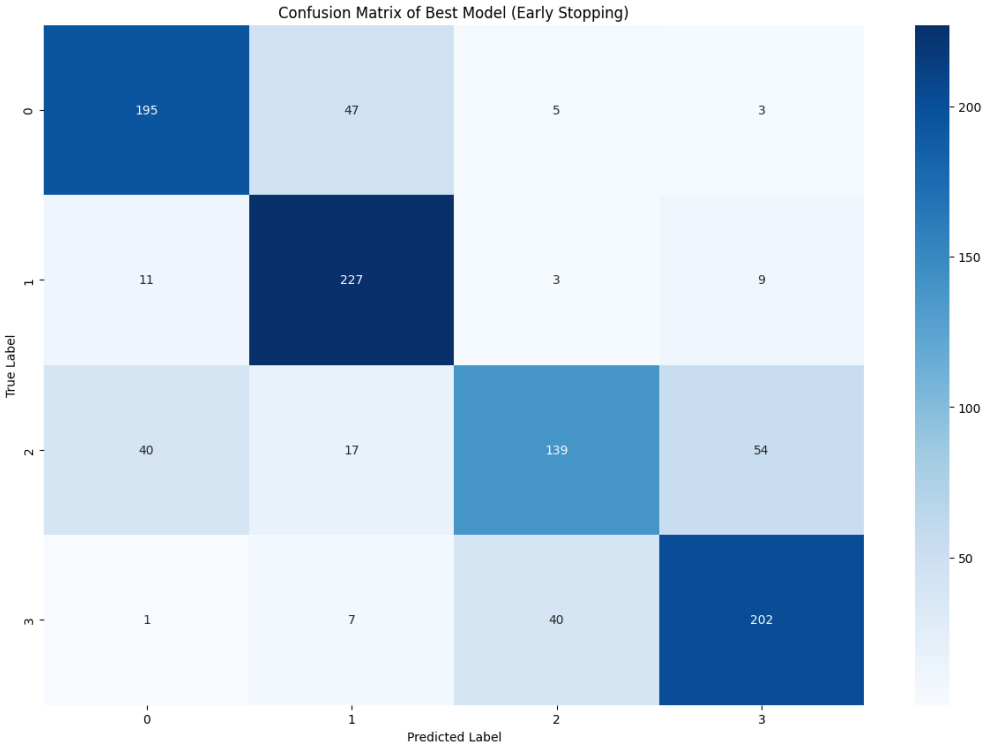
```
Predicted: [3 2 3 3 0 3 1 0 0 3]
Actual Labels: [3 2 3 3 0 2 1 0 0 3]
Test Accuracy Batch Normalization: 74.80%
Precision: 0.7905, Recall: 0.7480, F1: 0.7122
```

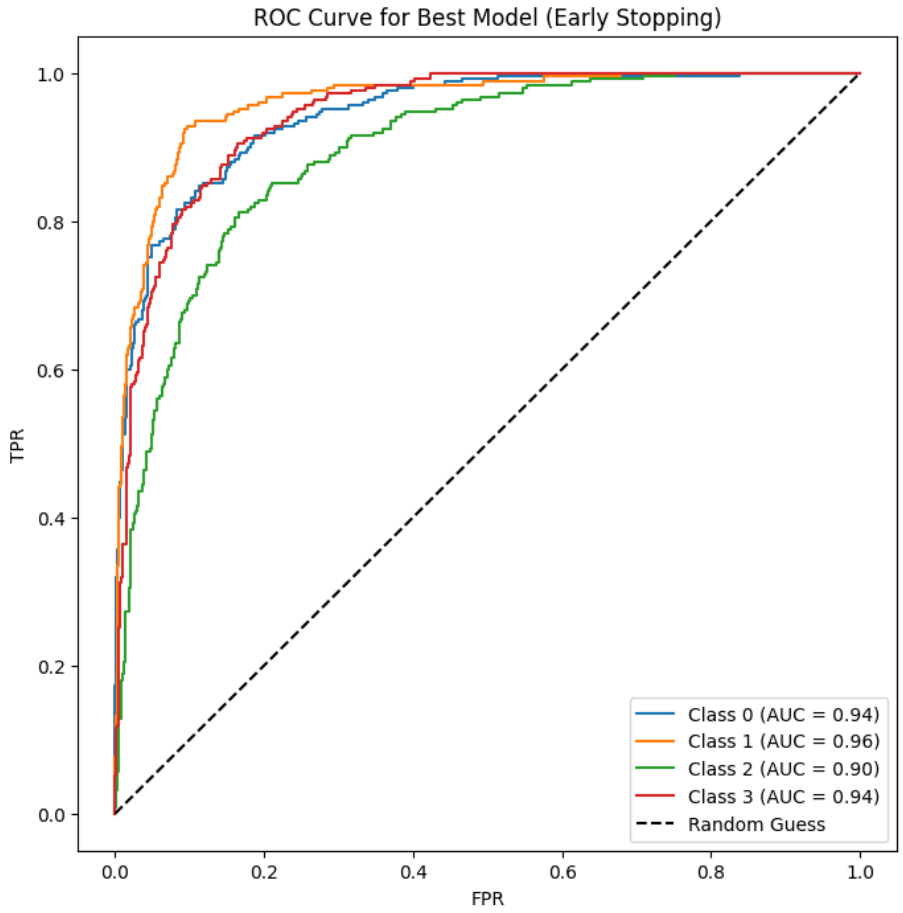




7. Provide a detailed description of your 'best model' that returns the best results. Discuss the performance and add visualization graphs with your analysis.

Based on the accuracy metric the best model out of the improved versions was the early stopping. Even though the learning scheduler seemed to have good overall values in all other metrics as well however the test accuracy is lower than early stopping. Attached are further visualizations of early stopping models.
Based on the attached confusion matrix we can see that the model's performance is spread through the classes. The diagonal values show the correctly classified cases and the other indicated misclassification. We can see that class 1 has higher correct classifications and 2 has the more wrong classification.
In the next graph, the ROC curve shows the model's capacity to differentiate between the classes. A higher AUC value indicates better separation and here also we see it perform better for class 1 and bad for class 2.
The overall model is strong, but it can use further tunning to increase the accuracy and better improve the classifications.


Confusion Matrix of Best Model (Early Stopping)
True Label
Predicted Label
0
1
2
3
195
47
5
3
11
227
3
9
40
17
139
54
1
7
40
202
200
150
100
50


ROC Curve for Best Model (Early Stopping)
TPR
FPR
0.0
0.2
0.4
0.6
0.8
1.0
Class 0 (AUC = 0.94)
Class 1 (AUC = 0.96)
Class 2 (AUC = 0.90)
Class 3 (AUC = 0.94)
Random Guess

PART IV

Completed and is a separate hand written pdf.