ASSIGNMENT 0 REPORT

PART 2

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.)**

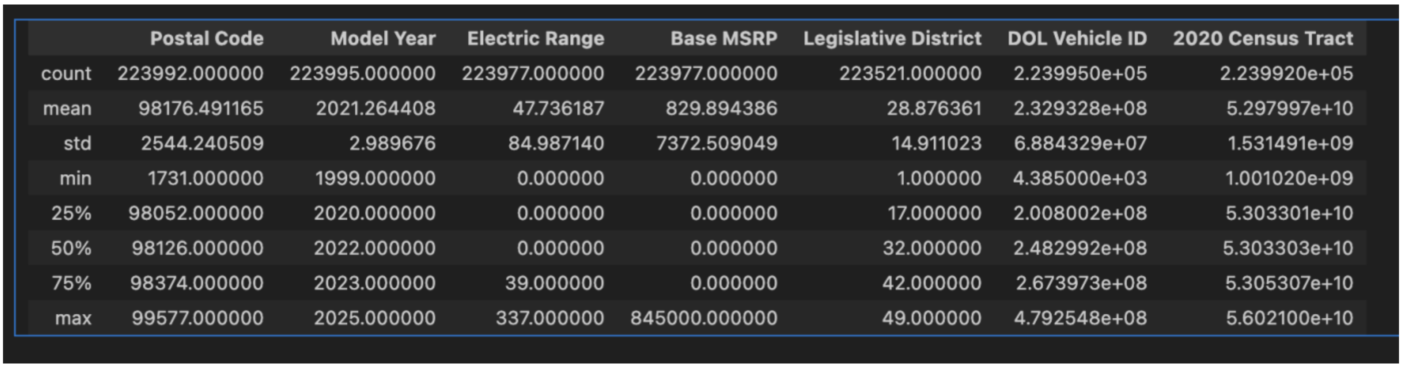
The dataset contains information on Battery Electric Vehicles (BEVs) and Plug-in Hybrid Electric Vehicles (PHEVs) that are currently registered through the Washington State Department of Licensing (DOL). It consists of 17 columns, which are: VIN (1-10), County, City, State, Postal Code, Model Year, Make, Model, Electric Vehicle Type, Clean Alternative Fuel Vehicle (CAFV) Eligibility, Electric Range, Base MSRP, Legislative District, DOL Vehicle ID, Vehicle Location, Electric Utility, and 2020 Census Tract.

Among these columns:

- **VIN (1-10)**, **County**, **City**, **State**, **Make**, **Model**, **Electric Vehicle Type**, **Clean Alternative Fuel Vehicle (CAFV) Eligibility**, **Vehicle Location**, and **Electric Utility** are string columns.

- **Postal Code**, **Model Year**, **Electric Range**, **Base MSRP**, **Legislative District**, **DOL Vehicle ID**, and **2020 Census Tract** are numerical columns.

The main Statistics of the dataset are:



	Postal Code	Model Year	Electric Range	Base MSRP	Legislative District	DOL Vehicle ID	2020 Census Tract
count	223992.000000	223995.000000	223977.000000	223977.000000	223521.000000	2.239950e+05	2.239920e+05
mean	98176.491165	2021.264408	47.736187	829.894386	28.876361	2.329328e+08	5.297997e+10
std	2544.240509	2.989676	84.987140	7372.509049	14.911023	6.884329e+07	1.531491e+09
min	1731.000000	1999.000000	0.000000	0.000000	1.000000	4.385000e+03	1.001020e+09
25%	98052.000000	2020.000000	0.000000	0.000000	17.000000	2.008002e+08	5.303301e+10
50%	98126.000000	2022.000000	0.000000	0.000000	32.000000	2.482992e+08	5.303303e+10
75%	98374.000000	2023.000000	39.000000	0.000000	42.000000	2.673973e+08	5.305307e+10
max	99577.000000	2025.000000	337.000000	845000.000000	49.000000	4.792548e+08	5.602100e+10

The missing values are:

VIN (1-10)	0
County	3
City	3
State	0
Postal Code	3
Model Year	0
Make	0
Model	0
Electric Vehicle Type	0
Clean Alternative Fuel Vehicle (CAFV) Eligibility	0
Electric Range	18
Base MSRP	18
Legislative District	474
DOL Vehicle ID	0
Vehicle Location	10
Electric Utility	3
2020 Census Tract	3

2. What kind of preprocessing techniques have you applied to this dataset?

To prepare the dataset for analysis, I performed the following preprocessing steps:

1.Handling Missing Values:

I dropped all rows containing null values to ensure the dataset is complete and free from missing data, which could otherwise lead to inaccuracies in analysis.

2.Removing Incorrect Entries:

I identified and dropped rows where the **Base MSRP (Manufacturer's Suggested Retail Price)** was listed as 0. Since the MSRP for a car cannot realistically be 0, these entries were considered incorrect or invalid and were removed from the dataset.

3. Dropping Irrelevant Columns:

I removed the following columns: **"VIN (1-10)"**, **"State"**, **"Postal Code"**, **"DOL Vehicle ID"**, **"Vehicle Location"**, and **"2020 Census Tract"**. These columns were deemed irrelevant to my target variable, **Electric Vehicle Type**, and their removal helps streamline the dataset for focused analysis.

4. Standardizing String Columns:

To ensure uniformity and improve readability, I converted all string columns to **title case**. This step standardizes the text format across the dataset, making it more consistent and easier to work with.

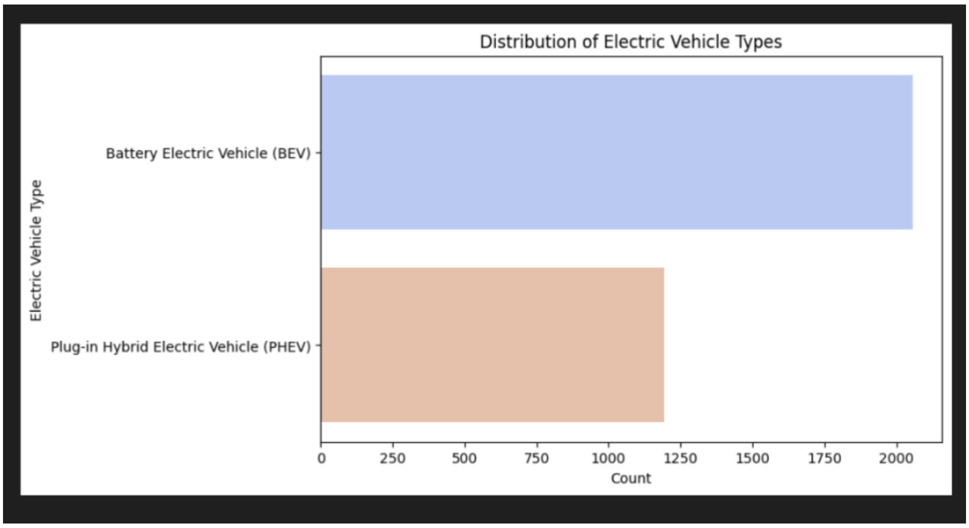
5. Encoding Categorical Columns:

I applied **Label Encoding** to convert categorical columns into numerical values. This transformation is essential for machine learning models, as they require numerical input to process and analyse the data effectively.

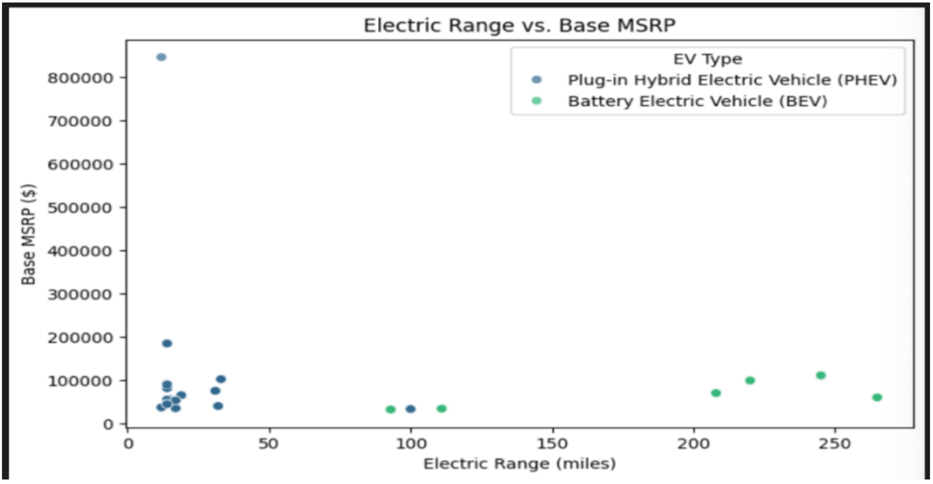
6. Normalizing Numerical Columns:

Finally, I normalized the numerical columns **"Electric Range"** and **"Base MSRP"**. Normalization scales these values to a standard range (typically between 0 and 1), which helps in improving the performance of machine learning algorithms by ensuring that no single feature dominates due to its scale.

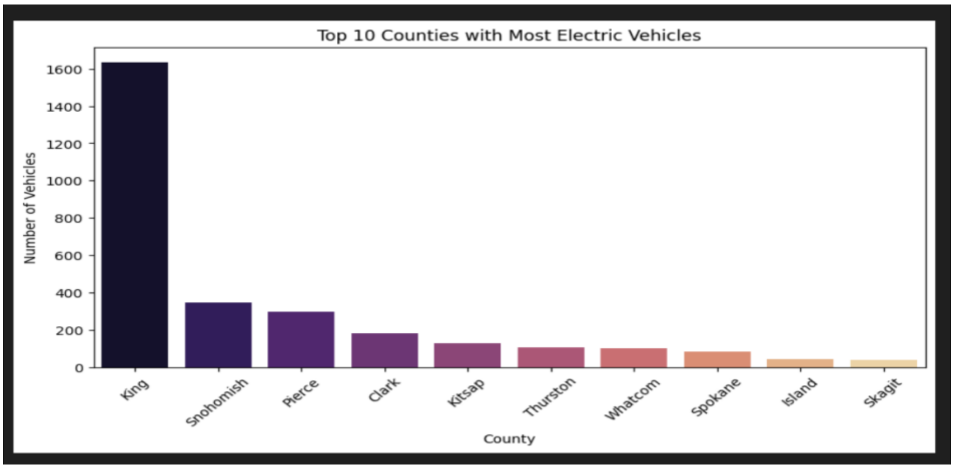
3. Provide at least 5 visualization graphs with a brief description for each graph, e.g. discuss if there are any interesting patterns or correlations.



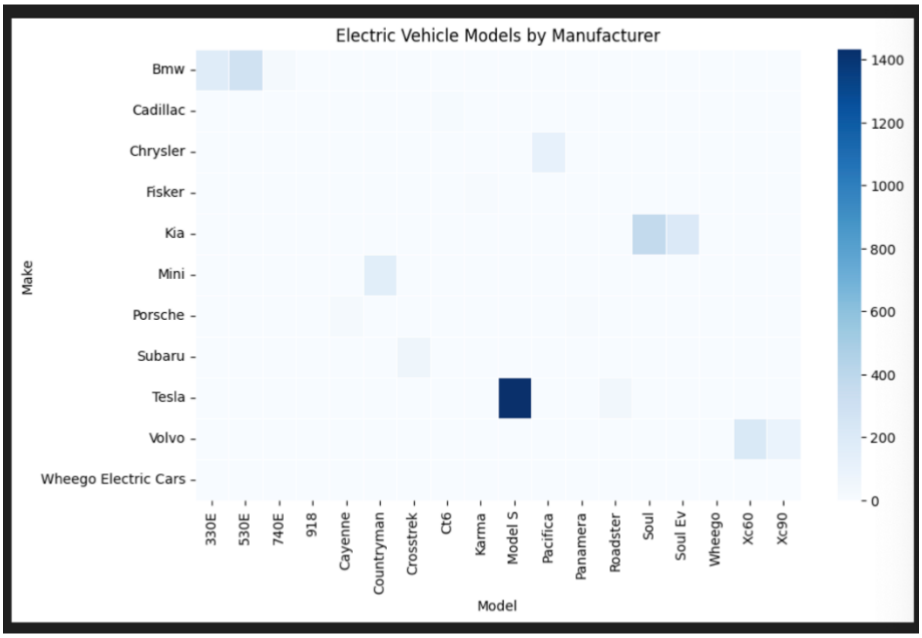
1. The graph illustrates that the dataset contains a higher number of Battery Electric Vehicles (BEVs) compared to Plug-in Hybrid Electric Vehicles (PHEVs). Specifically, the number of BEVs is approximately 2,000, while the number of PHEVs is around 1,250.



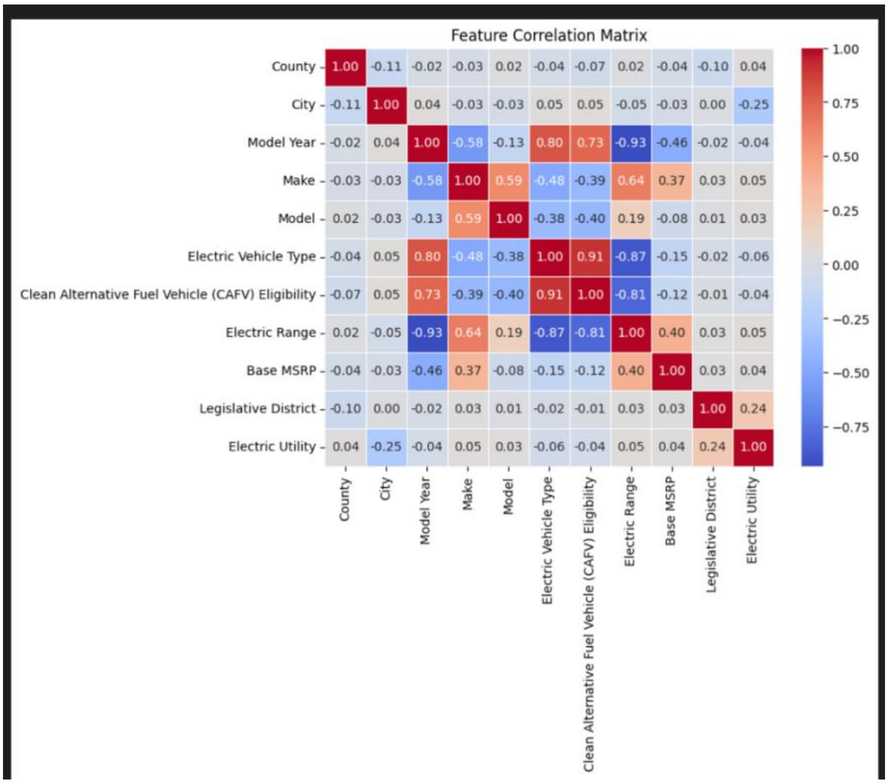
2. The graph presents the distribution of Base MSRP over Electric Range. It reveals that a large number of vehicles have an electric range of approximately 50 miles, while fewer vehicles fall around the 100-mile range. Additionally, a significant number of vehicles fall within the 200-250 mile range. An important insight from the graph is that vehicles with an electric range of 200-250 miles tend to have a higher Base MSRP compared to those with lower ranges.



3. The graph illustrates the top 10 counties with the highest number of registered Electric Vehicles (EVs). King County has the largest number of EVs, with approximately 1,600 vehicles. The second highest is Snohomish County, which has around 400 EVs. In contrast, Skagit County has fewer than 50 Electric Vehicles, making it the county with the lowest EV count among the top 10.



4.The Electric Vehicle Models by Manufacturer heatmap visualizes the distribution of various EV models across different manufacturers. The intensity of the color represents the frequency of each model, with darker shades indicating higher counts. Notably, Tesla's Model S has the highest count, followed by models from BMW, Kia, and Volvo. Other manufacturers, such as Porsche, Subaru, and Cadillac, have relatively fewer EV models in the dataset.



	County	City	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District	Electric Utility
County	1.00	-0.11	-0.02	-0.03	0.02	-0.04	-0.07	0.02	-0.04	-0.10	0.04
City	-0.11	1.00	0.04	-0.03	-0.03	0.05	0.05	-0.05	-0.03	0.00	-0.25
Model Year	-0.02	0.04	1.00	-0.58	-0.13	0.80	0.73	-0.93	-0.46	-0.02	-0.04
Make	-0.03	-0.03	-0.58	1.00	0.59	-0.48	-0.39	0.64	0.37	0.03	0.05
Model	0.02	-0.03	-0.13	0.59	1.00	-0.38	-0.40	0.19	-0.08	0.01	0.03
Electric Vehicle Type	-0.04	0.05	0.80	-0.48	-0.38	1.00	0.91	-0.87	-0.15	-0.02	-0.06
Clean Alternative Fuel Vehicle (CAFV) Eligibility	-0.07	0.05	0.73	-0.39	-0.40	0.91	1.00	-0.81	-0.12	-0.01	-0.04
Electric Range	0.02	-0.05	-0.93	0.64	0.19	-0.87	-0.81	1.00	0.40	0.03	0.05
Base MSRP	-0.04	-0.03	-0.46	0.37	-0.08	-0.15	-0.12	0.40	1.00	0.03	0.04
Legislative District	-0.10	0.00	-0.02	0.03	0.01	-0.02	-0.01	0.03	0.03	1.00	0.24
Electric Utility	0.04	-0.25	-0.04	0.05	0.03	-0.06	-0.04	0.05	0.04	0.24	1.00

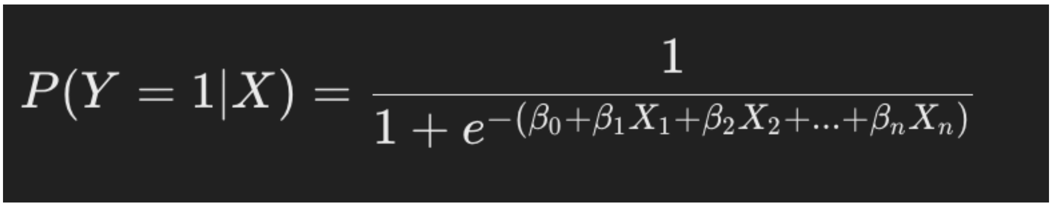
5. The plot represents a Feature Correlation Matrix**,** which visualizes the correlation between different variables in a dataset. Notable insights include a strong positive correlation between Electric Vehicle Type & CAFV Eligibility (0.91) and Model Year & CAFV Eligibility (0.73), indicating that newer models are more likely to qualify for clean fuel programs. Additionally, Electric Range & Model Year (-0.93) show a strong negative correlation, suggesting newer models have significantly different range distributions. Make & Model (0.59) also display a strong link, reflecting brand-specific model groupings.

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 used Logistic Regression, Random Forest, and K-Nearest Neighbors (KNN). Below is a brief overview of each method, their mathematical representations, key features, and advantages/disadvantages.

1. Logistic Regression

Mathematical Representation:
Logistic regression models the probability of a class label using the sigmoid function:



$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_n X_n)}}$$

where β are the model coefficients learned from data.

Key Features:

- Suitable for binary classification.
- Outputs probabilities, useful for decision thresholds.
- Assumes linear decision boundary.

Advantages:

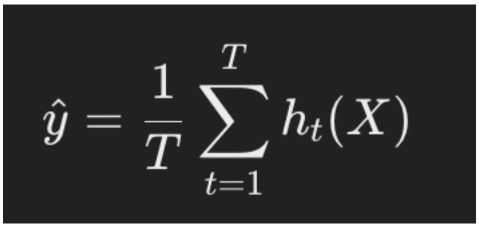
- Simple and interpretable.
- Efficient for large datasets.
- Provides probabilistic predictions.

Disadvantages:

- Assumes linear relationship between features and log-odds.
- Not suitable for highly complex, non-linear data.

2. Random Forest

Mathematical Representation:
Random Forest is an ensemble of decision trees, where predictions are made by averaging (regression) or voting (classification):



$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(X)$$

where ht(X) represents the prediction from the t-th decision tree.

Key Features:

- Uses multiple decision trees to improve performance.
- Reduces overfitting compared to individual decision trees.
- Handles both classification and regression tasks.

Advantages:

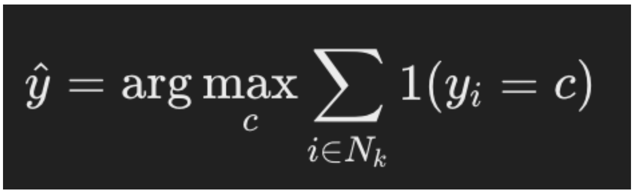
- Robust to overfitting due to averaging.
- Handles non-linear relationships well.
- Works well with high-dimensional data.

Disadvantages:

- Computationally expensive, especially for large datasets.
- Less interpretable than logistic regression.

3. K-Nearest Neighbors (KNN)

Mathematical Representation:
KNN classifies a new data point based on the majority vote of its k nearest neighbors:



$$\hat{y} = \arg\max_{c} \sum_{i \in N_k} 1(y_i = c)$$

where Nk is the set of k nearest neighbors, and 1 is an indicator function.

Key Features:

- A non-parametric, instance-based learning method.
- Classification is based on similarity (distance metric like Euclidean distance).
- Works well when decision boundaries are irregular.

Advantages:

- Simple and easy to implement.
- Works well for small datasets with complex decision boundaries.
- No training time required.

Disadvantages:

- Computationally expensive for large datasets (due to distance calculations).
- Sensitive to the choice of k and feature scaling.=

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

Neural Network Model Details

- **Number of Layers:** 4 (3 hidden layers + 1 output layer)
- **Activation Function:** ReLU (hidden layers), Sigmoid (output layer)
- **Dropout Rate:** 0.5 (applied after each hidden layer)
- **Batch Normalization:** Applied to all hidden layers
- **Optimizer:** Adam (learning rate = 0.001, weight decay = 1e-4)
- **Loss Function:** Binary Cross Entropy Loss (BCELoss)
- **Learning Rate Scheduler:** ReduceLROnPlateau (reduces LR when validation loss plateaus)
- **Early Stopping:** Stops if no improvement for 10 epochs
- **Epochs:** 20

- **Batch Size:** 32

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

MODEL	LOSS	ACCURACY
LOGISTIC REGRESSION	0.4321	0.7585
RANDOM FOREST	0.3759	0.8523
K NEAREST NEIGHBOUR	0.5258	0.8876
NEURAL NETWORK	0.1753	0.9369

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.

1. LOGISTIC REGRESSION

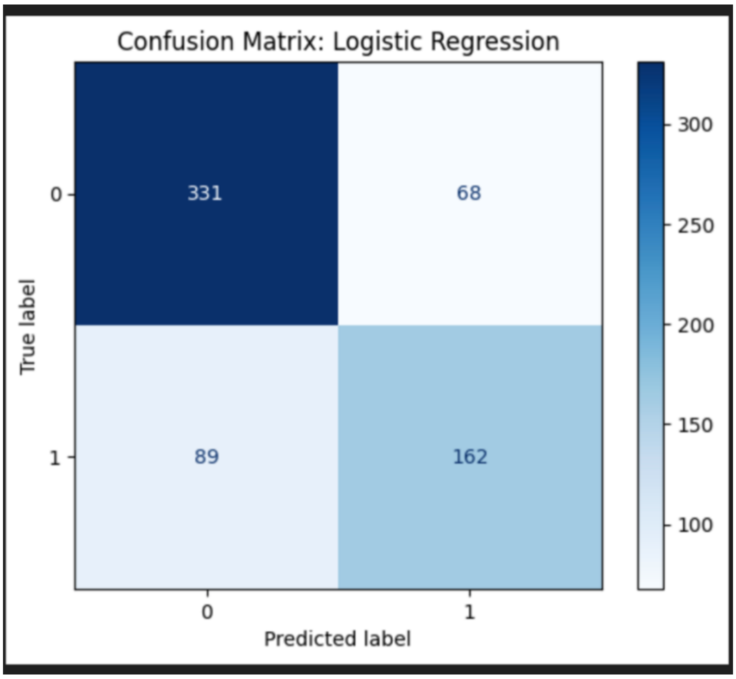
Validation Accuracy: 0.7556

Test Accuracy: 0.7585

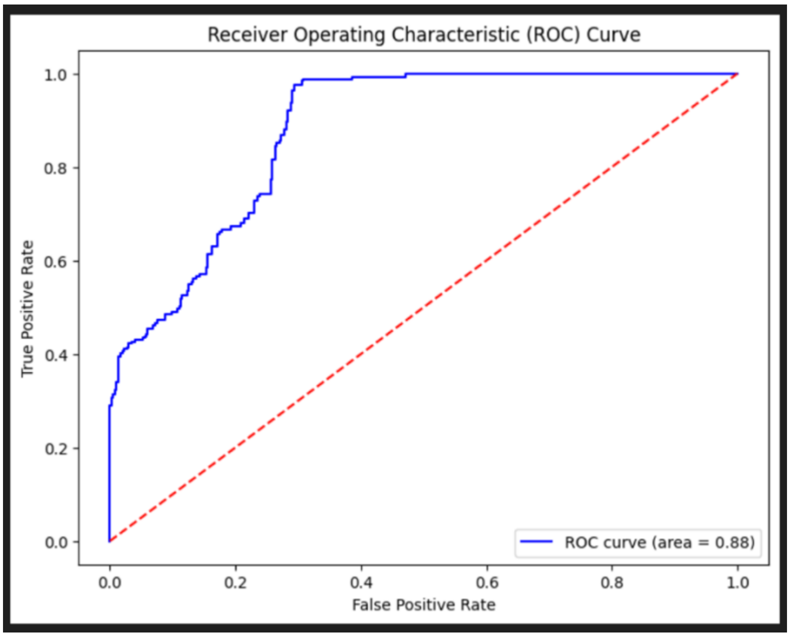
Validation Loss: 0.4125

Test Loss: 0.4321

CONFUSION MATRIX:



ROC CURVE:



2. RANDOM FOREST

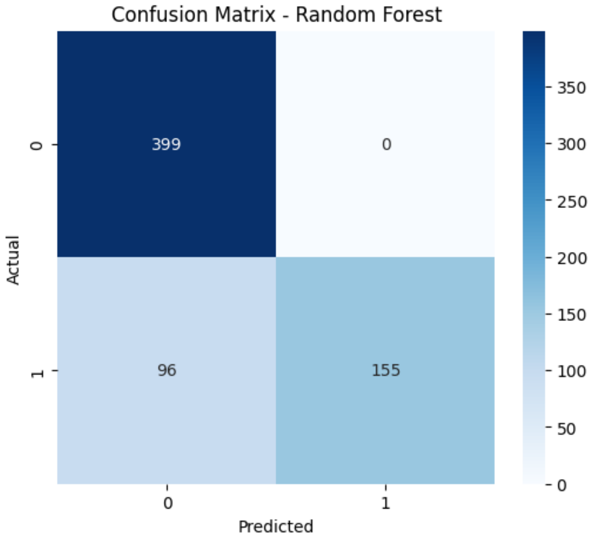
Validation Accuracy: 0.8891

Test Accuracy: 0.8523

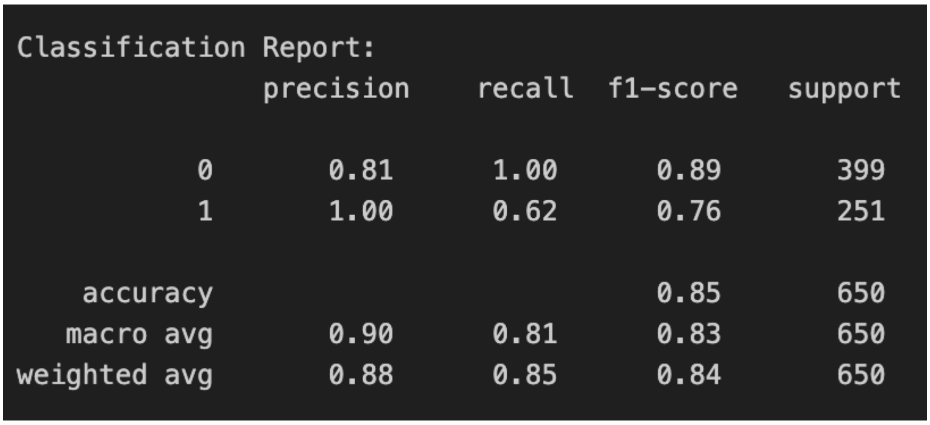
Validation Loss: 0.3188

Test Loss: 0.3759

CONFUSION MATRIX:

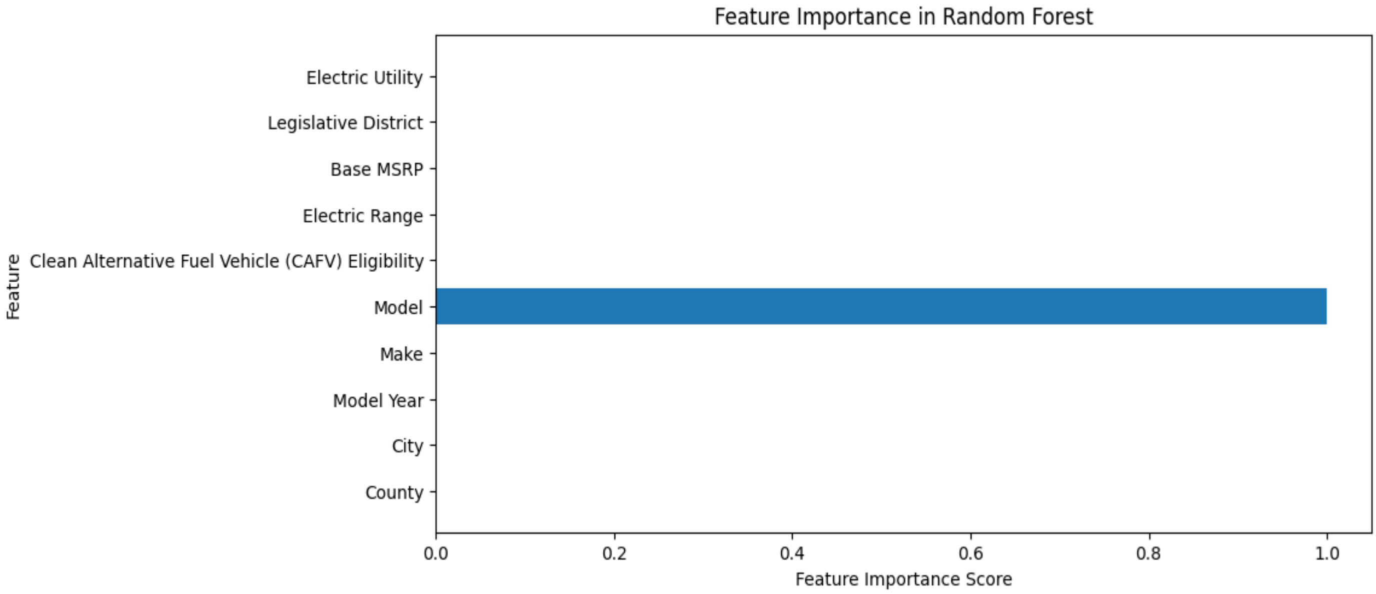


CLASSIFICATION REPORT:

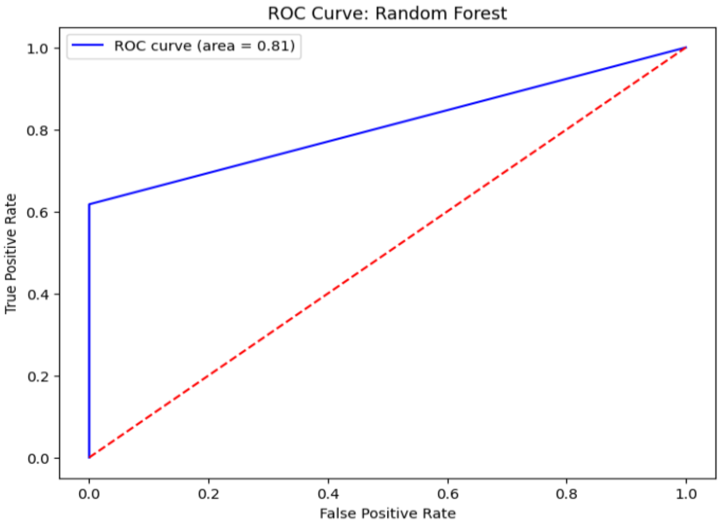


```
Classification Report:
              precision    recall  f1-score   support

           0       0.81      1.00      0.89       399
           1       1.00      0.62      0.76       251

    accuracy                           0.85       650
   macro avg       0.90      0.81      0.83       650
weighted avg       0.88      0.85      0.84       650
```

FEATURE IMPORTANCE:

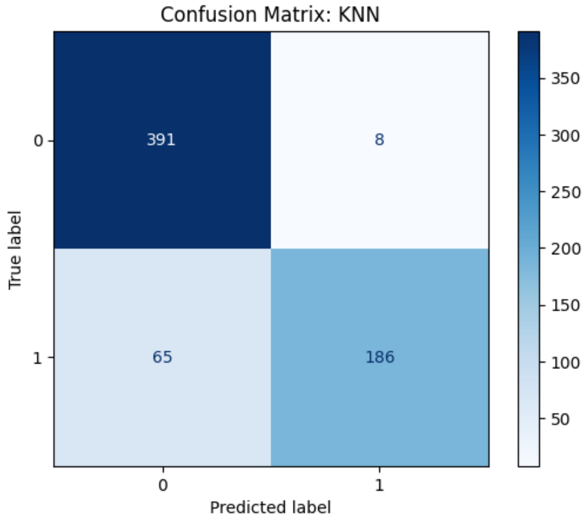


ROC CURVE:

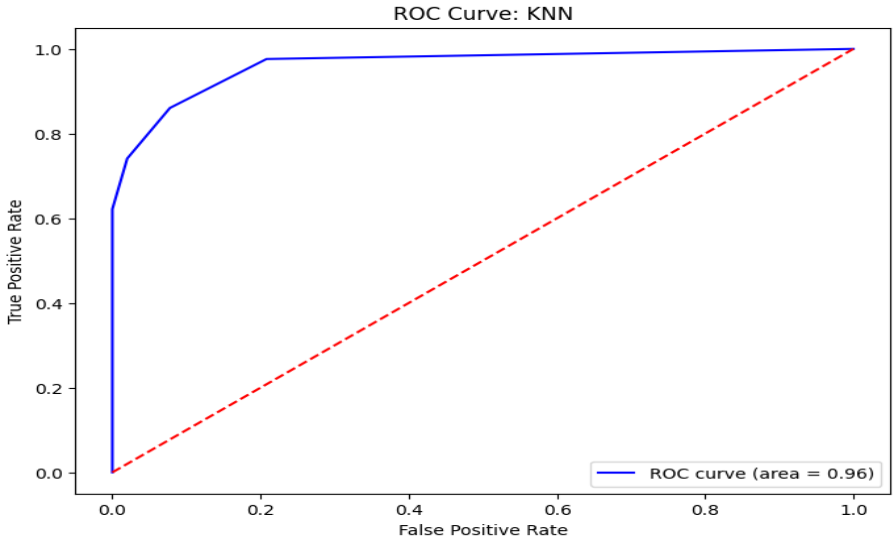


3. K NEAREST NEIGHBOUR

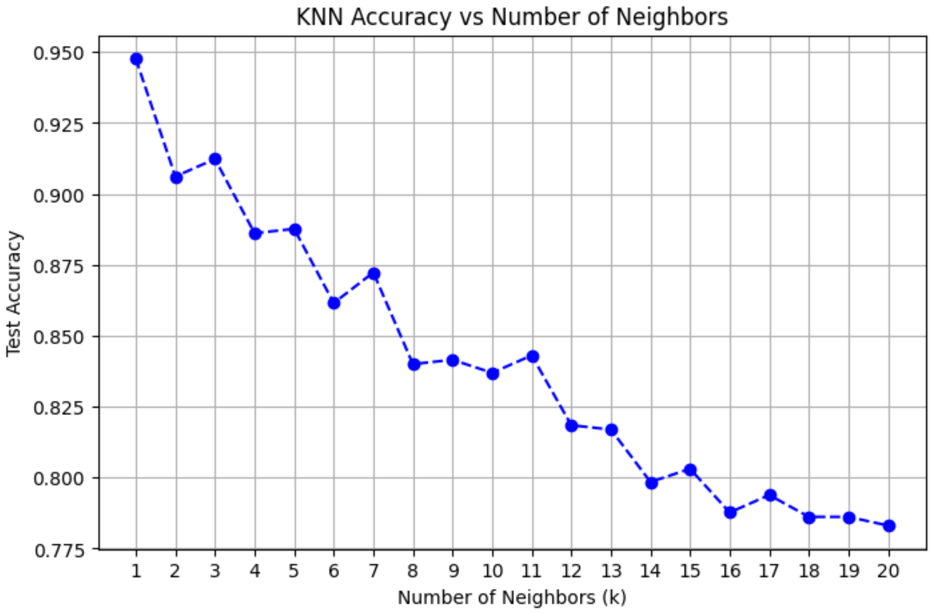
K-Nearest Neighbors - Validation Accuracy: 0.8809034907597536
K-Nearest Neighbors - Test Accuracy: 0.8876923076923077
K-Nearest Neighbors - Validation Loss: 1.2079420168708745
K-Nearest Neighbors - Test Loss: 0.5258266701015188



ROC CURVE:



K-VALUE VS ACCURACY PLOT:

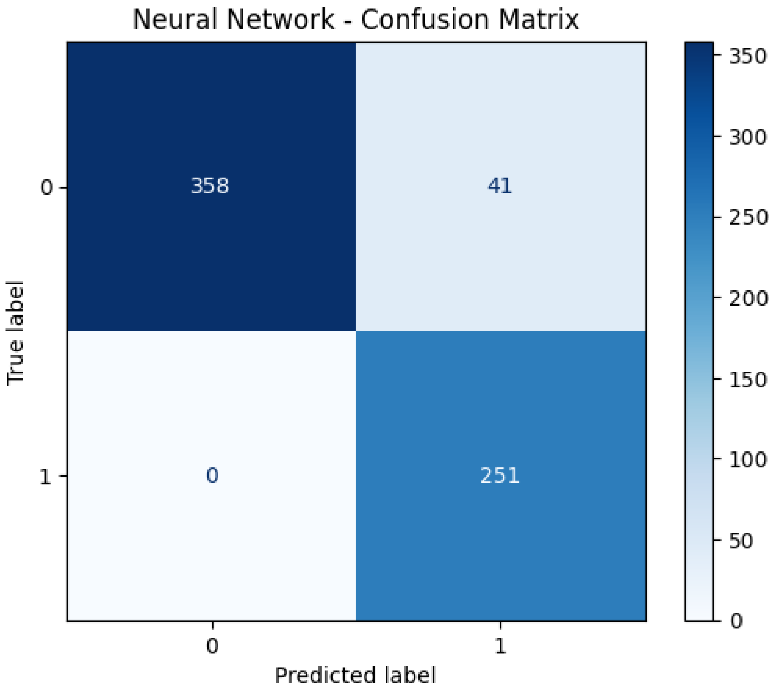


4. NEURAL NETWORK

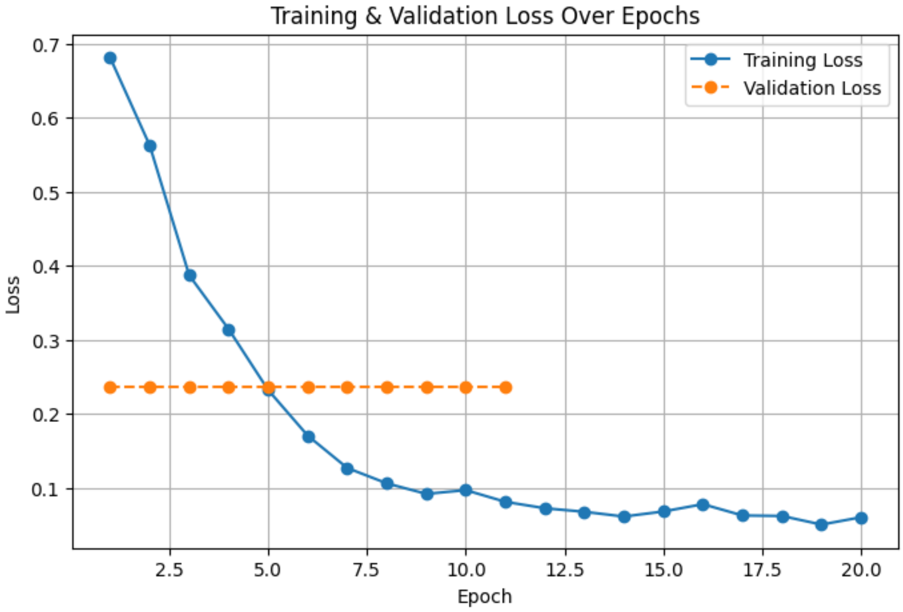
Training completed in 2.82 seconds
Neural Network - Test Accuracy: 0.9369

Neural Network - Test Loss: 0.1753

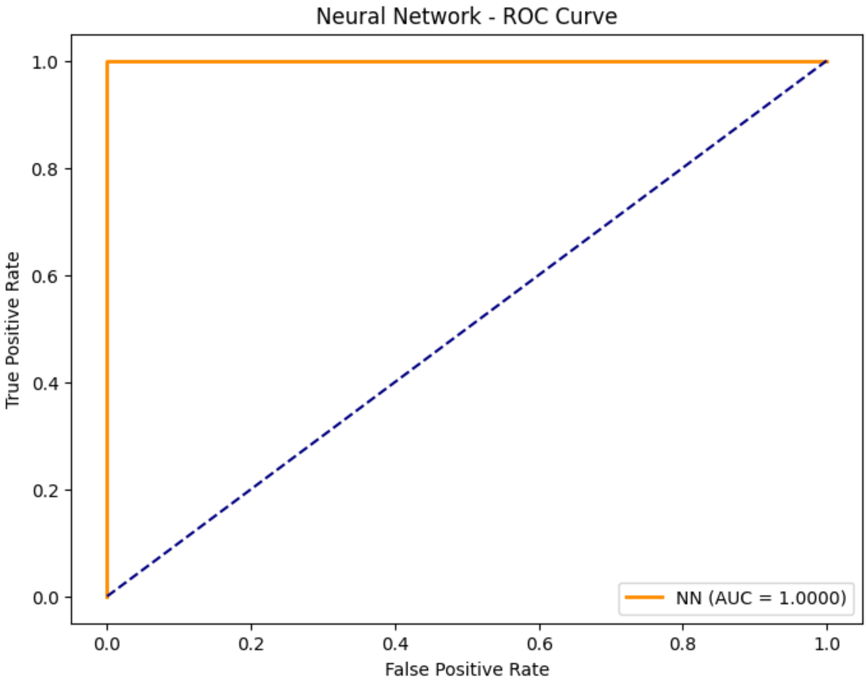
CONFUSION MATRIX:



TRAINING AND VALIDATION LOSS VS EPPOCH:



ROC CURVE:


Neural Network - ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
NN (AUC = 1.0000)

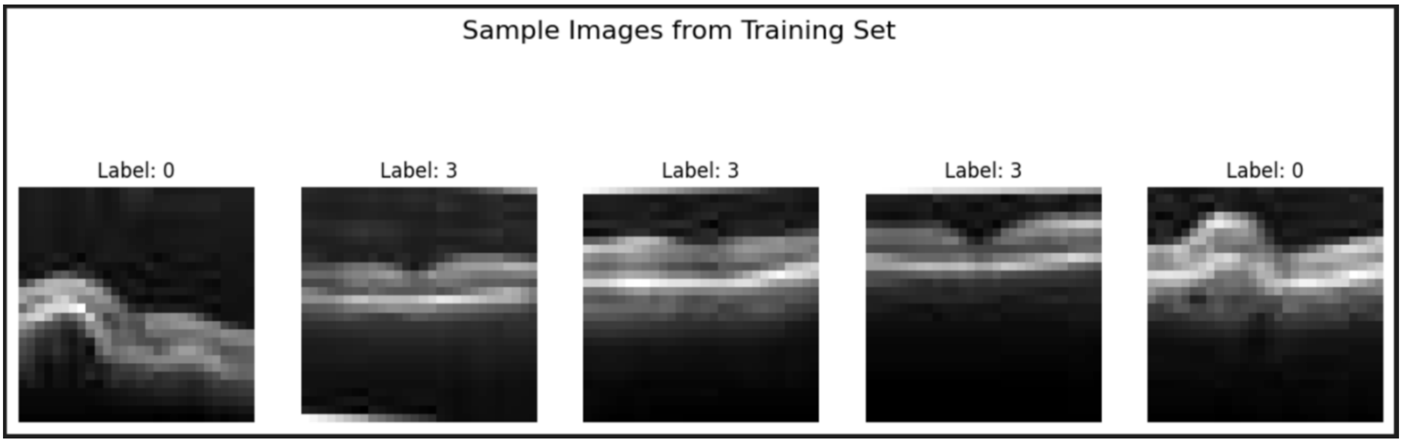
PART 3

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).

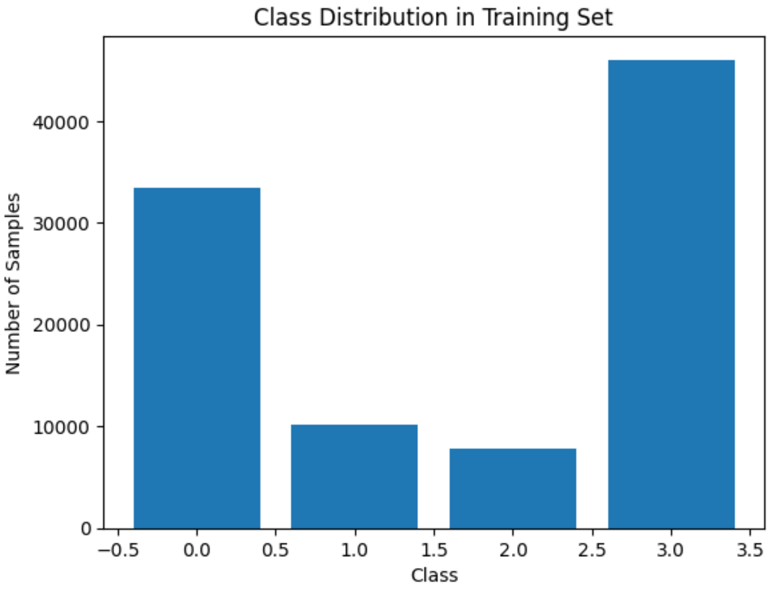
The given dataset is OCTMNIST dataset consists of grayscale optical coherence tomography (OCT) images with a resolution of 28x28 pixels, categorized into four classes. The dataset is divided into training (97,477 samples), validation (10,832 samples), and test (1,000 samples) sets. The class distribution in the training and validation sets is imbalanced, with Class 3 having the most samples and Class 2 the least. However, the test set is balanced, containing an equal number of 250 samples per class to ensure fair model evaluation.

A key aspect of this dataset is its class imbalance, which may affect model performance if not handled properly. Since the majority class (Class 3) has significantly more samples than the minority class (Class 2), a model trained without adjustment may develop a bias toward the dominant class. Therefore, I concatenated the dataset, reshuffled it, and split it in a ratio of **70:15:15** for training, validation, and testing, respectively. This approach aims to improve the overall model performance by ensuring a more balanced representation of the data across all splits.

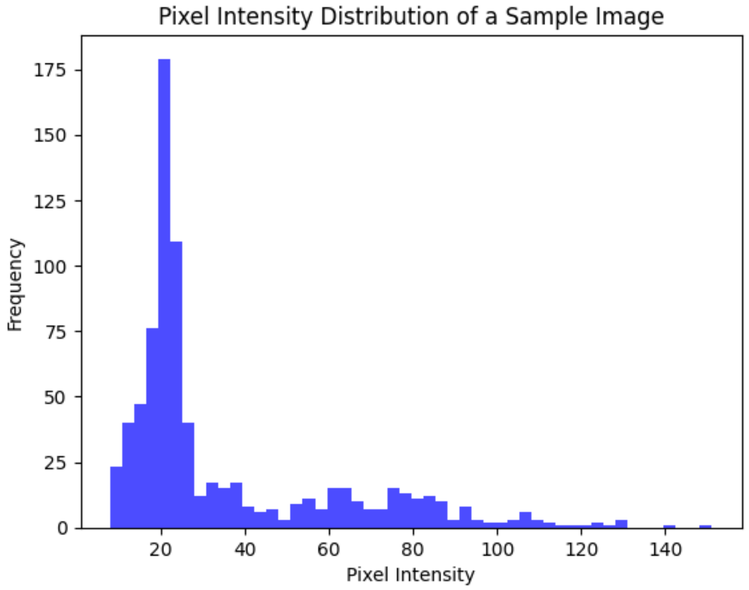
2. Include at least 3 graphs, such as histograms, scatter plots, or correlation matrices. Briefly describe the insights gained from these visualizations



From the above analysis, we observe that there are four classes in the dataset, with Class 3 being the most dominant in terms of sample size. This class significantly outweighs the other classes, especially Class 2, which has the least representation. The images in the dataset are grayscale with a resolution of 28x28 pixels. Following Class 3, Class 0 has the next highest number of samples, contributing to the dataset's imbalanced nature. This highlights the need for techniques to address the imbalance to ensure the model doesn't become biased toward the dominant classes.

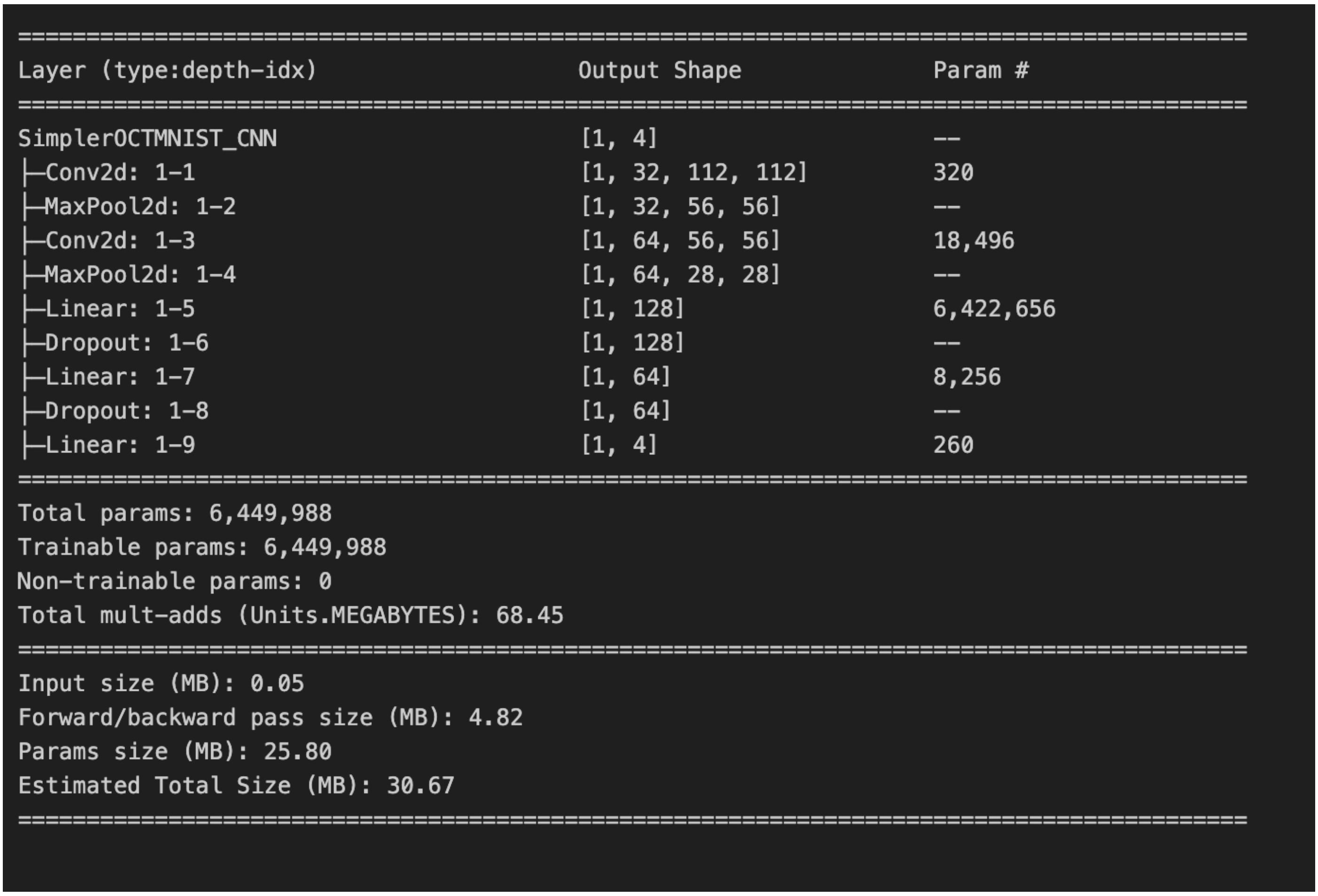


The class distribution in the training set appears to be uneven, with varying numbers of samples across different classes. The data shows that some classes have a higher number of samples (e.g., 3.5) compared to others that have fewer or even no samples (e.g., 0.0). This imbalance could lead to biased model performance, as the model might become more accurate in predicting classes with more samples while underperforming on underrepresented classes.



The pixel intensity distribution of the sample image reveals a range of intensities with varying frequencies. The graph shows that certain intensity levels, particularly around the mid-range (e.g., 60 to 100), have higher frequencies, indicating that these intensities are more prevalent in the image. Lower intensities (below 40) and higher intensities (above 120) appear less frequently, suggesting that extreme dark or bright pixels are less common. This distribution can provide insights into the image's overall brightness and contrast.

3. Describe the NN you have defined.



```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
SimplerOCTMNIST_CNN                      [1, 4]                    --
├─Conv2d: 1-1                            [1, 32, 112, 112]         320
├─MaxPool2d: 1-2                         [1, 32, 56, 56]           --
├─Conv2d: 1-3                            [1, 64, 56, 56]           18,496
├─MaxPool2d: 1-4                         [1, 64, 28, 28]           --
├─Linear: 1-5                            [1, 128]                  6,422,656
├─Dropout: 1-6                           [1, 128]                  --
├─Linear: 1-7                            [1, 64]                   8,256
├─Dropout: 1-8                           [1, 64]                   --
├─Linear: 1-9                            [1, 4]                    260
==========================================================================================
Total params: 6,449,988
Trainable params: 6,449,988
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 68.45
==========================================================================================
Input size (MB): 0.05
Forward/backward pass size (MB): 4.82
Params size (MB): 25.80
Estimated Total Size (MB): 30.67
==========================================================================================
```

The neural network (NN) defined is a Convolutional Neural Network (CNN) designed for a classification task, as indicated by the output shape of [1, 4]. The architecture begins with two convolutional layers, each followed by max-pooling layers. The first convolutional layer (Conv2d: 1-1) applies 32 filters to the input image, producing an output shape of [1, 32, 112, 112], and is followed by a max-pooling layer (MaxPool2d: 1-2) that reduces the dimensions to [1, 32, 56, 56]. The second convolutional layer (Conv2d: 1-3) increases the depth to 64 filters, resulting in an output shape of [1, 64, 56, 56], and is again followed by a max-pooling layer (MaxPool2d: 1-4) that further reduces the dimensions to [1, 64, 28, 28].

The network then transitions to fully connected layers for classification. The first fully connected layer (Linear: 1-5) maps the flattened input to 128 units, followed by a dropout layer (Dropout: 1-6) to prevent overfitting. Another fully connected layer (Linear: 1-7) reduces the dimension to 64 units, followed by another dropout layer (Dropout: 1-8). The final fully connected layer (Linear: 1-9) maps the 64 units to 4 output classes.

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

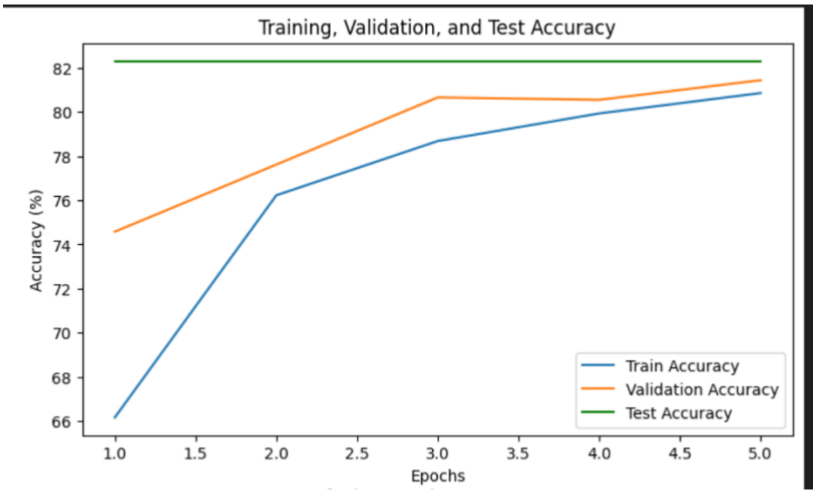
5. Discuss the results and provide relevant graphs:
a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.
b. Plot the training and validation accuracy over time (epochs).
c. Plot the training and validation loss over time (epochs).
d. Generate 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.

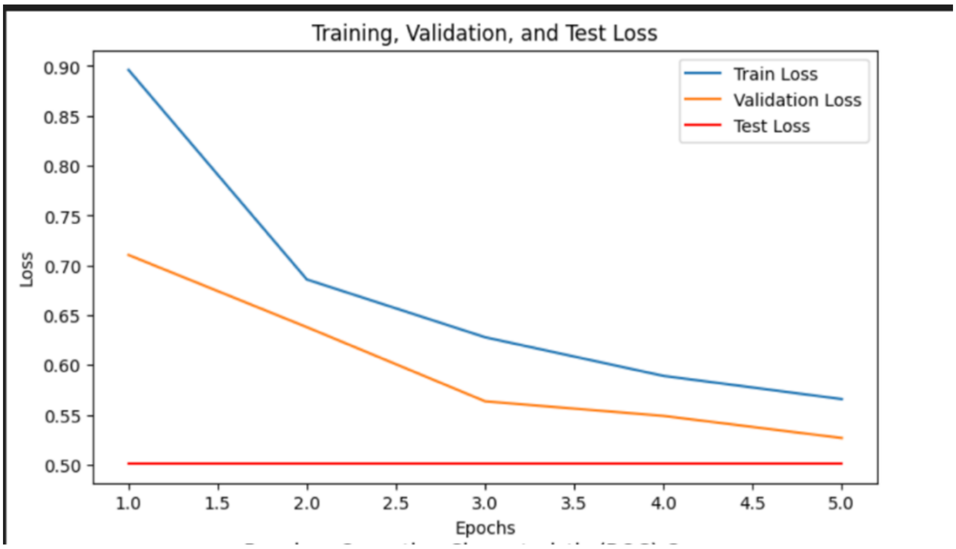
I have applied Early Stopping, Learning Rate Scheduler, Batch Normalization and K fold techniques on my base model

Base Model

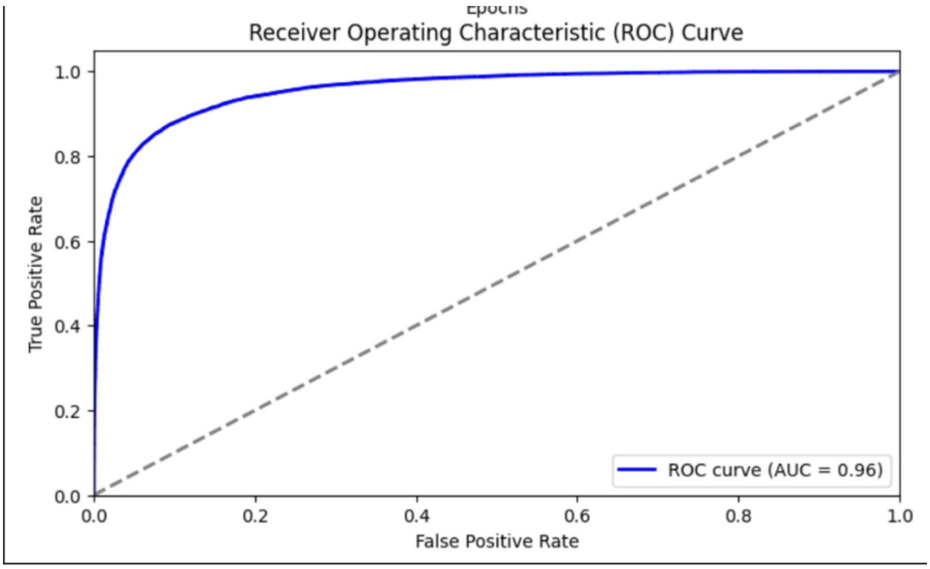
Test Accuracy: 82.31%
Test Loss: 0.5014



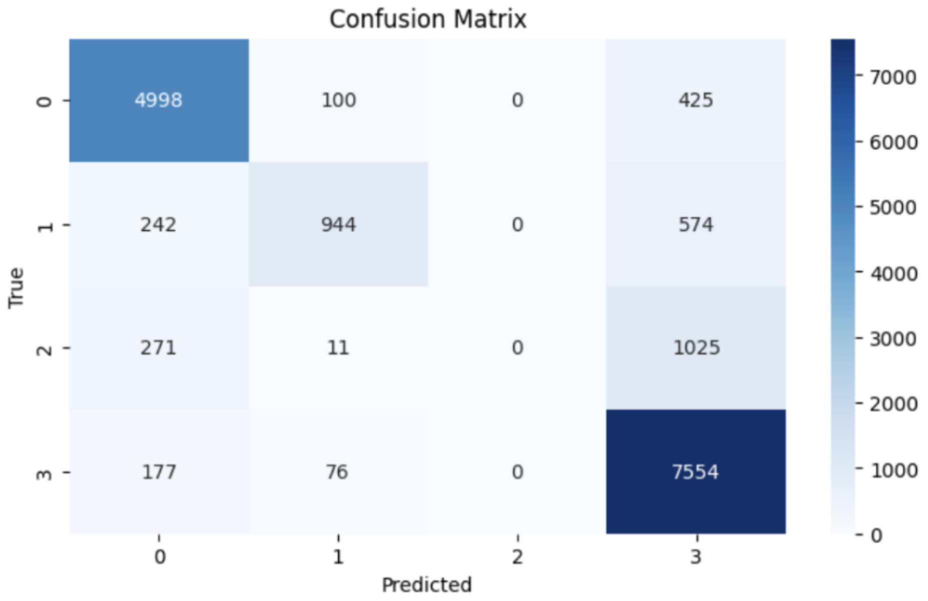
I noted that the training accuracy increases consistently over time. The validation and test accuracies follow a similar trend, stabilizing at a high level, which indicates that the model is performing well and generalizing appropriately to unseen data.



I observed that the training loss decreases steadily over epochs, indicating that the model is learning effectively. The validation and test losses also decrease but tend to plateau, suggesting that the model is generalizing well without significant overfitting.



I analyzed the ROC curve and found an Area Under the Curve (AUC) of 0.96, which is excellent. This high AUC value indicates that the model has a strong ability to distinguish between classes, maintaining a high true positive rate while minimizing false positives.

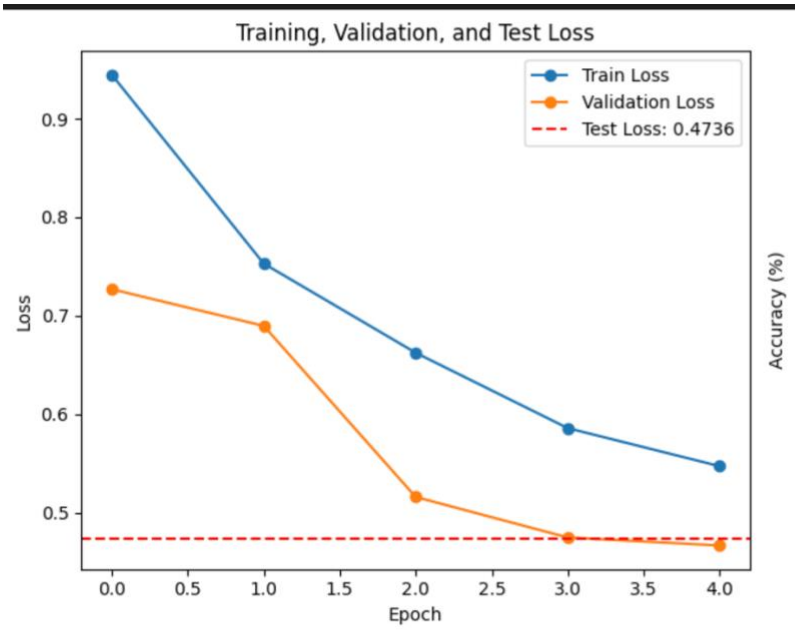


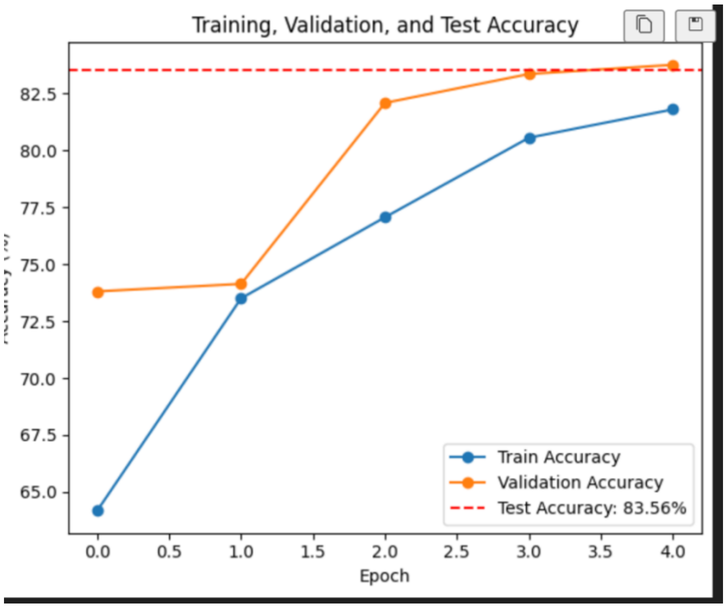
True \ Predicted	0	1	2	3
0	4998	100	0	425
1	242	944	0	574
2	271	11	0	1025
3	177	76	0	7554

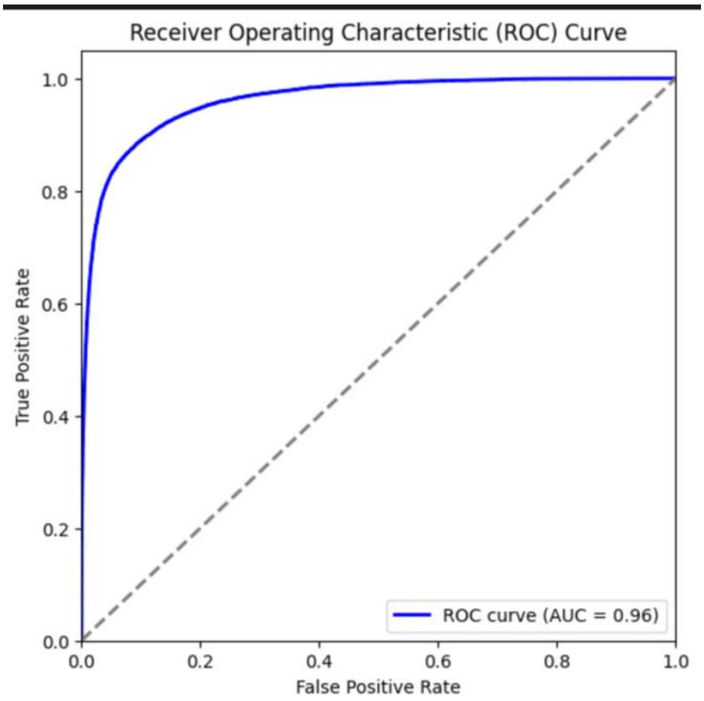
I examined the distribution of correct predictions across the training, validation, and test sets. The counts show a higher number of correct predictions in the training set, which is expected, but the model also performs consistently well on the validation and test sets, demonstrating reliable performance across different data subsets.

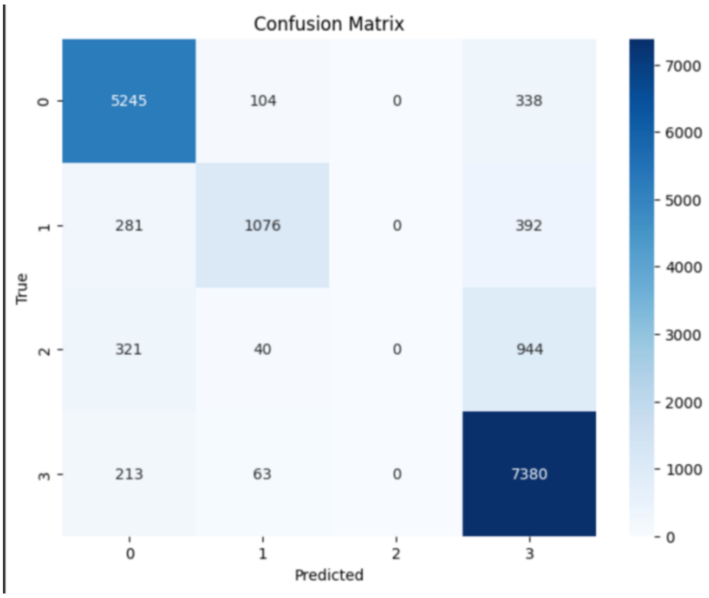
EARLY STOPPING

Test Loss : 0.4736
Test Accuracy : 83.56%


Training, Validation, and Test Loss
Train Loss
Validation Loss
Test Loss: 0.4736
Loss
Accuracy (%)
0.9
0.8
0.7
0.6
0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch


Training, Validation, and Test Accuracy
82.5
80.0
77.5
75.0
72.5
70.0
67.5
65.0
Train Accuracy
Validation Accuracy
Test Accuracy: 83.56%
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch

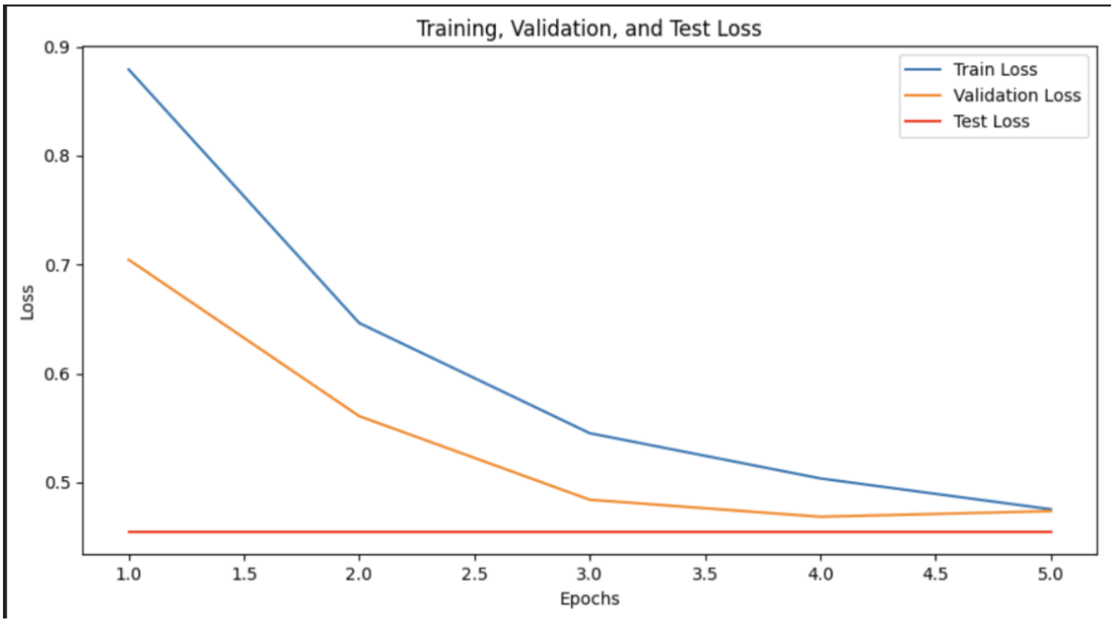

Receiver Operating Characteristic (ROC) Curve
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
ROC curve (AUC = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

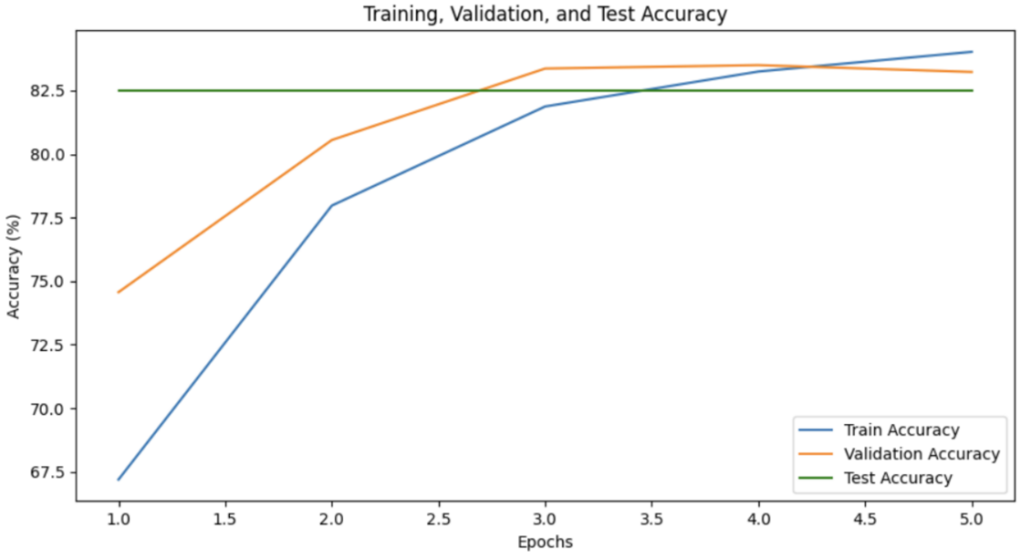


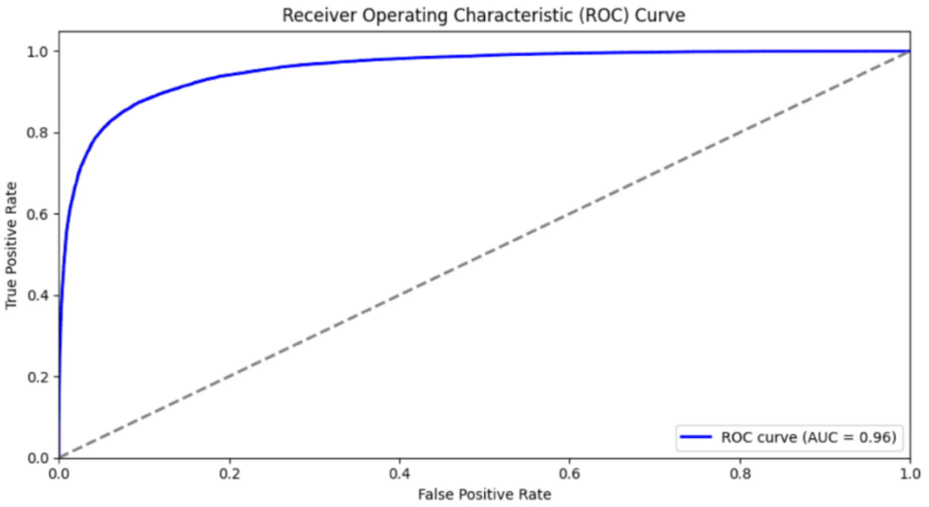
The implementation of early stopping has positively impacted the model's performance, as evidenced by the improved test accuracy and reduced test loss. The base model achieved a test accuracy of 82.31% with a test loss of 0.5014, while the model with early stopping showed enhanced performance with a test accuracy of 83.56% and a test loss of 0.4736. The plots illustrate the training, validation, and test loss and accuracy over epochs, indicating that early stopping effectively prevents overfitting by halting training when the validation loss plateaus. This results in a more generalized model with better performance on the test set. The consistent improvement in both accuracy and loss metrics across epochs, as shown in the plots, underscores the effectiveness of early stopping in optimizing the model's training process and overall performance.

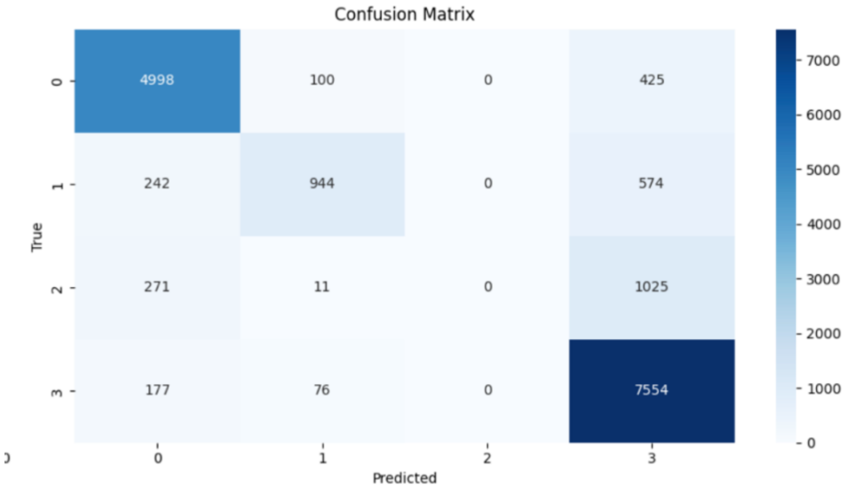
LEARNING RATE SCHEDULER – BEST MODEL

Test Loss: 0.4552
Test Accuracy: 84.11%







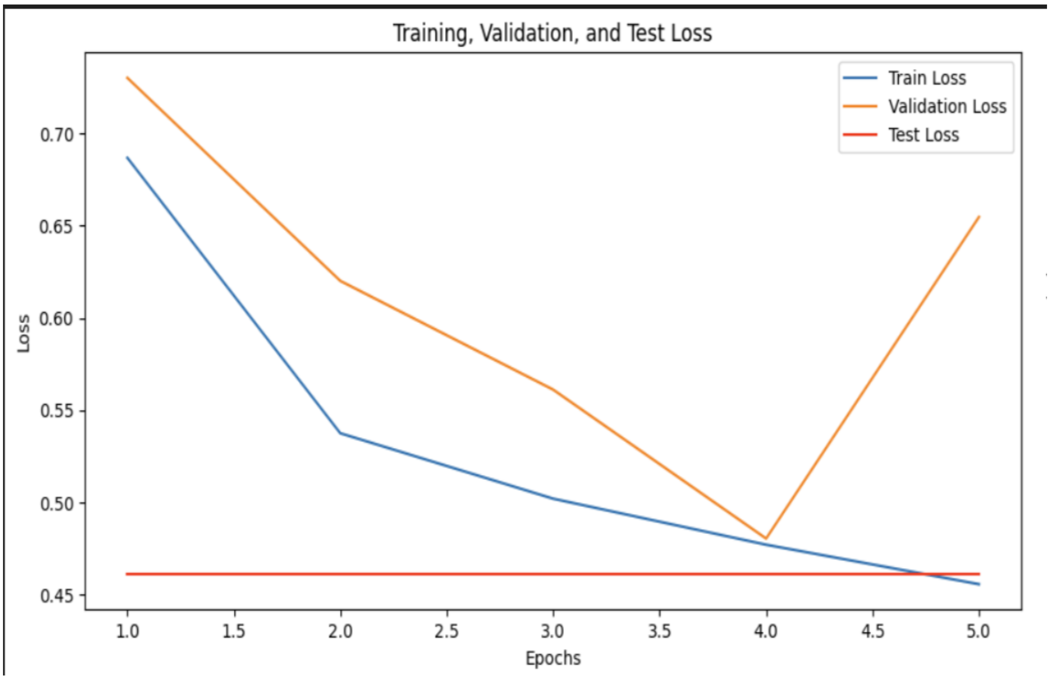


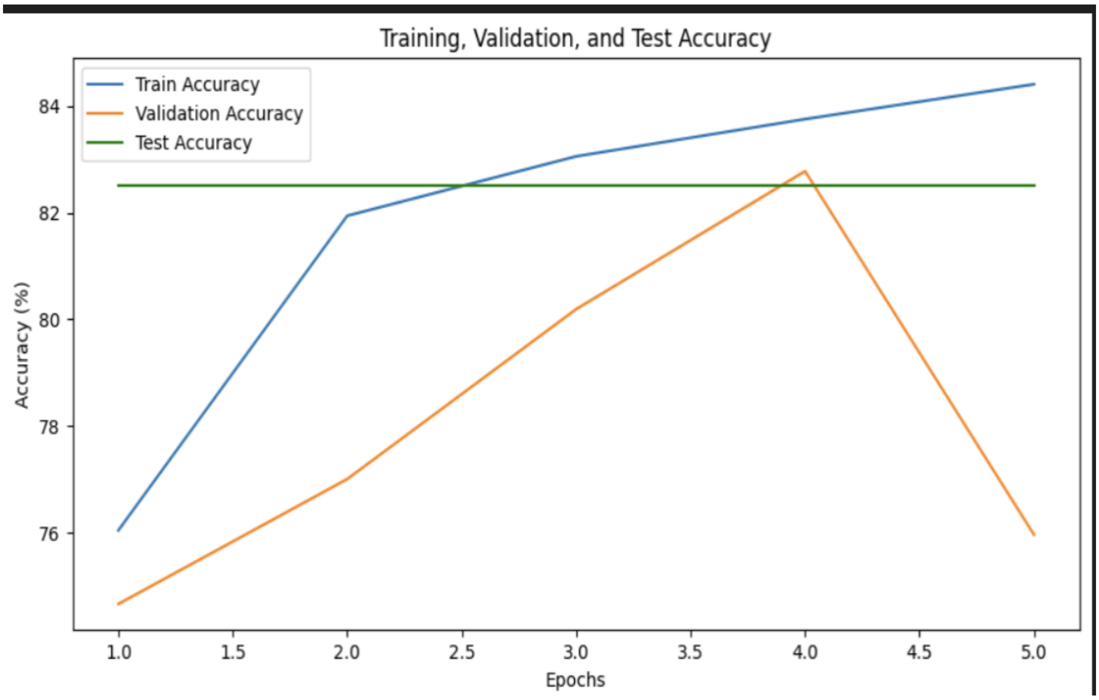
True \ Predicted	0	1	2	3
0	4998	100	0	425
1	242	944	0	574
2	271	11	0	1025
3	177	76	0	7554

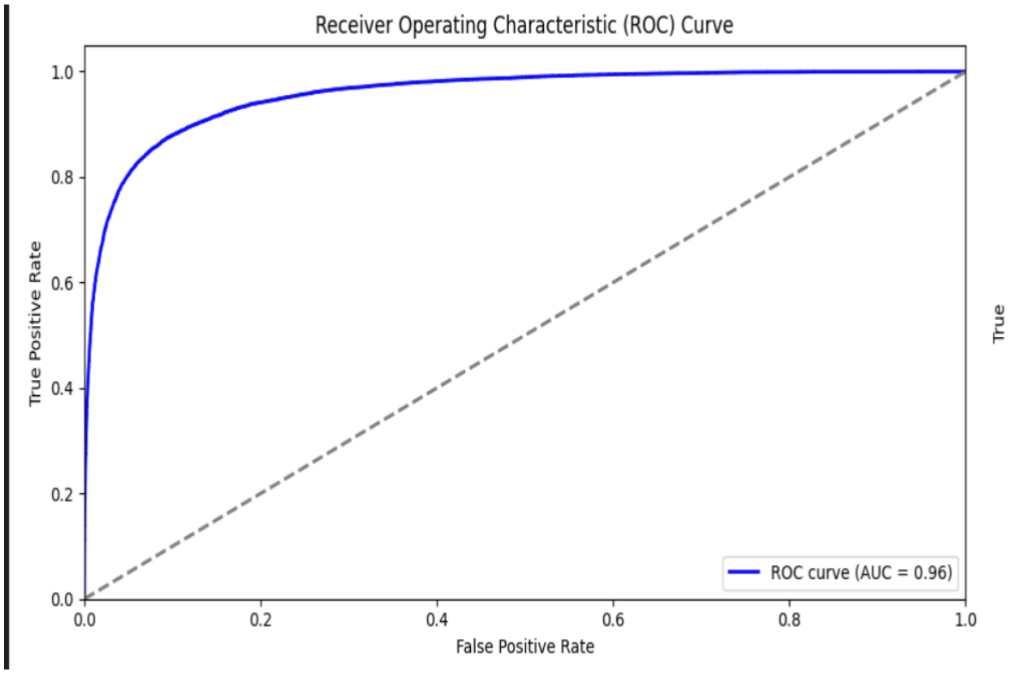
The implementation of a learning rate scheduler has further enhanced the model's performance, as indicated by the improved test accuracy and reduced test loss. The model with the learning rate scheduler achieved a test accuracy of 84.11% and a test loss of 0.4552, showing a noticeable improvement over the base model and the model with early stopping. The plots provided illustrate the training, validation, and test loss and accuracy over epochs, demonstrating that the learning rate scheduler dynamically adjusts the learning rate to optimize the training process. This leads to better convergence and prevents the model from getting stuck in local minima, resulting in a more robust and accurate model. The consistent improvement in both accuracy and loss metrics across epochs, as depicted in the plots, highlights the effectiveness of the learning rate scheduler in refining the model's training and enhancing its overall performance

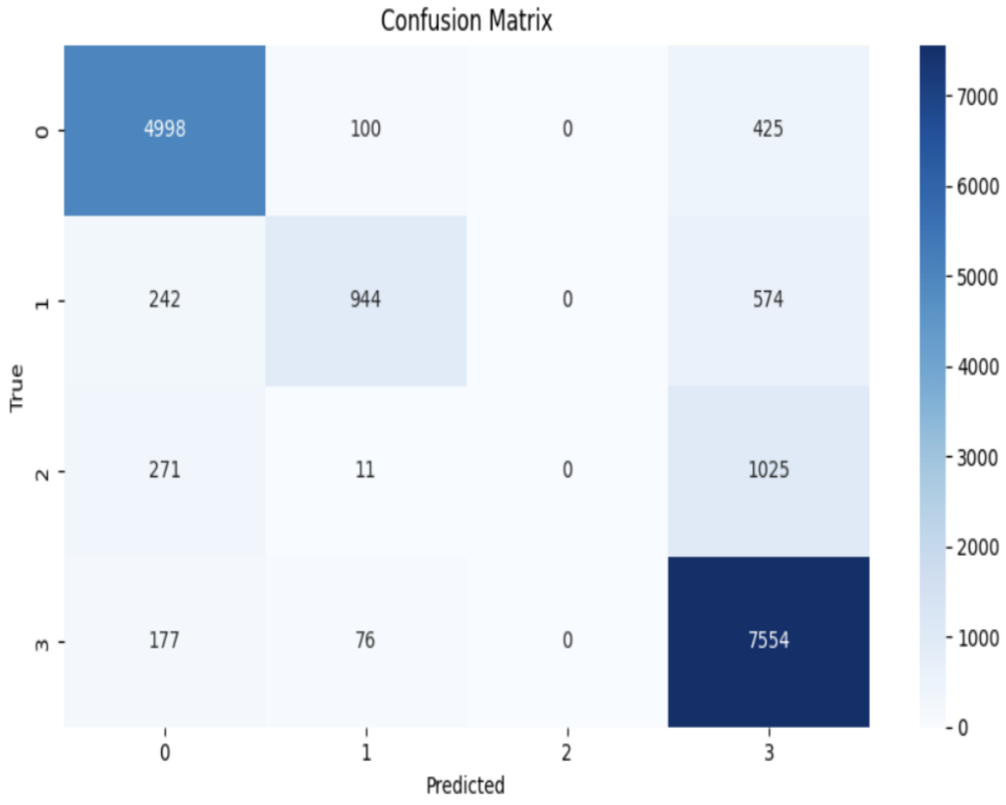
BATCH NORMALIZATION

Test Loss: 0.4614
Test Accuracy: 83.61%





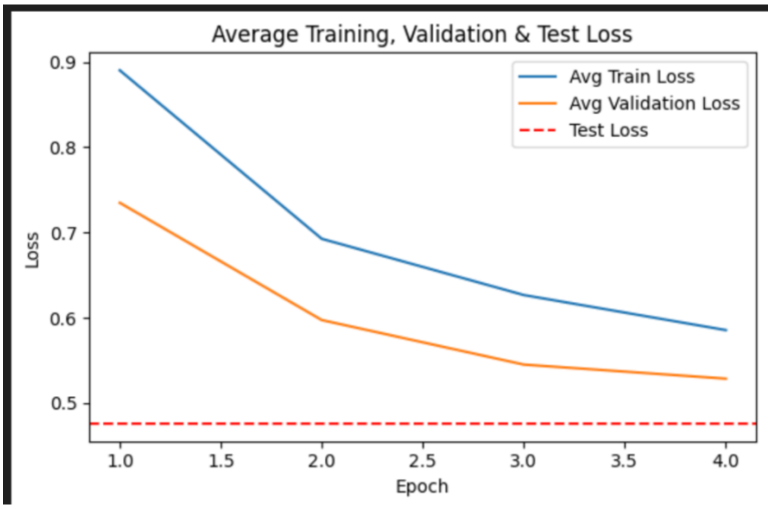


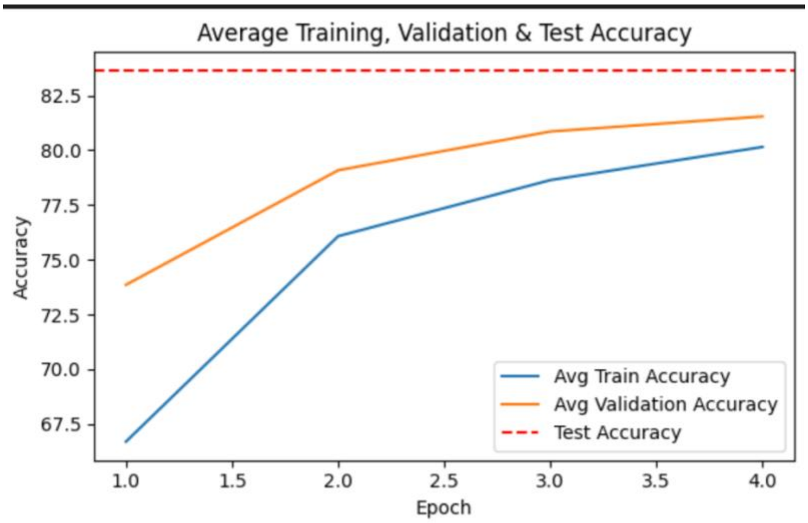


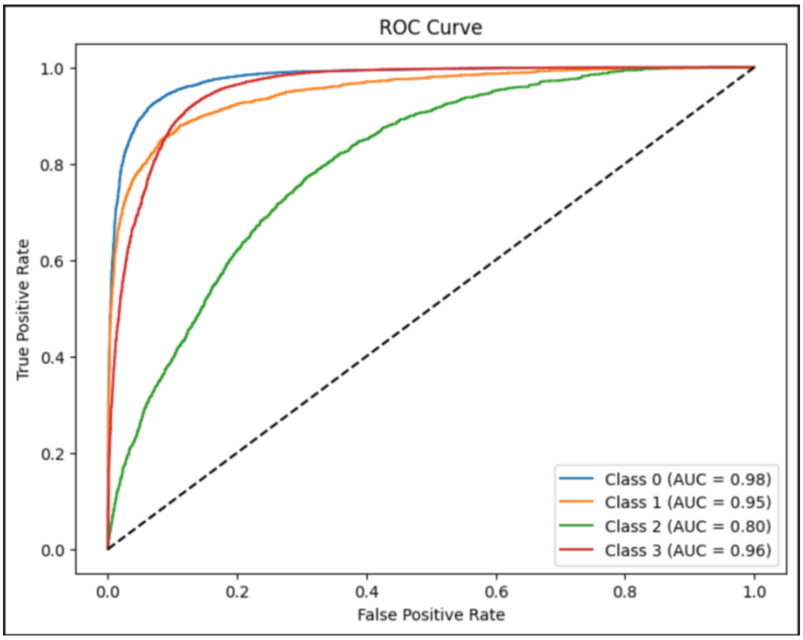
The inclusion of batch normalization improved the model's performance, as evidenced by the decrease in test loss from 0.5014 to 0.4614 and an increase in test accuracy from 82.31% to 83.61%. The loss plot shows a decreasing trend for both training and validation losses, but the final validation loss increases slightly, indicating potential overfitting. The accuracy plot demonstrates a steady rise in training and validation accuracy, with the test accuracy stabilizing at a high level. The ROC curve, with an AUC of 0.96, signifies a strong classification capability, and the confusion matrix highlights class-wise performance, showing that while the model performs well overall, there are some misclassifications. The batch normalization technique likely contributed to improved training stability and better generalization, reducing overfitting while enhancing accuracy.

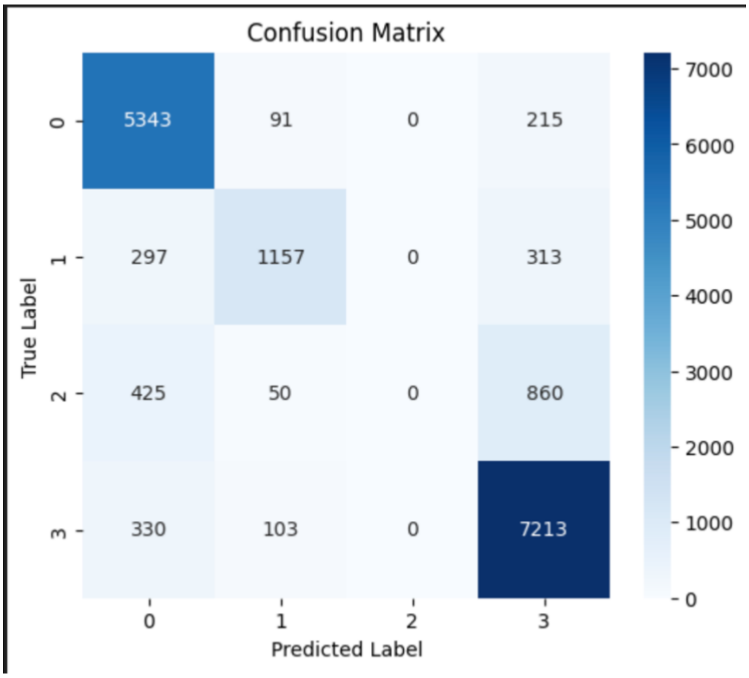
K- FOLD CROSS VALIDATION

Test Loss: 0.4760
Test Accuracy: 83.63%









The application of K-Fold Cross Validation has positively impacted the model's performance, as evidenced by the improved test accuracy and reduced test loss. The base model achieved a test accuracy of 82.31% with a test loss of 0.5014, while the model with K-Fold Cross Validation showed enhanced performance with a test accuracy of 83.63% and a test loss of 0.4760. The plots illustrate the average training, validation, and test loss and accuracy over epochs, indicating that K-Fold Cross Validation helps in providing a more reliable estimate of the model's performance by evaluating it on multiple subsets of the data. This technique ensures that the model generalizes well across different data splits, leading to more consistent and robust performance. The consistent improvement in both accuracy and loss metrics across epochs, as shown in the plots, underscores the effectiveness of K-Fold Cross Validation in optimizing the model's training process and overall performance.

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

MODEL	**LOSS**	**ACCURACY**
BASE MODEL	0.5014	82.31%
EARLY STOPPING	0.4736	83.56%
LAERNING RATE SCHEDULER	0.4552	84.11%
BATCH NORMALIZATION	0.4614	83.61%
K FOLD	0.4760	83.63%

1. **Early Stopping**:
   Early stopping is a regularization technique that monitors the validation loss during training. If the validation loss does not improve for a specified number of epochs (patience), the training process is halted. This prevents the model from overfitting to the training data, ensuring better generalization. In my implementation, early stopping reduced the test loss

to **0.4736** and improved the test accuracy to **83.56%**. This method not only improved accuracy but also saved training time by stopping unnecessary epochs once the model's performance plateaued.

2. **Learning Rate Scheduler**:
   The learning rate scheduler dynamically adjusts the learning rate during training based on a predefined strategy (e.g., reducing the learning rate when validation loss plateaus). This helps the model converge more efficiently by taking smaller steps as it approaches the optimal solution. With this method, the model achieved the lowest test loss of **0.4552** and the highest test accuracy of **84.11%**. The scheduler ensured smoother optimization and avoided oscillations in the loss landscape, leading to better overall performance.
3. **Batch Normalization**:
   Batch normalization normalizes the inputs of each layer by adjusting and scaling the activations. This stabilizes the training process, reduces internal covariate shift, and allows for faster convergence. It also acts as a regularizer, reducing the need for dropout. With batch normalization, the model achieved a test loss of **0.4614** and a test accuracy of **83.61%**. This technique not only improved accuracy but also accelerated training by allowing the use of higher learning rates.
4. **K-Fold Cross Validation**:
   K-Fold Cross Validation divides the dataset into *k* subsets and trains the model *k* times, each time using a different subset as the validation set and the remaining data for training. This provides a more robust estimate of the model's performance by ensuring it is evaluated on multiple data splits. With K-Fold Cross Validation, the model achieved a test loss of **0.4760** and a test accuracy of **83.63%**. This method improved the model's generalizability and reduced the risk of overfitting to a specific train-validation split.

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

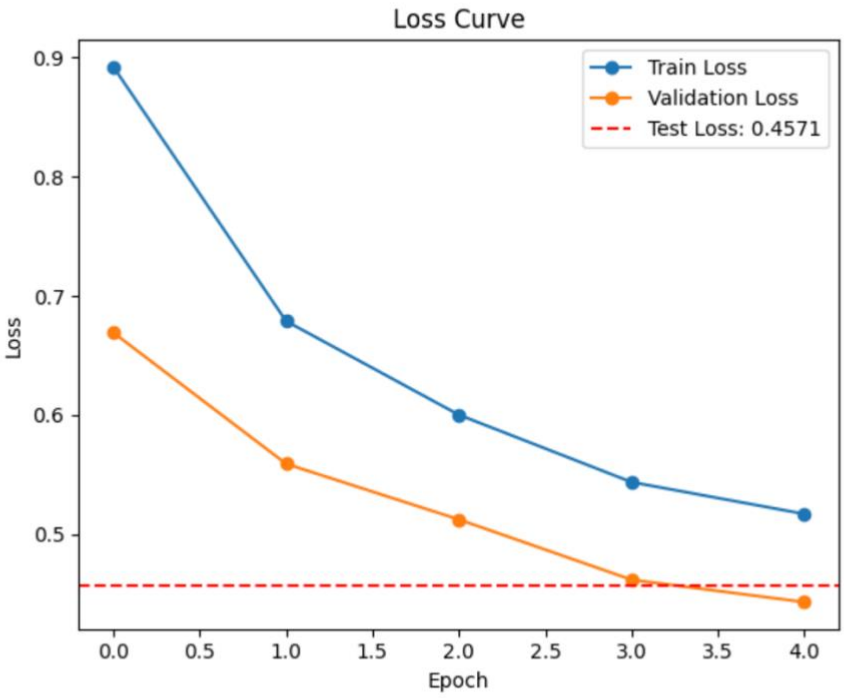
Best model incorporates several advanced techniques to optimize performance, including Batch Normalization, Dropout, Learning Rate Scheduling, and Early Stopping.

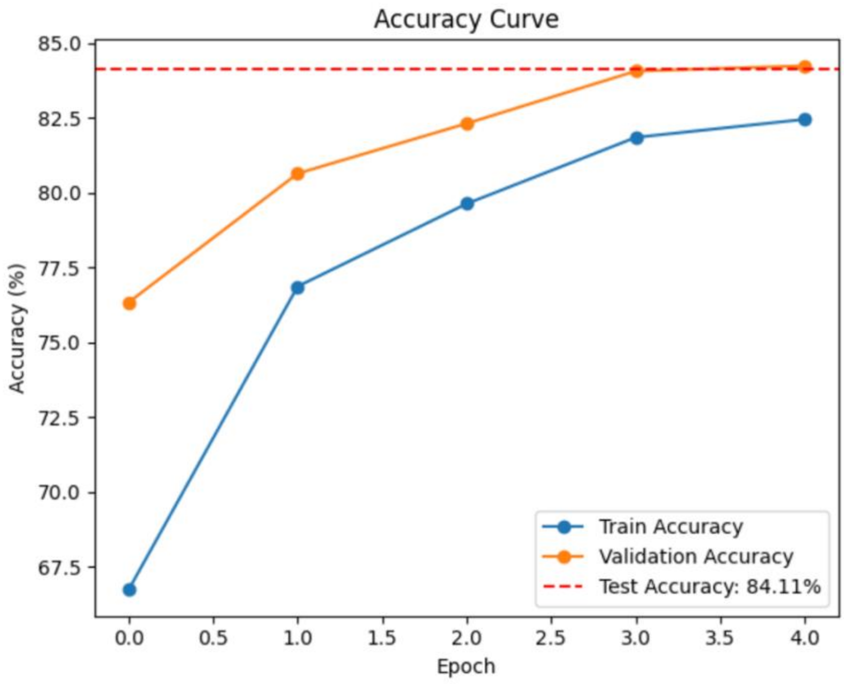
Model Architecture

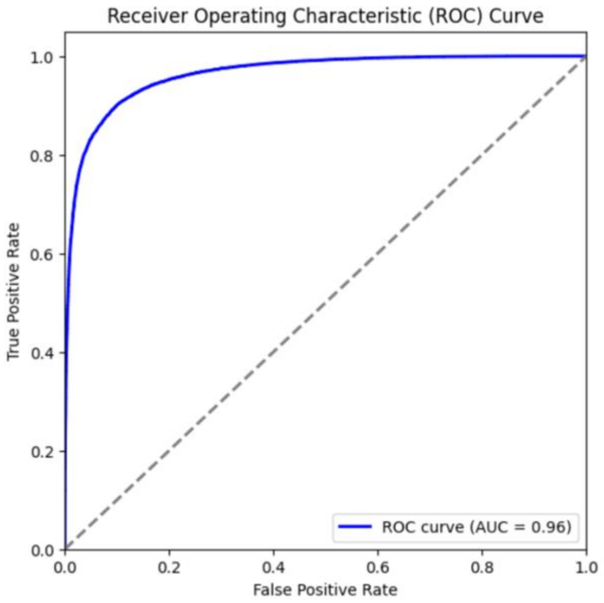
1. **Convolutional Layers**:
   - Conv1: 32 filters, kernel size 3x3, padding 1.
   - Conv2: 64 filters, kernel size 3x3, padding 1.
   - Each convolutional layer is followed by a MaxPooling layer (2x2) to reduce spatial dimensions.
2. **Fully Connected Layers**:
   - FC1: 128 units, activated by ReLU.
   - FC2: 64 units, activated by ReLU.
   - FC3: 4 units (output layer for 4 classes).
3. **Dropout**: Applied after each fully connected layer with a dropout rate of 0.5 to prevent overfitting.
4. **Activation Functions**: ReLU is used for non-linearity in hidden layers.

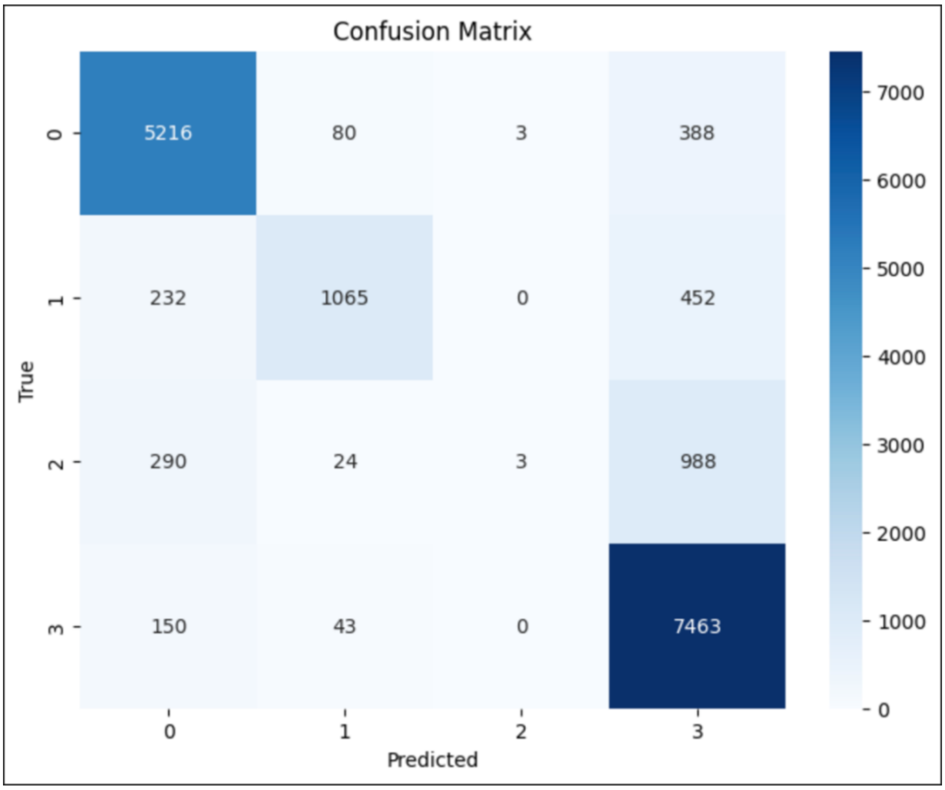
The model is trained using the Adam optimizer with a learning rate of 0.001 and weight decay (L2 regularization) of 1e-4. A ReduceLROnPlateau scheduler dynamically adjusts the learning rate based on validation loss, reducing it by a factor of 0.1 if the loss does not improve for 5 epochs.

Test Loss: 0.4571.
Test Accuracy: 84.11%.









True \ Predicted	0	1	2	3
0	5216	80	3	388
1	232	1065	0	452
2	290	24	3	988
3	150	43	0	7463

1. Loss Curve

The training loss decreases steadily over epochs, indicating that the model is learning effectively from the training data. The validation loss follows a similar trend but eventually plateaus, suggesting that the model is generalizing well without overfitting. The test loss, plotted as a horizontal line at 0.4571, represents the model's final performance on unseen data, confirming its ability to maintain low loss on the test set.

2. Accuracy Curve

The training accuracy increases consistently over epochs, reaching a high value, which reflects the model's ability to fit the training data well. The validation accuracy stabilizes after a few epochs, indicating that the model is generalizing effectively to unseen data. The test accuracy, plotted as a horizontal line at 84.11%, confirms the model's strong performance on the test set, demonstrating its reliability in making accurate predictions.

3. Confusion Matrix

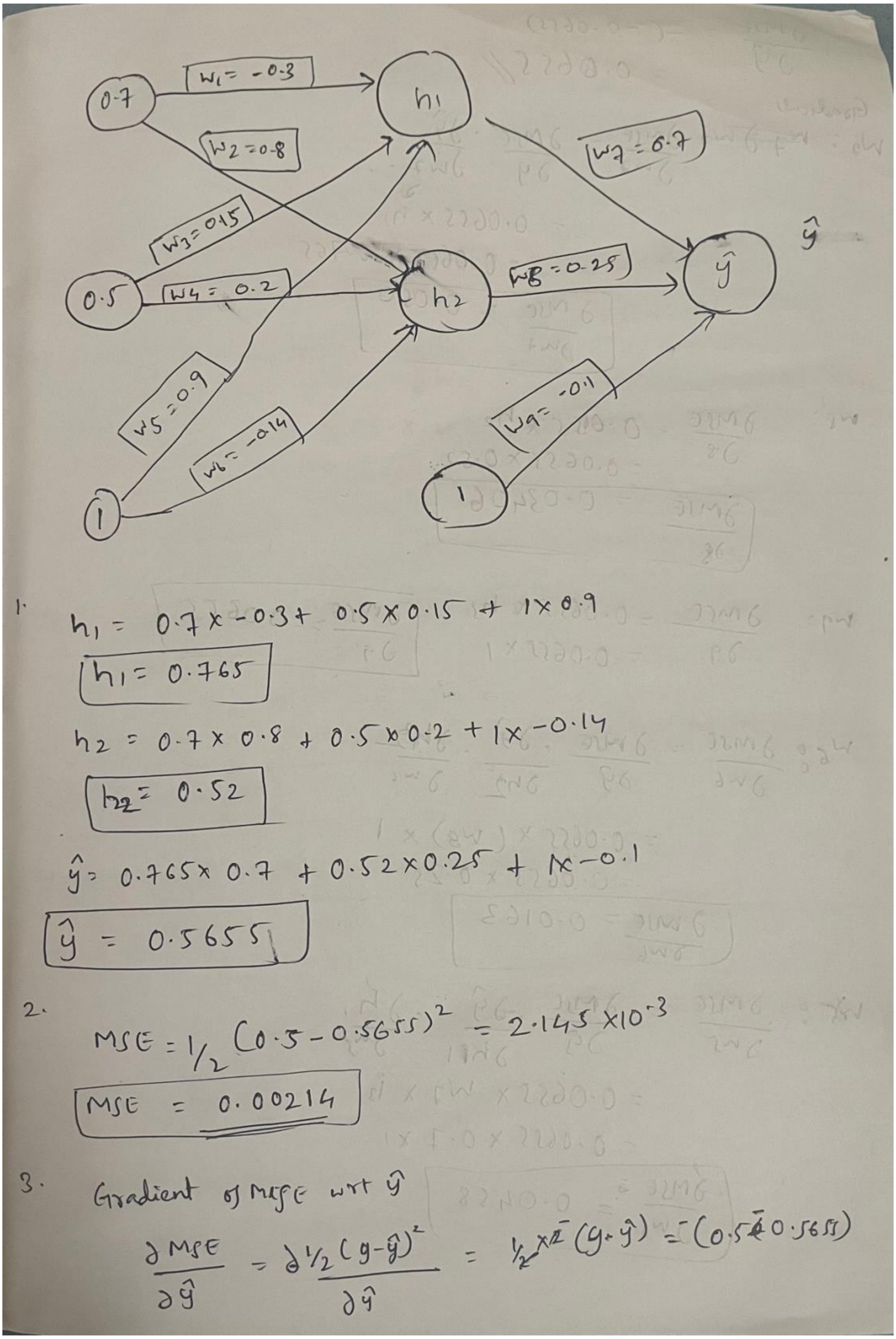
The confusion matrix provides a detailed breakdown of the model's predictions compared to the true labels. High values on the diagonal indicate correct predictions, while off-diagonal values represent misclassifications. The matrix confirms that the model performs well across all classes, with most predictions aligning with the true labels. This highlights the model's ability to handle multi-class classification effectively.

4. ROC Curve

The ROC curve has an AUC of 0.96, which indicates excellent classification performance. The curve shows a high true positive rate (TPR) and a low false positive rate (FPR), demonstrating the model's strong ability to distinguish between classes. This high AUC value underscores the model's effectiveness in balancing sensitivity and specificity, making it a robust classifier for the given task.

PART 4

1. Forward-backward Pass [15 points]


0.7
0.5
1
h1
h2
1
$\hat{y}$
$w_1 = -0.3$
$w_2 = 0.8$
$w_3 = 0.15$
$w_4 = 0.2$
$w_5 = 0.9$
$w_6 = -0.14$
$w_7 = 0.7$
$w_8 = 0.25$
$w_9 = -0.1$
$\hat{y}$

1\.

$h_1 = 0.7 \times -0.3 + 0.5 \times 0.15 + 1 \times 0.9$

$\boxed{h_1 = 0.765}$

$h_2 = 0.7 \times 0.8 + 0.5 \times 0.2 + 1 \times -0.14$

$\boxed{h_{22} = 0.52}$

$\hat{y} = 0.765 \times 0.7 + 0.52 \times 0.25 + 1 \times -0.1$

$\boxed{\hat{y} = 0.5655}$

2\.

$MSE = \frac{1}{2} (0.5 - 0.5655)^2 = 2.145 \times 10^{-3}$

$\boxed{MSE = \underline{0.00214}}$

3. Gradient of MSE wrt $\hat{y}$

$$\frac{\partial MSE}{\partial \hat{y}} = \frac{\partial \frac{1}{2} (y - \hat{y})^2}{\partial \hat{y}} = \frac{1}{2} \times -2 (y - \hat{y}) = -(0.5 - 0.5655)$$

$$\frac{\partial MSE}{\partial \hat{y}} = -(-0.0655)$$

$$= 0.0655 //$$

Gradients

W_7 : ~~$W_7\ \partial W_7$~~ $$\frac{\partial MSE}{\partial w_7} = \frac{\partial MSE}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_7}$$

$$= 0.0655 \times h_1$$

$$= 0.0655 \times 0.765$$

$$\boxed{\frac{\partial MSE}{\partial w_7} = 0.050}$$

w_8 : $$\frac{\partial MSE}{\partial 8} = 0.0655 \times h_2$$

$$= 0.0655 \times 0.52$$

$$\boxed{\frac{\partial MSE}{\partial 8} = 0.03406}$$

w_9 : $$\frac{\partial MSE}{\partial 9} = 0.0655 \times h_3$$

$$= 0.0655 \times 1$$

$$\boxed{\frac{\partial MSE}{\partial 9} = 0.0655}$$

w_6 : $$\frac{\partial MSE}{\partial w_6} = \frac{\partial MSE}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_6}$$

$$= 0.0655 \times (w_8) \times 1$$

$$= 0.0655 \times 0.25$$

$$\boxed{\frac{\partial MSE}{\partial w_6} = 0.0163}$$

w_5 : $$\frac{\partial MSE}{\partial w_5} = \frac{\partial MSE}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_5}$$

$$= 0.0655 \times W_7 \times i_3$$

$$= 0.0655 \times 0.7 \times 1$$

$$\boxed{\frac{\partial MSE}{\partial w_5} = 0.0458}$$

w_4 : $\frac{\partial MSE}{\partial w_4} = \frac{\partial MSE}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_4}$

$= 0.0655 \times \underset{w_8}{\cancel{0.025}} \times i_2$

$= 0.0655 \times 0.25 \times 0.5$

$$\boxed{\frac{\partial MSE}{\partial 4} = 8.18 \times 10^{-3} = 0.0081}$$

w_3 : $\frac{\partial MSE}{\partial w_3} = \frac{\partial MSE}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_3}$

$= 0.0655 \times w_7 \times i_2$

$= 0.0655 \times 0.7 \times 0.5$

$$\boxed{\frac{\partial MSE}{\partial w_3} = 0.0229}$$

w_2 : $\frac{\partial MSE}{\partial w_2} = \frac{\partial MSE}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2}$

$= 0.0655 \times w_8 \times i_1$

$= 0.0655 \times 0.25 \times 0.7$

$$\boxed{\frac{\partial MSE}{\partial w_2} = 0.0114}$$

w_1 = $\frac{\partial MSE}{\partial w_1} = \frac{\partial MSE}{\partial \hat{y}} \cdot \frac{\partial \hat{y}_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial m}$

$= 0.0655 \times w_7 \times i_1$

$= 0.0655 \times 0.7 \times 0.7$

$$\boxed{\frac{\partial MSE}{\partial w_1} = 0.032}$$

4. Update weighs

$$W = W - \alpha\left(\frac{\partial MSE}{\partial W}\right) \qquad \alpha = 0.03$$

$$w_9 = w_9 - \alpha\left(\frac{\partial MSE}{\partial w_9}\right)$$

$$= -0.1 - 0.03(0.0655)$$

$$\boxed{w_9 = -0.101}$$

$$w_8 = w_8 - \alpha\left(\frac{\partial MSE}{\partial w_8}\right)$$

$$= 0.25 - 0.03(0.034)$$

$$\boxed{w_8 = 0.249}$$

$$w_7 = w_7 - \alpha\left(\frac{\partial MSE}{\partial w_7}\right)$$

$$= 0.7 - 0.03(0.05)$$

$$\boxed{w_7 = 0.6985}$$

$$w_6 = w_6 - \alpha\left(\frac{\partial MSE}{\partial w_7}\right)$$

$$= -0.14 - 0.03(0.016)$$

$$\boxed{w_6 = -0.1404}$$

$$w_5 = w_5 - \alpha\left(\frac{\partial MSE}{\partial 5}\right)$$

$$= 0.9 - 0.03(0.046)$$

$$\boxed{w_5 = 0.8986}$$

$$w_4 = w_4 - \alpha$$

$$w_4 = w_4 - \alpha\left(\frac{\partial MSE}{\partial 4}\right)$$

$$= 0.2 - 0.03(0.008)$$

$$\boxed{w_4 = 0.1997}$$

$$w_3 = w_3 - \alpha\left(\frac{\partial MSE}{\partial w_3}\right)$$

$$= 0.15 - 0.03(0.0233)$$

$$\boxed{w_3 = 0.1493}$$

$$w_2 = w_2 - \alpha\left(\frac{\partial MSE}{\partial w_2}\right)$$

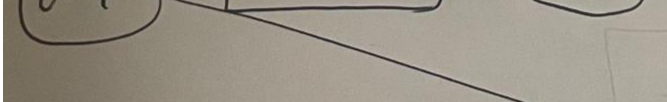
$$= 0.8 - 0.03(0.0114)$$

$$\boxed{w_2 = 0.7996}$$

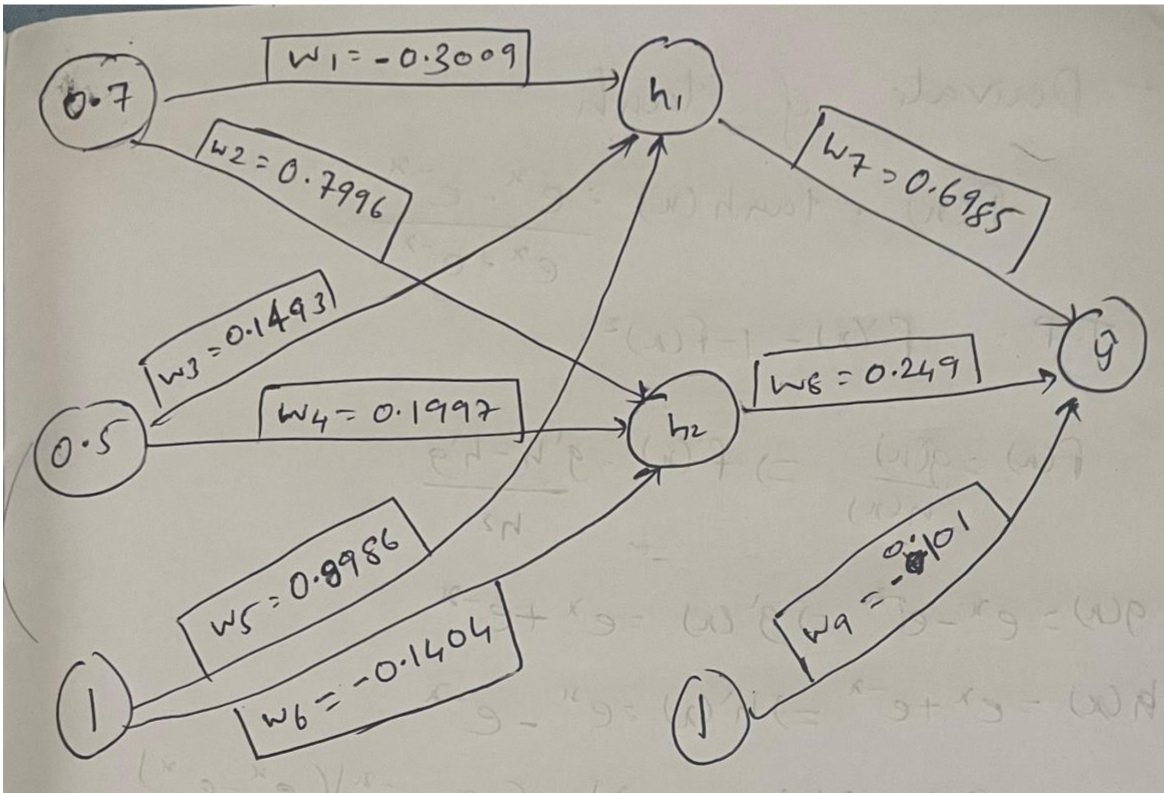
$$w_1 = w_1 - \alpha\left(\frac{\partial MSE}{\partial w_1}\right)$$

$$= -0.3 - 0.03(0.032)$$

$$\boxed{w_1 = -0.3009}$$



5.



6.

$h_1 = w_1 i_1 + w_3 i_2 + w_5 i_3$

$= 0.7 \times -0.3009 + 0.5 \times 0.1493 + 1 \times 0.8986$

$\boxed{h_1 = 0.7626}$

$h_2 = w_2 i_1 + w_4 i_2 + w_6 i_3$

$= 0.7996 \times 0.7 + 0.1997 \times 0.5 + (-0.1404) \times 1$

$\boxed{h_2 = 0.5191}$

$\hat{y} = h_1 w_7 + h_2 w_8 + h_3 w_9$

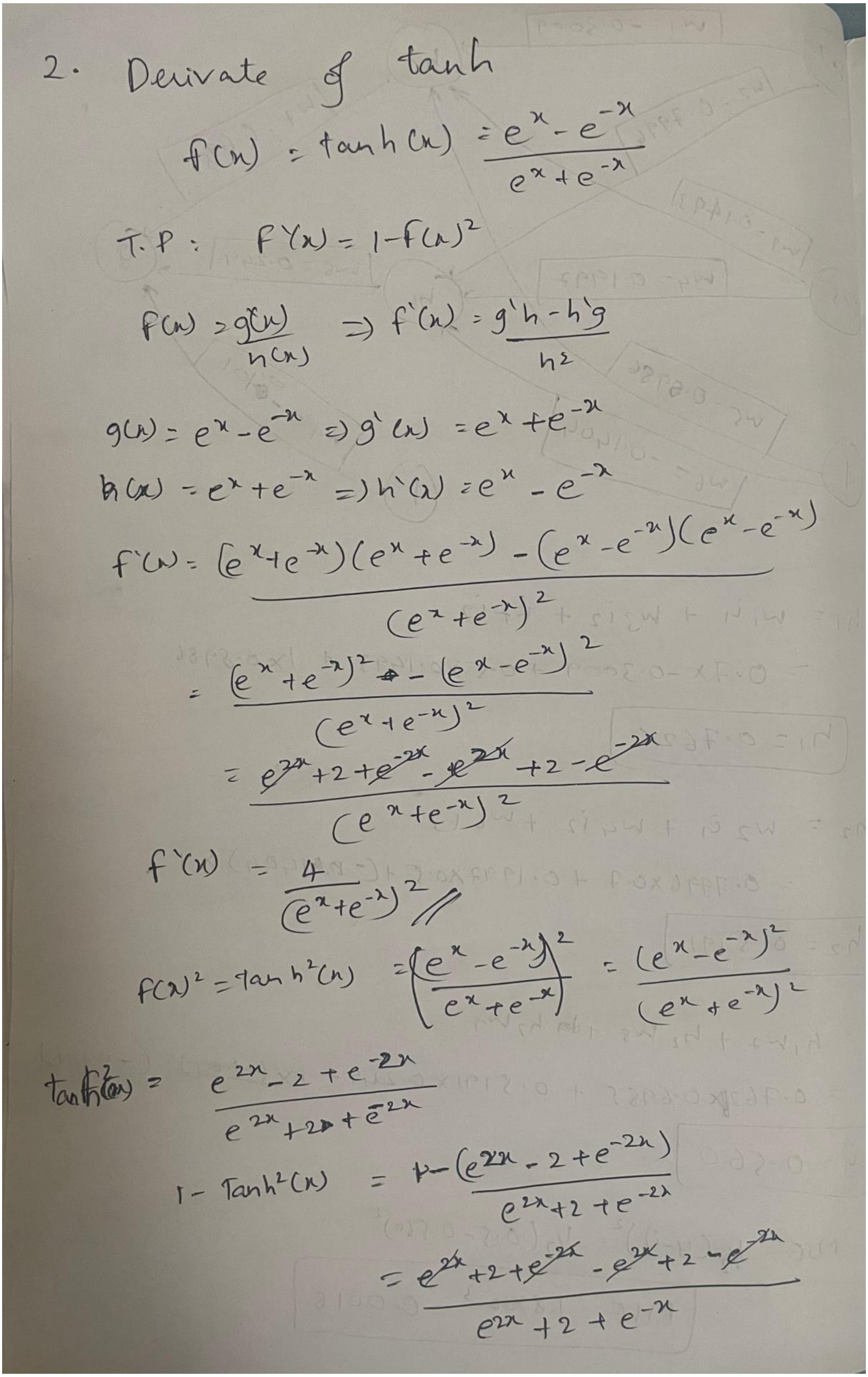
$= 0.762 \times 0.6985 + 0.5191 \times 0.249 + 1 \times \cancel{(-1.01)}\ (-0.101)$

$\boxed{\hat{y} = 0.560}$

7. $MSE = \frac{1}{2}(y-\hat{y})^2 = \frac{1}{2}(0.5-0.560)^2 =$

$\boxed{MSE = 1.8 \times 10^{-3} = 0.0018}$

2. Derivative of Tanh [5 points]



2. Derivate of tanh

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

T.P : $f'(x) = 1 - f(x)^2$

$$f(x) = \frac{g(x)}{h(x)} \Rightarrow f'(x) = \frac{g'h - h'g}{h^2}$$

$$g(x) = e^{x} - e^{-x} \Rightarrow g'(x) = e^{x} + e^{-x}$$

$$h(x) = e^{x} + e^{-x} \Rightarrow h'(x) = e^{x} - e^{-x}$$

$$f'(x) = \frac{(e^{x}+e^{-x})(e^{x}+e^{-x}) - (e^{x}-e^{-x})(e^{x}-e^{-x})}{(e^{x}+e^{-x})^2}$$

$$= \frac{(e^{x}+e^{-x})^2 - (e^{x}-e^{-x})^2}{(e^{x}+e^{-x})^2}$$

$$= \frac{\cancel{e^{2x}} + 2 + \cancel{e^{-2x}} - \cancel{e^{2x}} + 2 - \cancel{e^{-2x}}}{(e^{x}+e^{-x})^2}$$

$$f'(x) = \frac{4}{(e^{x}+e^{-x})^2} \;//$$

$$f(x)^2 = \tanh^2(x) = \left(\frac{e^{x} - e^{-x}}{e^{x}+e^{-x}}\right)^2 = \frac{(e^{x}-e^{-x})^2}{(e^{x}+e^{-x})^2}$$

$$\tanh^2(x) = \frac{e^{2x} - 2 + e^{-2x}}{e^{2x} + 2 + e^{-2x}}$$

$$1 - \tanh^2(x) = \frac{\cancel{1} - (e^{2x} - 2 + e^{-2x})}{e^{2x} + 2 + e^{-2x}}$$

$$= \frac{\cancel{e^{2x}} + 2 + \cancel{e^{-2x}} - \cancel{e^{2x}} + 2 - \cancel{e^{-2x}}}{e^{2x} + 2 + e^{-x}}$$

~~$= \frac{4}{e}$~~

$$1 - \tanh^2(a) = \frac{4}{e^{2x} + 2 + e^{-2x}}$$

$$\Rightarrow 1 - \tanh^2(x) = \frac{4}{(e^x + e^{-x})^2}$$

$\therefore$ LHS = RHS

$$\Rightarrow \quad f'(x) = 1 - f(x)^2 \text{ or } 1 - \tanh^2(x)$$

REFERNCES:

1. https://pytorch.org/tutorials/
2. https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/
3. https://catalog.data.gov/dataset/electric-vehicle-population-data
4. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html
5. https://www.geeksforgeeks.org/ml-linear-regression/
6. https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/
7. https://www.w3schools.com/python/python_ml_knn.asp
8. https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html
9. https://medmnist.com/
10. https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html
11. https://pytorch.org/docs/stable/optim.html
12.https://pytorch.org/ignite/generated/ignite.handlers.early_stopping.EarlyStopping.html
13. https://pytorch.org/docs/stable/generated/torch.nn.functional.batch_norm.html
14. https://discuss.pytorch.org/t/using-k-fold-cross-validation-to-train-my-model/196288

15 Piazza Course Slides -
spring25_cse676_b_lec_3_Activation_functions_Forward_Backward_Pass.pdf