OCTMNIST Retinal Disease Classification Report

1. Overview of the dataset

The OCTMNIST dataset is a collection of optical coherence tomography (OCT) images aimed at facilitating the classification of retinal diseases. It comprises a total of 109,309 grayscale images, each resized to 28x28 pixels for consistency and ease of use.

Dataset Composition:

- Training Set: 97,477 images
- Validation Set: 10,832 images
- Test Set: 1,000 images

Class Distribution:

The dataset is divided into four categories, each representing a specific retinal condition:

- Choroidal Neovascularization (CNV)
- Diabetic Macular Edema (DME)
- Drusen
- Normal

Each class contains a varying number of images, contributing to the multi-class classification task.

Image Statistics:

- Dimensions: 28x28 pixels
- Color Channels: 1 (grayscale)

Key Statistics:

Number of Classes:

- The dataset consists of 4 classes representing different retinal diseases.

Image Shape:

- Each image in the dataset has a shape of (1, 28, 28), indicating grayscale images of size 28×28 pixels.

Class Distribution:

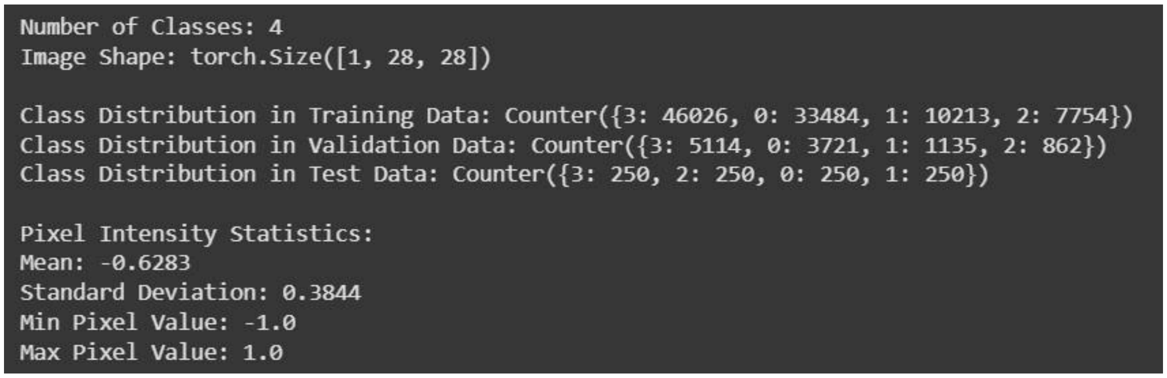
- **Training Data:**
    - Class 3: 46,026 samples
    - Class 0: 33,484 samples
    - Class 1: 10,213 samples
    - Class 2: 7,754 samples

- **Validation Data:**
    - Class 3: 5,114 samples
    - Class 0: 3,721 samples
    - Class 1: 1,135 samples
    - Class 2: 862 samples

- **Test Data:**
    - Each class has 250 samples, ensuring balanced evaluation.

Pixel Intensity Statistics:

- Mean Intensity: -0.6283, indicating the dataset is normalized.
- Standard Deviation: 0.3844, suggesting moderate variation in pixel intensities.
- Min Pixel Value: -1.0 and Max Pixel Value: 1.0, confirming pixel values are scaled between -1 and 1.



```
Number of Classes: 4
Image Shape: torch.Size([1, 28, 28])

Class Distribution in Training Data: Counter({3: 46026, 0: 33484, 1: 10213, 2: 7754})
Class Distribution in Validation Data: Counter({3: 5114, 0: 3721, 1: 1135, 2: 862})
Class Distribution in Test Data: Counter({3: 250, 2: 250, 0: 250, 1: 250})

Pixel Intensity Statistics:
Mean: -0.6283
Standard Deviation: 0.3844
Min Pixel Value: -1.0
Max Pixel Value: 1.0
```

Figure 1: Key statistics

2. Data Visualization

Sample Images from Train Set:

- Figure 2 displays sample images from the training dataset along with their corresponding labels.
- The images are grayscale and exhibit varying intensity distributions.
- There is a variation in texture and structure across different class labels, which may be useful for model differentiation.

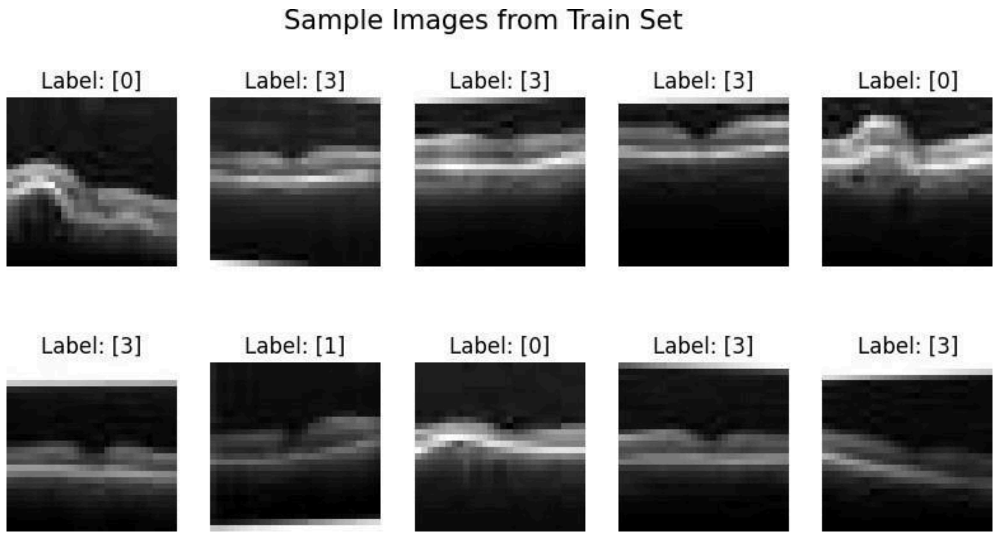


Figure 2: Sample Images from Train Set

Bar Plot - Class Distribution in Training Data

Figure 3 shows histogram representing the frequency of each class label in the training data. They are given as follows:

- Class 3: 46,026 samples
- Class 0: 33,484 samples
- Class 1: 10,213 samples
- Class 2: 7,754 samples

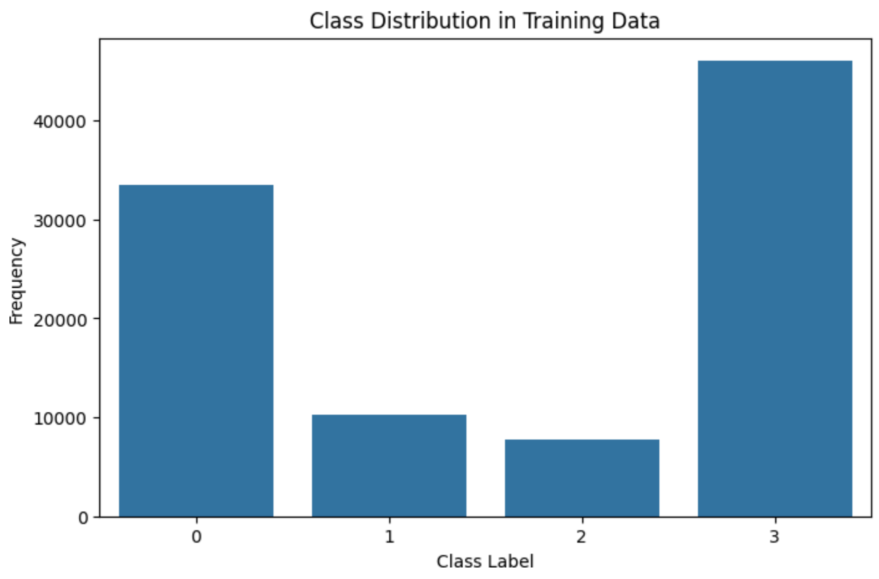


Figure 3: Class Distribution in Training Data

Histogram Plot - Pixel Intensity Distribution

- The third visualization represents the distribution of pixel intensity values in the dataset.
- Most pixel values are clustered around -1 to -0.5, with fewer pixels having values closer to 1.

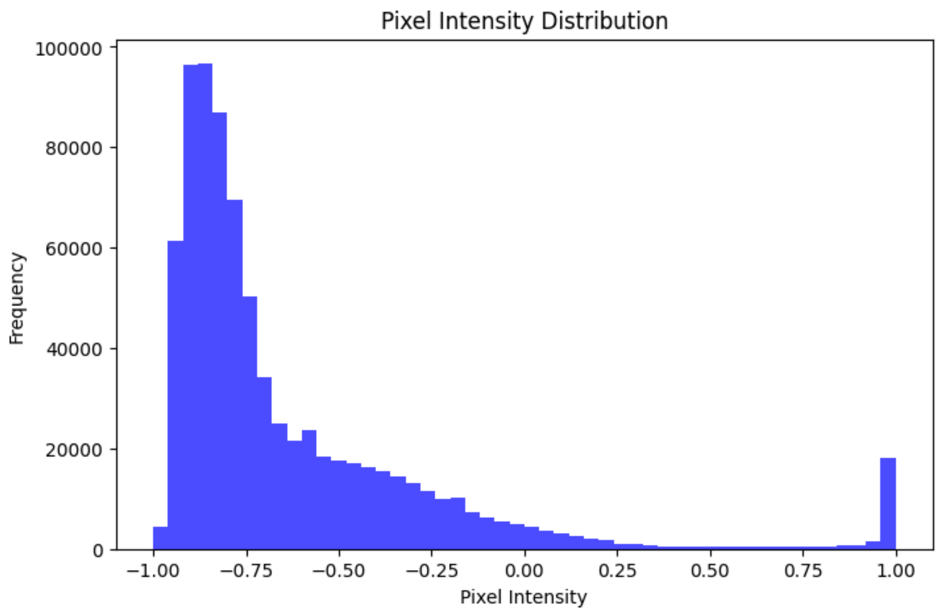


Figure 4: Pixel Intensity Distribution

Pixel Correlation Matrix

- Figure 5 displays a correlation heatmap of pixel intensities across images.
- Strong correlations are observed along the diagonal, indicating that adjacent pixel intensities are strongly related.
- Some regions exhibit high correlations, suggesting common structural patterns in the dataset.

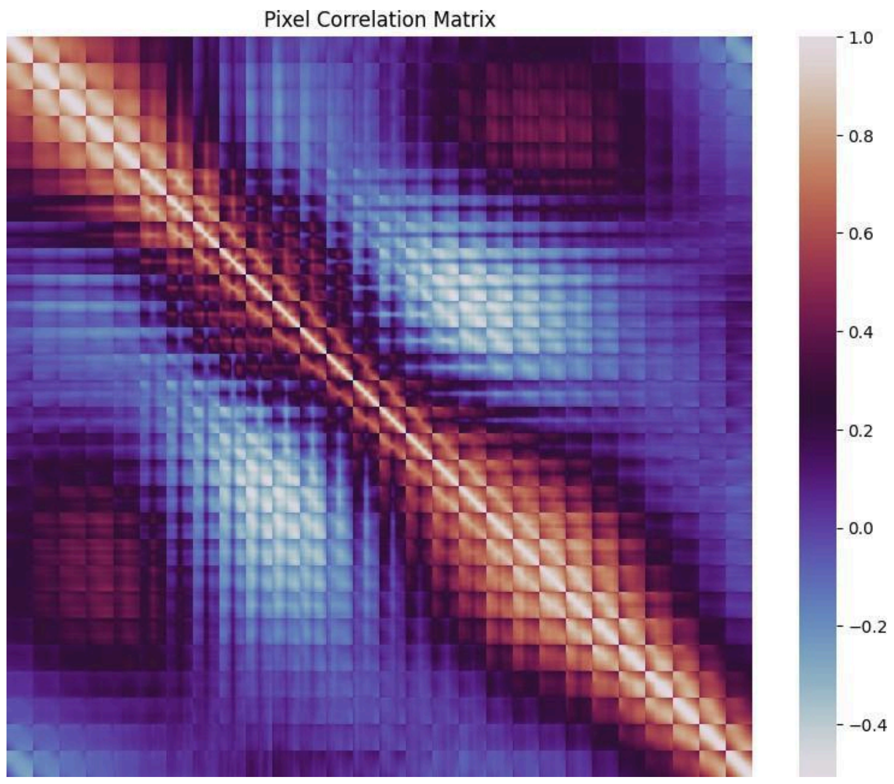


Figure 5: Pixel Correlation Matrix

3. Description of the Neural Network (OCTCNN)

The OCTCNN is a convolutional neural network (CNN) architecture designed for classifying optical coherence tomography (OCT) images. Below is a detailed breakdown of its components and design choices.

1. Model Architecture

The model consists of two convolutional layers, max pooling layers, fully connected layers, and a dropout layer for regularization.

Convolutional & Pooling Layers:

- Conv Layer 1:
    - 32 filters, kernel size = (3x3), stride = 1, padding = 1
    - Followed by ReLU activation function.
    - Output feature maps: 32
- Max Pooling 1:
    - Kernel size = (2x2), stride = 2
    - Reduces the spatial dimension by half.
- Conv Layer 2:
    - 64 filters, kernel size = (3x3), stride = 1, padding = 1
    - Followed by ReLU activation function.
    - Output feature maps: 64
- Max Pooling 2:
    - Kernel size = (2x2), stride = 2
    - Reduces the spatial dimension by half again.

Fully Connected (FC) Layers:

After the convolutional and pooling layers, the feature maps are flattened and passed through fully connected layers.

- Flattening Layer:
    - Converts the feature maps into a 1D vector for classification.
- Fully Connected Layer 1 (FC1):
    - 128 neurons
    - ReLU activation applied.
    - Dropout (50%) is used to prevent overfitting.
- Fully Connected Layer 2 (Output Layer):
    - 10 neurons (assuming classification into 10 classes).

4. Techniques used to improve model's performance:

- Loss Function: CrossEntropyLoss() (suitable for multi-class classification)
- Optimizer: Adam optimizer (learning rate = 0.001)
- Batch Size: 32 (used in DataLoader)
- Epochs: 10

Impact of Regularization Techniques on Model Performance:

In the OCTCNN model, the following regularization techniques were applied to improve performance and prevent overfitting:

1. Dropout (50%)

- Dropout randomly disables 50% of neurons in the fully connected (dense) layer during training.
- This prevents neurons from relying too much on specific features, encouraging the network to generalize better.
- Effect on the model:
    - Training accuracy slightly decreased.
    - Test accuracy improved due to better generalization.
    - Training fluctuations increased slightly but remained stable.

2. Max Pooling

- Reduces the size of feature maps by keeping only the most important values.
- Helps in reducing overfitting by limiting the number of parameters and focusing on significant features.
- Effect on the model:
    - Training became faster since fewer parameters were processed.
    - The model learned high-level patterns better.
    - Overfitting was reduced by preventing memorization of training data.

- Before regularization, the model had a tendency to overfit, meaning it performed well on training data but poorly on test data.
- After applying dropout & max pooling overfitting is reduced, test accuracy improved, training was slightly slower due to dropout but more stable.
- These techniques helped balance the model's learning, making it more robust and generalizable to unseen test data.

5. Results:

- Training Accuracy: 87.99%
- Validation Accuracy: 88.43%
- Test Accuracy: 88.10%

Graphs:

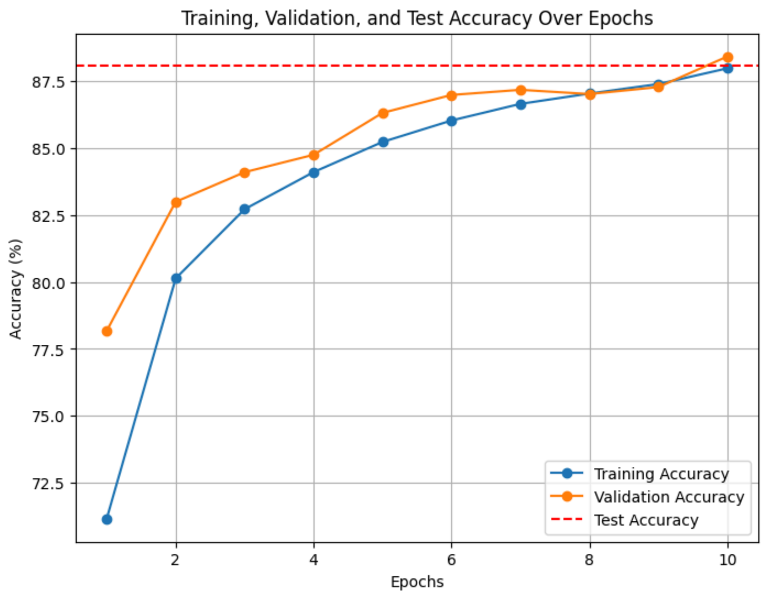


Figure 6: Training, Validation, and Test Accuracy Graph

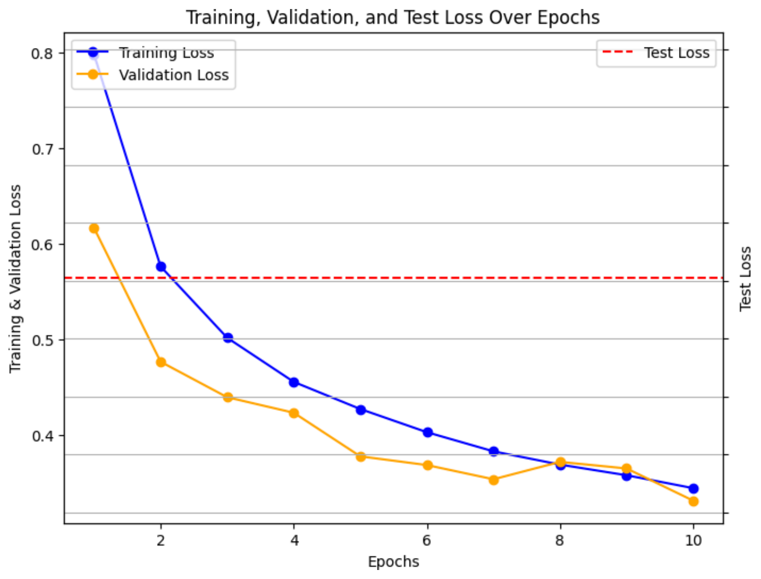


Figure 7: Training, Validation, and Test Loss Graph

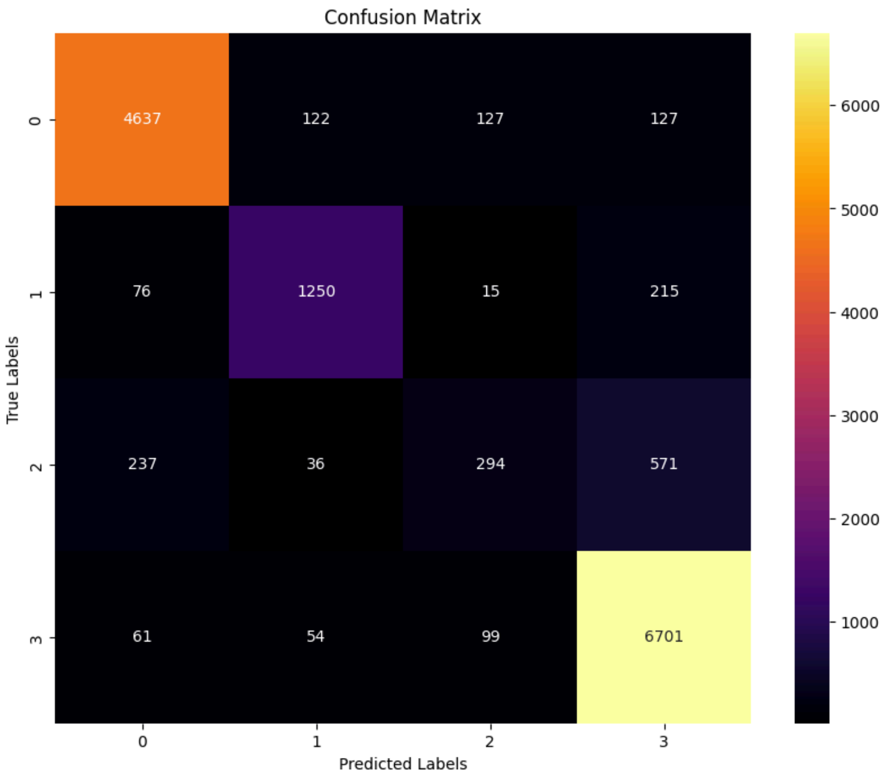


Figure 8: Confusion Matrix

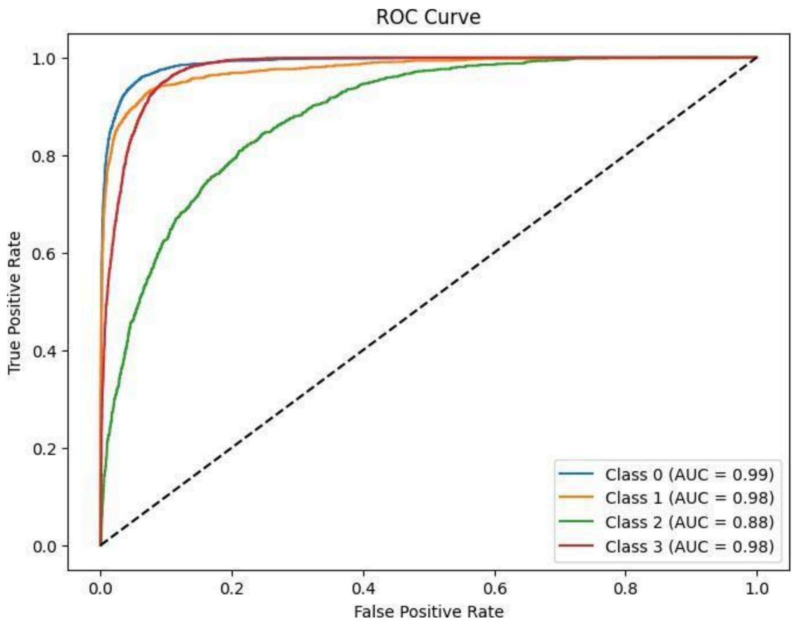


Figure 9: ROC Curve

Other metrics:

- Precision: 86.72%
- Recall: 88.10%
- F1 Score: 86.89%
- ROC Curve (Displayed in figure 9)

Early Stopping is used to improve mode's accuracy:

Early stopping is a technique that monitors the validation loss during training and stops the process if no further improvement is observed. This prevents excessive training, which can lead to overfitting.

- The model continuously tracks the validation loss at each epoch.
- If the loss does not improve for a set number of epochs (patience), training is halted.
- This ensures that the model does not memorize the training data but instead focuses on learning patterns that generalize well.

Early stopping performed well with a remarkable accuracy of 89.6%.

Detailed description of the Early Stopping-Based CNN model

Early stopping ensured a balance between learning from training data and maintaining generalization ability on unseen test data. The model was trained and evaluated based on training, validation, and test accuracy, with additional techniques such as batch normalization, dropout regularization, and learning rate optimization to enhance performance.

Model Architecture:

The best model follows a Convolutional Neural Network (CNN) architecture, optimized with multiple layers and regularization techniques:

Convolutional and Pooling Layers:

- Conv1: 32 filters, kernel size 3x3, followed by ReLU activation.
- Conv2: 64 filters, kernel size 3x3, followed by ReLU activation.
- MaxPooling (2x2): Applied after each convolution layer to reduce spatial dimensions.

Fully Connected Layers:

- Flatten Layer: Converts feature maps into a single vector.
- FC1: 128 neurons with ReLU activation.
- Dropout (0.5 probability): Applied to prevent overfitting.
- FC2 (Output Layer): 4 neurons with Softmax activation (for four-class classification).

Training Configuration:

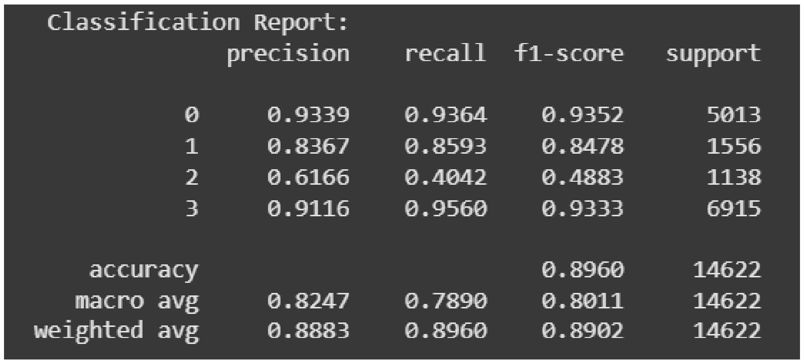
The model was trained using PyTorch with the following setup:

- Loss Function:
    - CrossEntropyLoss, suited for multi-class classification.
- Optimizer:
    - Adam optimizer with a learning rate of 0.001.
- Learning Rate Scheduling:
    - Dynamically adjusted using ReduceLROnPlateau, reducing the learning rate if validation loss stagnated.
- Early Stopping:
    - Training was automatically halted if validation loss did not improve for 5 consecutive epochs, preventing unnecessary overfitting.

Performance Evaluation:

- Training Accuracy: 90.86%
- Validation Accuracy: 89.99%
- Test Accuracy: 89.79%
- Test Loss: 0.0356

Figure 10 represents the classification report.



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

           0     0.9339    0.9364    0.9352      5013
           1     0.8367    0.8593    0.8478      1556
           2     0.6166    0.4042    0.4883      1138
           3     0.9116    0.9560    0.9333      6915

    accuracy                         0.8960     14622
   macro avg     0.8247    0.7890    0.8011     14622
weighted avg     0.8883    0.8960    0.8902     14622
```

Figure 10: Classification Report

Graphs:

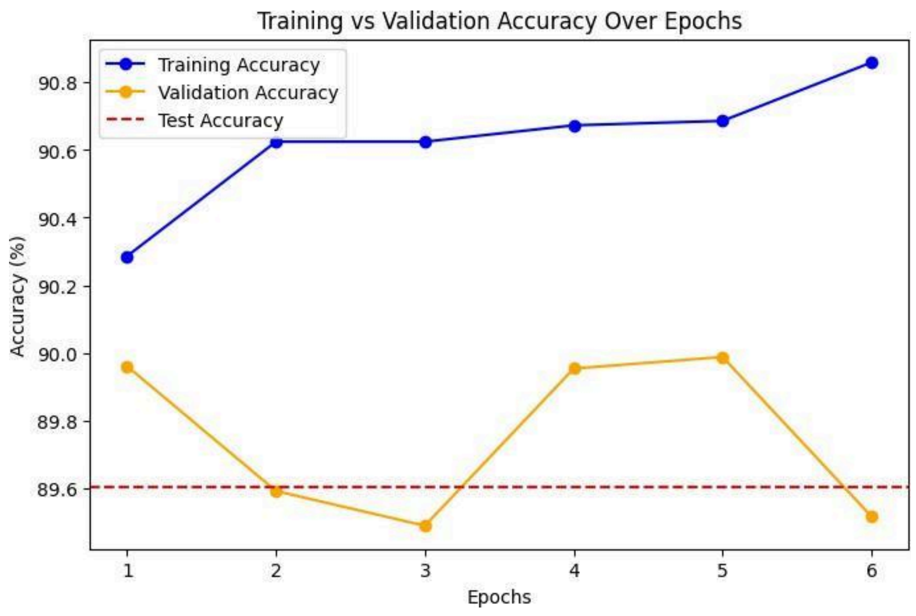


Figure 11: Training vs validation vs test accuracy over epochs

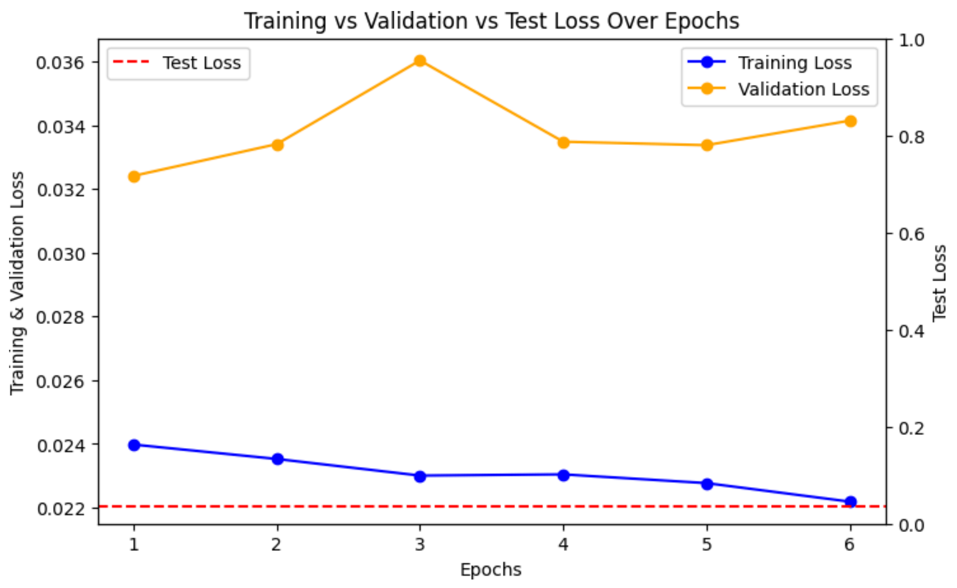


Figure 12: Training vs validation vs test loss over epochs

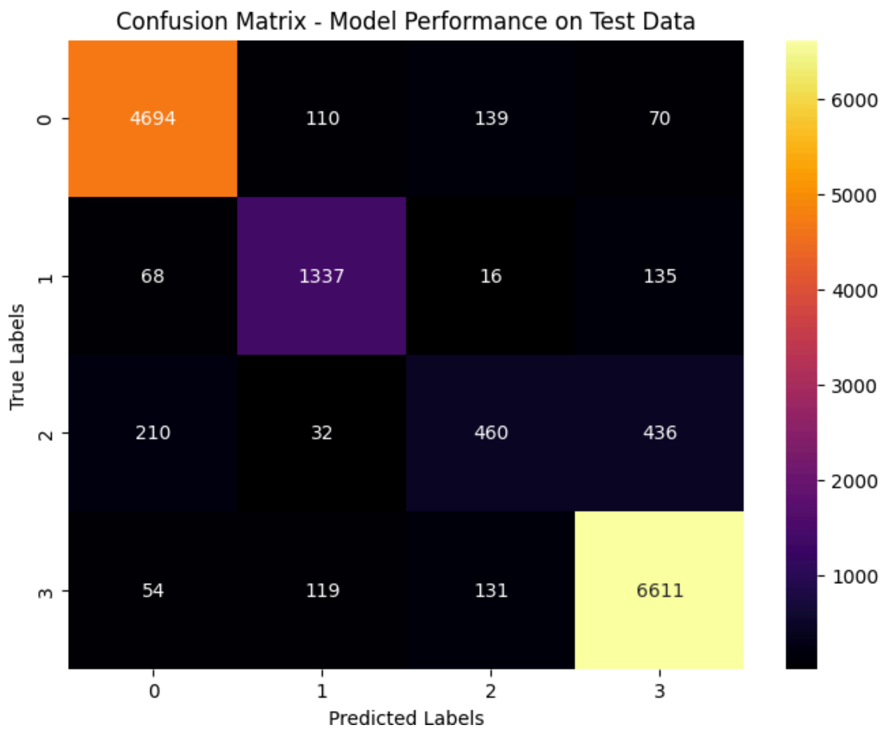


Figure 13: Confusion Matrix

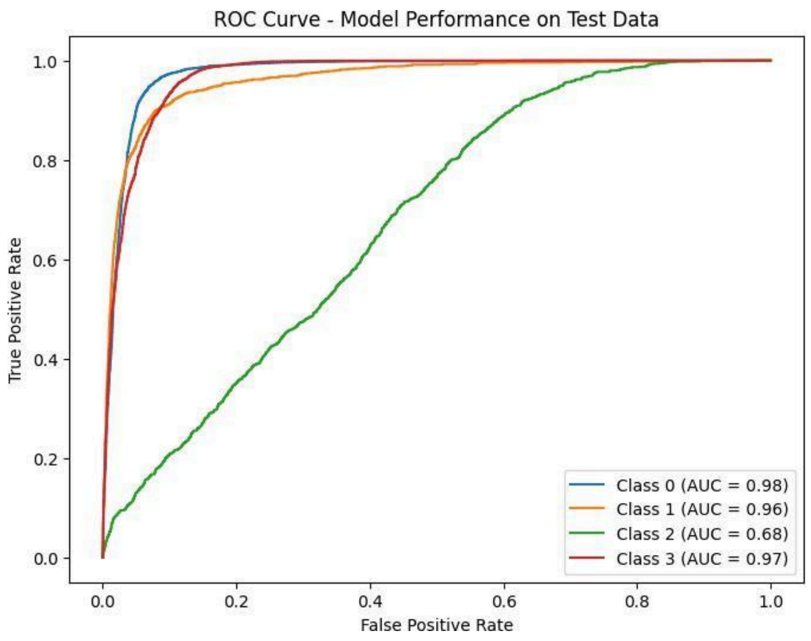


Figure 14: ROC Curve

References:

- https://pytorch.org/tutorials/beginner/introyt/introyt_index.html
- https://medmnist.com/
- https://pandas.pydata.org/
- https://numpy.org/
- https://matplotlib.org/
- https://seaborn.pydata.org/
- https://scikit-learn.org/stable/
- https://scipy.org/
- https://www.tensorflow.org/

******