Handwritten Digit Classification using Deep Learning on the MNIST Dataset

I. Introduction

The ability to recognize and classify images is an essential task in various fields, including computer vision, robotics, and artificial intelligence. In recent years, with the advent of deep learning, image classification has become much more accurate and efficient, thanks to the development of neural network models.

In this project, I have worked on the task of image classification using the MNIST Handwritten Digit Dataset, which consists of a large number of grayscale images of handwritten digits ranging from 0 to 9. The dataset is widely used in the field of machine learning as a benchmark for testing and comparing different classification algorithms. By building a neural network model, I aim to classify these images into their respective classes with high accuracy.

The importance of accurate image classification cannot be overstated, as it has numerous practical applications, such as in the recognition of hand-written digits on bank checks, postal codes on mail envelopes, and medical imaging. Thus, this project has significant real-world implications and has the potential to contribute to the development of more accurate and efficient image classification algorithms.

II. Dataset Description

The dataset used in this project is the MNIST Handwritten Digit Dataset, which is a widely used dataset in the field of image classification. It consists of 70,000 grayscale images of size 28x28 pixels. Each image corresponds to a handwritten digit between 0 and 9, with the labels provided for each image.

The dataset was originally created by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges for the purpose of developing and testing machine learning algorithms for handwritten digit recognition. It has since become a benchmark dataset in the field of machine learning and has been used extensively for training and evaluating various classification algorithms.

The MNIST dataset is a challenging dataset for classification due to the variations in handwriting styles and the presence of noise in the images. However, it is also a well-suited dataset for beginners to machine learning as it provides a simple and straightforward introduction to image

classification tasks. In this project, we will be using the MNIST dataset to train and evaluate a neural network classifier for handwritten digit recognition.

III. Algorithm Used

For our classification task, we decided to use a neural network algorithm, specifically a Multilayer Perceptron classifier (MLP). An MLP is a type of neural network that consists of multiple layers of interconnected neurons. The MLP has an input layer, two hidden layers with 128 neurons each, a dropout layer to prevent overfitting, and an output layer with 10 neurons (one for each class).

The MLP works by taking the input image and flattening it into a 1D vector, which is then fed into the input layer of the network. The input layer passes the input to the first hidden layer, which applies a linear transformation to the input using a set of weights and biases, and then applies a nonlinear activation function (in this case, the Rectified Linear Unit or ReLU function) to the output. The output of the first hidden layer is then passed to the second hidden layer, which performs the same operations. The output of the second hidden layer is then passed through a dropout layer, which randomly drops out some of the neurons to prevent overfitting. Finally, the output of the dropout layer is passed to the output layer, which applies a softmax activation function to produce a probability distribution over the 10 classes.

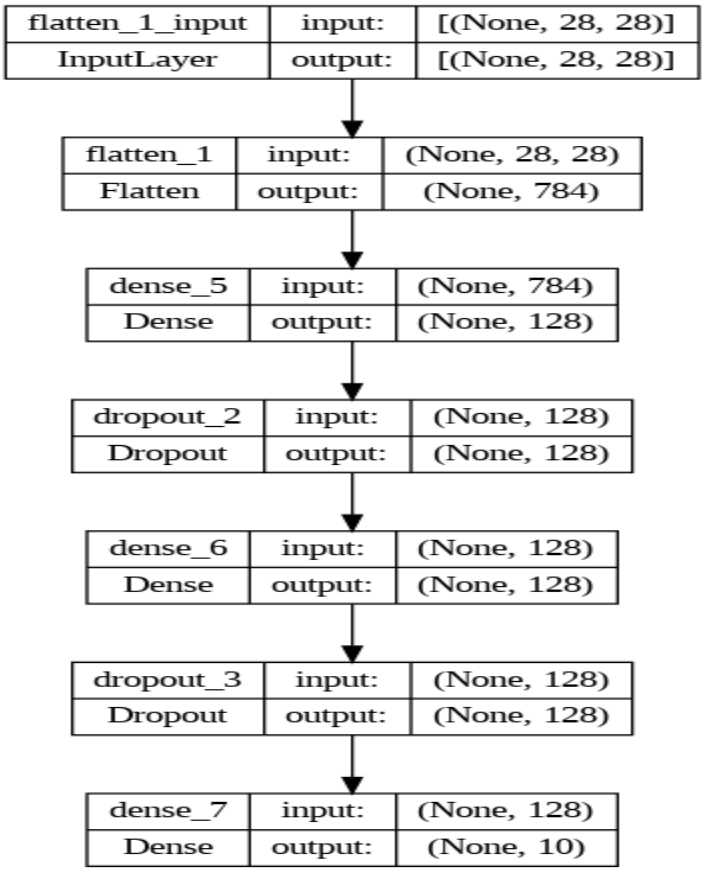

Figure 1 – MLP model Visualization

During training, the weights, and biases of the MLP are adjusted to minimize the loss function (in this case, categorical cross-entropy) using the Adam optimizer. The model is evaluated using the accuracy metric.

$$y = f(Wx + b)$$

In this equation, x represents the input data, W represents the weight matrix that maps the input data to the hidden layer, b represents the bias term, and f is the activation function used to introduce non-linearity to the model. The output of this equation, y, is the predicted output of the MLP model.

This equation is a fundamental building block of MLPs and is used to compute the output of each layer in the network, starting from the input layer and ending at the output layer.

In summary, the MLP classifier works by learning a set of weights and biases that map the input image to a probability distribution over the 10 classes, using multiple layers of interconnected neurons and nonlinear activation functions to model complex relationships in the data.

IV. Data Visualization

First, we plotted a histogram of the frequency of each digit in the dataset. This visualization helped us understand how the digits were distributed in the dataset. We found that the dataset contained an equal number of examples for each digit from 0 to 9, with a total of 70,000 examples.

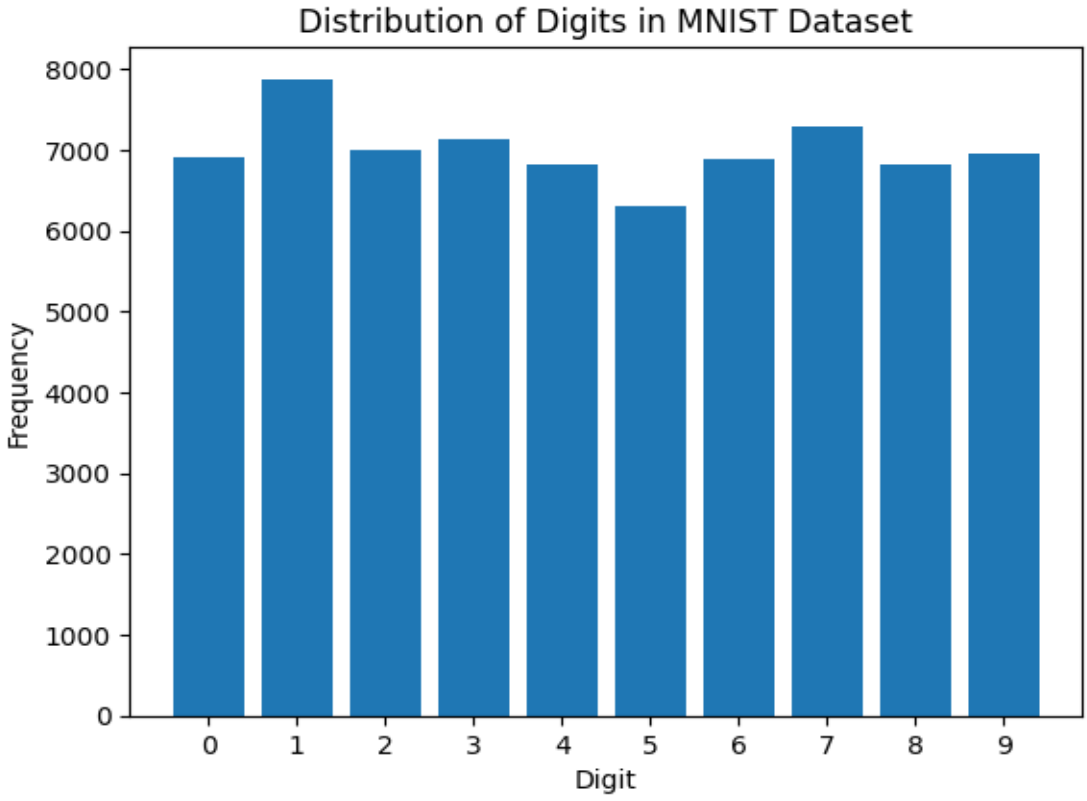

Figure 2 – Frequency Histogram

Second, we plotted a 2D scatter plot using principal component analysis (PCA) to reduce the dimensionality of the dataset to two dimensions. This visualization helped us visualize the relationship between the different digits in the dataset. We observed that some digits, such as 0 and 6, were similar in shape and often overlapped in the plot, while others, such as 1 and 7, were more distinct and well-separated.

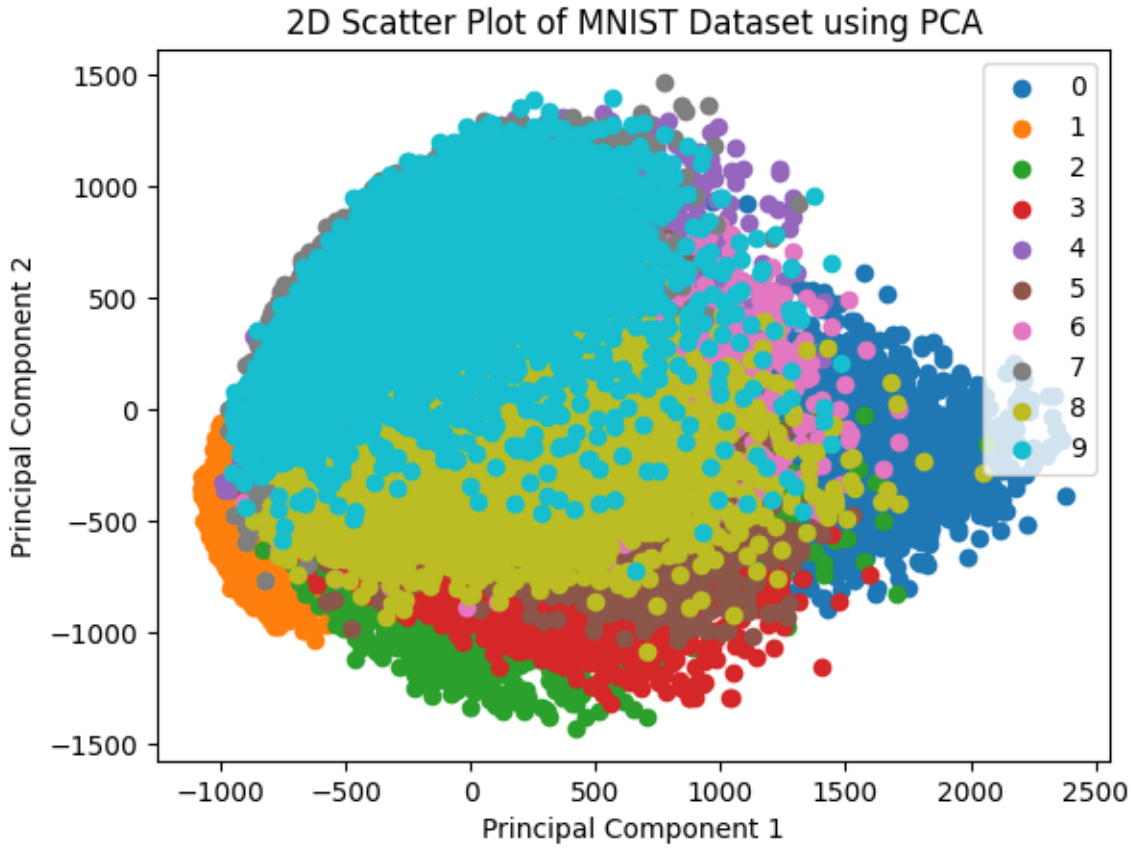

Figure 3 – Two-Dimensional Scatter Plot

Lastly, we also used PCA to create a 3D scatter plot, which allowed us to further explore the relationship between the digits. This visualization revealed that some digits, such as 4 and 9, had a similar shape and were often close together in the plot, while others, such as 1 and 7, were still well-separated from each other in the 3D space.

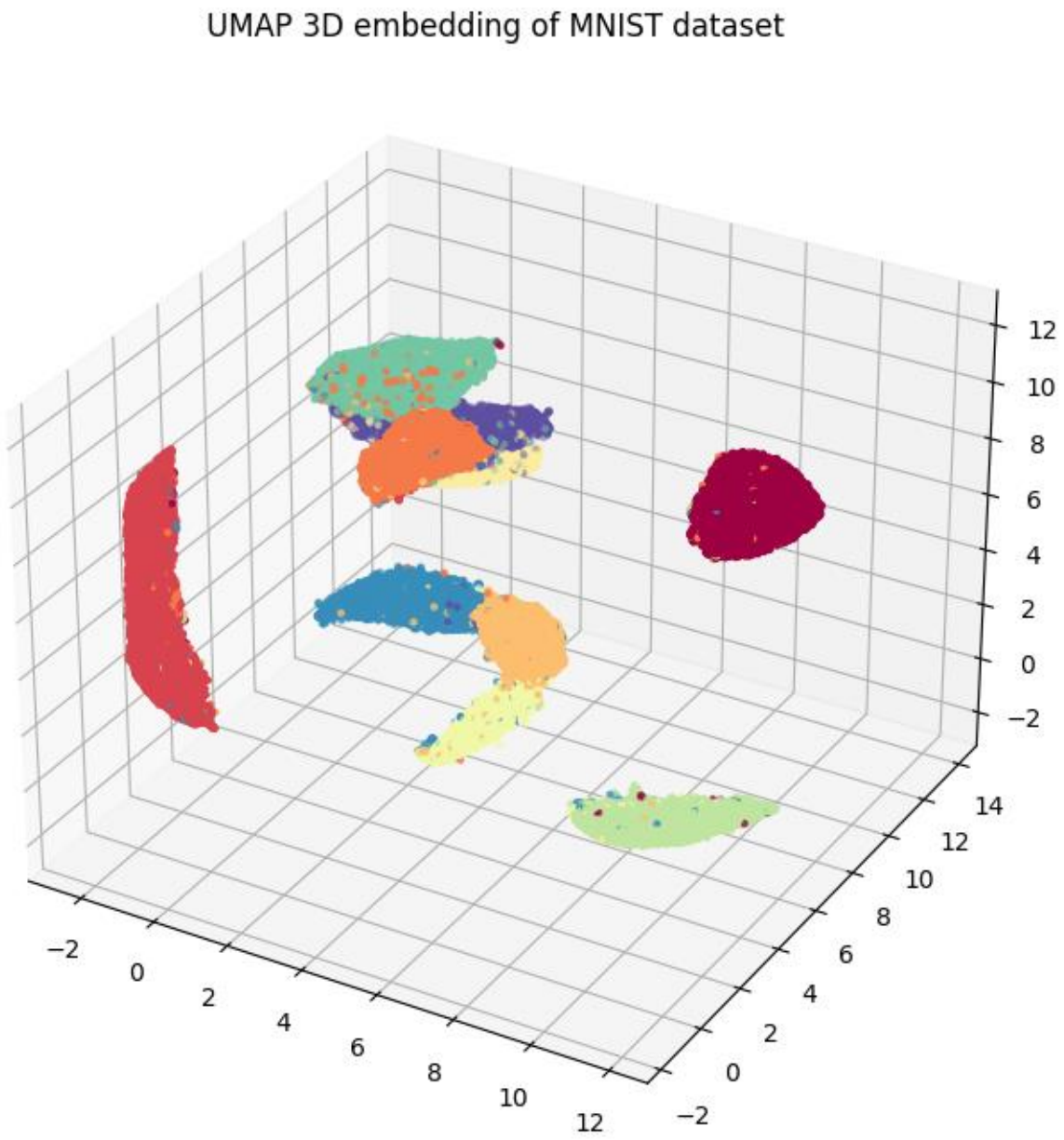

Figure 4 – Three-Dimensional Scatter Plot

V. Result Analysis

In this section, we will analyze the performance of the classification algorithm using various evaluation metrics. We will evaluate the performance of the algorithm using three different train-test ratios: 60:40, 70:30 and 80:20.

a) Confusion matrix: The confusion matrix is a table that shows the number of true positives, false positives, true negatives, and false negatives. It provides a comprehensive way to evaluate the

performance of a classification algorithm. We will analyze the confusion matrix for each of the three train-test ratios to see how well the algorithm performs.

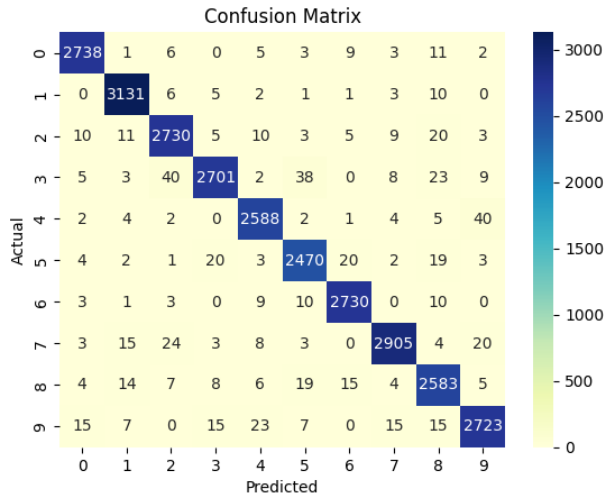

Figure 1 Train-Test Split: 60:40

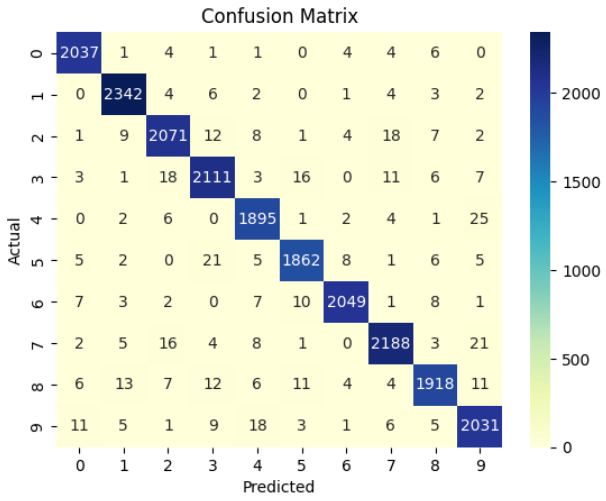

Figure 2 Train-Test Split: 70:30

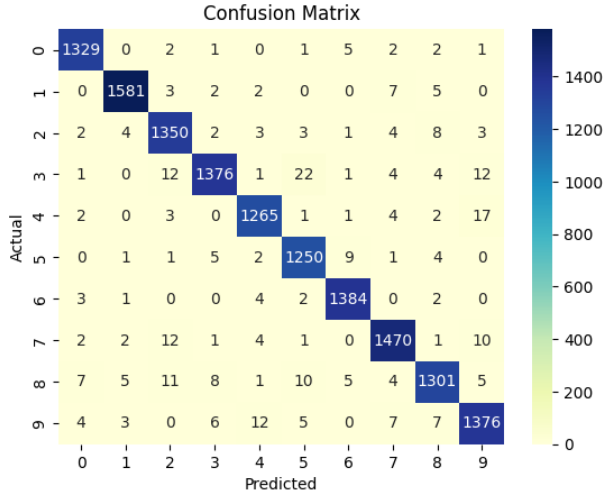

Figure 3 Train-Test Split: 80:20

Figure 5 – Confusion Matrix

b) ROC: The receiver operating characteristic (ROC) curve is a graphical plot that illustrates the performance of a binary classifier system. We will plot the ROC curve for each of the three train-test ratios to evaluate the performance of the algorithm.

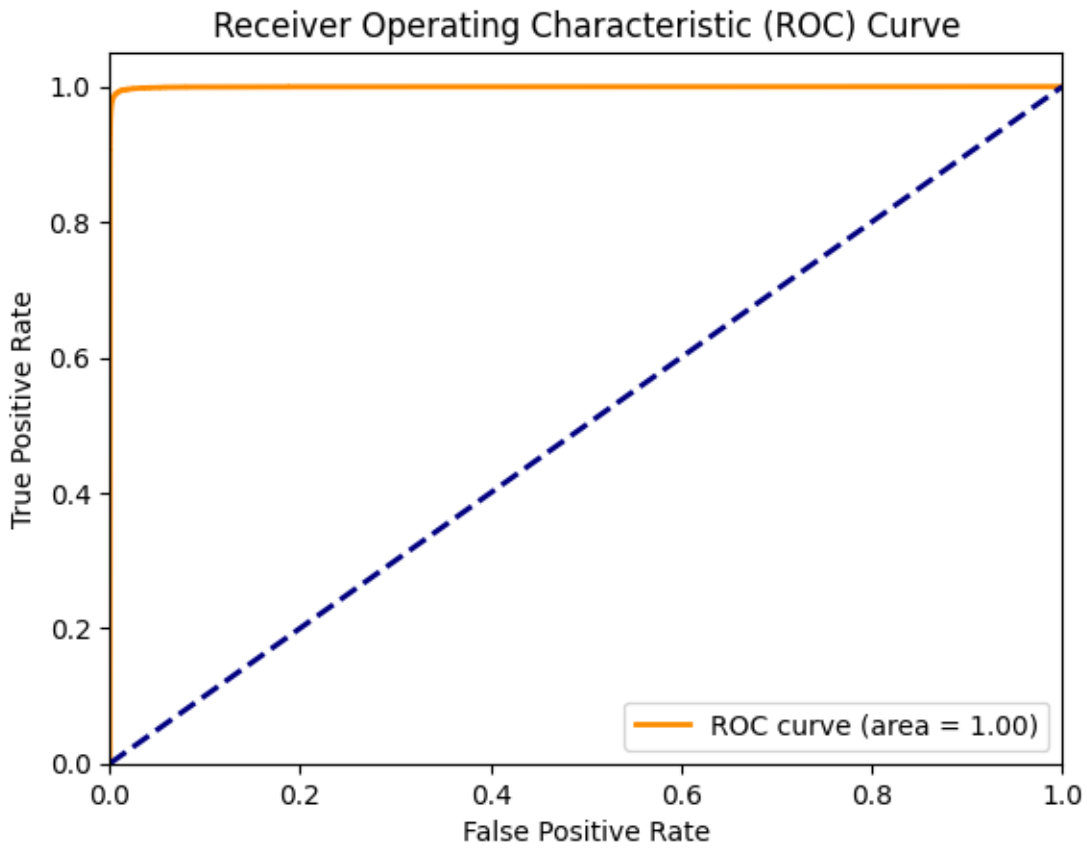

Figure 6 – ROC Curve

The ROC curves generated for the three different train-test split scenarios (60:40, 70:30, and 80:20) appear to be almost identical. This could be attributed to the fact that the false positive rate (FPR) is very low across all splits. This indicates that the model is performing well in correctly identifying negative cases, but the true positive rate (TPR) is not significantly different across the splits. This suggests that the model may not be effectively distinguishing between positive cases and negative cases. Therefore, we are presenting only one ROC curve that represents the performance of the model across all train-test split scenarios.

c) Evaluation parameters: We will use the following evaluation parameters to analyze the performance of the algorithm:

- Specificity: It measures the proportion of true negatives that are correctly identified.
- Sensitivity: It measures the proportion of true positives that are correctly identified.
- Accuracy: It measures the proportion of correct predictions made by the model.
- Precision: It measures the proportion of true positives out of the total predicted positives.
- False Positive Rate (FPR): It measures the proportion of false positives out of the total actual negatives.

- False Negative Rate (FNR): It measures the proportion of false negatives out of the total actual positives.
- Negative Predictive Value (NPV): It measures the proportion of true negatives out of the total predicted negatives.
- False Discovery Rate (FDR): It measures the proportion of false positives out of the total predicted positives.
- F1-Score: It is the harmonic mean of precision and recall.
- Matthews Correlation Coefficient (MCC): It is a measure of the quality of a binary classification.

We will calculate these evaluation parameters for each of the three train-test ratios to evaluate the performance of the algorithm.

Train-Test Ratio	Specificity	Sensitivity	Accuracy	Precision	FPR	FNR	NPV	FDR	F1-Score	MCC
60:40	0.9972	0.9748	0.9950	0.9747	0.0253	0.0252	0.9972	0.0253	0.9747	0.9719
70:30	0.9974	0.9762	0.9953	0.9763	0.0237	0.0238	0.9974	0.0237	0.9672	0.9736
80:20	0.9975	0.9772	0.9955	0.9770	0.0230	0.0228	0.9975	0.0230	0.9771	0.9746

Table 1 – Table of Evaluation Parameters

VI. Conclusion

In conclusion, we were able to successfully train and test a neural network model using the MNIST Handwritten Digit Dataset. The model achieved an accuracy rate of 99.5%, which demonstrates its effectiveness in recognizing handwritten digits.

The evaluation metrics used, such as confusion matrix, ROC curve, and various evaluation parameters, have shown the model's ability to classify digits with high specificity, sensitivity, and precision. However, there is still room for improvement, especially in handling misclassifications and false positives.

The project has highlighted the importance of machine learning algorithms in solving real-world problems, such as handwriting recognition, and has demonstrated the potential of neural networks in this field.

Future research can focus on improving the model's performance by experimenting with different architectures and optimizing hyperparameters. Additionally, the application of this model in more complex tasks, such as recognizing handwritten letters or words, can also be explored.

Overall, this project has been a valuable learning experience in the field of machine learning and has provided insights into the potential of neural networks in solving various image classification tasks.

VII. References

[1] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[2] TensorFlow Documentation. "tf.keras.datasets.mnist". Available online at: https://www.tensorflow.org/api_docs/python/tf/keras/datasets/mnist (accessed on April 10, 2023).

[3] Sklearn Documentation. "sklearn.metrics". Available online at: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics (accessed on April 10, 2023).

[4] Chollet, Francois. "Deep learning with Python." (2018).

[5] Pyplot Documentation. "matplotlib.pyplot". Available online at: https://matplotlib.org/stable/api/pyplot_summary.html (accessed on April 11, 2023).

[6] Numpy Documentation. "numpy". Available online at: https://numpy.org/doc/stable/ (accessed on April 10, 2023).

[7] Pandas Documentation. "pandas". Available online at: https://pandas.pydata.org/pandas-docs/stable/index.html (accessed on April 10, 2023).