Project Report: MNIST Digit Classification Using Artificial Neural Network in C

1. Introduction

This project involves implementing a simple artificial neural network in C to classify handwritten digits from the MNIST dataset. The motivation behind this work is to understand the inner workings of neural networks at a low level, without relying on high-level libraries such as TensorFlow or PyTorch. The model was trained and evaluated entirely using C, providing full control over memory, weight initialization, forward pass, backpropagation, and file I/O.

---

2. Objective

- Develop a 3-layer neural network in C.
- Train the model on the MNIST training set (60,000 samples).
- Evaluate performance on the test set (10,000 samples).
- Calculate accuracy, precision, recall, F1-score, and generate a confusion matrix.
- Save predictions in a CSV file for further analysis.

---

3. Network Architecture

- **Input Layer**: 784 neurons (28x28 grayscale image)
- **Hidden Layer 1**: 512 neurons with ReLU activation
- **Hidden Layer 2**: 256 neurons with ReLU activation
- **Output Layer**: 10 neurons (one for each digit 0–9) using Softmax activation

---

4. Implementation Details

Programming Language

- Language: C
- No external machine learning libraries used
- File I/O handled manually for model weights, image loading, and CSV output

Training

- **Loss Function**: Directly Calculated Error
- **Optimizer**: Basic stochastic gradient descent (SGD)
- **Epochs**: 100

- **Dataset**: MNIST (converted from original .idx format to raw arrays and taken from kaggle)

Evaluation

- Accuracy calculation
- Confusion matrix (10x10)
- Precision, Recall, F1-score (per class)
- Output of predictions to predictions.csv

---

5. Results

Accuracy

- Achieved 89.05% accuracy on the MNIST test dataset

Sample Evaluation Output

```
$ ./new2.exe
Loading pre-trained model...
Model loaded from 10k.bin
Evaluating the model on the test set...

Evaluation Accuracy: 89.05%
Correct predictions: 8905 / 10000

Class-wise Precision, Recall, F1-score:
Class 0: Precision = 0.89, Recall = 0.97, F1-score = 0.93
Class 1: Precision = 0.93, Recall = 0.97, F1-score = 0.95
Class 2: Precision = 0.92, Recall = 0.81, F1-score = 0.86
Class 3: Precision = 0.88, Recall = 0.87, F1-score = 0.87
Class 4: Precision = 0.90, Recall = 0.91, F1-score = 0.91
Class 5: Precision = 0.87, Recall = 0.80, F1-score = 0.83
Class 6: Precision = 0.93, Recall = 0.93, F1-score = 0.93
Class 7: Precision = 0.95, Recall = 0.86, F1-score = 0.90
Class 8: Precision = 0.80, Recall = 0.86, F1-score = 0.83
Class 9: Precision = 0.84, Recall = 0.90, F1-score = 0.87

Confusion Matrix:
      0   1   2   3   4   5   6   7   8   9
 0: 954   0   1   1   0   6  10   2   5   1
 1:   0 1105   1   5   0   4   4   0  15   1
 2:  31  38 833  10  13   5  16  17  55  14
 3:  12   3  17 876   2  53   2  10  23  12
 4:   1   4   3   0 897   2  10   3  15  47
 5:  38   3   5  38   2 714  15   2  67   8
 6:  12   4  12   1  15  10 892   1  11   0
 7:   1  13  24  23  12   2   1 887   3  62
 8:  18  19   6  24   7  22   8   3 841  26
 9:   8   4   1  15  46   3   1   8  17 906
```

predictions.csv:

Index	Actual	Predicted
0	7	7
1	2	0
2	1	1
3	0	0
4	4	4
5	1	1
6	4	4
7	9	9
8	5	6
9	9	9
10	0	0

6. Conclusion

This project was a great exercise in understanding neural networks at a low level. The C implementation enforces a deeper appreciation for backpropagation, weight updates, and memory management. Despite the absence of high-level libraries, the model achieved competitive results and serves as a solid foundation for Machine Learning.