Manual Implementation of MLP and CNN on MNIST

1. Introduction

The objective of this project was to manually implement, train, and evaluate two neural network models — a Multi-Layer Perceptron (MLP) and a Convolutional Neural Network (CNN) — without utilizing any high-level deep learning libraries such as PyTorch's `torch.nn`, TensorFlow, or Keras.

The MNIST handwritten digit dataset was used for training and evaluation.
The focus was on implementing all core computations manually using only **NumPy** for model construction, training, and evaluation.

2. Methodology

2.1 Multi-Layer Perceptron (MLP)

- **Architecture**:
  - Input: 784 features (28×28 flattened images)
  - Hidden Layer: 128 neurons, sigmoid activation
  - Output Layer: 10 neurons, softmax activation
- **Training Setup**:
  - Batch size: 128
  - Learning rate: 0.1
  - Epochs: 10
- **Loss Function**:
  Cross-entropy loss, manually implemented with NumPy.
- **Forward and Backward Propagation**:
  - Manual computation of linear transformations, activations, and gradient updates for both layers.
- **Data Preprocessing**:
  - Normalized MNIST data (mean=0.1307, std=0.3081).

2.2 Convolutional Neural Network (CNN)

- **Architecture**:
  - Single 3×3 convolutional kernel, stride=1, no padding
  - ReLU activation after convolution
  - Fully connected output layer to 10 classes with softmax activation
- **Training Setup**:
  - Batch size: 128
  - Learning rate: 0.001

    - Epochs: 5
- **Loss Function**:
  Cross-entropy loss.
- **Forward and Backward Propagation**:
    - Manual implementation of 2D convolution, flattening, and fully connected gradient updates.
- **Data Preprocessing**:
    - Normalized MNIST data (mean=0.1307, std=0.3081).

3. Results

Model	Final Test Accuracy	Notes
MLP	95.73%	Stable convergence over 10 epochs
CNN	42.13%	Limited by simple architecture (1 conv filter, no pooling)

- The MLP achieved high classification accuracy typical for MNIST with a single hidden layer.
- The CNN demonstrated learning behavior but achieved lower accuracy due to its minimal architecture, which was intentionally restricted to comply with project guidelines.

4. Learning Outcomes

- Developed a detailed understanding of forward propagation, loss calculation, and backpropagation for both fully connected and convolutional layers.
- Learned how to implement all major components of neural network training manually without using any automatic differentiation or high-level abstractions.
- Gained hands-on experience managing batch processing, normalization, weight updates, and evaluation.
- Understood the computational trade-offs and challenges associated with manual implementation compared to framework-assisted modeling.