Deep Learning

Assignment 1

Saurabh Chavan[B22EE061]

Google Colab:
https://colab.research.google.com/drive/19IOO7z6qcfrStG3lOVPyIYzisi8e1deo?usp=sharing

Introduction: The objective of this project was to implement a feedforward neural network from scratch using Python to classify images in the MNIST dataset. The neural network contains at least two fully connected layers and uses the backpropagation algorithm for training. Additionally, the network was trained on different train-test splits (70:30, 80:20, and 90:10), and its performance was evaluated using accuracy, loss plots, and confusion matrices.

Network Architecture

- **Input Layer:** 784 neurons (corresponding to the 28x28 pixel images).
- **Hidden Layer:** 128 neurons with ReLU activation.
- **Output Layer:** 10 neurons with softmax activation (for 10-class classification).

Weights were initialized randomly using a Gaussian distribution scaled by 0.01, and biases were initialized to zero. The random seed value was fixed at 61 as my roll number is b22ee061.

Training and Optimization

- **Loss Function:** Categorical Cross-Entropy was used as the loss function to measure the performance of the model during training.

- **Optimization:** Gradient Descent was implemented manually to update the weights and biases during the backpropagation step.
- **Learning Rate:** Set to 0.01.
- **Batch Size:** 22(year of admission 2022)
- **Epochs:** 25.

Experimental Setup The network was trained and evaluated on three different train-test splits:

1. **70:30 Split:** 42,000 training samples, 18,000 validation samples.
2. **80:20 Split:** 48,000 training samples, 12,000 validation samples.
3. **90:10 Split:** 54,000 training samples, 6,000 validation samples.

For each split, the dataset was shuffled, and the training was carried out using mini-batches. The performance was tracked in terms of:

- **Training Loss and Accuracy**
- **Validation Loss and Accuracy**

RESULTS

For each split, the loss and accuracy metrics were recorded over 25 epochs. The results showed a steady decrease in loss and an increase in accuracy, indicating effective learning by the network. Plots of accuracy and loss per epoch were generated to visualize the trends.

The model was evaluated on the test dataset after training, and the following results were observed:

- **70:30 Split:**
    - Test Accuracy: ~96.2%

    - Confusion Matrix: A confusion matrix was generated, highlighting the classification performance across all 10 classes.
- **80:20 Split:**
    - Test Accuracy: ~96.7%
    - Confusion Matrix: Improved performance due to more training data.
- **90:10 Split:**
    - Test Accuracy: ~97.1%
    - Confusion Matrix: Highest performance due to the largest training dataset.

Confusion Matrices

Confusion matrices were plotted for each split, illustrating class-wise prediction performance. Most misclassifications occurred between visually similar digits such as 3 and 8.

Plots

- **Loss vs Epochs:** Training and validation loss decreased steadily for all splits.
- **Accuracy vs Epochs:** Training and validation accuracy improved consistently, with the 90:10 split showing the best performance.

Total Parameters

- **Trainable Parameters:**
    - Weights (Input to Hidden):
    - Biases (Hidden Layer):

    - Weights (Hidden to Output):
    - Biases (Output Layer):
    - **Total Trainable Parameters:** 101,770
- **Non-Trainable Parameters:** 0 (no parameters are frozen).

Limitations:

- The model lacks regularization techniques (e.g., dropout) to prevent overfitting.
- It does not utilize convolutional layers, which are better suited for image data.