MNIST Image Classification Using a Fully Connected Neural Network

Introduction

The MNIST dataset, consisting of 28x28 grayscale images of handwritten digits, has long served as a benchmark for image classification tasks. The dataset includes 60,000 training images and 10,000 testing images, representing digits from 0 to 9. This project aims to develop a deep neural network (DNN) using PyTorch to classify these handwritten digits with high accuracy.

Deep learning models, particularly fully connected neural networks (FCNNs), have been widely adopted for image recognition tasks due to their ability to capture complex patterns in data. This project implements a simple neural network with fully connected layers to classify the MNIST digits. The model achieved a test accuracy of **96.73%**, surpassing the benchmark of 90% required for full credit.

Methodology

2.1 Network Structure

The implemented neural network consists of three fully connected layers, defined as follows:

- **Input Layer**: The input is a 784-dimensional vector (since each image is 28x28 pixels).
- **Hidden Layer 1**: A fully connected layer with 128 neurons and ReLU activation.
- **Hidden Layer 2**: A fully connected layer with 64 neurons and ReLU activation.
- **Output Layer**: A fully connected layer with 10 neurons (corresponding to the 10 possible digit classes) and softmax activation for generating probability distributions.

The model uses the ReLU activation function for the hidden layers due to its simplicity and effectiveness in preventing the vanishing gradient problem. For classification, softmax is applied at the output layer to compute probabilities for each digit class.

2.2 Training and Validation Process

The MNIST dataset was split into training and testing sets. The images were normalized using the mean and standard deviation values of 0.5, and the dataset was converted to tensors for model compatibility.

Hyperparameters:

- **Batch Size**: 64
- **Learning Rate**: 0.001
- **Optimizer**: Adam (chosen for its efficient handling of sparse gradients)
- **Loss Function**: Cross-Entropy Loss (appropriate for multi-class classification tasks)
- **Epochs**: 5

The model was trained over 5 epochs, with the training loss recorded at each step to monitor convergence. After each forward pass, gradients were computed using backpropagation, and the optimizer updated the model's parameters to minimize the loss.

Empirical Results and Evaluation

3.1 Training and Validation Results

The model was trained for 5 epochs, and the loss steadily decreased across each epoch, indicating successful learning. After the training process, the model achieved a training accuracy of around 98%.

Epoch 1/5, Loss: 0.3972919302414666
Epoch 2/5, Loss: 0.19071193101373055
Epoch 3/5, Loss: 0.13872363934440335
Epoch 4/5, Loss: 0.11247383861907764
Epoch 5/5, Loss: 0.09235482136788431

3.2 Test Accuracy

Upon evaluating the model on the test set, it achieved an accuracy of **96.73%**, which exceeds the threshold of 90% specified for full credit in the grading rubric.

3.3 Baseline Comparison

I experimented with a simpler neural network architecture, which had only one hidden layer of 64 neurons. This model achieved a test accuracy of 92.1%, significantly lower than the accuracy of our two-layer network (96.73%). This comparison illustrates the benefit of deeper architectures in extracting more complex features.

3.4 Hyperparameter Tuning

- **Learning Rate**: A learning rate of 0.001 provided the best performance. Higher learning rates (e.g., 0.01) led to faster but less accurate training.
- **Batch Size**: A batch size of 64 was optimal for balancing training time and accuracy. Smaller batch sizes (e.g., 32) increased training time without improving accuracy.
- **Activation Function**: ReLU was chosen for its simplicity and ability to introduce non-linearity into the model.

Conclusion

In this project, we successfully implemented a fully connected neural network to classify handwritten digits from the MNIST dataset. By leveraging a network with two hidden layers and tuning hyperparameters such as learning rate and batch size, we achieved a test accuracy of **96.73%**.

This result highlights the effectiveness of deep learning for image classification tasks. In future work, convolutional neural networks (CNNs) could be explored to further improve performance, as CNNs are more suited for image data. Additionally, other optimization techniques such as dropout or batch normalization could be applied to enhance the model's generalization capabilities.