Handwritten Digit Recognition Using Neural Networks

Ethan Alexander

Project Overview

This project demonstrates the application of Convolutional Neural Networks (CNNs) to accurately recognize handwritten digits using the MNIST dataset. The workflow involved data preprocessing, model construction, training, evaluation, and visualization to optimize predictive performance.

Data Loading and Preprocessing

- **Dataset**:
  - Utilized the MNIST dataset containing 28x28 grayscale images of digits (0–9).
- **Preprocessing Steps**:
  - **Normalization**:
    - Scaled pixel values to a range of [0, 1] by dividing by 255.
  - **Reshaping**:
    - Adjusted input data dimensions to meet convolutional layer requirements (60000 x 28 x 28 x 1).
  - **Label Encoding**:
    - Converted categorical labels to one-hot encodings.
- **Data Visualization**:
  - Displayed 10 random samples of images with their corresponding labels to confirm dataset integrity.

Neural Network Construction

- **Architecture**:
  - Built a Sequential CNN model with the following components:
    1. **Convolutional Layers**: Automatically learned spatial hierarchies of features.
    2. **ReLU Activation**: Introduced non-linearity, enabling the network to learn complex patterns.
    3. **Max Pooling**: Reduced spatial dimensions while retaining key features.

4. **Dropout Layers**: Minimized overfitting by randomly deactivating a fraction of neurons during training.
5. **Dense Layers**: Aggregated learned features for final digit classification.

- **Activation Function Comparison**:
  - Tested ReLU, tanh, and sigmoid.
  - Found **ReLU** to be the most effective due to faster computation and robustness for image data.

Model Training and Evaluation

- **Training Setup**:
  - Used 90% of the dataset for training and reserved 10% for validation.
- **Performance Visualization**:
  - Plotted training and validation loss and accuracy over epochs using Matplotlib.
  - Results showed significant improvement in the first epoch, with diminishing returns in subsequent epochs.
- **Confusion Matrix**:
  - Constructed a confusion matrix that demonstrated high classification accuracy, with the majority of images correctly labeled.

Experiments and Insights

1. **Batch Size**:
   - Smaller batch sizes introduced more noise but stabilized learning, while larger batch sizes accelerated training but risked overfitting.
2. **Epochs**:
   - Increasing epochs improved accuracy initially but reached a point of diminishing returns and risked overfitting.
3. **Model Architecture**:
   - CNNs effectively captured spatial features, significantly improving accuracy for image classification tasks.

Key Takeaways

- **Effectiveness of CNNs**: The architecture was highly successful in processing and classifying image data, highlighting the power of convolutional layers and non-linear activation functions.
- **Optimization Matters**: Experimenting with hyperparameters like batch size, learning rate, and epoch count led to meaningful performance improvements.
- **Real-World Applications**: This project demonstrates how deep learning can be leveraged for tasks like optical character recognition and other image-based classification problems.