MNIST Image Classification Using Classical and Deep Learning Methods

Abstract

This project implements image classification for the MNIST handwritten digit dataset, comparing classical machine learning techniques (K-Nearest Neighbors, Naive Bayes, Linear Regression) with modern deep learning approaches (Linear Model, Multilayer Perceptron, Convolutional Neural Network). The models were evaluated on their accuracy and ability to generalize to unseen data. Results show a clear progression in performance from simpler classical models to deep learning methods, illustrating the effectiveness of neural networks in visual pattern recognition tasks.

1. Project Description

The goal of this project is to classify handwritten digits from the MNIST dataset, a standard benchmark in machine learning, and in doing so addresses the problem of automated digit recognition. Applications of this technology include automated check processing, postal mail sorting, and as an educational tool for teaching AI techniques. By implementing multiple models, the project demonstrates both classical and modern approaches to pattern recognition and shows a clear and concise comparison of their effectiveness.

2. AI Techniques

- **K-Nearest Neighbors (KNN):** Predicts the label of a test sample based on the majority label among its k closest training samples.

- **Naive Bayes (Bernoulli):** Models each pixel as an independent Bernoulli variable and computes the class with maximum probability.

- **Linear Classifier (NumPy and PyTorch):** Maps flattened pixel intensities using a linear transformation. The PyTorch implementation enables GPU acceleration.

- **Multilayer Perceptron (MLP):** A fully connected neural network with non-linear activation functions, allowing learning of more complex patterns than linear models.

- **Convolutional Neural Network (CNN):** Uses convolutional layers to extract local spatial features, followed by fully connected layers for classification.

3. Datasets

- **Raw MNIST Dataset:**
    - 60,000 training images and 10,000 test images of grayscale digits (28×28 pixels).
    - Images were normalized to [0,1] for NumPy models and [-1,1] for PyTorch models.
    - Data splits: 80% training, 20% validation, with optional subsampling for testing & debugging experiments.

4. Implementation Tools

- **Programming Language:** Python 3.13
- **Libraries and Packages:**
    - NumPy for array operations and machine learning implementation
    - PyTorch and torchvision for deep learning models and data loading
    - PIL for image processing
    - Matplotlib for weight visualization
- **Hardware:** CPU for classical models; GPU acceleration (if available) for deep learning models.

6. References and AI Assistance

- MNIST Dataset
- AI Assistance: Guidance and code structuring suggestions were provided by ChatGPT.

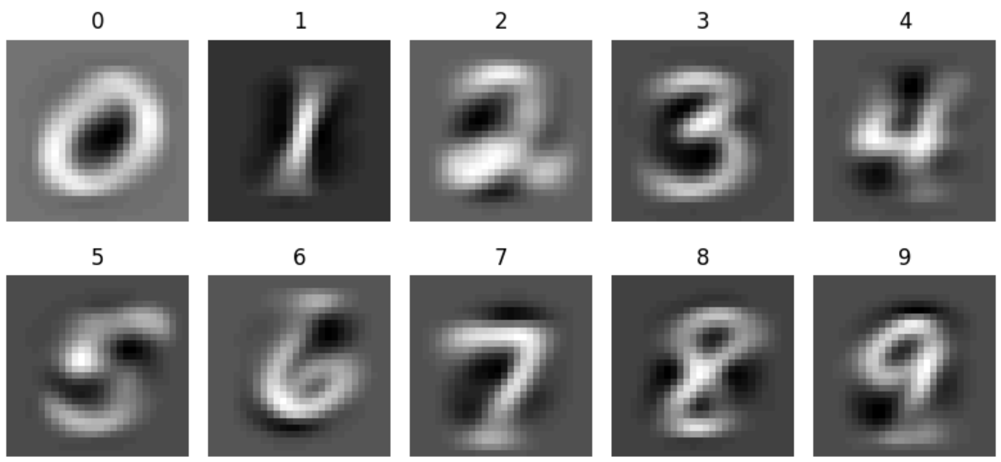
Appendix

Results Summary

- KNN (k = 1): 97.11%
- KNN (k = 3): 97.15%
- KNN (k = 5): 96.93%

- Naive Bayes: 83.24%
- Linear Classifier (Numpy): 75.28%
- Linear Model (PyTorch): 91.44%
- MLP: 95.53%
- CNN: 98.04%

The results show a consistent performance increase from simpler classical models to deep learning models. CNN achieved the best accuracy, demonstrating the benefit of convolutional feature extraction for image classification and its ability to capture hierarchical spatial patterns.



The above screenshot shows a visual of the learned weight vectors for the Linear Classifier (NumPy). Each image corresponds to digits 0–9. Brighter pixels indicate stronger positive weights.