AI 100 Midterm Project Report: MNIST Digit Classification

Name:
Date:

1) Problem definition and dataset curation

Goal: classify a 28x28 grayscale image of a handwritten digit into one of 10 classes (0–9).

Dataset: MNIST (60,000 train, 10,000 test) of handwritten digits. We load it using torchvision’s MNIST dataset loader and split the training set into 55,000 train and 5,000 validation samples.

Preprocessing: ToTensor() to map pixels to [0,1], then Normalize using standard MNIST mean/std.

2) Deep learning model

We use a small Convolutional Neural Network (CNN), which is well-suited for images because it learns local spatial features and shares parameters across the image.

Architecture: Conv(1→32, 3x3)+ReLU → Conv(32→64, 3x3)+ReLU → MaxPool(2x2) → MaxPool(2x2) → Flatten → FC(64*7*7→128)+ReLU+Dropout → FC(128→10 logits).

Loss: Cross-Entropy. Optimizer: Adam. Default hyperparameters: epochs=5, batch_size=128, lr=0.001, dropout=0.25.

3) Results and how they are presented

The training script logs per-epoch loss/accuracy and saves plots and evaluation artifacts under runs/.

Artifacts: metrics.json, training_curves.png (train/val loss), confusion_matrix.png, sample_predictions.png, and best_model.pt.

Typical outcome: a simple CNN on MNIST often reaches around ~99% accuracy after a few epochs with standard settings (see references).

4) Lessons and experience

CNNs are a strong baseline for image classification because they exploit local patterns.

A validation split is important for model selection and to detect overfitting.

Even on an easy dataset like MNIST, good hygiene (normalization, seeds, saved metrics) makes results reliable and easy to explain.

References

1. Yann LeCun et al., “Gradient-Based Learning Applied to Document Recognition,” 1998.

2. Google Developers Codelab: “TensorFlow, Keras and deep learning, without a PhD” (MNIST, 99% accuracy).

3. PyTorch documentation for torchvision datasets and training workflows.