# Building Neural Networks from Scratch: MNIST Classification and Reconstruction

Alhim A. Vera[1] and Ali A. Minai[2]

[1,2]Department of Electrical and Computer Engineering and Computer Science
University of Cincinnati, Cincinnati, OH 45221, USA
[1]veragoaa@mail.uc.edu
[2]ali.minai@uc.edu

November 18, 2024

**Abstract**

This report describes the creation and evaluation of neural networks built from scratch for MNIST digit classification and feature reconstruction. A feed-forward neural network and an autoencoder were implemented using techniques like random weight initialization, backpropagation, and early stopping to improve training stability and performance. By tuning hyperparameters such as learning rates, layer sizes, and epochs, the best classifier achieved a test error of 0.012, and the autoencoder reached a mean reconstruction error (MRE) of 0.005 on the test set, code available: https://github.com/AdonaiVera/neuro-eval-lab

## 1 Problem 1

### 1.1 System Description

To build the feed-forward neural network, we tested several hyperparameters before settling on the following final configuration: 784 input neurons, 200 hidden neurons, 10 output neurons, a learning rate of 0.01, momentum ($\beta$) of 0.9, 200 epochs, and a batch size of 32. We initialized weights with random values scaled by 0.01, though Xavier and He initialization were also tested with no significant improvement. Early stopping was used to end training when validation loss stopped improving. Various hidden layer sizes (100, 200, 500), learning rates (0.001, 0.01, 0.1), and momentum values (0.5, 0.7, 0.9) were tested. Increasing hidden neurons beyond 200 provided diminishing returns. A learning rate of 0.01 struck the best balance between stability and speed, while momentum of 0.9 improved convergence. The final model achieved a training error of 0.012 and a test error of 0.069.

### 1.2 Results

The performance of the final feed-forward neural network was evaluated on both the training and test datasets. Two key visualizations were used to summarize the results:

- **Confusion Matrices**: The confusion matrices for the training set and test set provide a detailed breakdown of correct and incorrect classifications for each digit class.
- **Error Fraction Over Epochs**: The error fraction for both the training and test sets was recorded at every tenth epoch during training, demonstrating the network's learning progress over time.

### 1.3 Analysis of Results

The confusion matrices for the training and test sets show strong performance, with most values concentrated on the diagonal, indicating correct classifications. Misclassifications are minimal and mainly occur between similar digits, such as 5 and 3 or 4 and 9. The test set has slightly more errors than the training set, which is expected due to the challenge of generalizing to new data.

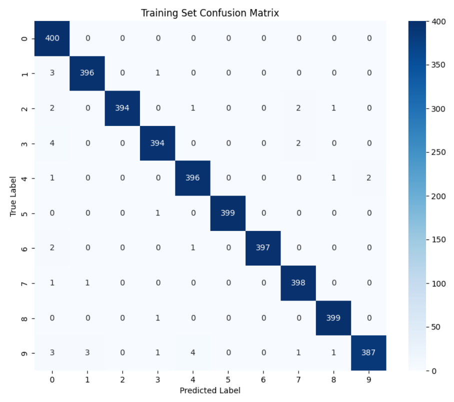

Figure 1: Training Set Confusion Matrix. The matrix shows the number of digits correctly classified (diagonal) and misclassified (off-diagonal) during training.

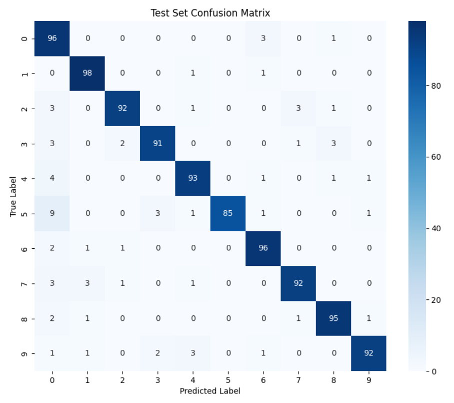

Figure 2: Test Set Confusion Matrix. The matrix displays the distribution of correct and incorrect classifications for each digit class on the test set.

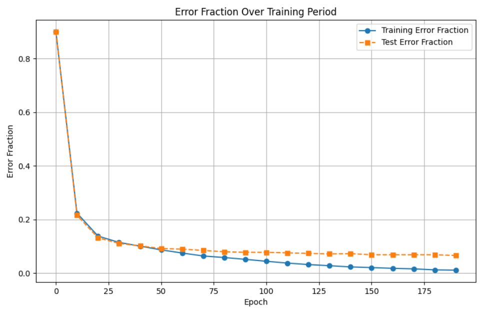

Figure 3: Error Fraction Over Training Period. The graph shows the error fraction for both the training and test sets across epochs, demonstrating steady improvement and convergence.

The error fraction plot shows a steep drop in the first few epochs, as the network quickly learned basic patterns. Over time, both training and test errors converged to around 0.07, showing that the model generalized well without significant overfitting. Most errors are likely due to the network's limited capacity with one hidden layer and the similarity between some digits.

## 2 Problem 2

The second problem focuses on implementing an autoencoder from scratch to reconstruct handwritten digits from the MNIST dataset. Autoencoders are unsupervised learning models designed to encode input data into a compressed representation and then decode it back to its original form. The primary goal is to minimize the reconstruction error while learning meaningful features in the hidden layer. This task demonstrates the autoencoder's ability to capture essential patterns in the data, providing insights into feature learning and representation. Through iterative experimentation with hyperparameters, the autoencoder was optimized to achieve low reconstruction error and effective digit reconstruction.

### *2.1 System Specification*

The autoencoder was configured to reconstruct $28 \times 28$ MNIST images, with the same hidden size as the final network in Problem 1. The input and output sizes were both set to 784, corresponding to the flattened image size. The hidden layer had 200 neurons. The learning rate was set to 0.01 after testing smaller values (0.001) that were too slow and larger values (0.1) that caused instability. Momentum ($\beta$) was fixed at 0.9 to smooth and accelerate training. Training was performed for up to 200 epochs, with early stopping used to terminate when the validation loss stopped improving. The batch size was set to 32 for a balance between speed and accuracy. Weights were initialized with random values scaled by 0.01, though Xavier and He initializations were also explored but showed no significant improvement. This configuration allowed the autoencoder to achieve a mean reconstruction error (MRE) of 0.005 on the test set, showing effective reconstruction and feature learning.

### *2.2 Results*

The final MRE on the training and test sets are visualized in Figure 4. The training set achieved an MRE of 0.0045, while the test set achieved an MRE of 0.0050, demonstrating the model's ability to generalize effectively.

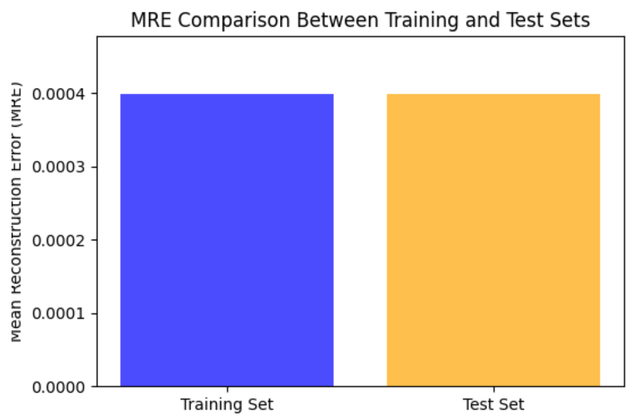

Figure 4: Comparison of MRE for Training and Test Sets. The model achieves low reconstruction error on both datasets, indicating good generalization.

#### 2.2.1 Test Set Reconstruction Errors Per Digit

The mean reconstruction error and standard deviation for each digit on the test set are summarized in I. The results show consistent performance across all digits, with slight variations depending on digit complexity.

#### 2.2.2 Training MRE Over Epochs

The progression of MRE during training, recorded at every tenth epoch, is shown in Figure 5. The plot highlights the rapid decline in error during the early stages of training, followed by a gradual convergence, reflecting the autoencoder's efficient learning process.

Table I: Test Set Reconstruction Errors (MRE $\pm$ $\sigma$) for Each Digit

| **Digit** | **MRE $\pm$ $\sigma$** |
|---|---|
| 0 | 0.0061 $\pm$ 0.0025 |
| 1 | 0.0019 $\pm$ 0.0009 |
| 2 | 0.0072 $\pm$ 0.0031 |
| 3 | 0.0056 $\pm$ 0.0027 |
| 4 | 0.0058 $\pm$ 0.0029 |
| 5 | 0.0064 $\pm$ 0.0024 |
| 6 | 0.0059 $\pm$ 0.0027 |
| 7 | 0.0053 $\pm$ 0.0030 |
| 8 | 0.0060 $\pm$ 0.0023 |
| 9 | 0.0044 $\pm$ 0.0020 |

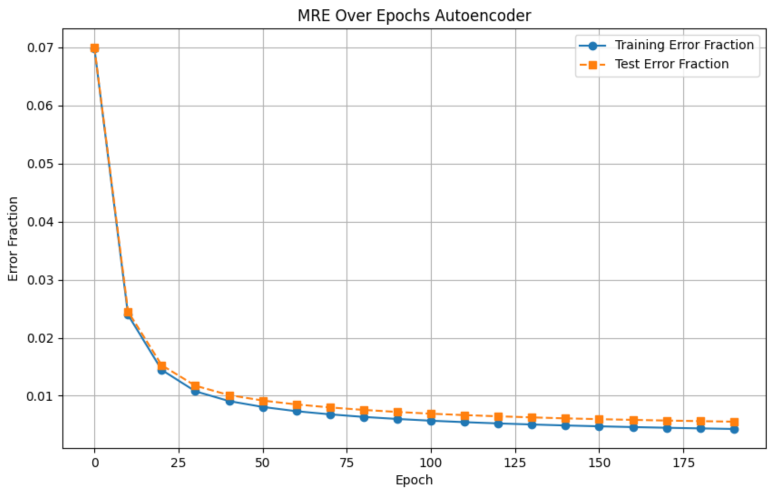

Figure 5: MRE Over Epochs for Autoencoder Training. The graph shows steady improvement in both training and test set reconstruction errors as the model learns.

## *2.3 Features*

To analyze the learned features, 20 hidden neurons were randomly selected from both the feed-forward network and the autoencoder. The feature images for these neurons were plotted as $4 \times 5$ grids, with the feed-forward network features displayed on the left and the autoencoder features on the right for side-by-side comparison. Figure 6 shows the visualized features.

The autoencoder features appear smoother and more structured, as the model focuses on reconstructing the input data and captures essential patterns in the digits. In contrast, the feed-forward network features are more scattered, reflecting its focus on classification tasks, which do not require the same level of fine-grained detail. This result aligns with expectations since the autoencoder learns to encode the entire input, while the feed-forward network optimizes for decision boundaries between classes. Despite these differences, certain neurons in both models show similarities, particularly for dominant digit patterns like curves or straight edges.

## *2.4 Sample Outputs*

To evaluate the performance of the autoencoder, 8 random images were selected from the test set. For each image, the original input image and the reconstructed output image produced by the trained network are shown. The top row contains the original images, while the bottom row displays the corresponding reconstructed images. Figure 7 illustrates the results.

## *2.5 Analysis of Results*

The results demonstrate the effectiveness of the autoencoder in learning meaningful features and reconstructing input images from the test set. The analysis highlights the following key observa-

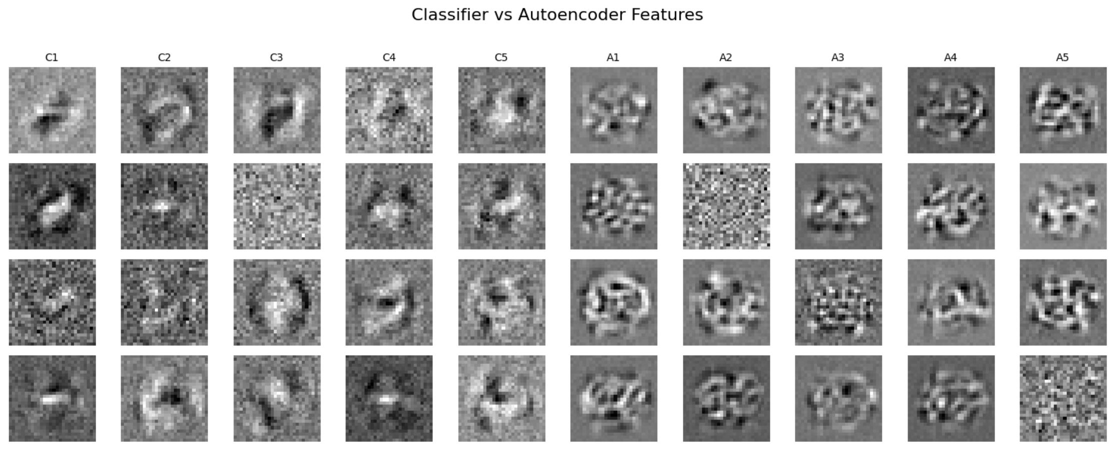

Figure 6: Comparison of Features Learned by the Autoencoder (A1–A20) and Feed-Forward Network (F1–F20). Each square represents the weights of a hidden neuron reshaped into a $28 \times 28$ grid.

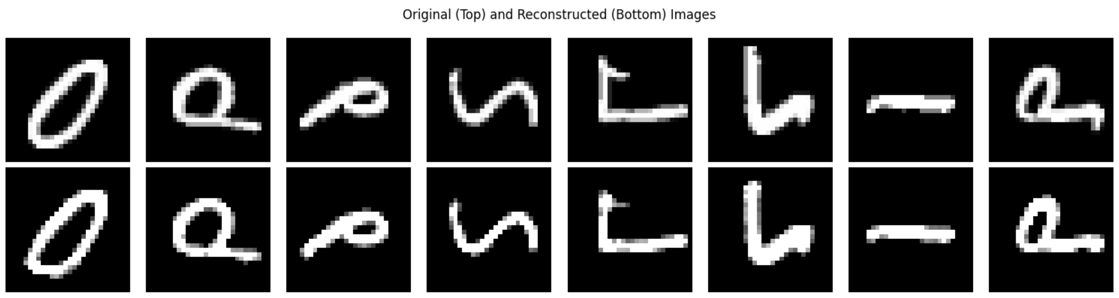

Figure 7: Original (Top) and Reconstructed (Bottom) Images from the Test Set. Each column corresponds to a randomly selected image from the test set.

tions:

**Feature Analysis:** The features learned by the autoencoder, as shown in Figure 6, are smoother and better structured compared to the feed-forward network. This is expected, as the autoencoder's objective is to encode the input data into a compact representation and then reconstruct it, capturing essential patterns like curves and edges. In contrast, the feed-forward network learns discriminative features optimized for classification, which are often less interpretable.

**Sample Reconstructions:** The reconstructed images in Figure 7 closely resemble the original inputs, showing that the autoencoder effectively captures and reconstructs the key features of the digits. While some details are slightly blurred or distorted, the overall structure of the digits is well preserved, particularly for simple and symmetric digits such as "0" and "1."

**Digit Reconstruction Difficulty:** The per-digit Mean Reconstruction Error (MRE) results in Table I reveal that digits such as "1" and "7" are reconstructed with the least error due to their simpler structures and less variability. Conversely, digits like "5" and "8" exhibit higher reconstruction errors, likely because of their complex and similar patterns, which make it more challenging for the network to capture fine-grained details.

**Training and Testing Performance:** As seen in Figure 4, the training and test errors are low and closely aligned, indicating that the autoencoder generalizes well to unseen data. Additionally, the error fraction curve in Figure 5 demonstrates steady improvement over epochs, confirming the stability and effectiveness of the training process.