Exploring Autoencoders and Transfer Learning: Classification and Feature Generalization on MNIST Digits

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 26, 2024

Abstract

This report explores the use of autoencoders and transfer learning with the MNIST dataset. In Problem 1, initializing weights from a pre-trained autoencoder significantly improved classification accuracy and training speed. Problem 2 demonstrated the transferability of features, with the autoencoder effectively reconstructing unseen digits with varying success depending on digit similarity. These results showcase the strengths and limitations of feature transfer in neural networks, code available: https://github.com/AdonaiVera/neuro-eval-lab

1 Problem 1

In this problem, we aim to explore the concept of transfer learning by leveraging features learned in one task (image reconstruction) to improve performance in another task (image classification). Specifically, we utilize the weights from the input-to-hidden layer of an autoencoder, trained on the reconstruction task, as a starting point for a neural network tasked with classifying digits from the MNIST dataset.

1.1 Results

The table below summarizes the overall performance metrics for Case I, Case II, and the baseline network from HW 4, highlighting the differences in initial and final training and test errors. This comparison demonstrates the impact of transfer learning on network performance.

Table I: Overall Results for Case I, Case II, and Baseline (HW 4)

Metric	**Case I**	**Case II**	**HW 4 (Baseline)**
Initial Training Error	0.914	0.878	0.868
Initial Test Error	0.915	0.877	0.862
Final Training Error	0.016	0.011	0.012
Final Test Error	0.019	0.016	0.069

1.1.1 Confusion Matrices

The confusion matrices provide a detailed view of the classification performance for Case I, Case II, and the baseline network (Homework 4). These matrices display the distribution of correctly and incorrectly classified samples across all digit classes, highlighting the strengths and weaknesses of each model in recognizing specific digits. They serve as a crucial tool for evaluating the effectiveness of the learned features and training strategies.

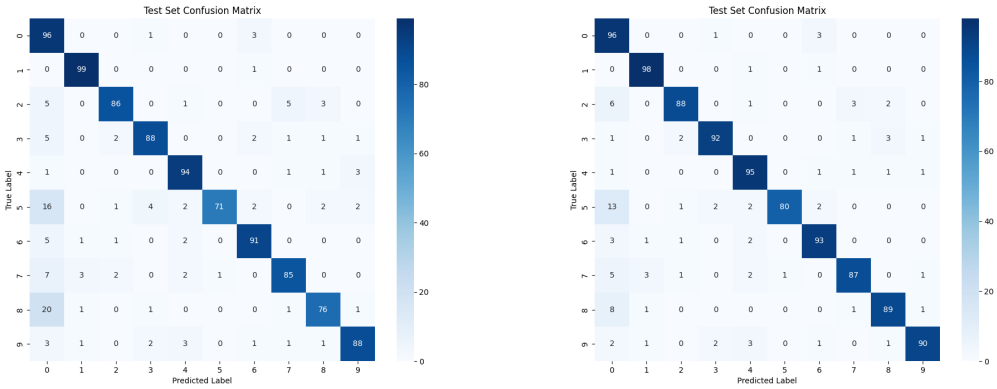

Figure 1: Confusion Matrices for the Test Set: **Left:** Case I. **Right:** Case II.

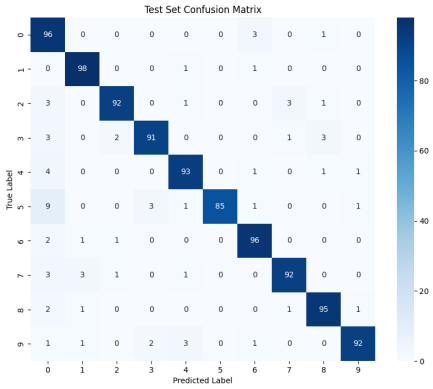

Figure 2: Confusion Matrix for the Test Set: Baseline (HW 4).

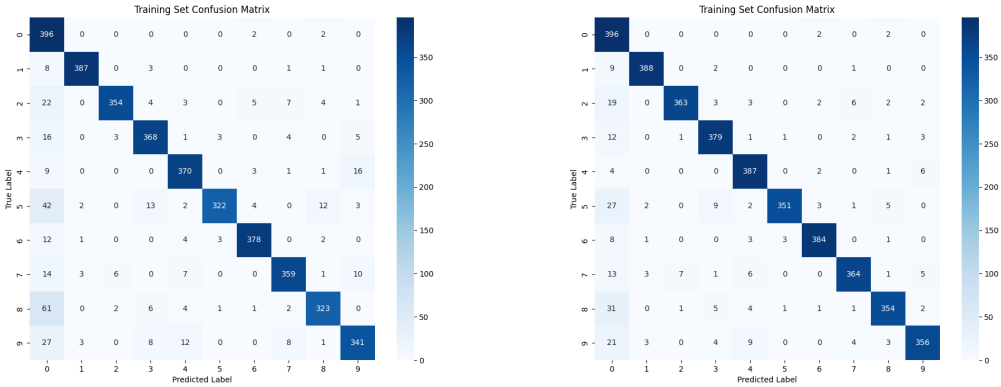

Figure 3: Confusion Matrices for the Training Set: **Left:** Case I. **Right:** Case II.

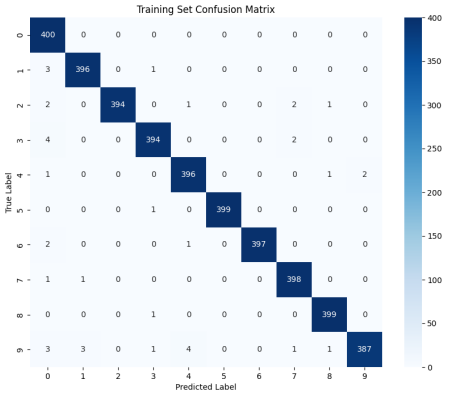

Figure 4: Confusion Matrix for the Training Set: Baseline (HW 4).

1.1.2 Mean Errors Per Digit and Overall Performance

The mean error plots for Case I and Case II provide a comprehensive view of the training and test errors for each digit, alongside the overall error. These plots offer valuable insights into the performance differences between the two training strategies. The overall training and test errors for Case I were **0.0167** and **0.0197**, respectively, while Case II achieved superior performance with overall training and test errors of **0.0122** and **0.0173**, respectively. The improved results in Case II highlight the benefits of training all layers of the network, enabling it to refine features for better generalization and accuracy.

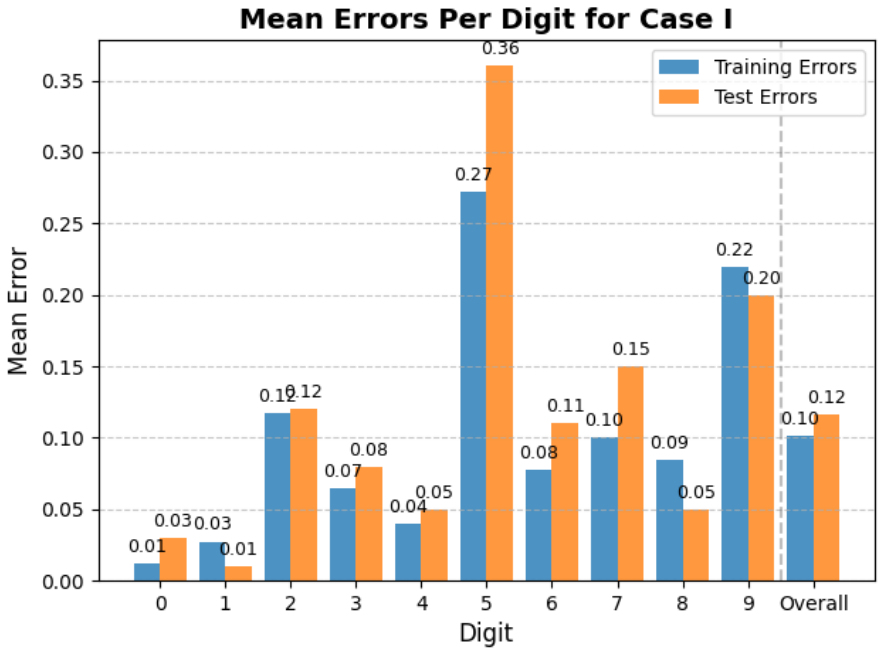

Figure 5: Mean Errors Per Digit for Case I. Training and test errors are displayed for each digit and the overall performance.

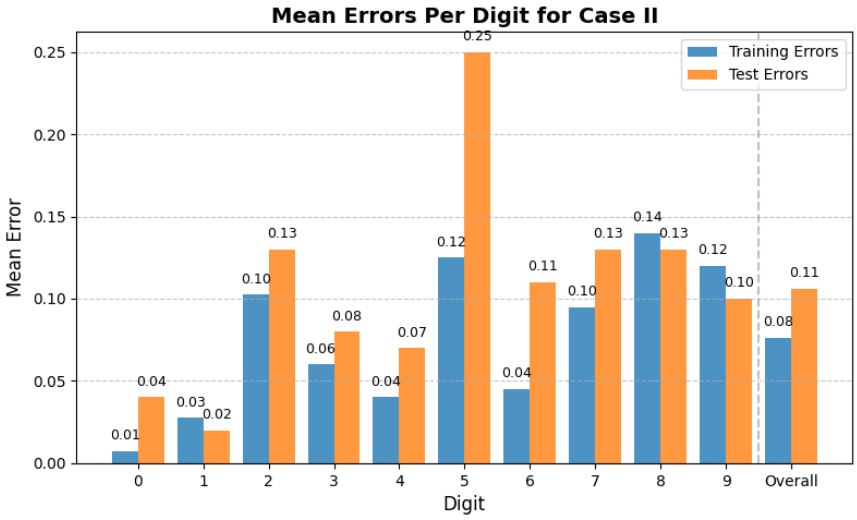

Figure 6: Mean Errors Per Digit for Case II. Training and test errors are displayed for each digit and the overall performance.

1.1.3 Error Fraction Over Training Epochs

The error fraction plots illustrate the progression of training and test errors across epochs for Case I, Case II, and the baseline network. These visualizations highlight the rate of convergence and the overall effectiveness of each training strategy, providing insights into how quickly and accurately the models learn over time.

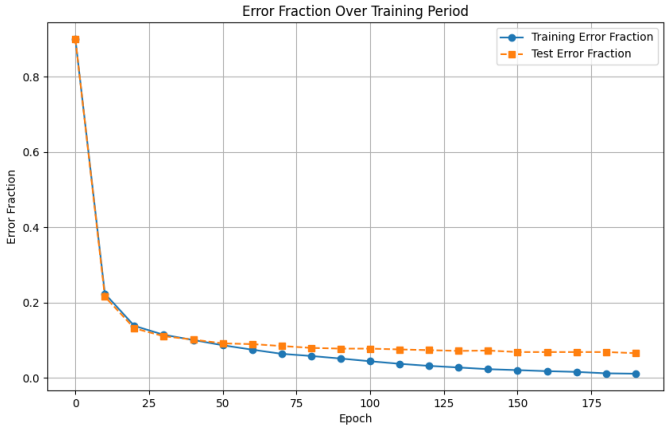

Figure 7: Error Fraction Over Training Period: Baseline (HW 4).

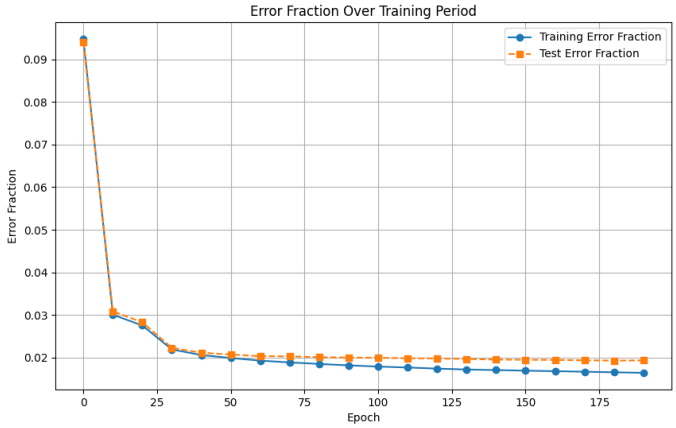

Figure 8: Error Fraction Over Training Period: Case I.

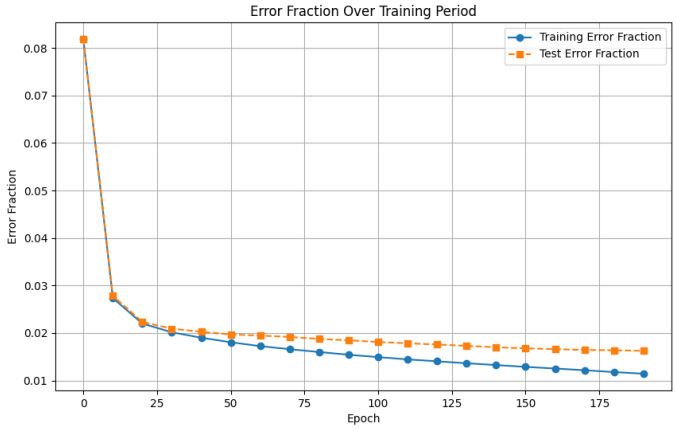

Figure 9: Error Fraction Over Training Period: Case II.

1.2 Analysis of Results

The results clearly demonstrate the benefits of transfer learning when initializing the hidden weights from the autoencoder. Both Case I and Case II outperformed the baseline network from Homework 4 in terms of error reduction, convergence speed, and classification accuracy. The use of pre-trained weights provided a strong starting point for the models, allowing them to bypass the need to learn basic feature representations from scratch, which significantly accelerated the training process and improved final performance metrics.

In Case I, where the input-to-hidden weights were frozen and only the hidden-to-output weights were trained, the network achieved a final test error of 0.01935. This marks a significant improvement over the baseline's test error of 0.069. The error fraction plots show a much faster convergence for Case I, with the model reaching low error levels in fewer epochs compared to the baseline. The confusion matrix for Case I highlights improved classification accuracy across most digits, particularly for challenging digits like '8' and '9'. However, some misclassifications persisted, likely due to the inability to refine the pre-trained input-to-hidden weights to better adapt to the classification task.

Case II, which allowed training of both input-to-hidden and hidden-to-output weights, achieved the best overall performance with a final test error of 0.01624. The model not only converged faster than both Case I and the baseline but also demonstrated the lowest misclassification rates in the confusion matrix. This result underscores the advantage of fine-tuning pre-trained weights, as it enables the network to refine the features learned during the reconstruction task and better adapt them to the specific requirements of the classification problem. The reduced error rates for digits like '5' and '9' further highlight the improved generalization and adaptability of Case II.

The baseline network, on the other hand, struggled with slower convergence and higher error rates. Its final test error of 0.069 reflects the inefficiency of learning entirely from scratch with randomly initialized weights. The error fraction plots show a gradual decline in training and test errors, but the lack of meaningful initial features led to longer training times and poorer final performance. The confusion matrix for the baseline reveals higher misclassification rates for several digits, especially for visually similar ones, which the pre-trained networks in Case I and Case II handled more effectively.

Overall, the results strongly support the effectiveness of transfer learning. Initializing the hidden weights with features learned by the autoencoder provided a significant boost in training efficiency and model performance. Case II, which trained all layers, demonstrated the best results by refining these pre-trained features, achieving the lowest errors and the most accurate classifications. These findings highlight the importance of leveraging pre-trained weights in neural networks, particularly when transitioning from one task (e.g., reconstruction) to another (e.g., classification).

2 Problem 2

The second problem explores the ability of an autoencoder to generalize its learned representations from one subset of digits to another. Specifically, the autoencoder is trained using images of digits 0 through 4 from the MNIST dataset and evaluated based on its reconstruction accuracy for these digits. The trained network is then tested, without further adjustments, on digits 5 through 9 to assess its capacity to reconstruct previously unseen data. This approach evaluates how effectively the autoencoder's learned features can be transferred across similar but distinct subsets of data. By analyzing reconstruction errors and visual outputs, we aim to uncover insights into the network's generalization capabilities and its inherent limitations.

2.1 Results

In this section, we present the results for the second problem, including reconstruction errors for each digit and visual comparisons of original and reconstructed images for digits 5 through 9.

2.1.1 Reconstruction Errors for All Digits

The reconstruction errors (MRE $\pm$ σ) were computed for all digits in the MNIST dataset. Digits 0 through 4 were used for training, and their reconstruction errors are expected to be low. For digits 5 through 9, the network's performance was evaluated without additional training, leading to higher reconstruction errors due to their absence in the training set. The results are summarized in Table II.

Table II: Reconstruction Errors (MRE $\pm$ σ) for Each Digit

Digit	**MRE $\pm$ σ**
0	0.0083 $\pm$ 0.0039
1	0.0023 $\pm$ 0.0013
2	0.0094 $\pm$ 0.0040
3	0.0083 $\pm$ 0.0039
4	0.0082 $\pm$ 0.0038
5	0.0134 $\pm$ 0.0053
6	0.0121 $\pm$ 0.0051
7	0.0114 $\pm$ 0.0060
8	0.0127 $\pm$ 0.0051
9	0.0082 $\pm$ 0.0032

2.1.2 Visual Comparison of Reconstruction for Digits 5 to 9

To illustrate the network's reconstruction capabilities, we randomly selected 5 samples from the test set for each digit between 5 and 9. The original images and their corresponding reconstructions are displayed in two rows of 5 images per digit. The top row represents the original images, while the bottom row represents the reconstructed outputs from the autoencoder. Figures 10 through 14 show these visual comparisons.

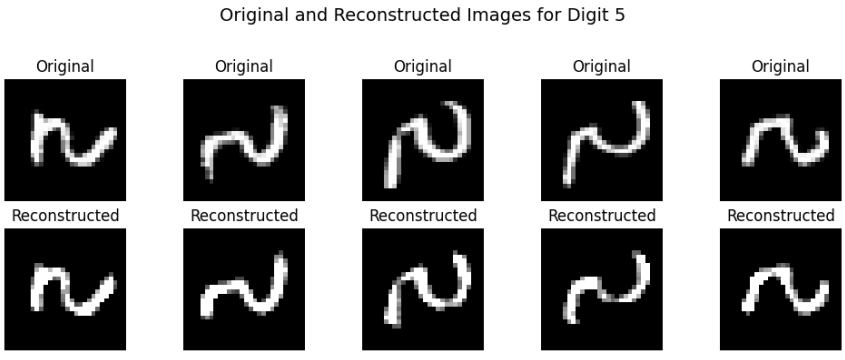

Figure 10: Original and Reconstructed Images for Digit 5

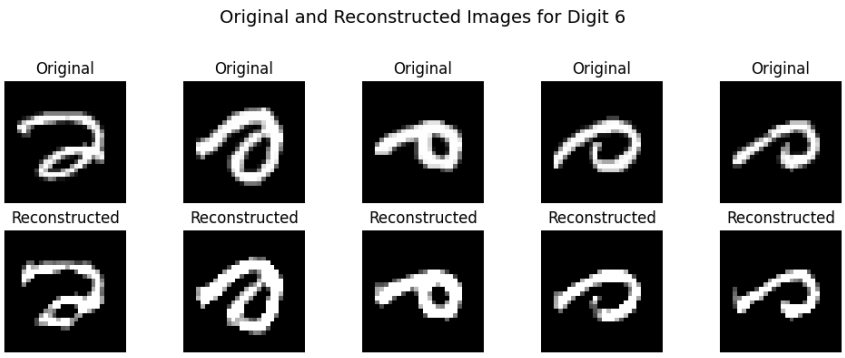

Figure 11: Original and Reconstructed Images for Digit 6

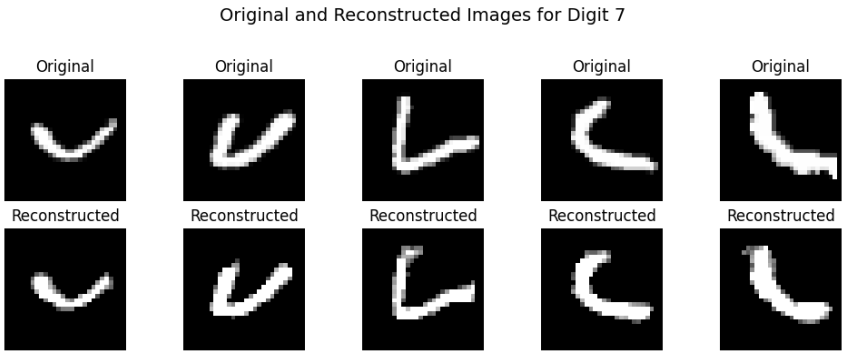

Figure 12: Original and Reconstructed Images for Digit 7

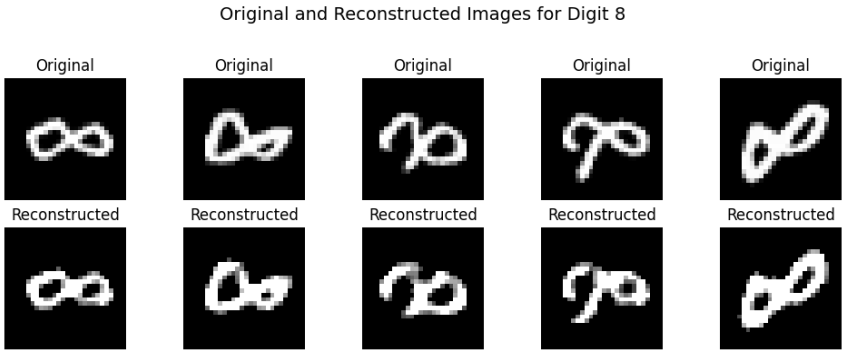

Figure 13: Original and Reconstructed Images for Digit 8

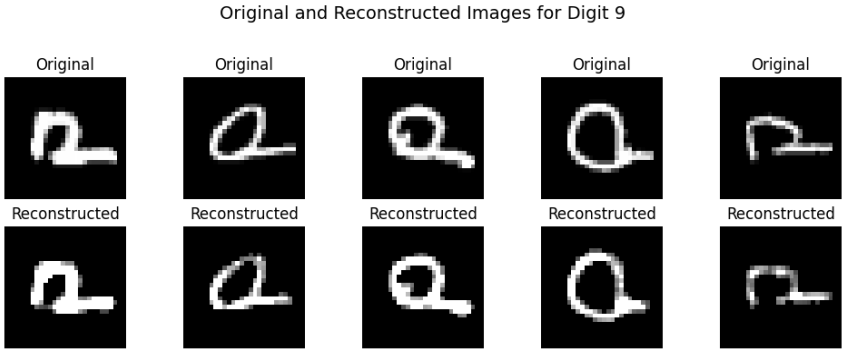

Figure 14: Original and Reconstructed Images for Digit 9

2.2 Discussion

The results of this problem demonstrate the autoencoder's ability to transfer learned features from one subset of digits (0 through 4) to another subset (5 through 9). For digits 0 through 4, which were part of the training set, the reconstruction errors are consistently low. This indicates that the autoencoder successfully captured meaningful feature representations for these digits. On the other hand, the reconstruction errors for digits 5 through 9 are higher, reflecting the challenges associated with applying learned features to unseen data. However, the modest increase in reconstruction errors suggests that the features learned by the autoencoder are partially transferable to unseen digits.

A closer examination of the results reveals variability in the transfer performance across digits 5 through 9. As shown in Table II, digit 9 achieves the lowest reconstruction error among the unseen digits (MRE = 0.0082 ± 0.0032), while digit 5 exhibits the highest error (MRE = 0.0134 ± 0.0053). This variability can be attributed to the structural similarities between the unseen digits and the training digits. For example, digit 9 shares visual features, such as loops and curves, with training digits like 0 and 4, making it easier for the autoencoder to reconstruct. In contrast, digit 5 contains more distinct structural elements, such as sharp angles and straight lines, which are less represented in the training set, leading to higher reconstruction errors.

Overall, the results underscore the moderate success of the autoencoder in transferring features to new digits. The effectiveness of the transfer appears to depend on the degree of similarity between the unseen and the training digits. This variability emphasizes the importance of including diverse examples in the training set to ensure robust generalization. In practical applications, where unseen categories are expected, strategies such as data augmentation, semi-supervised learning, or fine-tuning on additional data could be explored to improve the model's ability to handle novel inputs. The findings from this analysis provide valuable insights into the strengths and limitations of using autoencoders for feature transfer across similar datasets.