Bilkent University

CS 464 - Introduction to Machine Learning

Homework 02 Report

Emre Karataş

Section 01

22001641

December 13, 2023

Contents

1 PCA Analysis [50 pts]

I analyzed the MNIST dataset using Principal Component Analysis (PCA) in this question. The MNIST dataset comprises 70,000 grayscale digit images, split into 60,000 training and 10,000 test images. Each image is a 28x28 pixel resolution grayscale representation of handwritten digits. For this analysis, I used only the training data. The dataset files are:

- train-images-idx3-ubyte.gz
- train-labels-idx1-ubyte.gz

My approach involves implementing PCA from scratch, as library use for PCA calculations is not permitted. The analysis starts by flattening each 28x28 image into a 784-dimensional vector, resulting in a data shape of 60,000 x 784, where 60,000 represents the number of samples.

1.1 Question 1.1 [15 pts]

The PCA (Principal Component Analysis) applied to the MNIST dataset showed the following Proportion of Variance Explained (PVE) for the first ten principal components:

Component	PVE (%)
1	9.70
2	7.10
3	6.17
4	5.39
5	4.87
6	4.31
7	3.27
8	2.88
9	2.76
10	2.36

Table 1: Proportion of Variance(PVE) Explained by PCA Components

These results indicate a variance dispersion across multiple dimensions, characteristic of complex datasets like MNIST. The relatively balanced variance distribution among the top components suggests that no single piece captures a dominant part of the dataset's variance. This reflects the multi-dimensional and diverse nature of the handwritten digits in the MNIST dataset.

1.2 Question 1.2 [5 pts]

After applying Principal Component Analysis (PCA) to the MNIST dataset, I have determined the following:

1. *Number of Principal Components:* The analysis revealed that **26 components** are sufficient to explain over **70%** of the total variance in the dataset.
2. *Cumulative Variance:* These components cumulatively account for **70.02%** of the variance, underscoring PCA's ability to condense dataset complexity significantly.

This result indicates that a substantial portion of the dataset's variance can be captured with a relatively small number of components, demonstrating the efficacy of PCA in reducing the dimensionality of the data set while retaining most of the information.

1.3 Question 1.3 [10 points]

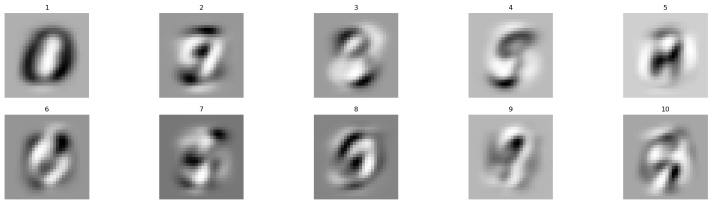

Figure 1: Grayscale Principal Component Images of Size 28 × 28

After reshaping principal components to 28x28 matrices and applying minmax scaling, these are the images that I got from the code. We can see that some of the The first 10 principal components indeed look like numbers. For example, the more black regions actually show weights and indicate the handwritten digit lines, even though they are blurry to see. This stems from the matrix size itself, which is very small for that visualization.

1.4 Question 1.4 [10 points]

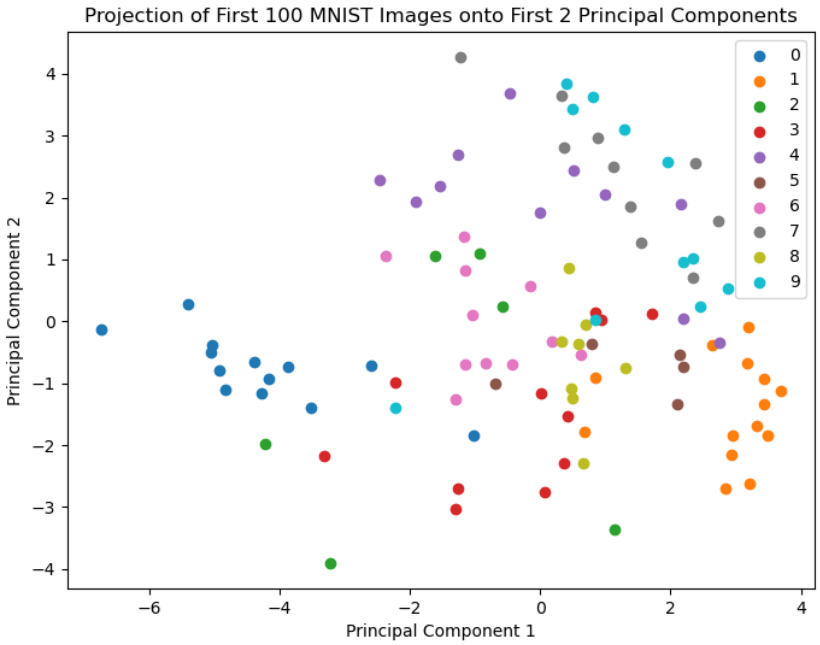

Figure 2: Projected Data Points on the 2-D Space

In this plot, distinct clusters of the same hue indicate that the respective digit representations share commonalities in variance that are captured by the principal components. Conversely, the degree of overlap between different colored clusters may suggest feature similarities across those digits, potentially indicating areas where the classifier might confuse one digit for another. The relative positioning of the clusters along the principal components can provide insights into the most significant axes of variation among the digits. For instance, a tight cluster suggests low-class variance, whereas widely dispersed points within the same color reflect greater diversity in the digit's representation.

1.5 Question 1.5 [10 points]

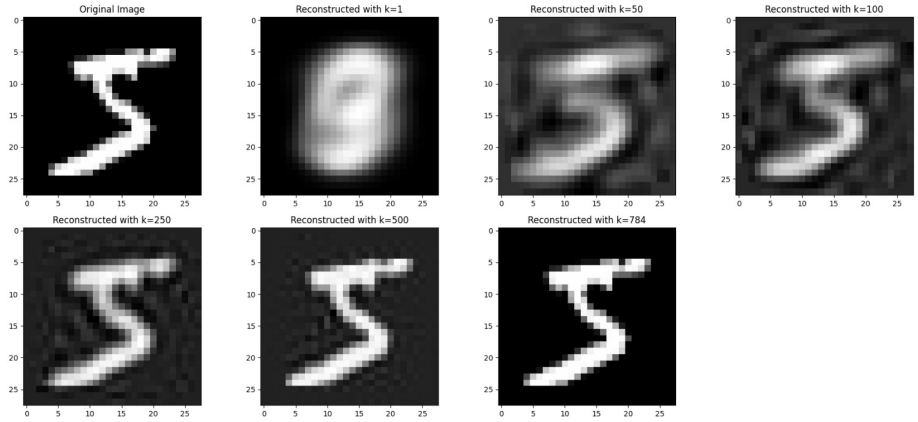

Figure 3: Reconstructed Images of Different 'k' Values

As the number of principal components k used for reconstruction increases, the clarity and detail of the MNIST digit images improve. At very low k values, such as 1, the images are highly abstract, emphasizing only the broadest strokes of variance across the dataset. Intermediate k values like 50 and 100 begin to reveal more specific characteristics of each digit, though with some residual blurring, reflecting a mix of major and minor features. High k values, such as 250 and 500, offer reconstructions with fine detail, closely approximating the original images, as most variations specific to individual digits are captured. At the maximum k of 784, the reconstruction is virtually identical to the original, but this includes noise as well, illustrating the point of diminishing returns where additional components fail to significantly enhance the essential data structure but may introduce unnecessary complexity or noise.

2 Logistic Regression [50 points]

In this part of the homework, I have developed a Multinomial Logistic Regression Classifier model aimed at classifying digit images from the MNIST database. The MNIST dataset, known for its collection of handwritten digits, originally included only training and test data. To enhance the model's validation process, I partitioned the first 10,000 images from the training data set, along with their corresponding labels, to create a validation set. This resulted in a distribution of 50,000 images for training, 10,000 for testing, and 10,000 for validation.

The model uses one-hot encoding for the labels. Furthermore, the weight matrices, crucial for the classification task, are initialized with their bias terms. The following subsections provide detailed insights and responses to various questions concerning the application of Logistic Regression to the MNIST data set.

2.1 Question 2.1 [15 pts]

Upon training, the model's performance was evaluated on the test data set. The key metrics for evaluation included:

- **Test Accuracy**: This metric quantifies the proportion of correct predictions made by the model out of all predictions.

- **Confusion Matrix**: A matrix that visualizes the performance of the classification model. Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class.

2.1.1 Model Training Results

The model was trained over a series of epochs, and the loss was recorded at regular intervals. Following the training, the model achieved a **test accuracy of 90.90%**, indicating a high level of predictive performance on the unseen test data. Here are some key points observed during the training phase:

- Epoch 0, Loss: 3.28
- Epoch 10, Loss: 1.29
- Epoch 20, Loss: 1.13
- Epoch 30, Loss: 1.04
- Epoch 40, Loss: 0.99
- Epoch 50, Loss: 0.94
- Epoch 60, Loss: 0.90
- Epoch 70, Loss: 0.87
- Epoch 80, Loss: 0.84
- Epoch 90, Loss: 0.82

2.1.2 Confusion Matrix Visualization

The confusion matrix of the model on the test data set is visualized in the figure below:

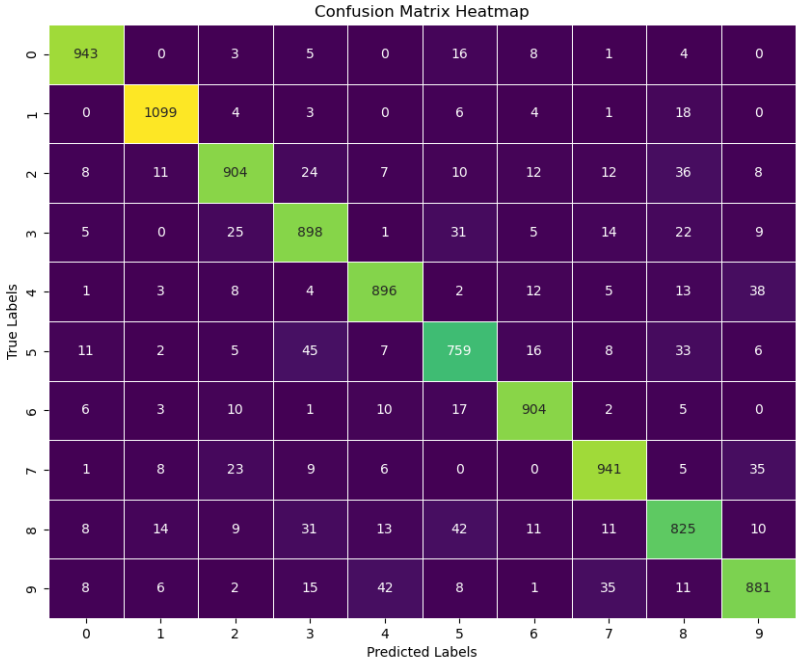

Figure 4: Confusion Matrix of the Default Logistic Regression Model

2.2 Question 2.2 [15 points]

For this part of the question, I did separate experiments on the hyper-parameters mentioned. I developed 13 different scenarios for different hyper-parameter effect testing. These hyper-parameters and their corresponding test values are as follows:

- **Batch Size**: I experimented with batch sizes of 1, 64, and 50000.
- **Weight Initialization Technique**: Different initialization techniques were tested, including zero initialization, uniform distribution, and normal distribution.
- **Learning Rate**: The model was trained with varying learning rates, specifically 0.1, 10^{-3}, 10^{-4}, and 10^{-5}.
- **Regularization Coefficient** (λ): Regularization coefficients tested were 10^{-2}, 10^{-4}, and 10^{-9}.

Each hyper-parameter impacts the model's learning dynamics and overall performance in distinct ways, as detailed in the following subsections and corresponding graphical analyses.

2.2.1 Hyperparameter: Batch Size

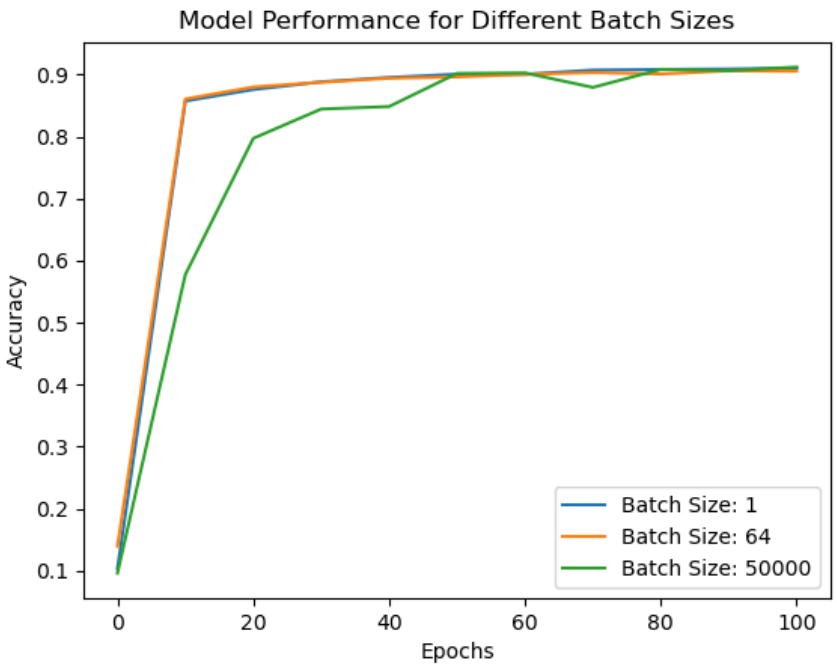

Figure 5: Accuracy vs Epoch for Different Batch Sizes

The first graph displays the performance of a model across different batch sizes: 1, 64, and 50,000. The model with a batch size of 1 demonstrates rapid learning in the early epochs, which is expected and beneficial. However, training with batch size 1 is significantly slower, taking about 10 times longer than other batch sizes. On the other hand, larger batch sizes, particularly 50,000, show slower learning rates as the number of epochs increases. This suggests that very large batch sizes might not be ideal for this model, even though they still perform adequately. These trends indicate a trade-off between learning speed and computational efficiency.

2.2.2 Hyperparameter: Weight Initialization Technique

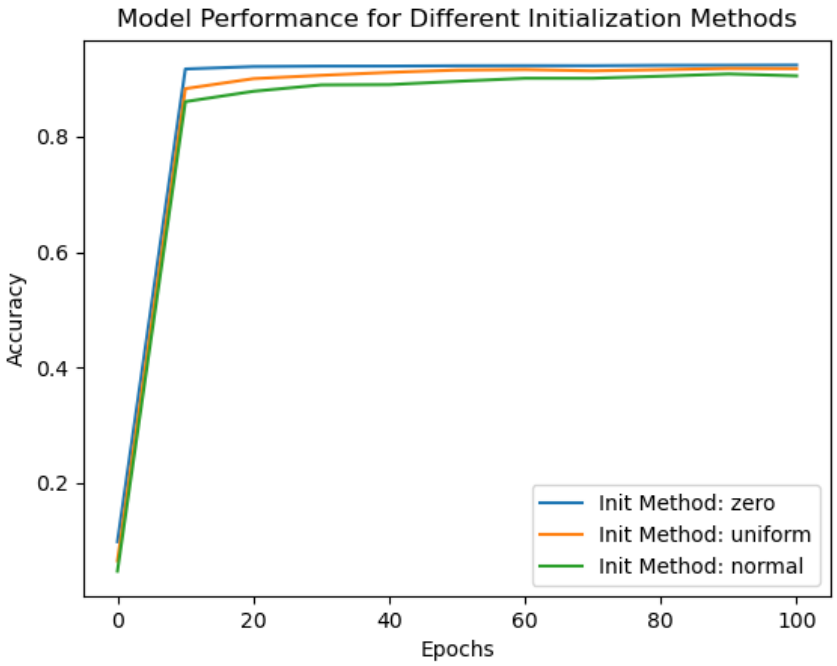

Figure 6: Accuracy vs Epoch for Different Weight Initialization Techniques

This graph indicates that the model's accuracy reaches a plateau early on during training for all three weight initialization methods: zero, uniform, and normal. The steep increase at the beginning suggests rapid learning for each method, but as the epochs progress, the differences in accuracy between the methods diminish, with all three converging to a similar value. This convergence implies that the choice of weight initialization might not significantly impact the final performance of the model.

2.2.3 Hyperparameter: Learning Rate

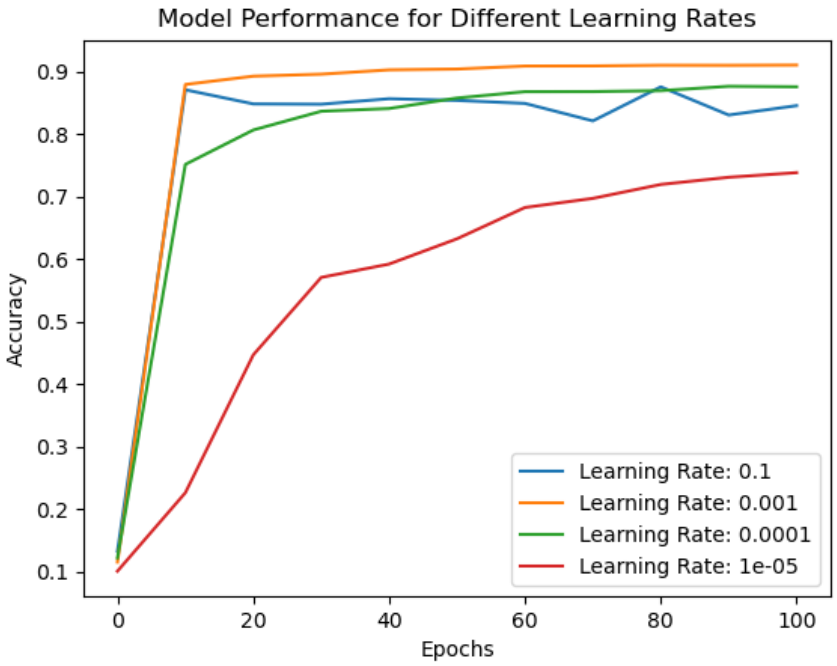

Figure 7: Accuracy vs Epoch for Different Learning Rates

The first graph presents the model's performance over different learning rates: specifically 0.1, 10^{-3}, 10^{-4}, and 10^{-5}. The learning rate of 0.1 shows rapid improvement in accuracy initially, but it plateaus quickly and then declines, suggesting that such a high learning rate may lead to instability or overfitting in later epochs. In contrast, a learning rate of 10^{-3} demonstrates a more controlled and consistent improvement, ultimately achieving the highest accuracy, which indicates it may be the optimal learning rate for this model. Meanwhile, the learning rate of 10^{-4} exhibits a steady but slower increase in accuracy, implying that while it may eventually converge to a similar accuracy given more epochs, it is less efficient. The smallest learning rate, 10^{-5}, increases accuracy very slowly, not converging within the given 100 epochs, which may be impractically slow for most applications.

2.2.4 Hyperparameter: Regularization Coefficient

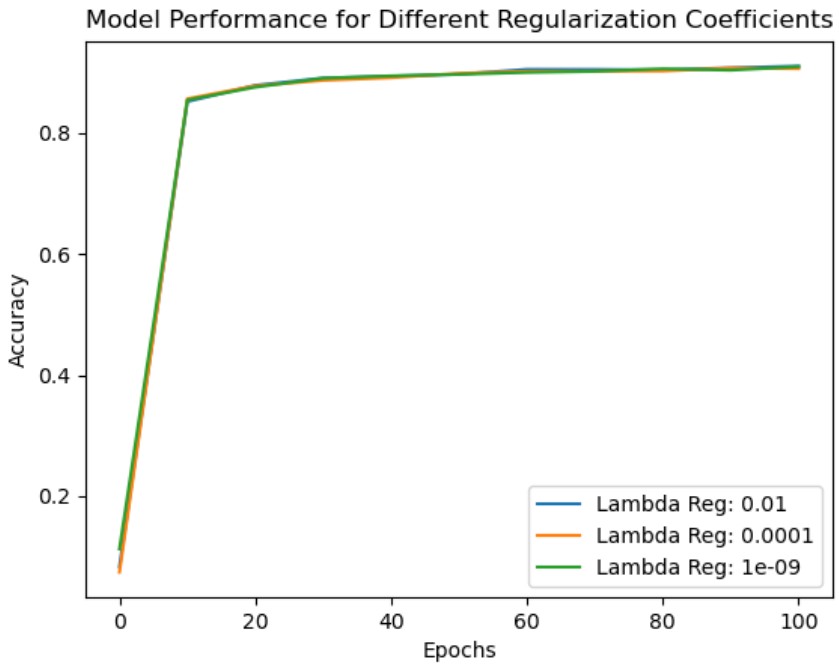

Figure 8: Accuracy vs Epoch for Different Regularization Coefficients

The graph illustrates the impact of different regularization coefficients on model accuracy over 100 epochs. The regularization coefficients compared are $\lambda = 0.01$, $\lambda = 0.0001$, and $\lambda = 1 \times 10^{-9}$. It is observed that the model with $\lambda = 0.01$ experiences a sharp increase in accuracy initially but then plateaus, suggesting that while this level of regularization prevents overfitting, it may also be constraining the model's ability to fit the training data closely. For $\lambda = 0.0001$, the accuracy improves at a similar pace but continues to increase slightly longer before plateauing, indicating a better balance between bias and variance. The model with $\lambda = 1 \times 10^{-9}$, which applies a very minimal regularization effect, shows an almost indistinguishable performance from the model with $\lambda = 0.0001$, implying that the impact of such a small regularization term is negligible. This graph explains that the MNIST dataset is sufficiently large and representative such that regularization does not play a significant role in model performance.

2.3 Question 2.3 [5 pts]

After conducting the aforementioned experiments in section 2.2, the optimal hyperparameters and weight initialization technique must be selected based on model performance. This involves choosing the values that yield the highest validation accuracy. The finalized model, incorporating these parameters, will then be evaluated on the test dataset.

According to results obtained from 2.2, selected hyperparameters and initialization weight are given as follows:

- **Batch Size**: 64
- **Initialization Method**: zero
- **Learning Rate**: 0.001
- **Regularization Coefficient** (λ): 0.01

Furthermore, the test accuracy of the optimal model can be displayed as:

Test Accuracy: 92.39%

In addition to that, the confusion matrix for the best model, which provides insight into the true versus predicted classifications, can be represented as follows in 10x10 heatmap image format:

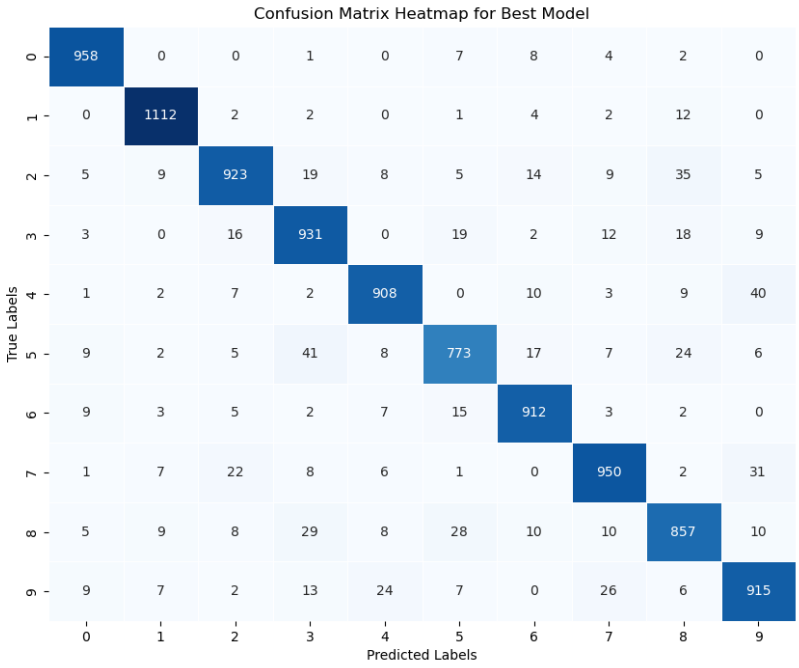

Figure 9: Confusion Matrix Heatmap for Best Model

2.4 Question 2.4 [10 pts]

In this section of homework, I visualized the weights as images, as can be seen below:

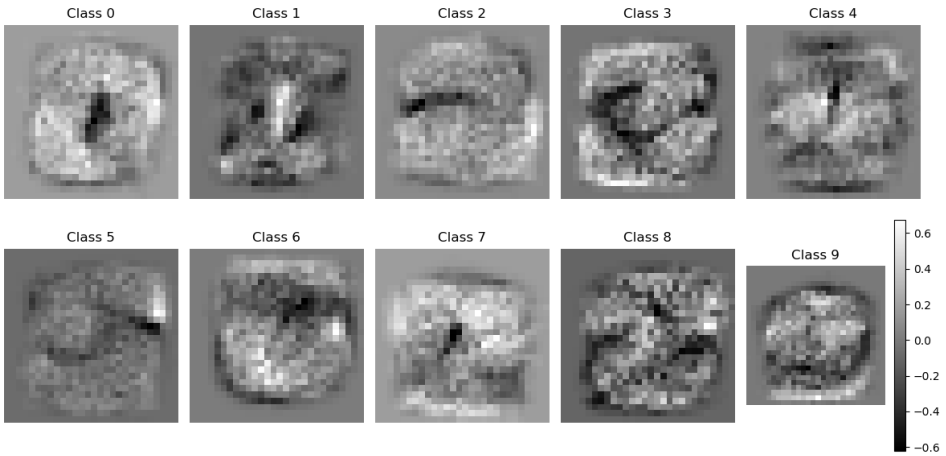

Figure 10: Weight Vectors of Each Digit as an Image

When visualizing weight vectors as images post-model training, each image roughly signifies the features the model correlates with a particular handwritten digit. Brighter regions in these weight images highlight pixels significantly influencing the model's classification decision, embodying a kind of abstract, averaged representation of the class. Typically, these visualizations appear blurry or abstract as seen above, as they reflect the model's internal, high-dimensional representation rather than direct visual features. This abstraction can lead to patterns or shapes that are not immediately recognizable, sometimes containing noise or irregularities.

2.5 Question 2.5 [5 pts]

In this section of the homework, I calculated precision, recall, F1 score and F2 score for each class by using the best model found in 2.3.

Class	Precision	Recall	F1 Score	F2 Score
0	0.958	0.9776	0.9677	0.9736
1	0.9661	0.9797	0.9729	0.9770
2	0.9323	0.8944	0.9130	0.9017
3	0.8884	0.9218	0.9048	0.9149
4	0.9370	0.9246	0.9308	0.9271
5	0.9030	0.8666	0.8844	0.8736
6	0.9335	0.9520	0.9426	0.9482
7	0.9259	0.9241	0.9250	0.9245
8	0.8862	0.8799	0.8830	0.8811
9	0.9006	0.9068	0.9037	0.9056

Table 2: Precision, Recall, F1 Score, and F2 Score for Each Class

In this table, each row corresponds to a handwritten digit number (class), and the columns represent the precision, recall, F1 score, and F2 score for that class. The values are rounded to four decimal places for clarity.

Based on the calculated metrics and the visual insights obtained from the confusion matrix and the weight images, several observations can be made about the model's performance. The high precision and recall values for classes such as 0 and 1, which correspond to weight images of classes 0 and 1 in section 2.4, indicate that the model effectively identifies these classes with minimal confusion. This is further supported by the distinct patterns observed in their respective weight images, suggesting a strong representation of features that are unique to these classes.

In contrast, classes such as 3 and 8, which have relatively lower scores, might be experiencing higher misclassification rates. This could be a result of the weight images for these classes being more abstract or less distinct, implying that the model struggles to capture the defining features of these classes accurately. The confusion matrix corroborates this, showing a higher number of off-diagonal elements for these classes, which signifies misclassifications.