CS464 Introduction to Machine Learning

Fall 2024

Homework 2

01.12.2024

Elif Sorguç
22003782

1. Introduction

This document contains solutions to the PCA analysis and Logistic Regression tasks. Concepts of dimensionality reduction using PCA and classification using multinomial logistic regression are learned.

2. PCA Analysis

The images are resized to 64×64 pixels using bilinear interpolation and flattened into 4096×3 matrices. Since PIL reads images in the uint8 format, which may cause issues with calculations, the data type is converted to float32. The dataset consists of 3-channel RGB images, stacked into a 3D array X of size 10000×4096×3. Each color channel is sliced as Xi = X[:,:,i] (0: Red, 1: Green, 2: Blue), creating a 10000×4096 matrix for each channel.

Question 1.1: PVE and Minimum Components for 70%

Red Values (X_R):

	PC Number	Eigenvalue	PVE	Cumulative PVE
0	1	4600717.59	0.289	0.289
1	2	1490755.88	0.094	0.383
2	3	1079651.58	0.068	0.451
3	4	931817.15	0.059	0.510
4	5	861886.54	0.054	0.564
5	6	697277.54	0.044	0.608
6	7	459258.50	0.029	0.636
7	8	325894.55	0.020	0.657
8	9	267429.29	0.017	0.674
9	10	259337.43	0.016	0.690
10	11	198816.80	0.013	0.703

Total PVE of top 10 principal components for Red: 0.690
Total PVE of top 11 principal components for Red: 0.703

Green Values (X_G):

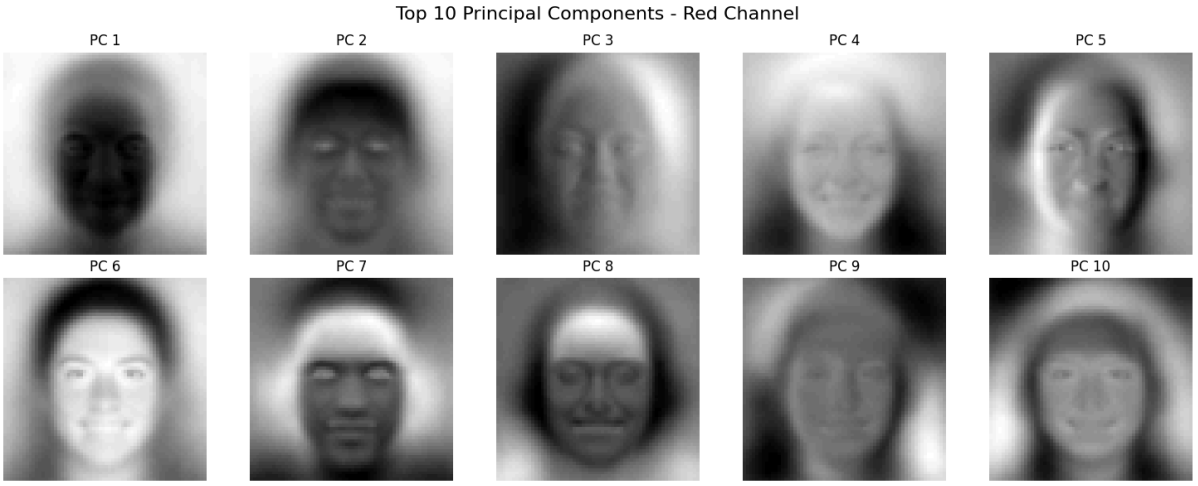
	PC Number	Eigenvalue	PVE	Cumulative PVE
0	1	4621729.32	0.320	0.320
1	2	1234628.46	0.086	0.406
2	3	1179543.76	0.082	0.488
3	4	836123.85	0.058	0.546
4	5	592944.48	0.041	0.587
5	6	570948.14	0.040	0.626
6	7	370480.42	0.026	0.652
7	8	267659.54	0.019	0.671
8	9	239956.93	0.017	0.687
9	10	231448.25	0.016	0.703
10	11	199401.98	0.014	0.717

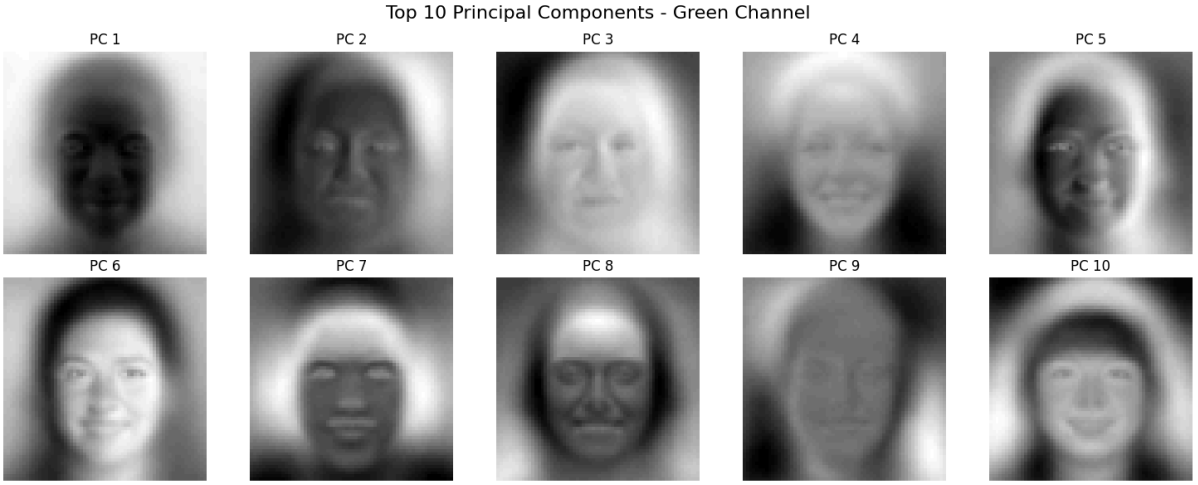
Total PVE of top 10 principal components for Green: 0.703
Total PVE of top 11 principal components for Green: 0.717

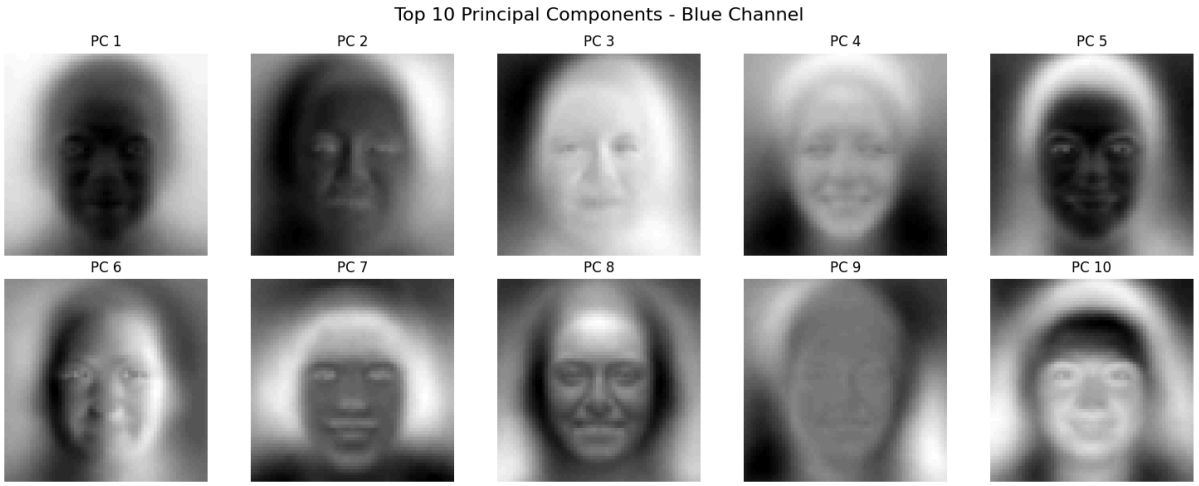
Blue Values (X_B):

	PC Number	Eigenvalue	PVE	Cumulative PVE
0	1	5119412.26	0.344	0.344
1	2	1319275.20	0.089	0.432
2	3	1263102.26	0.085	0.517
3	4	852713.11	0.057	0.574
4	5	567015.78	0.038	0.612
5	6	499813.03	0.034	0.646
6	7	370474.40	0.025	0.671
7	8	255939.48	0.017	0.688
8	9	242112.74	0.016	0.704
9	10	231403.20	0.016	0.720
10	11	200046.19	0.013	0.733

Total PVE of top 10 principal components for Blue: 0.720
Total PVE of top 11 principal components for Blue: 0.733



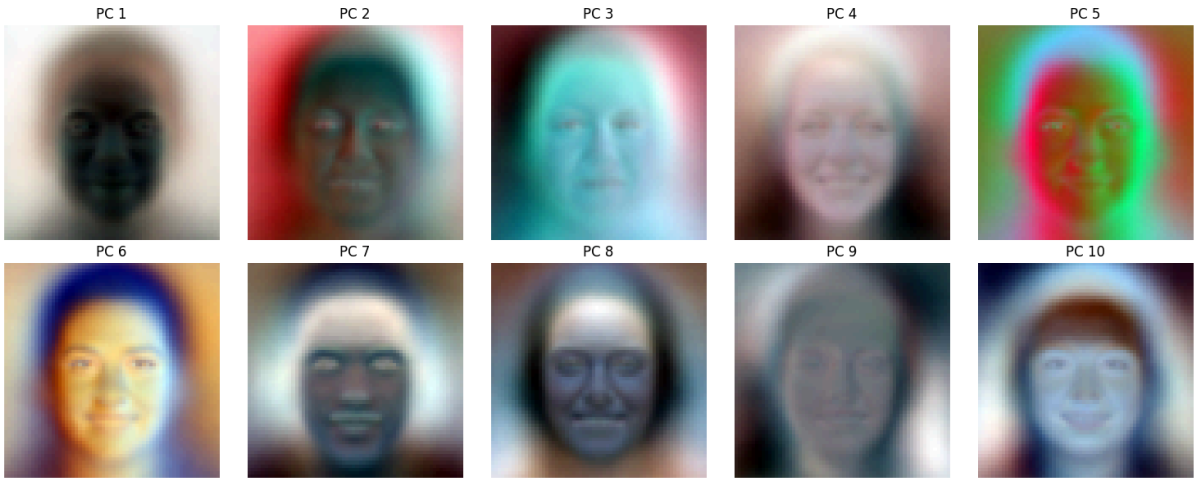




For the PCA analysis, the Red, Green, and Blue channels showed different levels of variance distribution. In the **Red channel**, the first 10 components explained 69.0% of the variance, requiring **11 components to reach the 70% threshold**. The **Green channel** was more efficient, with **70.3% cumulative PVE achieved in 10 components**. Similarly, the **Blue channel** reached **72.0% PVE with 10 components**, making it the most compact in terms of variance representation.

These results show that the Blue channel has simpler variance, while the Red and Green channels have more distributed features. PCA effectively reduced dimensionality while preserving key features, with each channel requiring slightly different numbers of components to meet the 70% threshold.

Question 1.2: Top 10 Principal Components

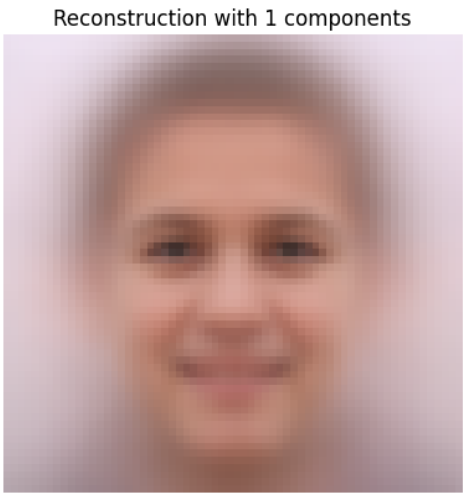


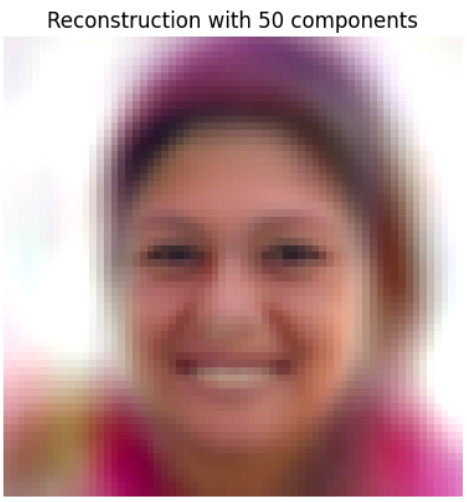
The top 10 principal components obtained for each of the red, green, and blue color channels are visualized by reshaping their eigenvectors into 64×64 matrices. After normalizing these matrices using min-max scaling, the corresponding RGB

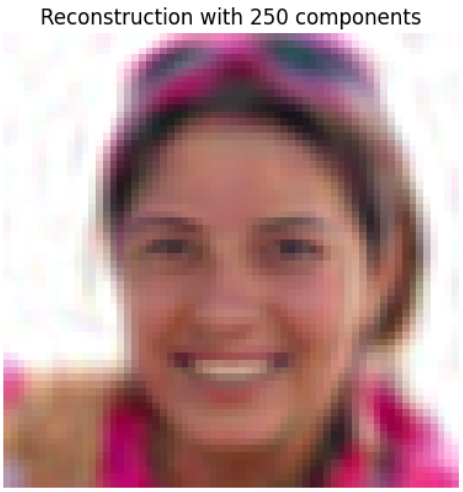
components are stacked to form **64×64×3 images representing the eigenvectors**. These visuals clearly demonstrate the features captured by each principal component across the color channels.

From the images, PC1 shows the most dominant and smooth intensity variations, capturing overall brightness and global color information. As it is moved to **higher principal components, like PC5 and PC10, finer details and localized features become apparent**, such as gradients, edges, and high-frequency noise. The RGB stacking highlights that each color channel contributes uniquely to the overall feature extraction, resulting in colorful and visually distinct representations of the eigenvectors. This analysis emphasizes how PCA prioritizes variance, with earlier components explaining global patterns and later ones capturing subtler details. These visuals are helpful for understanding how PCA decomposes the original data into distinct features.

Question 1.3







Reconstructing the first image in the dataset using the first k principal components $k \in \{1,50,250,500,1000,4096\}$. The reconstruction was performed by projecting the centered image data onto the eigenvectors corresponding to the top k components, then transforming the projected data back into the original space and adding the mean values for each channel as it is shown below.

```
def reconstruct_image(self, channel_data, eigenvalues, eigenvectors, mean, k):
    """Reconstruct the image using the top k principal components."""
    top_k_vectors = eigenvectors[:, :k]
    projection = np.dot(channel_data - mean, top_k_vectors)
    reconstruction = np.dot(projection, top_k_vectors.T) + mean
    return reconstruction
```

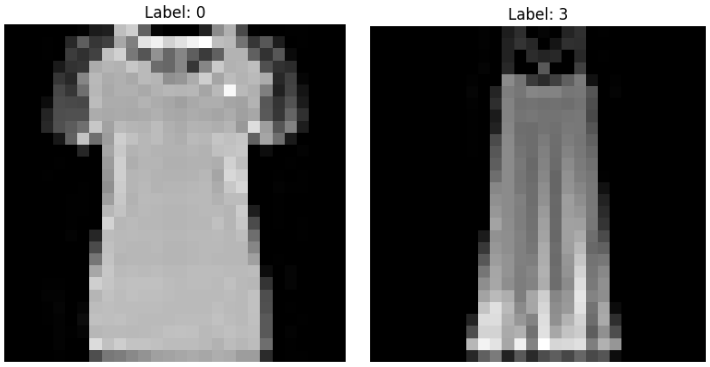
The reconstructions demonstrate the trade-off between the number of components and the image quality. With k=1, the reconstruction captures only the global intensity, resulting in a blurry, low-detail image. As k increases to 50, 250, and 500, more details, such as edges and textures, are progressively restored. By k=1000, the reconstructed image closely resembles the original, showing clear features and accurate color information. Finally, at k=4096, the reconstruction is nearly identical to the original image, confirming that PCA can effectively represent the entire dataset when all components are included.

This experiment may show PCA's efficiency in dimensionality reduction: a relatively small number of components (k=500) can achieve a visually accurate reconstruction, significantly reducing the computational complexity compared to using all components. The reconstruction quality at each k illustrates the contribution of higher-index principal

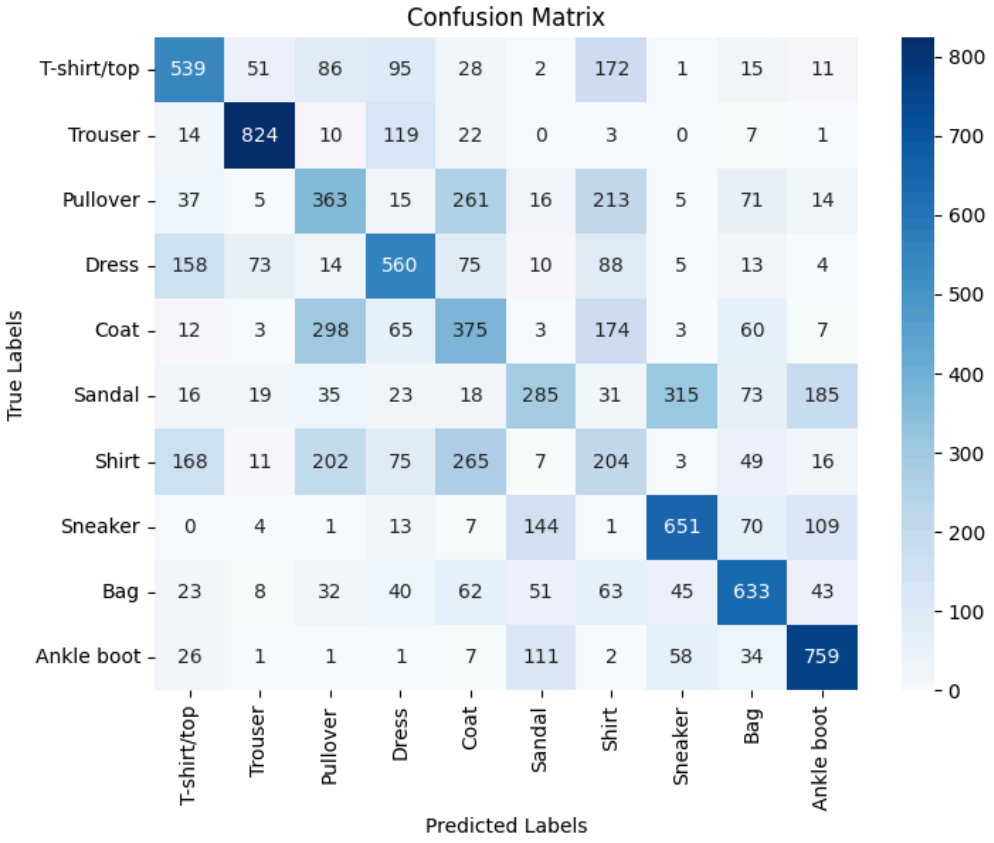
components in capturing finer details and noise, demonstrating PCA's ability to balance compression and data fidelity.

3. Logistic Regression

Evaluation of a **Multinomial Logistic Regression Classifier** to classify images from the FASHION MNIST dataset into 10 distinct fashion categories. The dataset consists of 60,000 training and 10,000 test images, with an additional validation set created by splitting 10,000 samples from the training set. The task involves training a default model with predefined hyperparameters, experimenting with various hyperparameter settings to optimize performance, and analyzing the results. Key evaluations include test accuracy, confusion matrices, precision, recall, F1 scores, and weight visualizations, providing insights into model performance and interpretability. Below classified examples of images 0 for T-shirt and 3 for dress.



Question 2.1: Test accuracy and confusion matrix




```
Epoch 100/100

Validation
Accuracy:
0.5598

Test
Accuracy:
0.5443
```

Question 2.2: Hyperparameter Tuning

Batch Size Comparison

Batch 1

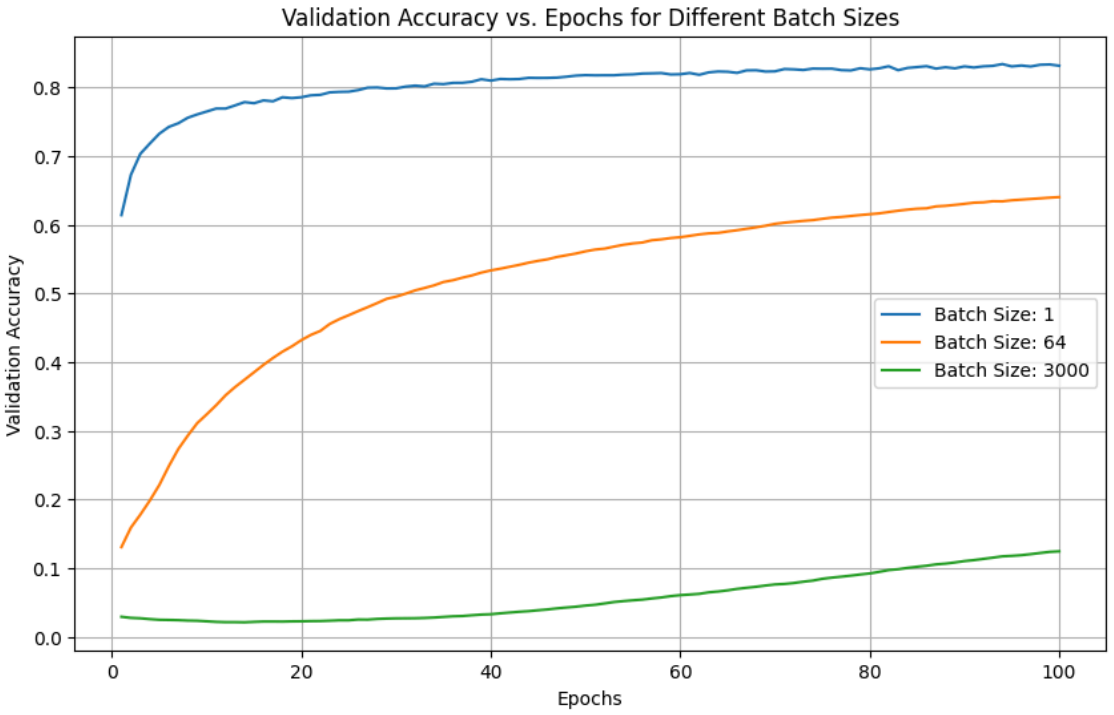
- Epoch 100/100, Validation Accuracy: 0.8497

Batch 64

- Epoch 100/100, Validation Accuracy: 0.6861

Batch 3000

- Epoch 100/100, Validation Accuracy: 0.1989



Weight Initialization Techniques Comparison

Zero Initialization

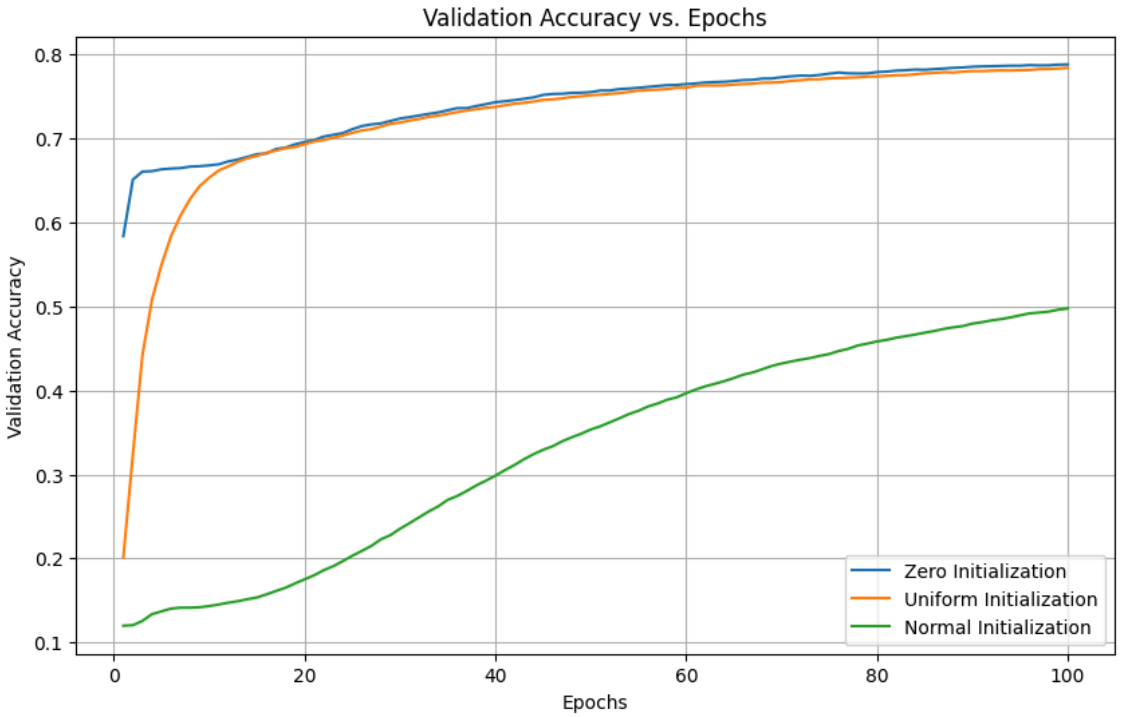
- Epoch 100/100, Zero Initialization, Validation Accuracy: 0.7883

Uniform Initialization

- Epoch 100/100, Uniform Initialization, Validation Accuracy: 0.7842

Normal Initialization

- Epoch 100/100, Normal Initialization, Validation Accuracy: 0.4977



Learning Rates Comparison

Learning Rate: 0.01

- Epoch 100/100, Validation Accuracy: 0.7696

Learning Rate: 0.001

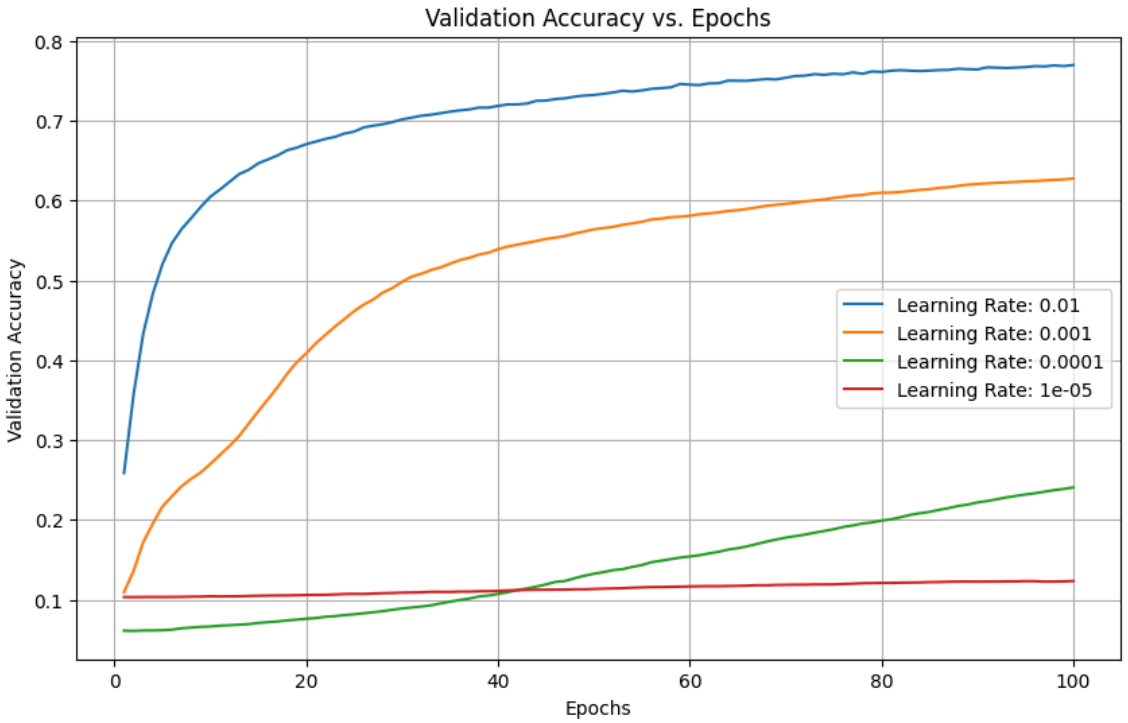
- Epoch 100/100, Validation Accuracy: 0.6276

Learning Rate: 0.0001

- Epoch 100/100, Validation Accuracy: 0.2407

Learning Rate: 1e-05

- Epoch 100/100, Validation Accuracy: 0.1236



Regularization Coefficients Comparison

Regularization Coefficient: 0.01

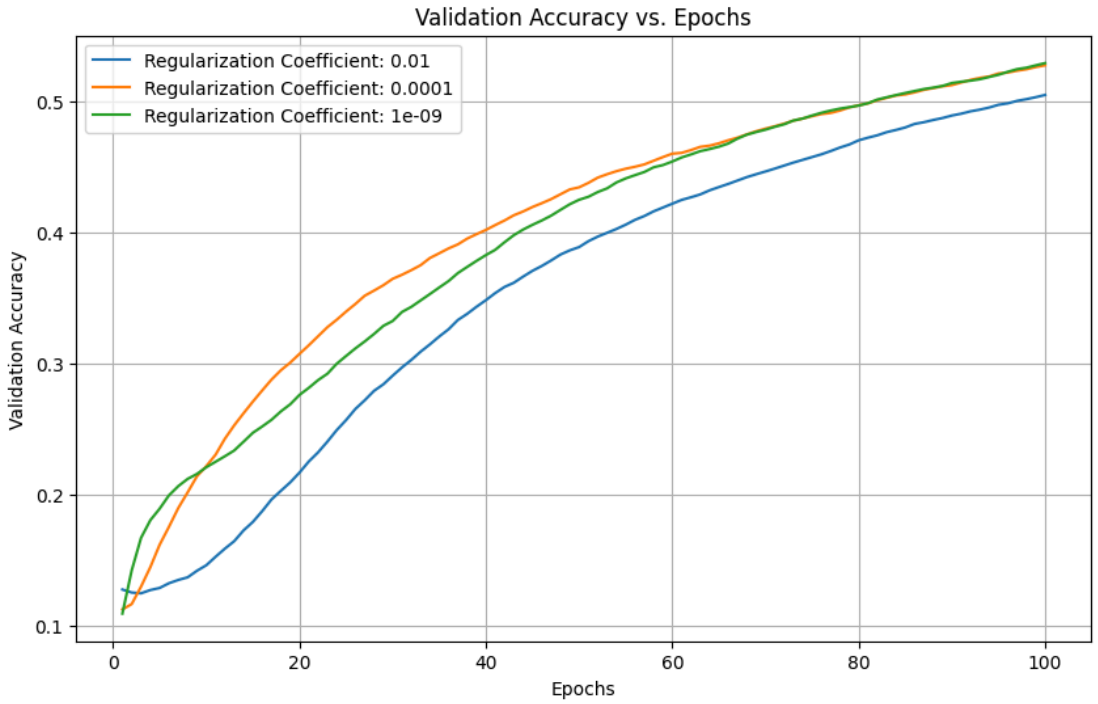
- Epoch 100/100, Validation Accuracy: 0.5046

Regularization Coefficient: 0.0001

- Epoch 100/100, Validation Accuracy: 0.5272

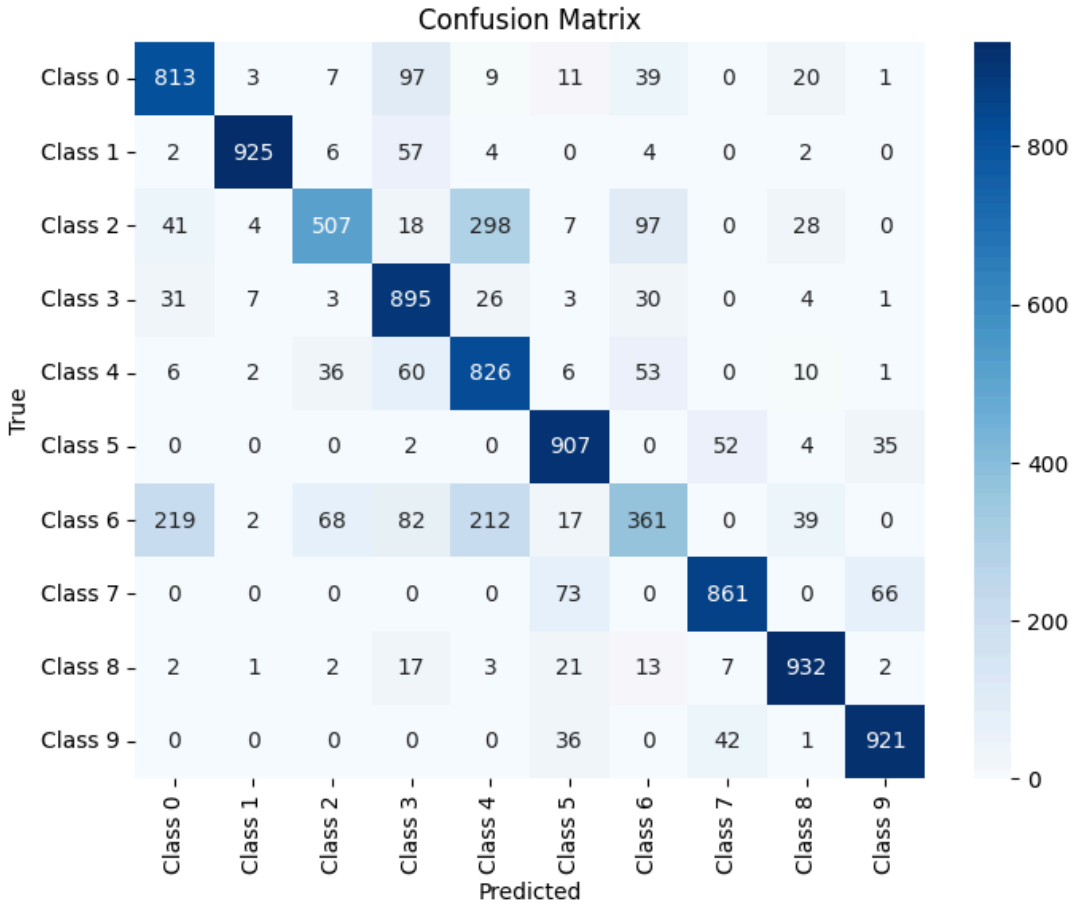
Regularization Coefficient: 1e-09

- Epoch 100/100, Validation Accuracy: 0.5289



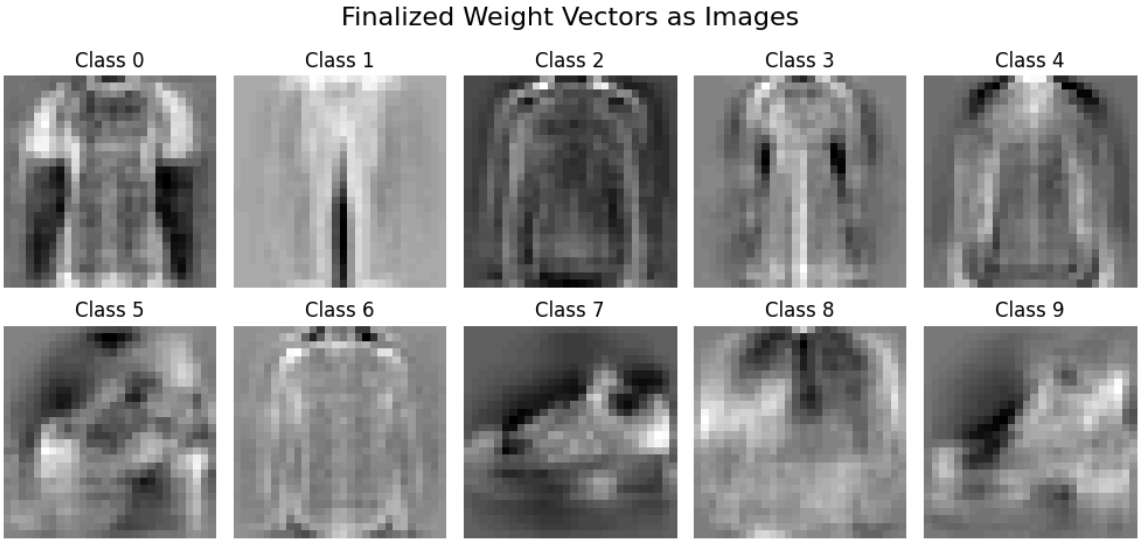
Question 2.3: Best Model

Optimal Model Test Accuracy: 0.7948



The optimal logistic regression model for MNIST was configured with the following hyperparameters: learning rate (lr) = 0.01, regularization coefficient (reg) = 0.01, batch size = 1, epochs = 100, and zero weight initialization. This configuration achieved the best trade-off between convergence, generalization, and accuracy during experiments. The test accuracy was **79.48**%, and the confusion matrix demonstrated strong classification performance across all classes. These choices were validated by systematic experimentation and careful tuning of hyperparameters.

Question 2.4: Weight Visualization



Each weight image represents the learned features for a specific digit (0-9). The weights are visualized as 28x28 grayscale images, corresponding to the MNIST input dimensions. The images are blurry but show distinguishable patterns associated with each class. For example, the weight image for the digit '0' may show a circular shape, while '1' has a vertical line. These patterns represent the features the model has learned to identify the respective classes.

Question 2.5: Metrics based on performance

```
Precision, Recall, F1 Score, and F2 Score for Each Class:
Class 0: Precision=0.9419, Recall=0.3890, F1 Score=0.5506, F2 Score=0.4407
Class 1: Precision=0.9669, Recall=0.9340, F1 Score=0.9502, F2 Score=0.9404
Class 2: Precision=0.7670, Recall=0.5760, F1 Score=0.6579, F2 Score=0.6062
Class 3: Precision=0.6057, Recall=0.9370, F1 Score=0.7358, F2 Score=0.8446
Class 4: Precision=0.7054, Recall=0.6490, F1 Score=0.6760, F2 Score=0.6596
Class 5: Precision=0.8957, Recall=0.7900, F1 Score=0.8395, F2 Score=0.8091
Class 6: Precision=0.4687, Recall=0.6210, F1 Score=0.5342, F2 Score=0.5831
Class 7: Precision=0.5646, Recall=0.9870, F1 Score=0.7183, F2 Score=0.8586
Class 8: Precision=0.8881, Recall=0.9050, F1 Score=0.8965, F2 Score=0.9016
Class 9: Precision=0.9837, Recall=0.4220, F1 Score=0.5906, F2 Score=0.4764
```

Class-to-Clothing Mapping:

Class 0: T-shirt/top
Class 1: Trouser
Class 2: Pullover
Class 3: Dress
Class 4: Coat
Class 5: Sandal
Class 6: Shirt
Class 7: Sneaker
Class 8: Bag
Class 9: Ankle boot

The optimal logistic regression model for the MNIST dataset was configured with the following hyperparameters: learning rate = 0.01, regularization coefficient = 0.01, batch size = 1, epochs = 100, and zero weight initialization. This setup achieved a test accuracy of **79.48%**. The confusion matrix demonstrates relatively strong classification performance, with especially high precision and recall for classes like **Class 1 (Trouser)**, **Class 5 (Sandal)**, **Class 7 (Sneaker)**, and **Class 9 (Ankle Boot)**, reflecting the model's ability to distinguish these classes effectively.

The weight visualizations show meaningful patterns learned by the model. For instance, the weights for **Class 0 (T-shirt/top)** display an outline of a shirt, while **Class 1 (Trouser)** gives a vertical shape corresponding to pants. These visuals indicate that the model has successfully identified class-specific features. However, some classes, such as **Class 6 (Shirt)**, came out with more blurred weight patterns and relatively lower precision and recall, suggesting difficulties in differentiation due to overlapping features with other categories like **Class 0 (T-shirt/top)** and **Class 2 (Pullover)**.

The strong performance for classes like **Trouser** and **Ankle Boot** is due to their distinct, non-overlapping features, while the confusion observed for **Shirt**, **Pullover**, and **Coat** reflects the inherent similarity in their visual patterns. The optimal model configuration demonstrates that careful hyperparameter tuning is critical for achieving a balance between generalization and class-specific accuracy, as evidenced by the clear patterns in weight images and robust overall performance metrics.