Homework 2 Report

1 PCA Analysis

Question 1.1

Channels	**PVEs of Top 10 PCs**	**Sum of PVEs of Top 10 PCs**	**Minimum Number of PCs Required to Obtain 70% PVE**
Red	0.28929081 0.09373798 0.06788795 0.05859219 0.05419499 0.04384446 0.02887794 0.02049209 0.01681582 0.01630701	0.690041248204085	11
Green	0.32041101 0.08559319 0.08177433 0.05796603 0.04110711 0.03958217 0.02568433 0.01855605 0.01663551 0.01604563	0.7033553538090799	10
Blue	0.34357938 0.08854058 0.08477065 0.05722818 0.03805416 0.03354398 0.02486367 0.01717688 0.01624892 0.01553017	0.7195365740418399	9

Table 1: PC Analysis of RGB Channels of Image Data

As shown in Table 1, each of the color channels (Red, Green, Blue) is able to explain a significant amount of variance with a few components, especially the first. The first principal component has the highest PVE in Blue channel (34.36%), Green channel (32.04%) and Red channel (28.93%), which means the Blue channel has the most dominant direction of variance. However, the summation of the PVEs from the top 10 components is effectively 100% across all channels, implying that these components explain close to all the variability in the data. The minimum number of components needed to reach at least 70% PVE varies slightly. The variance of the Blue channel is more concentrated in fewer components than the other two, i.e the Blue channel has 9 components, the Green has 10, and the Red has 11. This means that the distribution of information across the color channels is slightly different, such that the Blue channel is summarized more effectively with fewer principal components.

Question 1.2

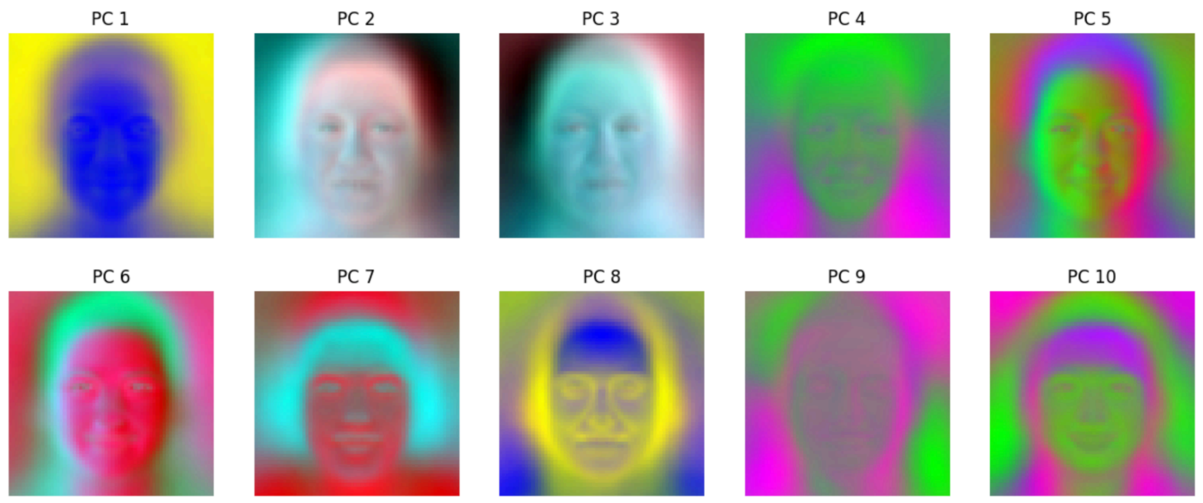

Figure 1: Corresponding Top 10 PCs for 3 Color Channels Overlapped

The principal components of the dataset which have the highest variance,for each color channel, are represented as the RGB images in Figure 1. Since the principal components with the highest variance correspond to the first few images, we often see the most noticeable patterns in the first few images, such as lighting and major facial features. When we move to higher principal components, the images begin to capture more subtle details or noise, which corresponds to less important but still present variations in the data. This visualization reveals what is most variable in the fake faces, for example, color gradient changes, facial shapes, and textures.

Question 1.3

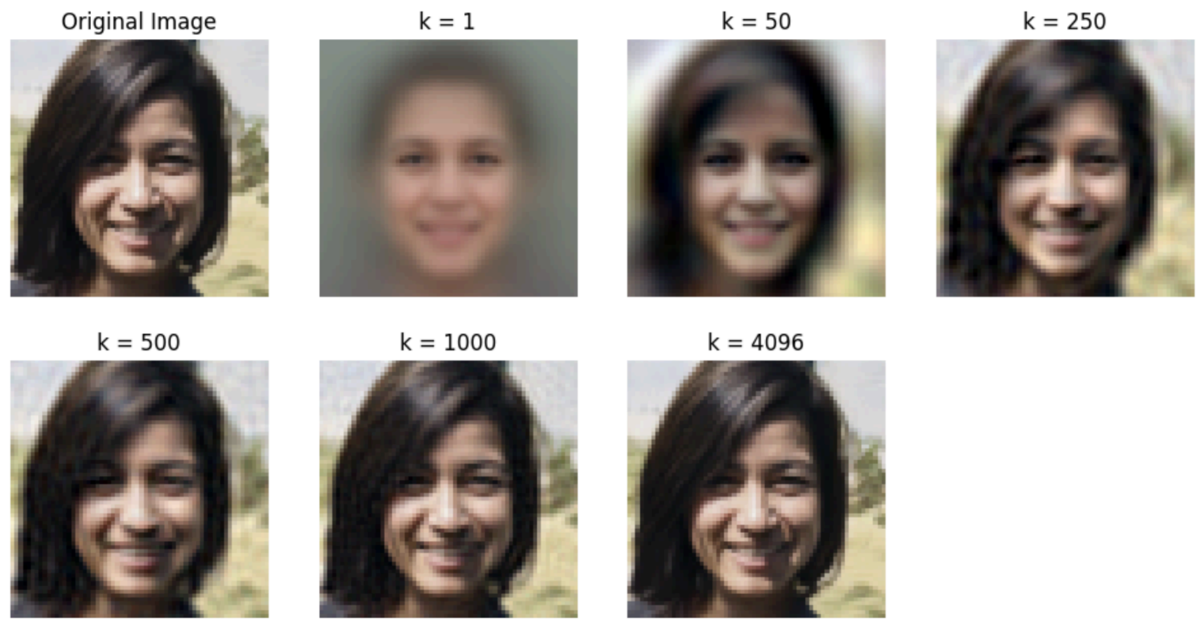

Figure 2: Reconstructed Image According to Number of PCs Used (k)

A face image is constructed by applying PCA separately on each color channel, capturing key features with reduced dimensionality. It first centers the data by subtracting the mean, then computes the covariance matrix to understand the variance between pixels. Through eigen decomposition, the principal components (eigenvectors) are extracted and sorted based on their corresponding eigenvalues. The top k components are selected, and the centered image is projected onto this reduced space to obtain the projection coefficients. Using these coefficients, the image is reconstructed by mapping back to the original feature space and adding the mean. The reconstructed channels are reshaped and combined into an RGB image, with pixel values clipped for valid display. This process highlights PCA's ability to retain critical information while reducing noise, resulting in a visually similar, compressed approximation of the original image.

2 Logistic Regression

Question 2.1

Default Model Test Accuracy: 0.8131
Confusion Matrix (rows represent ground truths, columns represent predictions):

```
[[782   6  21  41  10   0 121   0  19   0]
 [  7 945   5  33   4   0   4   0   2   0]
 [ 25   4 668  12 146   1 126   0  18   0]
 [ 43  16  18 798  48   0  65   0  12   0]
 [  2   4 110  36 716   2 116   0  14   0]
 [  0   1   0   1   3 868   1  51  19  56]
 [137   6 100  37 108   1 582   0  29   0]
 [  0   0   1   0   0  35   0 930   2  32]
 [ 10   1  10   8   7   7  28   7 919   3]
 [  0   0   0   0   0  24   1  51   1 923]]
```

Question 2.2

Experiment with Batch Size = 1
Test Accuracy = 0.8235
Confusion Matrix:

```
[[809   8  14  49  14   0  87   0  17   2]
 [  5 954   3  26   4   0   7   0   1   0]
 [ 34   3 727  16 151   2  57   0  10   0]
 [ 44  20  18 820  48   0  39   0  11   0]
 [  3   1 102  33 779   1  71   0  10   0]
 [  0   0   1   3   0 888   0  57  10  41]
 [158   4 131  46 142   1 487   0  30   1]
 [  0   0   0   0   0  44   0 915   0  41]
 [  8   2  11   9   5   5  22   5 931   2]
 [  0   0   0   0   0  26   3  44   2 925]]
```

Experiment with Batch Size = 64
Test Accuracy = 0.8065
Confusion Matrix:

```
[[759   6  20  51   9   0 139   1  15   0]
 [  3 951   3  30   8   0   5   0   0   0]
 [ 25   5 691   9 129   1 124   0  16   0]
 [ 44  30  22 795  45   0  48   0  16   0]
 [  1   1 131  33 659   1 155   0  19   0]
 [  2   0   1   0   1 858   3  63  17  55]
 [132   2 114  35  98   0 582   0  37   0]
 [  0   0   0   0   0  41   0 910   1  48]
 [  8   3  12  12   6   7  22   5 922   3]
 [  0   0   0   0   0  17   0  43   2 938]]
```

Experiment with Batch Size = 3000
Test Accuracy = 0.8202
Confusion Matrix:

```
[[827   7   8  71  13   0  56   0  18   0]
 [  2 962   1  28   3   0   1   0   3   0]
 [ 40  11 609  33 223   2  62   0  20   0]
 [ 35  22   5 863  39   1  28   1   6   0]
 [  2   6  47  59 805   1  70   0  10   0]
 [  1   2   0   4   1 860   0  59  22  51]
 [171   6  90  60 141   2 496   0  33   1]
 [  0   0   0   1   0  44   0 916   2  37]
 [  8   2   4  11   6   5   8   8 948   0]
 [  0   1   0   0   0  25   2  56   0 916]]
```

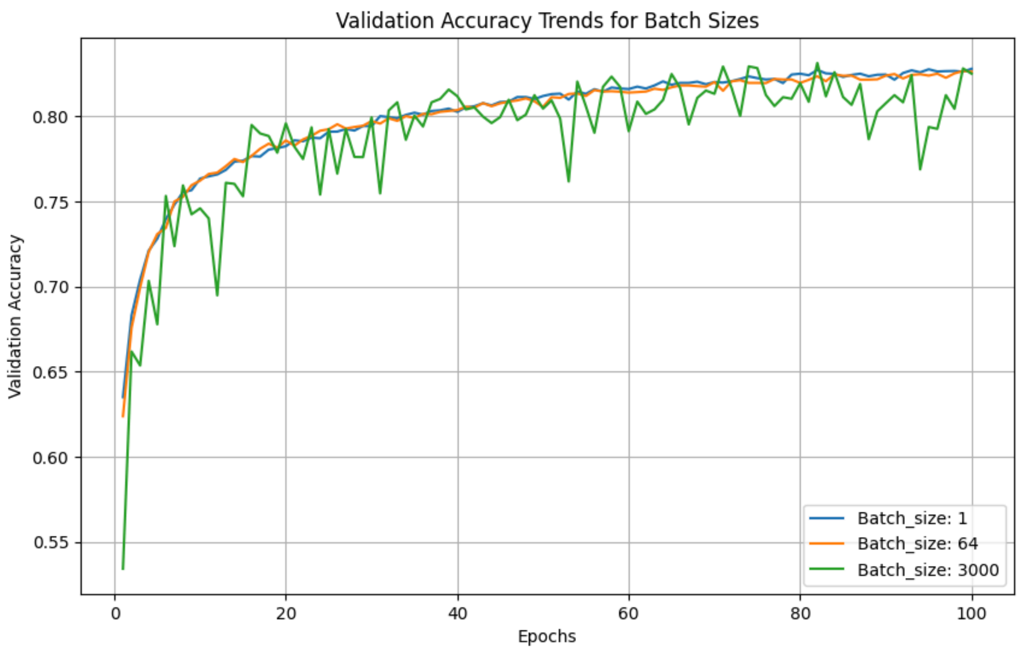

Figure 3: Batch Size Test

As batch size increases, test accuracy decreases. This can be due to reduced gradient variability, poor generalization, and limited regularization effects.

Experiment with Weight Initialization Technique = Zero Initialization
Test Accuracy = 0.8382
Confusion Matrix:
```
[[753   5  17  66   7   0 140   0  12   0]
 [  3 956   4  28   4   0   3   0   2   0]
 [ 15   6 761  12 107   1  86   0  12   0]
 [ 12  19  12 876  32   1  41   0   7   0]
 [  0   1 126  44 708   1 110   0  10   0]
 [  1   1   0   0   0 902   0  48   6  42]
 [101   1 129  51  86   0 604   0  28   0]
 [  0   0   0   0   0  41   0 926   0  33]
 [  4   1   5  10   3   4  18   5 949   1]
 [  0   0   0   0   0  14   0  37   2 947]]
```

Experiment with Weight Initialization Technique = Uniform Distribution
Test Accuracy = 0.8346
Confusion Matrix:
```
[[765   4  13  46   7   0 154   0  11   0]
 [  3 955   4  26   4   0   6   0   2   0]
 [ 13   6 701   7 107   1 154   0  11   0]
 [ 19  19  12 850  34   1  59   0   6   0]
 [  0   1  97  35 674   1 183   0   9   0]
 [  1   0   0   1   0 890   0  55   5  48]
 [102   1  95  31  69   0 681   0  21   0]
 [  0   0   0   0   0  29   0 936   0  35]
 [  4   1   5   9   2   3  30   6 940   0]
 [  0   0   0   0   0   7   1  38   0 954]]
```

Experiment with Weight Initialization Technique = Normal Distribution
Test Accuracy = 0.8073
Confusion Matrix:

```
[[758   6  29  53   7   0 127   0  20   0]
 [  5 952   3  31   1   0   7   0   1   0]
 [ 25   5 786  14  67   0  89   0  13   1]
 [ 36  27  22 823  31   1  52   0   8   0]
 [  1   2 229  53 563   1 143   0   8   0]
 [  0   3   0   1   1 861   0  59  16  59]
 [138   6 154  49  61   2 564   0  26   0]
 [  0   1   0   0   0  35   0 924   0  40]
 [ 10   2  14  12   5   9  24   6 918   0]
 [  1   1   0   0   0  25   1  47   1 924]]
```

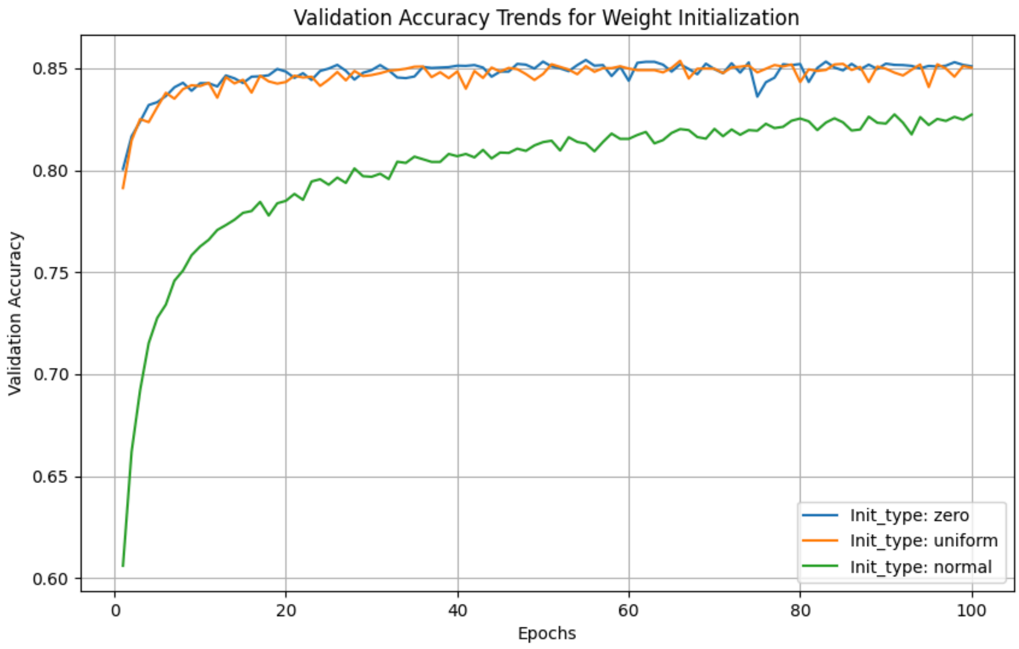

Figure 4: Weight Initialization Test

Results indicate that zero initialization is best (76.73%), followed by uniform (76.17%) and normal (49.95%) distribution respectively. Here, zero initialization may be working well since logistic regression does not suffer from symmetry issues.

Experiment with Learning Rate = 0.01
Test Accuracy = 0.8178
Confusion Matrix:

```
[[794   4  16  22  21   0 125   1  17   0]
 [  2 957   2  19  12   0   5   0   3   0]
 [ 22   9 610   7 225   1 115   0  11   0]
 [ 53  29  20 751  95   1  43   0   8   0]
 [  1   3  45  18 851   0  77   0   5   0]
 [  0   0   0   0   0 931   0  26  10  33]
 [144   4  80  21 147   0 582   0  22   0]
 [  0   0   0   0   0 101   0 835   0  64]
 [  5   1   4   8  15  11  26   4 924   2]
 [  0   0   0   0   0  36   1  20   0 943]]
```

Experiment with Learning Rate = 10^{-3}
Test Accuracy = 0.8253
Confusion Matrix:

```
[[829   9  17  45  13   0  70   0  15   2]
 [  5 960   2  26   3   0   3   0   1   0]
 [ 33   5 703  10 171   0  57   0  21   0]
 [ 44  21  19 816  51   0  37   0  12   0]
 [  3   1  94  39 792   1  52   0  18   0]
 [  0   0   0   0   1 885   0  52  13  49]
 [170   3 124  38 157   1 473   0  34   0]
 [  0   0   0   0   0  38   0 924   0  38]
 [  9   1   8  11   3   8  14   6 939   1]
 [  0   1   0   0   0  15   2  49   1 932]]
```

Experiment with Learning Rate = 10^{-4}
Test Accuracy = 0.7749
Confusion Matrix:

```
[[757  10  27  55  10   0 115   2  24   0]
 [ 11 935   5  35   5   0   5   0   3   1]
 [ 28   7 630  16 177   1 128   0  13   0]
 [ 51  26  27 791  41   0  50   1  11   2]
 [  7   5 115  42 703   2 115   0  11   0]
 [  2   4   4   4   0 805   4  86  26  65]
 [163   6 128  38 159   1 469   1  35   0]
 [  0   0   0   1   0  62   0 864   4  69]
 [ 13   2   5  14  11  15  27   9 903   1]
 [  0   1   0   0   0  41   1  62   3 892]]
```

Experiment with Learning Rate = 10^{-5}
Test Accuracy = 0.6629
Confusion Matrix:

```
[[687  23  37  76  23   3 122   2  25   2]
 [  7 898  20  49  19   1   2   0   1   3]
 [ 44   4 481  19 235   5 159   0  46   7]
 [ 60  43  27 720  60   1  72   3  14   0]
 [  9  10 201  45 554   0 160   0  16   5]
 [  2   3  11   5   4 535  14 185  74 167]
 [153  14 160  43 219   4 355   0  43   9]
 [  0   1   0   1   0 114   0 765   6 113]
 [ 11   1  30  21  19  42  39  10 815  12]
 [  4   0   3   2   5  66   4  82  15 819]]
```

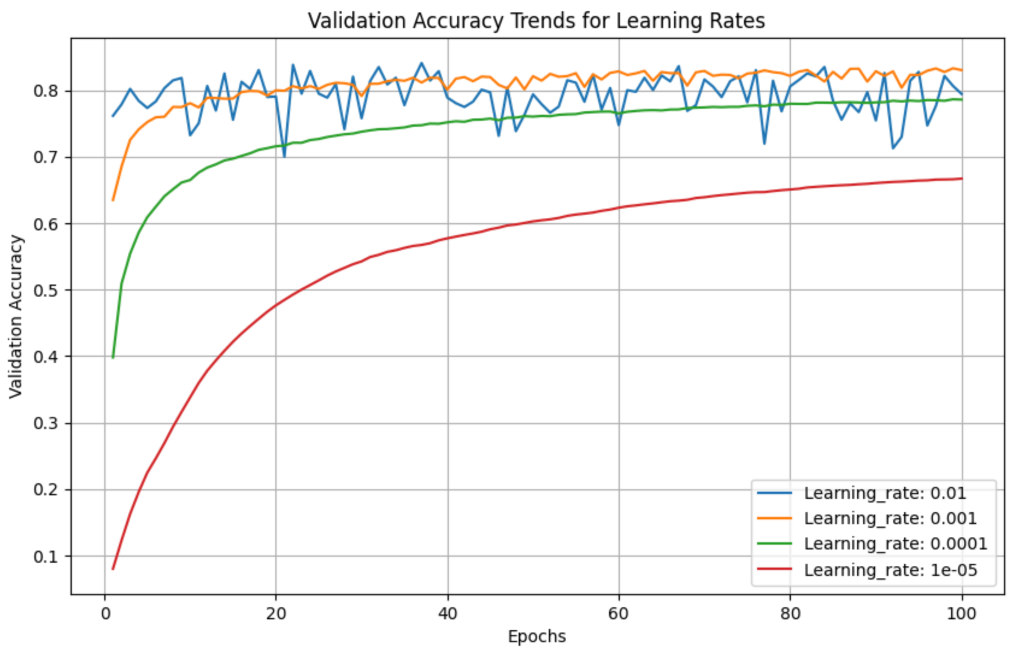

Figure 5: Learning Rate Test

A learning rate of 10^{-3}achieved the highest test accuracy of 82.53%, demonstrating optimal performance. Increasing the rate to 0.01 slightly reduced accuracy to 81.78%, likely due to overshooting. Lower rates of 10^{-4} and 10^{-5} caused significant accuracy declines to 77.49% and 66.29%, respectively, indicating underfitting. Therefore, 1e-3 is the most effective learning rate among those tested.

Experiment with Regularization Coefficient = 10^{-2}
Test Accuracy = 0.8166
Confusion Matrix:

```
[[789   9  17  62  12   0  98   0  12   1]
 [  4 953   3  29   3   0   5   0   2   1]
 [ 24   9 702  14 148   1  90   0  12   0]
 [ 46  26  18 818  36   1  43   0  11   1]
 [  3   4 102  45 723   0 110   0  13   0]
 [  0   1   1   1   0 884   0  48  16  49]
 [153   4 128  44 104   1 540   0  25   1]
 [  0   0   0   0   0  40   0 900   0  60]
 [  9   1  12  13   9   8  25   5 917   1]
 [  0   0   0   0   0  13   0  46   1 940]]
```

Experiment with Regularization Coefficient = 10^{-4}
Test Accuracy = 0.8184
Confusion Matrix:

```
[[780   6  23  59  19   0 101   0  12   0]
 [  4 950   3  29   5   0   6   0   3   0]
 [ 28   6 738  15 143   1  60   1   8   0]
 [ 38  24  22 829  41   1  37   0   8   0]
 [  2   2 133  41 740   2  72   0   8   0]
 [  3   3   0   0   1 880   0  54  13  46]
 [145   2 147  45 135   1 496   0  29   0]
 [  0   0   0   0   0  50   0 911   0  39]
 [ 10   2  10   8   9   9  25   5 920   2]
 [  0   0   0   0   0  16   0  43   1 940]]
```

<u>Experiment with Regularization Coefficient =</u> 10^{-9}
Test Accuracy = 0.8160
Confusion Matrix:

```
[[781   7  18  37   6   0 136   0  15   0]
 [  7 955   7  20   4   0   4   0   2   1]
 [ 25   5 719   9 138   1  91   0  12   0]
 [ 50  27  23 794  42   1  51   1  10   1]
 [  3   7 126  30 705   2 121   0   5   1]
 [  0   1   0   0   2 870   2  56  15  54]
 [140   4 113  26 119   1 566   0  28   3]
 [  0   0   0   0   0  41   0 919   1  39]
 [  9   1  17   9   6   7  24   6 920   1]
 [  0   0   0   0   0  21   0  45   3 931]]
```

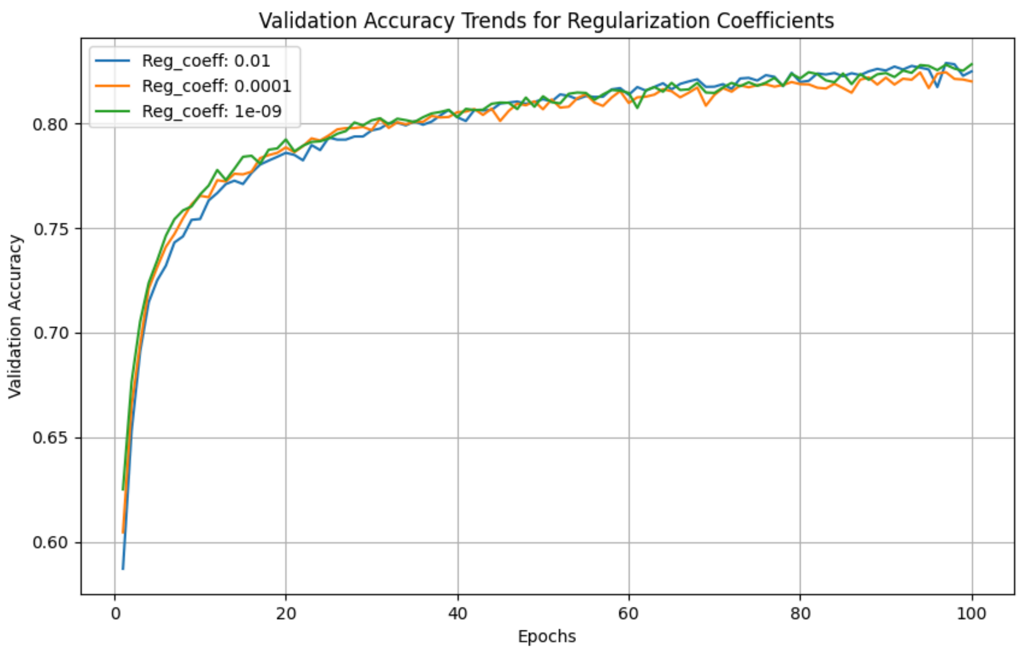

Figure 6: Regularization Coefficient Test

Higher regularization achieves the best accuracy maybe by preventing overfitting. Proper regularization is expected to balance generalization and stability.

Question 2.3

Best Model’s Hyperparameters:

Batch Size: 1
Weight Initialization Technique: Zero Initialization
Learning Rate: 0.001
Regularization Coefficient: 0.0001

Best Model’s Test Accuracy: 0.8383
Confusion Matrix:

```
[[794   5  15  54   5   0 115   0  12   0]
 [  3 955   5  29   3   0   3   0   2   0]
 [ 21   6 775  12  85   1  90   0  10   0]
 [ 18  16  14 873  32   1  39   0   7   0]
 [  0   1 157  45 659   1 128   0   9   0]
 [  1   1   0   0   0 895   0  53   5  45]
 [128   1 129  46  72   0 603   0  21   0]
 [  0   0   0   0   0  32   0 928   0  40]
 [  4   1   5  10   2   3  26   5 943   1]
 [  0   0   0   0   0   8   1  33   0 958]]
```

Question 2.4

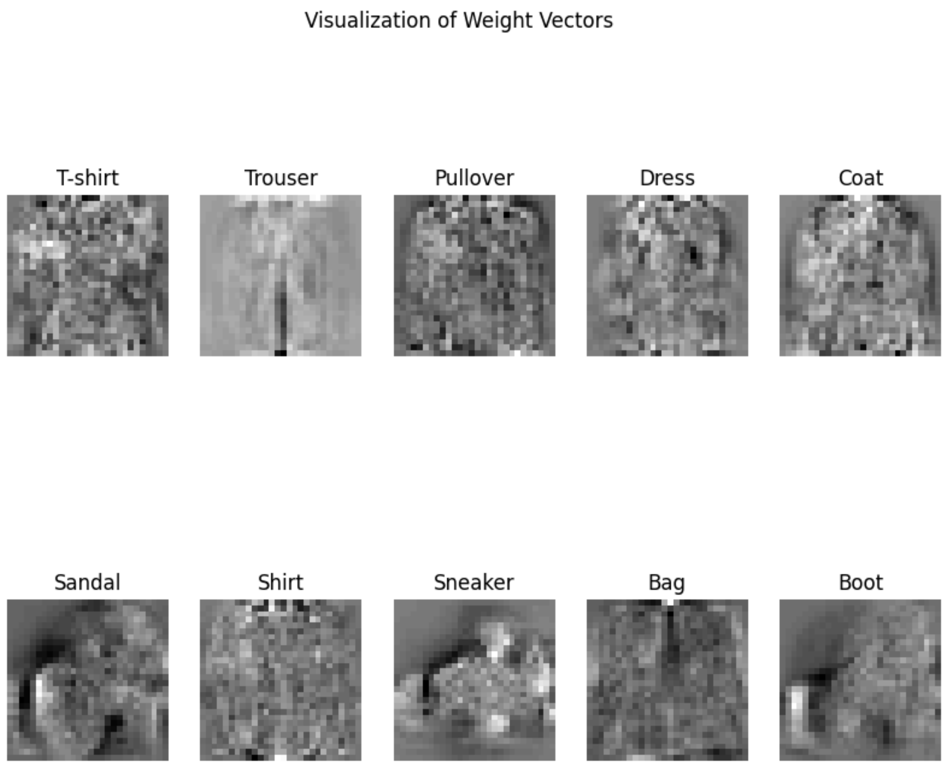

Figure 7: Visualization of Finalized Weight Vectors

The weight visualizations are very useful in that they help to bring focus on the features learned by the logistic regression model to classify between different classes. For instance, weights related to the “Trouser” class are seen to give vertical lines, probably representing the extended nature of trouser cuts and straight edges. Likewise, weights for “Sandal” and “Sneaker” have shapes that are similar to the form of a shoe including curved line and rough surface area, which are decided by the structure of the shoes. The weights of the “Bag” class emphasise rectangular patterns, which probably corresponds to the general, smooth shapes of bags. On the other hand, the “Boot” class weights emphasize more on the heavy parts and darker area, which might mean the higher and more prominent part of boots. For the “Dress” and “Coat” classes, the model provides centralized and smoother feature maps, reflecting the continuity and symmetry of these types of garments. Nevertheless, weights corresponding to more ambiguous classes, namely ‘Shirt’ and ‘Pullover’ are less interpretable, indicating that the model fails to distinguish these classes because the objects of these classes have similar shapes and textures. This means that there can be shared similarity in these classes in the input space and less distinguishable weight representations. In general, these visualizations suggest that the model learns class-specific structures, although some of these plots also demonstrate the model’s inability to distinguish between similar objects, which also might mean that better features or more preprocessing is necessary.

Question 2.5

Class	Precision	Recall	F1	F2
T-shirt	0.8194	0.7940	0.8065	0.7990
Trouser	0.9686	0.9550	0.9617	0.9577
Pullover	0.7045	0.7750	0.7381	0.7598
Dress	0.8167	0.8730	0.8439	0.8611
Coat	0.7681	0.6590	0.7094	0.6783
Sandal	0.9511	0.8950	0.9222	0.9057
Shirt	0.6000	0.6030	0.6015	0.6024
Sneaker	0.9107	0.9280	0.9193	0.9245
Bag	0.9346	0.9430	0.9388	0.9413
Boot	0.9176	0.9580	0.9374	0.9496

Table 2: Performance Metrics of Each Class

The findings also show that the model is accurate in classifying visually dissimilar classes like Trouser, Sandal, and Bag, as measured by precision, recall, and F1. For example, the Trouser class is well-identified and has vertical patterns detected in weight visualization; the F1 score of this model is 0.9617. Likewise, Sandal and Bag have clear patterns of weight visualization, which has boosted their performances with F1 scores of 0.9222 and 0.9388 and very little confusion with other classes. Similarly, other footwear classes such as Sneaker and Boot also show good performance with precision values above 0.91 and very low interclass confusion, which is evident from the weight patterns.

However, the model performs badly where there are overlapping and complicated classes like Shirt, Pullover, Coat, etc., where the weight maps are much noisy and not easily distinguishable. For instance, the Shirt class has the smallest values of both precision and recall which are about 0.6, meaning there are significant misclassifications, most probably with Pullover and Coat classes as can be inferred from the confusion matrix. The Coat class also shows lower performance, which has the F1 Score of 0.7094, which implies that the model fails to identify specific characteristics of the Coat class because of its proximity to other outerwear categories. T-shirt, while slightly less frequent, also suffers from regular misclassification between Pullover and Shirt which reiterates the problem of poor feature differentiation in these examples.

In general, the results suggest that the logistic regression model is successful in training features for different and clearly separable classes, but it struggles with similar or overlapping classes. These results indicate that the model is effective in recognizing general trends, although they can be improved by feature engineering, using more complex structures for the model, such as deep neural networks, or by expanding the data set. This is particularly important for classes such as Shirt, Pullover, and Coat, where there is a need to enhance the feature representation in order to enhance the classification.