CS 391L Machine Learning **Nidhi Kadkol**
Assignment 1 **UT EID: nk9368**
February 2019 **Email id: nidhik@cs.utexas.edu**

1 Introduction

The aim of this assignment is to understand decomposition of vectors into their eigen spaces, and how this is a method of extracting useful features for classification in machine learning. In this project, we are given images which have handwritten digits from 0 to 9 and we have to identify which digit is written from the picture. For this, we use k-nearest neighbours classification. But instead of directly applying it on the images pixel by pixel, we transform the images into their eigen space and classify them there. Thus, we project the data from a higher dimension into a lower dimension to extract only the important information for accurate classification of the data. All the experiments have been conducted in MATLAB.

2 Method

2.1 Computing Eigenvectors

Given a square matrix A, an eigenvector of A is a vector x such that

$$Ax = \lambda x$$

where λ is a scalar known as the eigenvalue corresponding to x. An n x n matrix can have upto n linearly independent eigenvectors. We can use these n linearly independent vectors as a basis for a new coordinate system to represent other vectors. Essentially, we can decompose any vector into a linear combination of the eigenvectors, and this represents a projection of a vector from the original coordinate system into the new coordinate system having the eigenvectors as basis vectors. If A is a symmetric matrix, it has the property that all its eigenvectors are perpendicular to each other. Then we can represent a vector in terms of orthogonal linearly independent basis vectors. This is the idea behind Principal Component Analysis, where we convert a set of observations having correlated features into a smaller set of uncorrelated observations called principal components.

In order to carry out our transformation, we use the covariance matrix of the training data. The resulting eigenvectors are ordered by their corresponding eigenvalues in descending order. The first eigenvector accounts for maximum possible variance in the data, the second eigenvector accounts for the second highest possible variance, and so on. Thus the first few eigenvectors capture most of the information and variability of the data.

To carry out eigen decomposition, we first reshape each 28x28x1 image into a 784x1 column vector. We then subtract the mean column vector among all the image samples from each of these images. Let x be the dimension of each of the training images (which is 784) and k be the number of training samples. If $x \geq k$, we calculate the covariance matrix of the matrix of column vectors after subtracting the mean by multiplying it with its transpose. The covariance matrix is a symmetric matrix, so its eigenvectors will be orthogonal to each other. We then find the eigenvectors of this covariance matrix and sort them in descending order of their corresponding eigenvalues. The resulting matrix of sorted column vectors is our matrix of eigenvectors. If $x < k$, then to avoid high dimensions, we use a trick as described below:

Consider an n x m matrix A, with n being significantly larger than m. Its covariance matrix is AA^T. The resulting covariance matrix is of size n x n. If n is large, finding so many eigenvectors is computationally expensive. However, consider the eigenvector v and the corresponding eigenvalue μ for the matrix $A^T A$.

$$\begin{aligned} A^T A v &= \mu v \qquad \text{Pre-multiplying both sides by } A, \\ AA^T A v &= \mu A v \end{aligned}$$

Thus, if v is an eigenvector of $A^T A$, Av is an eigenvector for AA^T with the same eigenvalue μ. $A^T A$ is a matrix of size m x m. So we can find the m eigenvectors of $A^T A$ and pre-multiply them with A to give us m eigenvectors of the covariance matrix of A. Since these m eigenvectors correspond to the m highest eigenvalues, they also capture most of the information, so we can use these m vectors for eigen decomposition of our images.

2.2 Projecting and Reconstructing Data Samples

After we find the matrix V whose columns are the eigenvectors arranged in descending order of their eigenvalues, we can project data points onto these eigenvectors. We subtract the mean column vector of the training images from the images that we want to project. If A represents the matrix whose columns are the images we want to project, then we get the projection weights as $P = A^T V$. Each row of the resulting matrix corresponds to an image, and the values in the row are the projections of the image along the different eigenvectors. To reconstruct the image from this decomposition, we have to multiply these weights with each of the eigenvectors and add them. Thus, $(PV^T)^T$ gives us a matrix where each column is the reconstructed image. The reconstruction of the images is a linear combination of the eigenvectors, which means we sum the projections along each eigenvector direction.

2.3 K Nearest Neighbours

After projecting a test image into the eigenspace, we apply the k-nearest neighbours algorithm to classify the new image. This algorithm calculates the distance between the test image and all of the training images and finds the top K closest training images. It then predicts the class of the test image based on the majority class of the K closest train images. To find the optimal value of K, it should be at least as big as the number of class labels that are present, and we can run a number of experiments from there to find the best suited value of K. In our experiments, setting K to 10 worked well. The way we proceeded was as follows - We calculate the eigen weights, that is the projection weights or factors for each eigenvector of the test image. We also do the same for every train image. We then calculate the euclidean distance between the test eigen weights and each of the train eigen weights. We look at the K train eigen weight vectors which have the least distance to the test eigen weight vector, and the majority class label of these K training samples is our predicted test label.

3 Results

3.1 Eigenvectors

We visualized the first 4 eigenvectors corresponding to the 4 highest eigenvalues for different number of training samples. As seen in figures 1, 2, 3, 3 and 4 the eigenvectors get more distinguished as we increase the number of training samples from which to construct them.

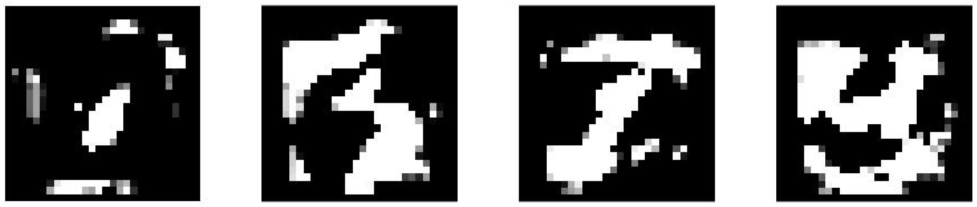


Figure 1: First 4 eigenvectors from 60 training samples

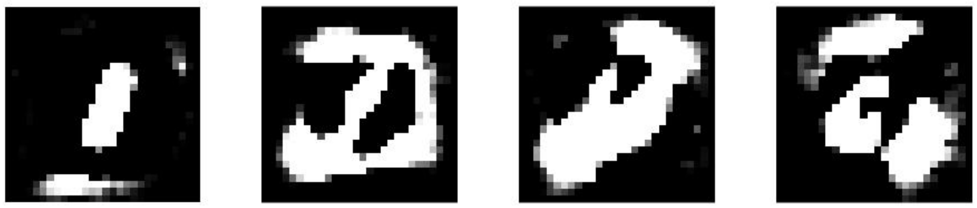


Figure 2: First 4 eigenvectors from 600 training samples

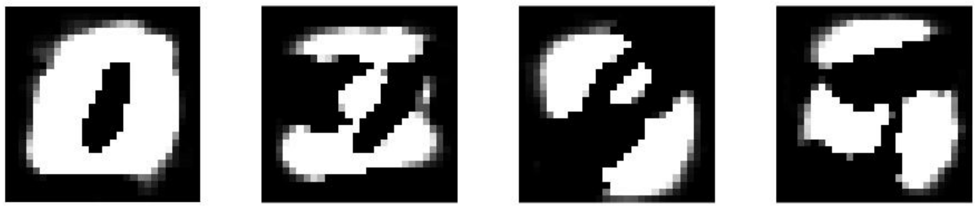


Figure 3: First 4 eigenvectors from 6000 training samples

These eigenvectors capture the most information about the training data as they are the first 4 eigenvectors. We also visualize eigenvectors that are lower in the sorted order, as in figure 5. We see that these eigenvectors are mostly noise.

3.2 Reconstruction of Test Images

We also visualize some of the reconstructed test images after they are projected into the eigenspace. From figures 6, 7, 8, and 9 we see that the quality of reconstruction depends a lot on the number of training samples

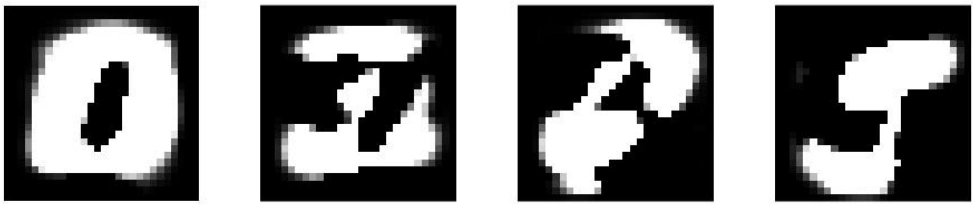


Figure 4: First 4 eigenvectors from all 60k training samples

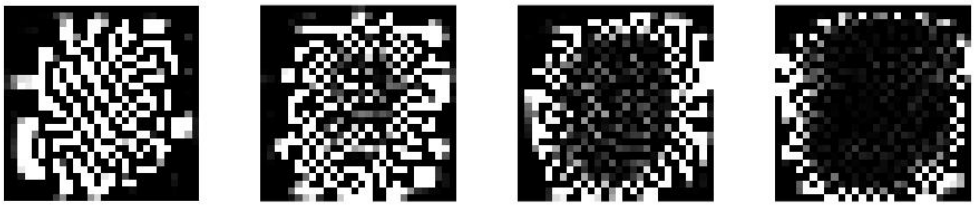


Figure 5: Eigenvectors 300, 400, 500, 600 from all 60k training samples

we use. With higher number of training samples, the reconstructed images come very close to the originals.

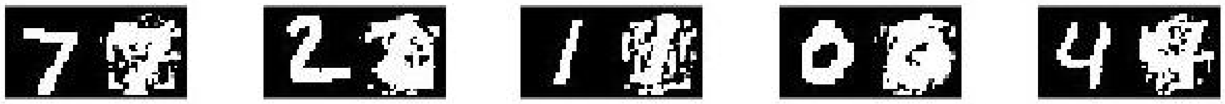


Figure 6: Projection of test images from all eigen vectors from 60 training samples (left: original test image, right: reconstructed image)

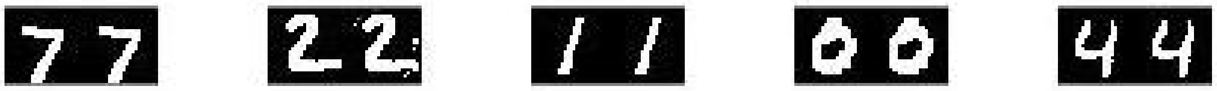


Figure 7: Projection of test images from all eigen vectors from 600 training samples (left: original test image, right: reconstructed image)

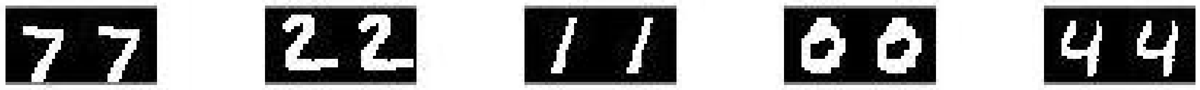


Figure 8: Projection of test images from all eigen vectors from 6000 training samples (left: original test image, right: reconstructed image)

We then decided to take only the top n eigenvectors and see how that affects the reconstruction. We see in figure 10 that 5 eigenvectors capture the main shape of the image though they are not enough to reconstruct a completely recognizable image in all cases. With 500 eigenvectors, we can reconstruct the images quite

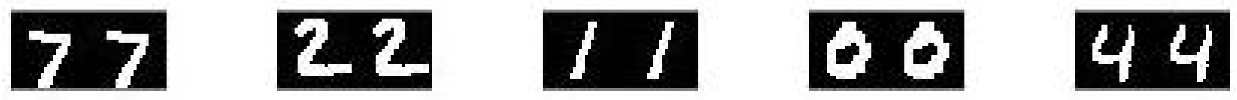


Figure 9: Projection of test images from all eigen vectors from all 60k training samples (left: original test image, right: reconstructed image)

well though there may be some noise in the background perhaps due to the influence of some irrelevant features captured by the later eigenvectors.

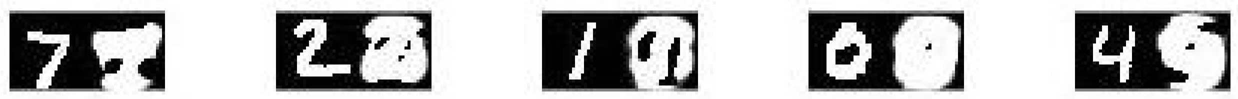


Figure 10: Projection of test images from first 5 eigen vectors from all 60k training samples (left: original test image, right: reconstructed image)

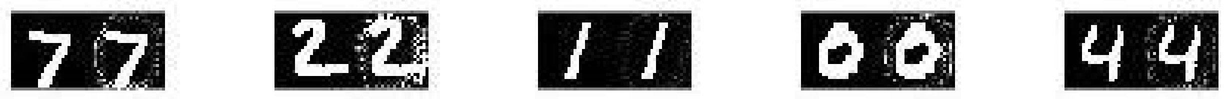


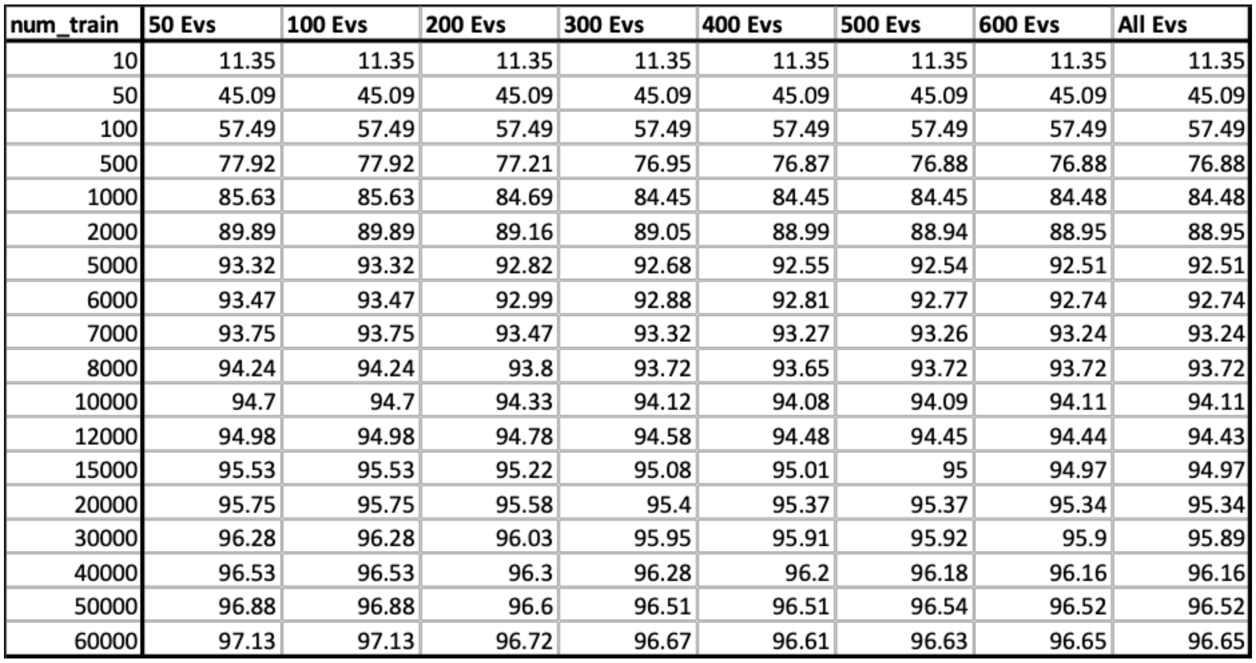
Figure 11: Projection of test images from first 500 eigen vectors from all 60k training samples (left: original test image, right: reconstructed image)

3.3 Classification with k Nearest Neighbours

In our experiments, setting K to 10 worked well, and so we carried out the rest of the experiments with that value. We first carried out experiments varying the number of training images and the number of eigenvectors under consideration for 2000 test images. The results of this are shown in the table in figure 12 and the graph in figure 13. We observe that for a given number of training samples, using a lower number of eigen vectors gives us better accuracy. This is because the later eigenvectors are noisy (as we saw in figure 5) and don't actually have any relevant information. When we extend the experiments to all the test images as shown in figures 14 and 15, we see similar trends. Since the accuracy values were mostly to within a single percent of each other across different eigen vectors, the graph lines are all closely set and almost overlap.

4 Summary

We calculated eigenvectors and eigenvalues for a set of training data points, and were able to project new points into this eigenspace which had the eigen vectors as basis vectors. We visualized the eigenvectors and saw that the initial few eigenvectors contribute most of the information about the data, while the ones towards the end are mostly noisy. We reconstructed and visualized the images we projected and saw that the quality of reconstruction improves with an increase in training samples. Finally, we applied the k-nearest neighbour classification technique in the reduced dimension space and got an accuracy of $> 97\%$ when we used all the training data and the first 50 eigen vectors.



num_train	50 Evs	100 Evs	200 Evs	300 Evs	400 Evs	500 Evs	600 Evs	All Evs
10	11.35	11.35	11.35	11.35	11.35	11.35	11.35	11.35
50	45.09	45.09	45.09	45.09	45.09	45.09	45.09	45.09
100	57.49	57.49	57.49	57.49	57.49	57.49	57.49	57.49
500	77.92	77.92	77.21	76.95	76.87	76.88	76.88	76.88
1000	85.63	85.63	84.69	84.45	84.45	84.45	84.48	84.48
2000	89.89	89.89	89.16	89.05	88.99	88.94	88.95	88.95
5000	93.32	93.32	92.82	92.68	92.55	92.54	92.51	92.51
6000	93.47	93.47	92.99	92.88	92.81	92.77	92.74	92.74
7000	93.75	93.75	93.47	93.32	93.27	93.26	93.24	93.24
8000	94.24	94.24	93.8	93.72	93.65	93.72	93.72	93.72
10000	94.7	94.7	94.33	94.12	94.08	94.09	94.11	94.11
12000	94.98	94.98	94.78	94.58	94.48	94.45	94.44	94.43
15000	95.53	95.53	95.22	95.08	95.01	95	94.97	94.97
20000	95.75	95.75	95.58	95.4	95.37	95.37	95.34	95.34
30000	96.28	96.28	96.03	95.95	95.91	95.92	95.9	95.89
40000	96.53	96.53	96.3	96.28	96.2	96.18	96.16	96.16
50000	96.88	96.88	96.6	96.51	96.51	96.54	96.52	96.52
60000	97.13	97.13	96.72	96.67	96.61	96.63	96.65	96.65

Figure 12: Classification accuracies on 2000 test images for K = 10 with varying number of training samples and eigen vectors (Evs = Eigenvectors)

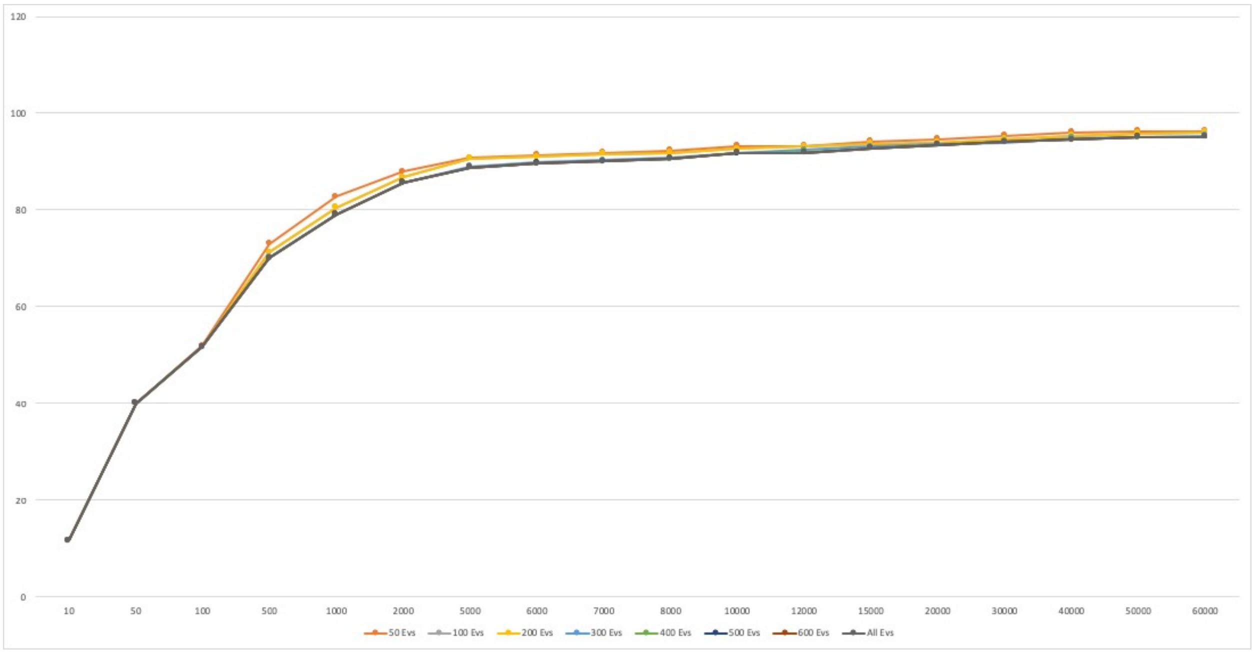
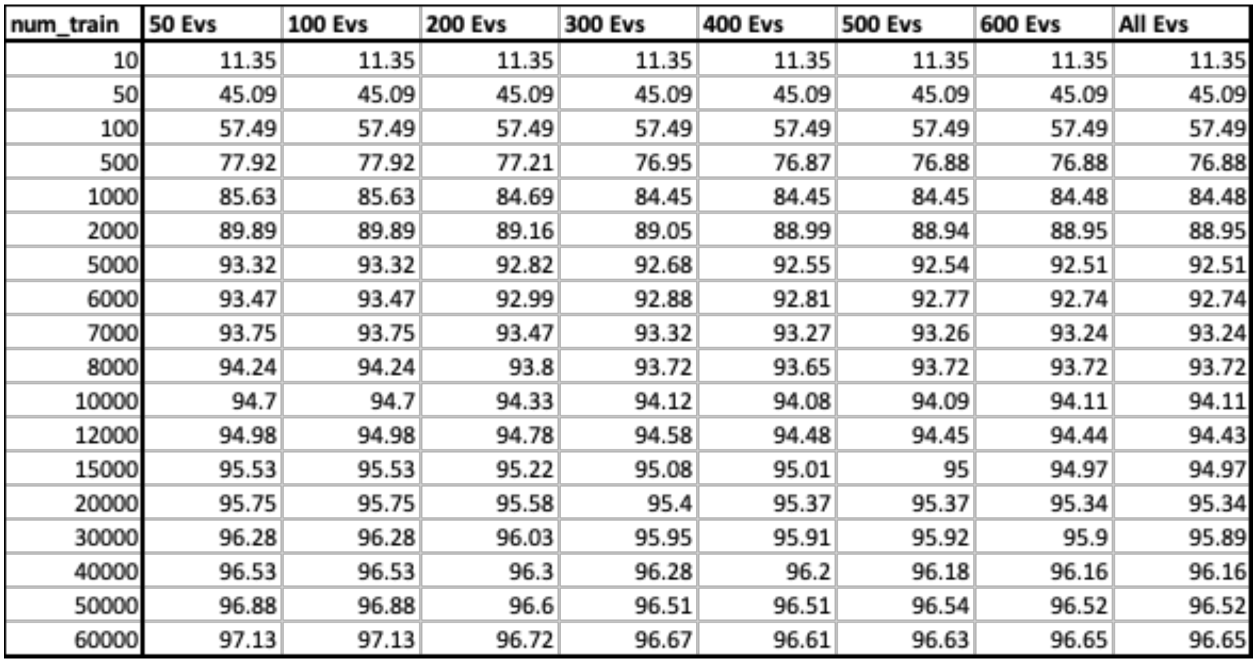


Figure 13: Graph of classification accuracies vs number of training samples for different eigenvector numbers. Experiments done on 2000 test images with K = 10. The orange line corresponds to 50 eigenvectors, yellow is 200, and grey is all the eigenvectors. The remaining eigenvector numbers have a large overlap.



num_train	50 Evs	100 Evs	200 Evs	300 Evs	400 Evs	500 Evs	600 Evs	All Evs
10	11.35	11.35	11.35	11.35	11.35	11.35	11.35	11.35
50	45.09	45.09	45.09	45.09	45.09	45.09	45.09	45.09
100	57.49	57.49	57.49	57.49	57.49	57.49	57.49	57.49
500	77.92	77.92	77.21	76.95	76.87	76.88	76.88	76.88
1000	85.63	85.63	84.69	84.45	84.45	84.45	84.48	84.48
2000	89.89	89.89	89.16	89.05	88.99	88.94	88.95	88.95
5000	93.32	93.32	92.82	92.68	92.55	92.54	92.51	92.51
6000	93.47	93.47	92.99	92.88	92.81	92.77	92.74	92.74
7000	93.75	93.75	93.47	93.32	93.27	93.26	93.24	93.24
8000	94.24	94.24	93.8	93.72	93.65	93.72	93.72	93.72
10000	94.7	94.7	94.33	94.12	94.08	94.09	94.11	94.11
12000	94.98	94.98	94.78	94.58	94.48	94.45	94.44	94.43
15000	95.53	95.53	95.22	95.08	95.01	95	94.97	94.97
20000	95.75	95.75	95.58	95.4	95.37	95.37	95.34	95.34
30000	96.28	96.28	96.03	95.95	95.91	95.92	95.9	95.89
40000	96.53	96.53	96.3	96.28	96.2	96.18	96.16	96.16
50000	96.88	96.88	96.6	96.51	96.51	96.54	96.52	96.52
60000	97.13	97.13	96.72	96.67	96.61	96.63	96.65	96.65

Figure 14: Classification accuracies on all 10k test images for K = 10 with varying number of training samples and eigen vectors (Evs = Eigenvectors)

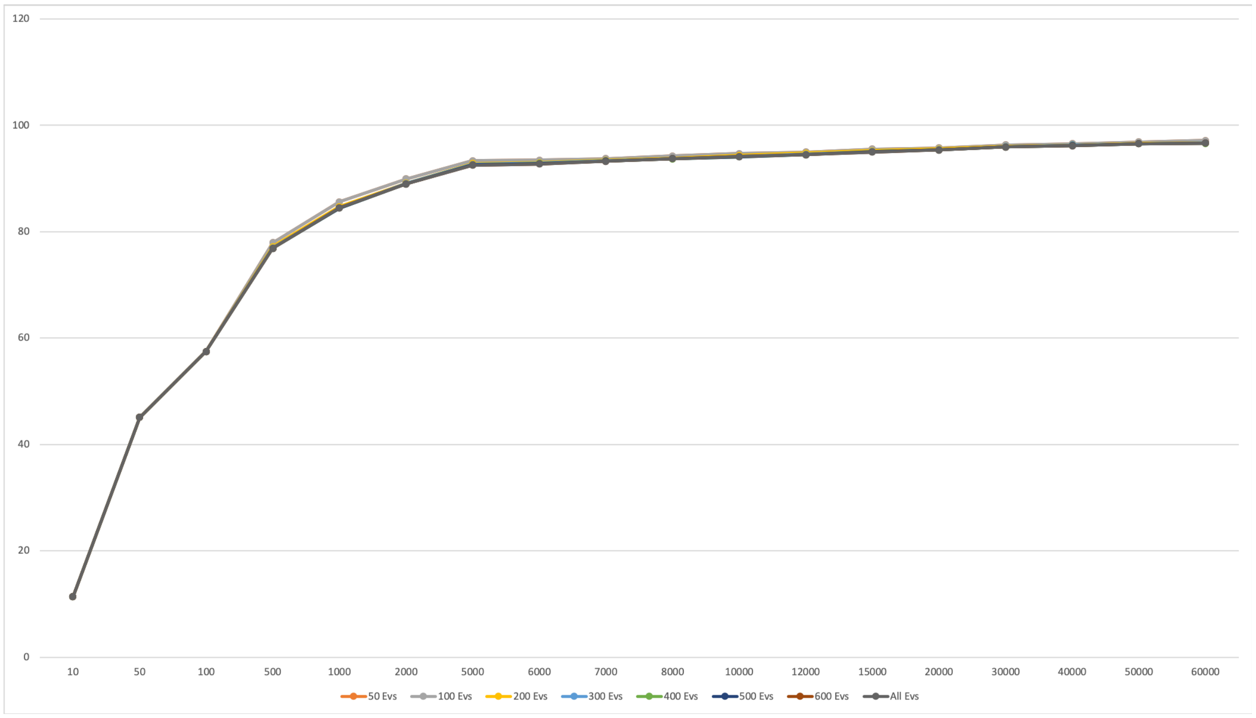


Figure 15: Graph of classification accuracies vs number of training samples for different eigenvector numbers. Experiments done on all 10k test images with K = 10. The light grey line corresponds to 100 eigenvectors, yellow is 200, and dark grey is all the eigenvectors. The remaining eigenvector numbers have a large overlap.