MAT 161 - APPLIED LINEAR ALGEBRA
FINAL PROJECT REPORT

ADITYA AGAGRWAL - 2110110854

INDEX

INTRODUCTION

The goal of this project is to build a face recognition that understands if the given image belongs to the person the model was trained on.

In this case, the person who we are testing on is our dear friend Pranav Dhar and we have 22 photos of him in total.

5 of them are being used as test images and 17 are being used for training and 3 other random images are being used for testing.

Below follows the explanation of my project in detail followed bya video and step by step explanation.

BEFORE READING

Before reading this document, we suggest that you look at the below attached video link to look at our understanding of math behind the concerned paper and how we have replicated it in our smaller scale model.

EXPLANATORY VIDEO

STEP 0

The step 0 in understanding and simulating the concepts shown in the research paper are as follows:

1. Reading the research paper
    a. This step was easy to finish with the help of the link provided to us here by Dr. Santosh Singh
2. Taking pictures of a test subject and saving them with appropriate names
    a. For this step we selected our friend Pranav Dharr a.k.a Tarzan who is a professional model for a lot of modeling agencies
3. Splitting the pictures into testing and training sets
    a. The next step was splitting them into 2 groups
    b. A total of 22 pictures were clicked out of which 17 were used for training the model and 5 were for testing it out
4. Converting them to black and white
    a. Function from the open_cv module were used to convert all images to black and white
    b. You can read up more about the module here
5. Resizing them to standard 1600x900px size
    a. This was also achieved with the help of the open_cv module

<u>STEP 1</u>

The next step is to read in all images that were selected for training and calculate the gray scale value for each pixel.

In other words, we are now converting a 900 x 1600 pixel image to 2d-matrix with each entry in the matrix representing the gray scale value of the corresponding pixel.

Since we have 17 such images we will now have 17 such 2d matrices.

The next step is to convert each of these 2d matrices into a 1440000 dimensional vector, and since there are 17 images we will have 17 such vectors.

Now arranging these 17 vectors as columns of a matrix resulting in a 1440000 x 17 matrix.

This matrix is then stored as a csv file in the train.csv file.

STEP 2

The next step is to calculate the mean vector.

As of now we have the train.csv file with all the images stored as column vectors in the file. We now load this file into a numpy dataframe and then calculate the column wise mean for each column.

This results in another numpy array of the size 1440000 x 1, which is them stored in the mean.csv and later referred to as the mean array for the training data.

The next thing we did was calculate the difference between each vector and the mean and again stored it in a csv file called difference.csv.

This difference matrix is a matrix of the size 1440000 x 17, since each image will generate one difference vector.

This difference matrix will hence forward be useful in calculating the weights which is a crucial component of the next step.

STEP 3

This is the most computationally taxing step with high requirements for memory needs.

First thing we do here is calculate the covariance matrix of the difference matrix.

This is done by multiplying the difference matrix with its transpose, however this will result in a 1440000 x 1440000 matrix which will be very very hard to store or process.

Thus to reduce complexity we instead calculate the covariance matrix of the transpose of difference matrix.

This is done by multiplying the transpose of the difference matrix with the original difference matrix, this will result in a 17x17 matrix which is much much smaller when compared to the previous matrix which was of 1440000 x 1440000.

We will henceforth refer the above matrix created as similar matrix

Now what we do is calculate the eigenvalues and eigenvectors for this similar matrix. This is done using the numpy module and the linalg.eig function in python.

The eigenvectors of the similar matrix and eigenvectors of the original covariance matrix will be related such that the product of eigenvectors of the similar matrix multiplied with the original difference matrix will result in the eigenvectors of the original covariance matrix.

Now we normalize these eigenvectors of the original matrix and call it eigenfaces.

Next step is now calculating weights for these 17 training photos.

This is done by calculating the product of transpose of eigenfaces with the all_differnece_matrix, which basically had all the differnece_vectors combined.

Now that, we have 17 sets of weights for my 17 training images. We are officially done with the training part and are ready to move ahead with the testing part once we set a threshold value.

To choose an optimum threshold value, we had decided to go ahead with 0.4225 * (maximum distance between the weights of the training set).

We can officially move on to testing.

Just to justify the accuracy of our weights we also calculate the mean square error.

TESTING

We now take the input path to a test image.

First thing we do is resize the image and convert it to black and white photo using the open_cv module.

Next thing we do is convert the image to a numpy array of size 1600 x 900 which will have each element representing the gray scale value of the image.

The next step is converting this matrix into a vector of 1440000 x 1 dimension.

We then load the mean vector that we calculated earlier and stored in a csv.

We now calculate the difference between the new input image vector and the mean vector.

We now calculate the distance between these new input weights and the previous training weights.

Followed by this we compare the minimum distance between these weights with the threshold that we had set earlier.

This comparison now helps us make the prediction that lets us decide if the photo entered is of our dear friend Tarzan or not.

TEST RESULTS

5 pictures of Tarzan and 3 random pictures were selected for testing this model.

And the model perfectly recognised the 8 images and were able to decide if the pictures were of Tarzan or not.