CSE 5523: FINAL PROJECT REPORT

Submitted by – Balavignesh Vemparala Narayana Murthy

1. Problem Description & Overview

For the Final Project, I have chosen the task of facial recognition, programmed in MATLAB. The task is, given a set of face images with different poses, expressions, with/without glasses, and. sizes, train a Machine Learning model which can identify the class given a new test image, based on the training data. The CMU Face images dataset, consisting of 640 images of 20 different subjects was chosen for the purpose. And two different techniques – Eigenfaces using Principal Component Analysis (PCA), Convolutional Neural Networks (CNNs) were used for classification, with the former serving as a baseline technique. Moreover, the dataset is comprised of three different image resolutions - 30×32 , 60×64, 120×128, and a pre-processing tool was coded which is capable of resizing images to the same dimensions.

2. CMU Face images dataset

The dataset consists of 640 black and white face images of people taken with varying poses (straight, left, right, up), facial expressions (neutral, happy, sad, angry) as well as with/without sunglasses, and image size (30×32 , 60×64, 120×128). Moreover, the dataset consists of images for 20 different subjects. Some of the images, which are suffixed with ".bad" were manually removed from the dataset, as these are the images with glitches due to issues with camera setup [1]. Figure 1 shows some images from the dataset.

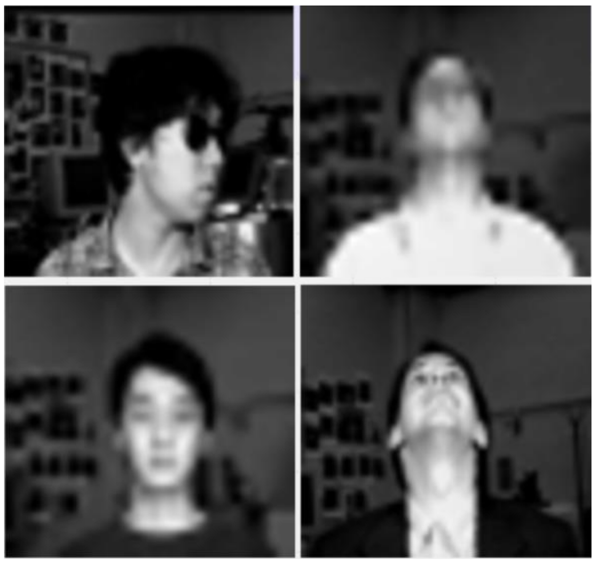


Figure 1: CMU Face images dataset

3. Techniques used

3.1. Pre-processing

First of all, a pre-processing tool was written in MATLAB, which reads all the images in the dataset, and reshapes them to a user-defined shape. This program, namely, "preprocessing.m" has been attached with the submission. This pre-processing step primarily uses the "imresize" function in MATLAB, which simply applies a scale transformation to the original image. Note that applying a scale transformation to an image doesn't really enhance the resolution, in the sense that we donot have any more information in the output than the input, however, it just interpolates to create a larger/smaller grid than the one that was started with [2]. The "imresize" function uses an inverse mapping to figure out where each point in the output image maps in the input image, and then interpolates it within the input grid to determine output grid values [2]. Bicubic interpolation is used by default, i.e., the output pixel value is a weighted average of pixels in the nearest 4-by-4 neighborhood [3].

3.2. Method of Eigenfaces using Principal Component Analysis (PCA)

Eigenfaces is an approach using dimensionality reduction techniques for face recognition, proposed by Turk & Pentland in 1991 [4,5]. In this project, Principal Component Analysis (PCA) was used to compute the Eigenfaces of a given human faces dataset, which were then used to recognize testing face images. The Eigenfaces algorithm is described as follows –

Step 1 – Given the training faces dataset, compute the mean face 'M'. So, for instance, if we have 'm' training images x_i of dimensions $N \times N$, then firstly each of the training images are converted into vectors of dimensions $N^2 \times 1$. Then, the mean face is computed as

$$M = \frac{1}{m}\sum_{1}^{m} x_i$$

Step 2 – Compute the deviations of the training faces x_i from the mean face 'M' as shown below –

$$\tilde{x_i} = x_i - M$$

Step 3 – Compute the Covariance matrix $\mathrm{S} = \tilde{X}'\tilde{X}$, where

$$\tilde{X} = [\tilde{x}_1\ \tilde{x}_2\ \tilde{x}_3\ \dots\ \tilde{x}_m] \text{ of size } N^2 \times m$$

Step 4 – Perform Eigenvalue Decomposition of Covariance matrix S to get the Eigenvalues λ_i and Eigenvectors v_i.

$$\tilde{X}'\tilde{X}v_i = \lambda_i v_i$$

$$\tilde{X}\tilde{X}'\tilde{X}v_i = \lambda_i \tilde{X}v_i$$

$$\mathrm{S}'u_i = \lambda_i v_i$$

where $\mathrm{S}' = \tilde{X}\tilde{X}'$. It can also be concluded that S' and S have same eigenvalues and their eigenvectors are related as $u_i = \tilde{X}v_i$.

Step 5 – Next select 'k' eigenvectors of S' corresponding to 'k' largest eigenvalues ($k < m$), such that sum of the corresponding eigenvalues is 99% of sum of all eigenvalues

Step 6 – Now, represent the normalized training faces as a linear combination of the best 'k' eigenvectors, also called eigenfaces. Now, each of the training face can be represented as a vector of coefficients $x_i \equiv [w_1^i, w_2^i, \dots, w_k^i]$

$$x_i - M = \sum_{j=1}^{k} w_j u_j$$

Step 7 – Next, represent any testing image as a vector of coefficients using the same procedure above. Next, find the best match from the training set using minimum distance as the criterion.

3.3. Convolutional Neural Networks (CNNs)

A CNN is a Deep Learning algorithm, popular in face detection/object recognition community, which is able to automatically learn low-level, mid-level features from a set of images, that are necessary for classification. Instead of using normal feed-forward Neural Networks over images (which would be computationally very expensive), CNNs convert images into a form that is easier to process (through the application of filters). This makes CNNs attractive as this makes it easily scalable to massive datasets.

The CNN used in this study looks as shown in Figure 2 below, with three Convolution, batch normalization and ReLU layers.

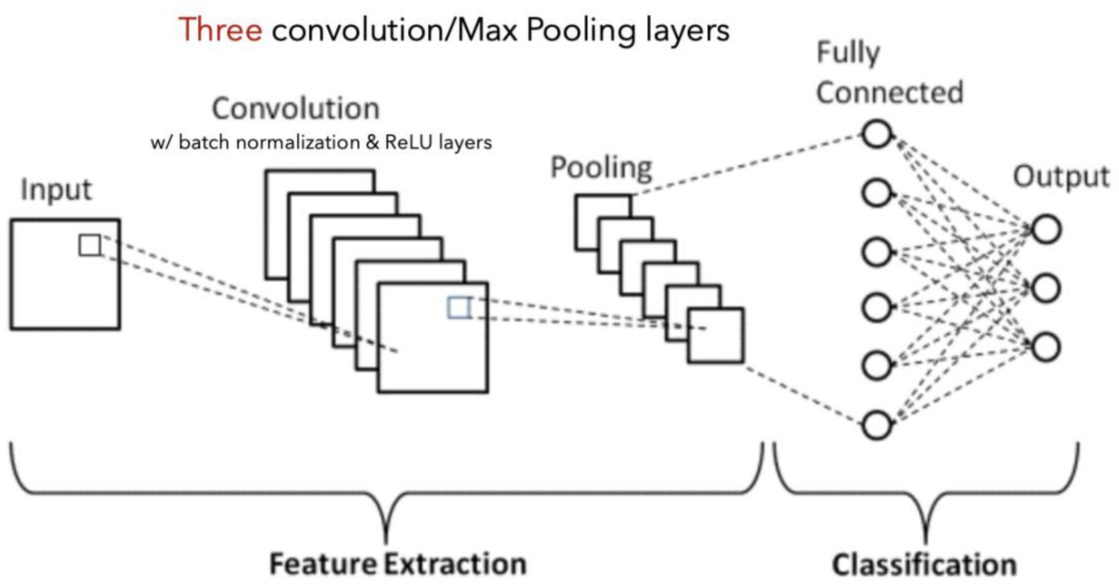


Image source - https://www.upgrad.com/blog/basic-cnn-architecture/

Figure 2: CNN used in this study

So, a convolution operation convolves an input image with a kernel and gives the output features, at each of the three layers. Batch normalization normalizes the activations and gradients propagating through a network, to stabilize or speed-up the convergence. ReLU activation is used to increase the non-linearity in our images, as real-world images data tend to be non-linear. And finally, a max-pooling layer follows, which applies a down-sampling operation that reduces the spatial size and removes redundant information. Moreover, this also reduces the computation needed in subsequent convolution layers. Finally, the fully connected layer combines all the features learned by the previous layers to classify the images, and the softmax activation function normalizes the output of this layer, which gives the probabilities for the different classes, which can be used for classification.

4. Results & Discussions

4.1. Method of Eigenfaces using PCA

Eigenfaces approach is scale-sensitive, and hence, all the images in the dataset were reshaped to the same dimensions using pre-processing tool (60×64). The mean face and the eigenfaces computed on the training data are shown in Figure 3 below –

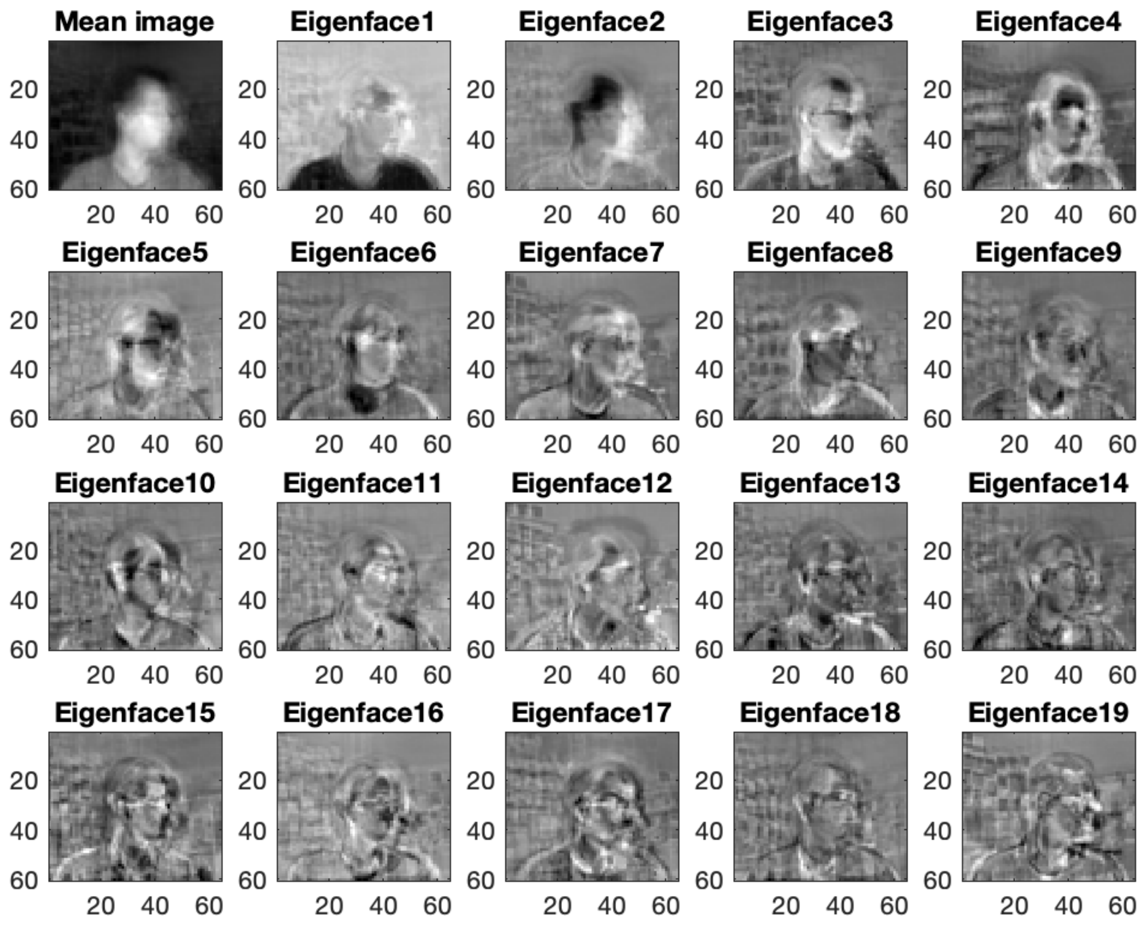


Figure 3: Mean face and Eigenfaces

When only five of the classes were used (an2i,at33,boland,bpm,ch4f), this method gave a classification accuracy of 100%. However, when all 20 classes were used, the classification accuracy dropped to 60%. **This is expected since PCA is equivalent to the search for a linear subspace which**

maximizes the variance between data points or in other words, minimizes the sum of squared distances from the data points to their projections onto the linear subspace, i.e., it is a linear model.

Figure 4 below shows examples of correct and wrong classifications using Eigenfaces method –

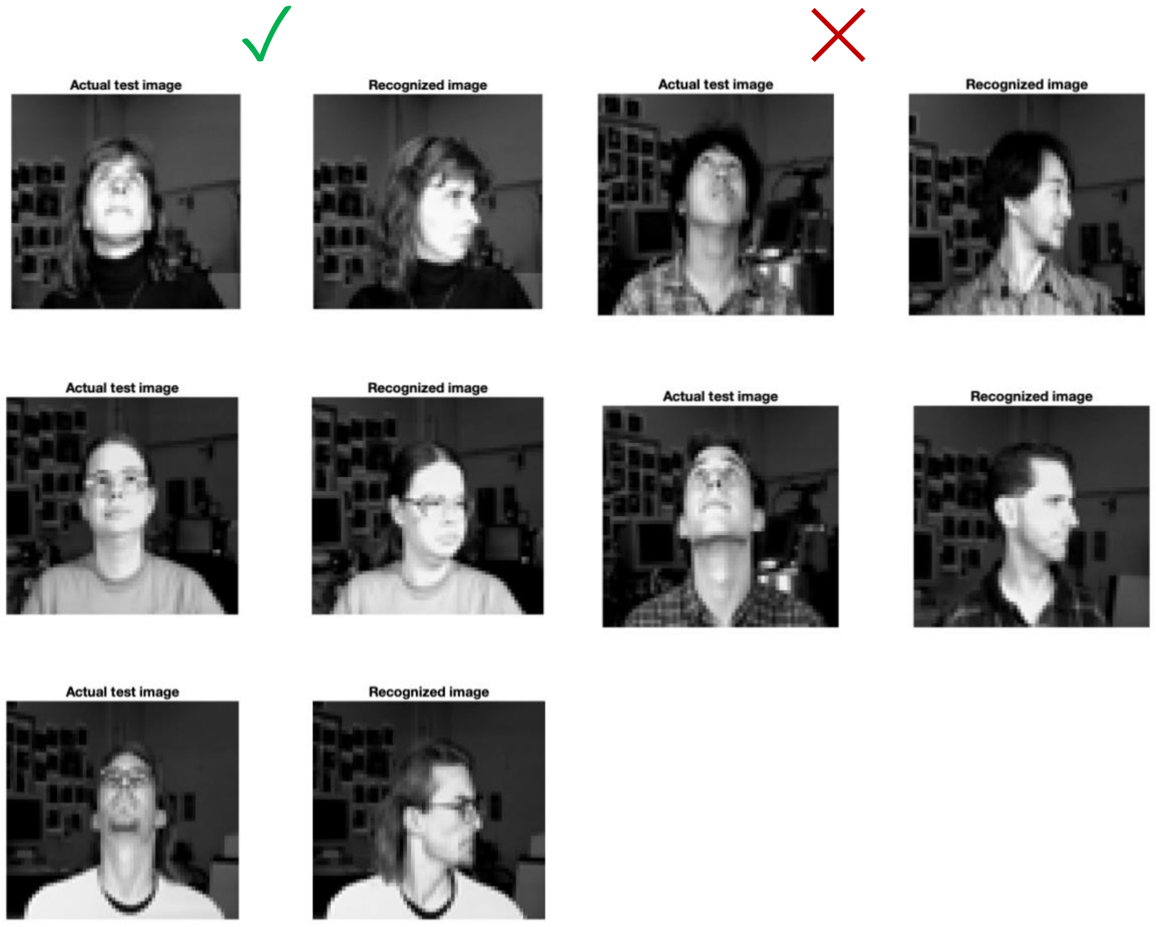


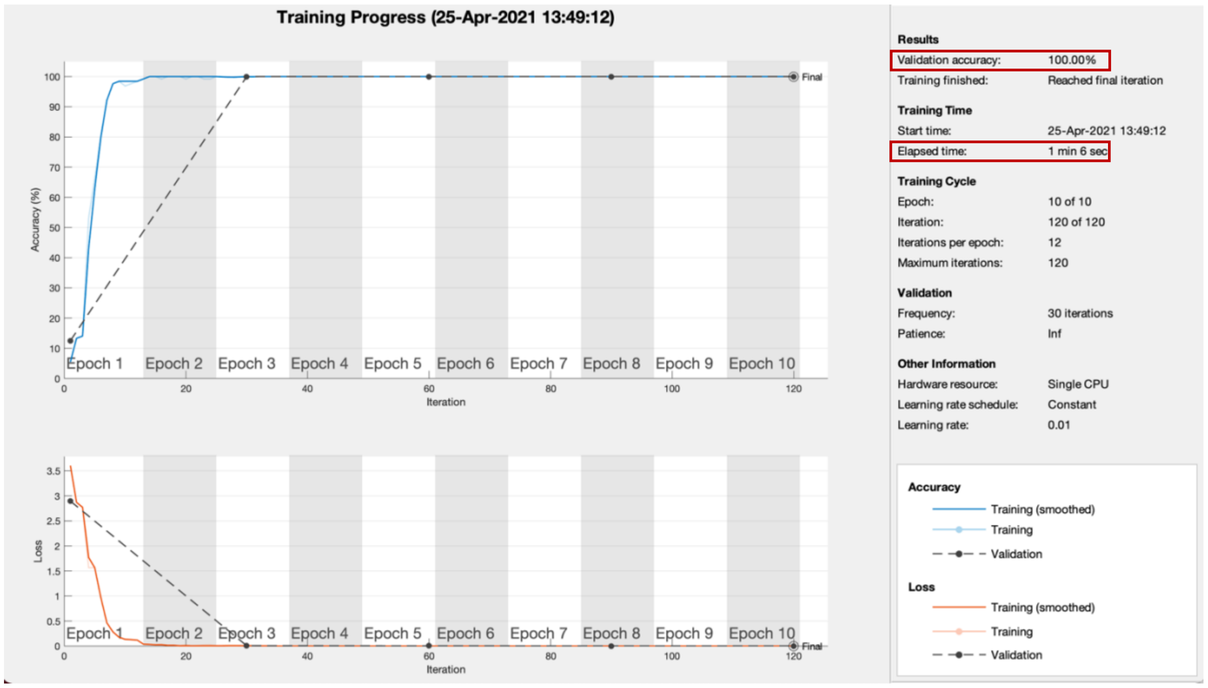
Figure 4: Correct and wrong predictions using Eigenfaces approach

Also, a sensitivity to the image dimensions was observed with PCA. As mentioned earlier, three different image dimensions 30×32 , 60×64, 120×128 are present in the dataset. I was able to run PCA with only images of dimensions 30×32 and 60×64 in this current study. When I tried running PCA on images with dimensions 120×128, it did not reach completion even after 30 mins and hence, I reshaped all images to 60×64 using the pre-processing tool developed in this study.

This is because, Eigenvalue decomposition is the step that takes the most time, as this was the step that lead to extremely slow progress using 120×128 images.

4.2. Convolutional Neural Networks (CNNs)

Using CNNs on the other hand, **a classification accuracy of 100%** was obtained. Moreover, the time taken using CNNs was also only around 1 minute, whereas it took several minutes using Eigenfaces approach (around 5 mins). The training progress using CNNs is shown in Figure 5 below –



```
Results
Validation accuracy:        100.00%
Training finished:          Reached final iteration

Training Time
Start time:                 25-Apr-2021 13:49:12
Elapsed time:               1 min 6 sec

Training Cycle
Epoch:                      10 of 10
Iteration:                  120 of 120
Iterations per epoch:       12
Maximum iterations:         120

Validation
Frequency:                  30 iterations
Patience:                   Inf

Other Information
Hardware resource:          Single CPU
Learning rate schedule:     Constant
Learning rate:              0.01
```

Figure 5: Training progress using CNNs

5. Conclusions

Concluding, comparing Eigenfaces approach with CNNs, we observe that using CNNs not only leads to a better classification accuracy (60% using Eigenfaces Vs 100% using CNNs), but also takes much less time (5 min using Eigenfaces Vs 1 min using CNNs). Moreover, Eigenvalue decomposition step in Eigenfaces approach is very slow (> 30 mins), which makes the use of 120×128 images impossible using Eigenfaces approach, but irrespective of the input image size, time taken using CNNs is only around 1 min consistently.

Also, in the current study, the images dataset consists of images from the same session with similar lighting, illumination. Using images with different poses and illumination will lead to much lesser classification accuracy according to [6]. However, this hasn’t been tried upon in the current project.

REFERENCES

[1] https://archive.ics.uci.edu/ml/datasets/CMU+Face+Images

[2] https://www.mathworks.com/matlabcentral/answers/82893-how-does-the-function-imresize-work-in-details

[3] https://www.mathworks.com/help/images/ref/imresize.html

[4] Turk, M. and Pentland, A., 1991. Eigenfaces for recognition. *Journal of cognitive neuroscience*, *3*(1), pp.71-86.

[5] https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/

[6] http://mathdesc.fr/documents/facerecog/AdvantagesLimitations.htm