DEEP LEARNING PROJECT REPORT

By Akhilesh Rajagopalan and Ryiad Hajji

I - Construction of Face Recognition Algorithm using Facenet Keras and MTCNN

Face recognition is the general task of identifying and verifying people from photographs of their face.
We will be using the FaceNet, which is a face recognition system that was described by Florian Schroff, et al. at Google in their 2015 paper titled "FaceNet: A Unified Embedding for Face Recognition and Clustering."

The model is a deep convolutional neural network trained via a triplet loss function that encourages vectors for the same identity to become more similar (smaller distance), whereas vectors for different identities are expected to become less similar (larger distance). The aim of this model was to create embeddings directly and is used as the one of the benchmark embedding models for training classifier systems on standard face recognition datasets.

The model was downloaded from here:

- Keras FaceNet Pre-Trained Model (88 megabytes)
  The Name of the file: '*facenet_keras.h5*'.

We then go on to use the MTCNN , which is a Multi-Task Cascaded Convolutional Neural Network, or MTCNN, for face detection, e.g. finding and extracting faces from photos. This is a state-of-the-art deep learning model for face detection, described in the 2016 paper titled "*Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks*." And we have installed this via pip.

Aim: We intend to use the dataset of 5 celebrities, where we have a training set and a test set of pictures of five famous pop culture figures.

Data link: https://www.kaggle.com/dansbecker/5-celebrity-faces-dataset

(Madonna, Jerry Seinfeld, Ben Affleck, Mindy Kaling, Elton John)

Training Set: 93 Images

Test set: 25 Images

1. We use the MTCNN package which is installed with pip, to identify the faces of these celebrities and isolate these areas and extract them as separate images. data.

2. We store these images with their respective labels(which we retrieved through the initial organization of the files as each celebrity had a separate file with their pictures.)

3. We then use the pretrained Facenet model, which is retrieved through the link above, This was used to retrieve the embedding of each image, which takes the faces image as input and spits out an output of a vector of 128 dimensons representing the most important features of the face in our case.

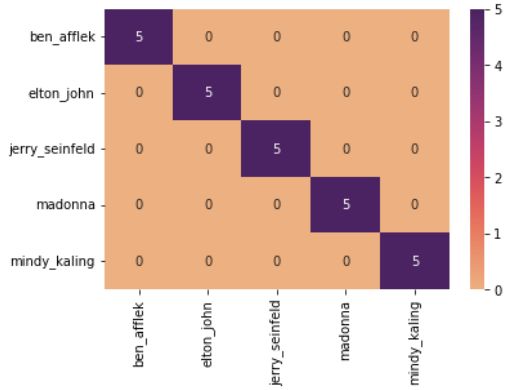
4. We the undertake the important step of pixel normalization is an important step which ensures that each pixel has a similar data distribution. This makes convergence faster while training the network, It is done by subtracting the mean from each pixel and then dividing the result by the standard deviation. The distribution of such data would resemble a Gaussian curve centered at zero. For image inputs we need the pixel numbers to be positive, so we choose to scale the normalized data in the range [0,1].

RESULTS

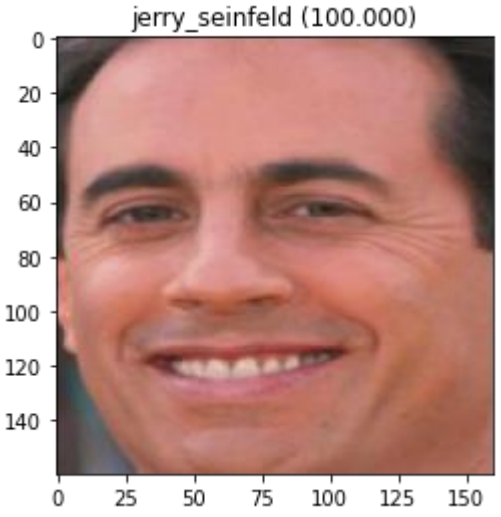
After these Preprocessing step, We go forward with the application of the final data onto three different Classification algorithms.

1. Linear Discriminant Analysis

Confusion Matrix

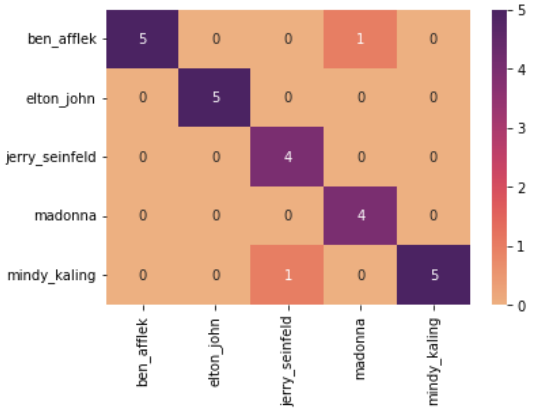


Example Image (Name and Prediction Probability)

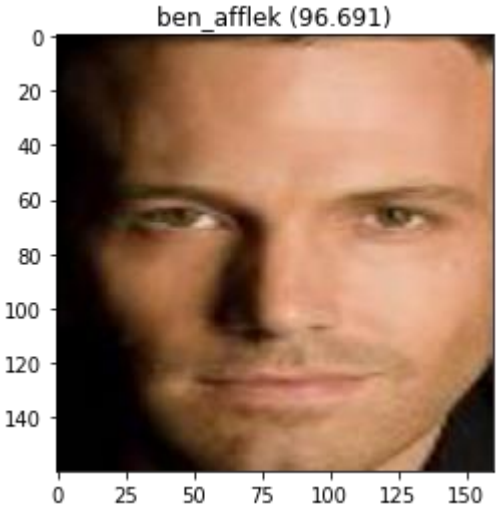


2. XGBOOST

Confusion Matrix

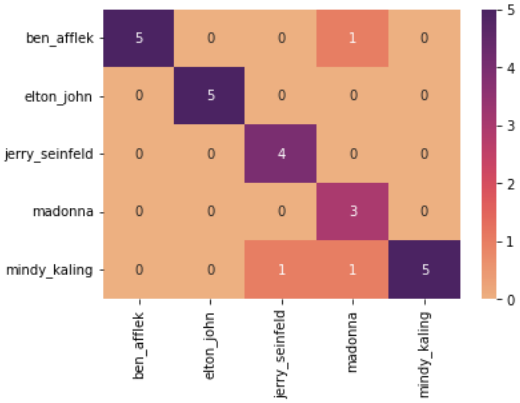


Example (Name and Prediction Probability)

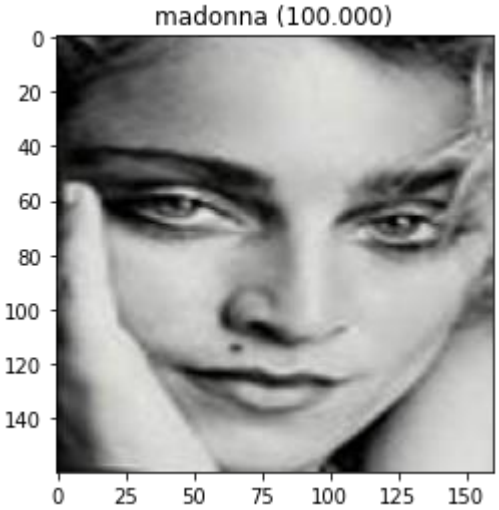


3. Gradient Boosting

Confusion Matrix

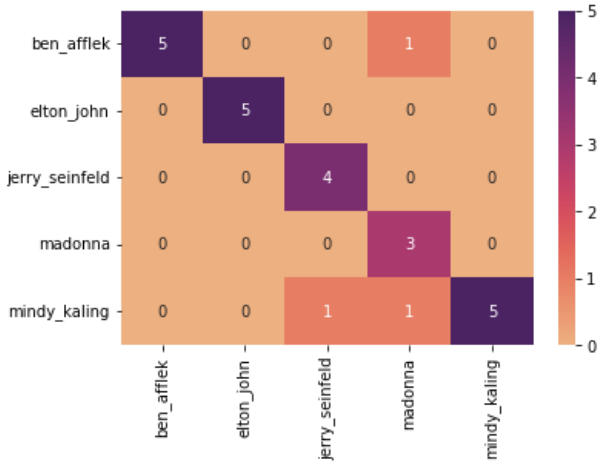


Example (Predicted Name and Predicted Probability)

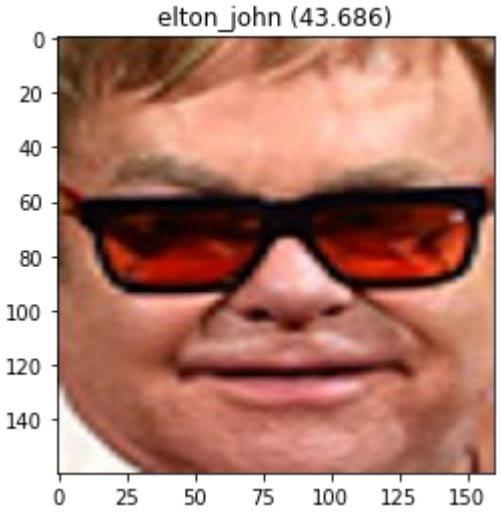


4. Random Forest

Confusion Matrix



Example (Predicted Name and Predicted Probability)



1. We see that amongst all of the classifiers, the best classifier algorithm was the simplest one which was linear discriminant analysis which also took the least amount of time to execute.
2. We see that as the classifying model used increased in its complexity, the results were also not as good as LDA as it was seen in Gradient boosting, XG Boost and Random Forest.
3. This assertion is conditional on the fact that the dataset we were using was very small and this cannot to be extended to larger and more complex datasets.

II – Face Recognition using eigenfaces method

1. The data

The data is called Olivetti's dataset, it contains 10 different pictures from 40 people. The photos were taken between April 1992 and April 1994 at AT&T Laboratories Cambridge and we can load it from sklearn via the function **sklearn.datasets.fetch_olivetti_faces**.

2. A simple application of eigenfaces method from Peter N. Belhumeur, Joao P. Hespanha and David J. Kriegman

Eigenfaces method uses PCA to reduce pictures dimensions. If $\{x_1, \ldots, x_N\}$ is a set of N images, the new features vectors are defined by a projection of the initial images :

$$\mathbf{y}_k = W^T \mathbf{x}_k \qquad k = 1, 2, \ldots, N$$

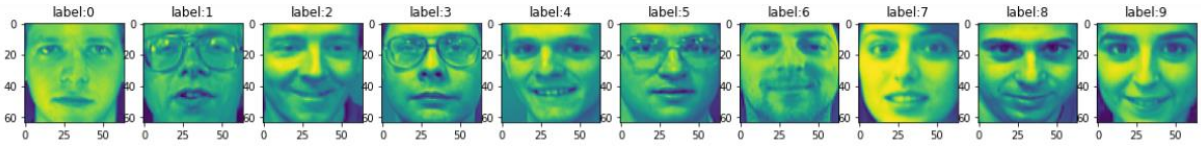
Next, the scatter matrix, which is a generalization of the variance covariance matrix records the link between variables in order to find in which way we are going to maximise the variance. This matrix is defined as follows :

$$S_T = \sum_{k=1}^{N} (\mathbf{x}_k - \boldsymbol{\mu})(\mathbf{x}_k - \boldsymbol{\mu})^T$$

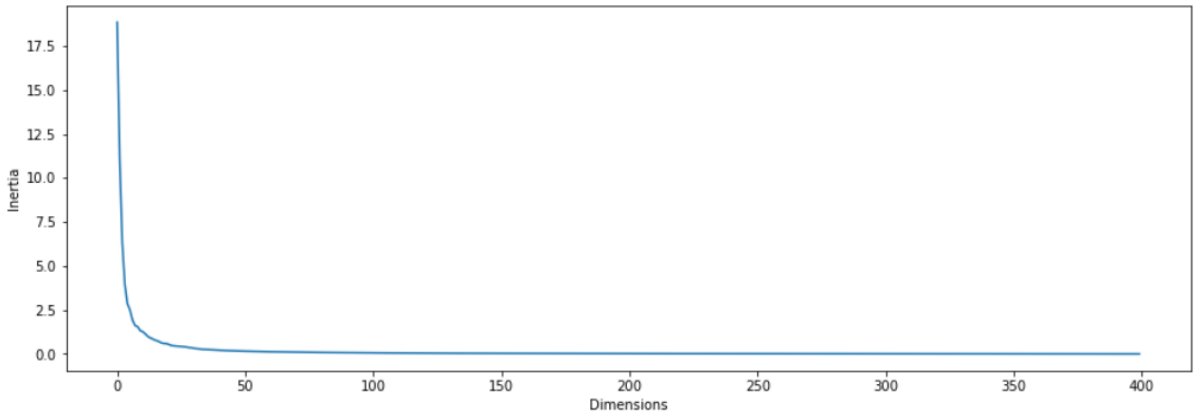
The optimal projection matrix is obtained by maximizing the expression in absolute value involving eigenvectors corresponding to the largest eigenvalues :

$$W_{opt} = \arg\max_W \left| W^T S_T W \right|$$

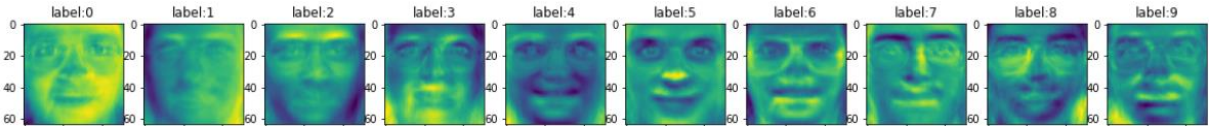
Before PCA, the face pictures are of this form :



Using PCA method provided by sklearn package, plotting the total inertia according to the number of dimensions provide us the optimal number to keep and we can see below that keeping 50 or a bit less is fair enough to don't loose to much information.



After dimension reduction by using 50 principal components, we can then obtain the eigenfaces, meaning the projection of the pictures over the space of dimensions kept :



Let's try to apply some algorithms to predict the label of the pictures, the methods used are :

- LDA
- Logistic Regression
- Decision Tree Classification
- Support Vector Machine

And the corresponding accuracy results are :

```
LR Result : 0.98
DT Result : 0.45
SVM Result : 0.94
LDA Result : 0.73
```

We can see that the Linear Regression and the Support Vector Machine methods performed well.

Now, let’s try to apply it on the 5 celebrities dataset :

The results are the following :

```
LDA Result : 0.88
LR Result : 0.72
DT Result : 0.44
SVM Result : 0.84
```

Linear Regression and Support Vector Machine are still the algorithms that provide the best results even though the precision is lower when applied on the 5 celebrities dataset.

REFERENCES

1. https://machinelearningmastery.com/how-to-develop-a-face-recognition-system-using-facenet-in-keras-and-an-svm-classifier/

2. https://github.com/loicbausor/face-recognition-and-embedding

3. https://towardsdatascience.com/step-by-step-face-recognition-code-implementation-from-scratch-in-python-cc95fa041120

4. https://www.kaggle.com/serkanpeldek/face-recognition-on-olivetti-dataset/notebook

5. https://cseweb.ucsd.edu/classes/wi14/cse152-a/fisherface-pami97.pdf