Biometrics System Concepts [H02C7a] - Assignment 3 (Facial Recognition)

Daniel Rey Šparemblek

r0883565

1 Introduction

Facial biometrics is becoming the most common biometric method due to its easy implementation and the lack of user interaction needed for use. It is used in many different sectors including security, health care, banking, retail, etc.

All facial recognition systems require three features on a high-level overview, those are:

- Detect a face from a given image.
- Extract feature vector descriptors from the raw face image.
- Matching the features with other facial features in the system for verification or identification purposes.

In this assignment, we look at all these steps but mostly focus on the last one - comparing different feature vector descriptors for verification/identification purposes.

The four procedures we look at are: Principal Components Analysis (Eigenfaces), Linear Discriminant Analysis (Fisherfaces), Local Binary Pattern (LBP), and deep learning.

2 Datasets

There are many different publicly available datasets that we can use. We choose The CALTECH Faces dataset which consists of 450 images of approximately 27 unique people - although we make certain deletions from this set to make it 440 images and 26 unique people. Each subject was captured under various lighting conditions, background scenes, and facial expressions.

We will later try two other publicly available datasets and measure the accuracy of our system on these other datasets.

3 Face detection

Although the dataset provides bounding boxes for the faces - they have already been detected and saved - we still choose to use one popular face detection method called HAAR-cascade to extract them. Later on, we will compare this method's accuracy with two other face detection methods.

The HAAR-cascade method is a popular face detection technique that uses Haar-like features and a cascading classifier. It involves training a classifier on a large dataset of positive and negative

samples, where the classifier learns to detect face-like patterns by efficiently evaluating Haar-like features in a cascading manner to achieve high detection accuracy.

4 Feature extraction

4.1 PCA - Eigenfaces

The Eigenfaces algorithm, using Principal Component Analysis (PCA), is a facial recognition technique that involves collecting a dataset of face images and applying eigenvalue decomposition to extract eigenvectors representing the most significant facial features. These eigenvectors, known as eigenfaces, can be used to represent and compare faces based on their linear combinations. Face identification is performed by computing the Euclidean distance between eigenface representations and applying classification algorithms such as k-Nearest Neighbors. Additionally, the Eigenfaces algorithm can be adapted to compare and identify objects other than faces, making it a great technique for various identification tasks.

In our implementation we choose 35 components, meaning each image will be decomposed into 35 key features (eigenvectors) that can then be compared to each other using Euclidean distance. Hence our 440 images are turned into a 440x35 matrix.

4.2 LDA - Fisherfaces

We choose to investigate this technique as PCA might not be the best for classification problems, which is what our problem is. Linear Discriminant Analysis finds a subspace that maps sample vectors from the same class in a single spot of the feature representation and those of different classes as far apart from each other as possible.

In our implementation we use 25 components, meaning each image will be decomposed into 25 key features (fisherfaces) that can then be compared to each other using Euclidean distance. Hence our 440 images are turned into a 440x25 matrix.

4.3 LBP - Local Binary Patterns

Local Binary Patterns, or LBPs for short, are a texture descriptor that compute a local representation of texture. This local representation is constructed by comparing each pixel with its surrounding neighborhood of pixels.

In our implementation, we will be using the Chi-Square metric to compute the distances between the texture descriptors. Hence our 440 images are turned into a 440x490 matrix.

4.4 DL using Siamese Network

In 2014, researchers at Facebook developed DeepFace, a deep learning based approach to feature extraction. In their original paper, they also tested a siamese network DNN architecture for generating embedded vector representations. This consists of two copies of the same CNN (sharing their weights) that are applied to pairs of images. During training the distance between the embedded representations of the same individual is minimized, and the distances between embedded representations of different individuals is maximized.

In our implementation of the siamese network, we will be using a simpler architecture - a shallow CNN model that is trained with contrastive loss. This subnetwork is copied twice and the output of both siamese copies is then passed onto a vector Euclidean distance (ED) calculation layer. The

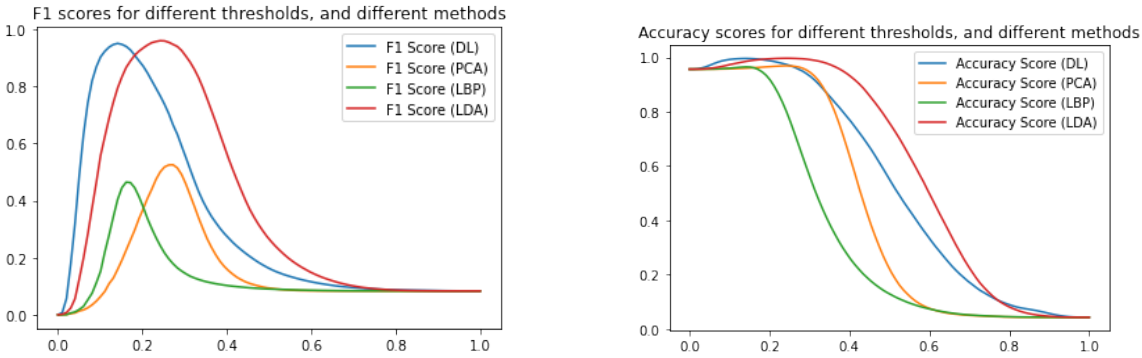


Figure 1: F1 Scores and Accuracy Scores for 4 methods

network gets input pairs of imposter or genuine images with the associated label (genuine = 0, imposter = 1). The figure shows the architecture of the network we used. Furthermore, the model is trained on 10 epochs and optimized using the Adam optimizer. The loss function we are optimizing is the contrastive loss function, but later on, we test another loss function and compare its results.

5 Evaluation

To apply any evaluation metric, we first compute the distance-based pair-wise matching scores for each image - for all 4 feature extraction methods used. For this, we use the built-in *scipy.spatial.distance* function *pdist*. This respectively gives us a vector of $\binom{440}{2}$ numbers.

5.1 F1 and Accuracy Scores

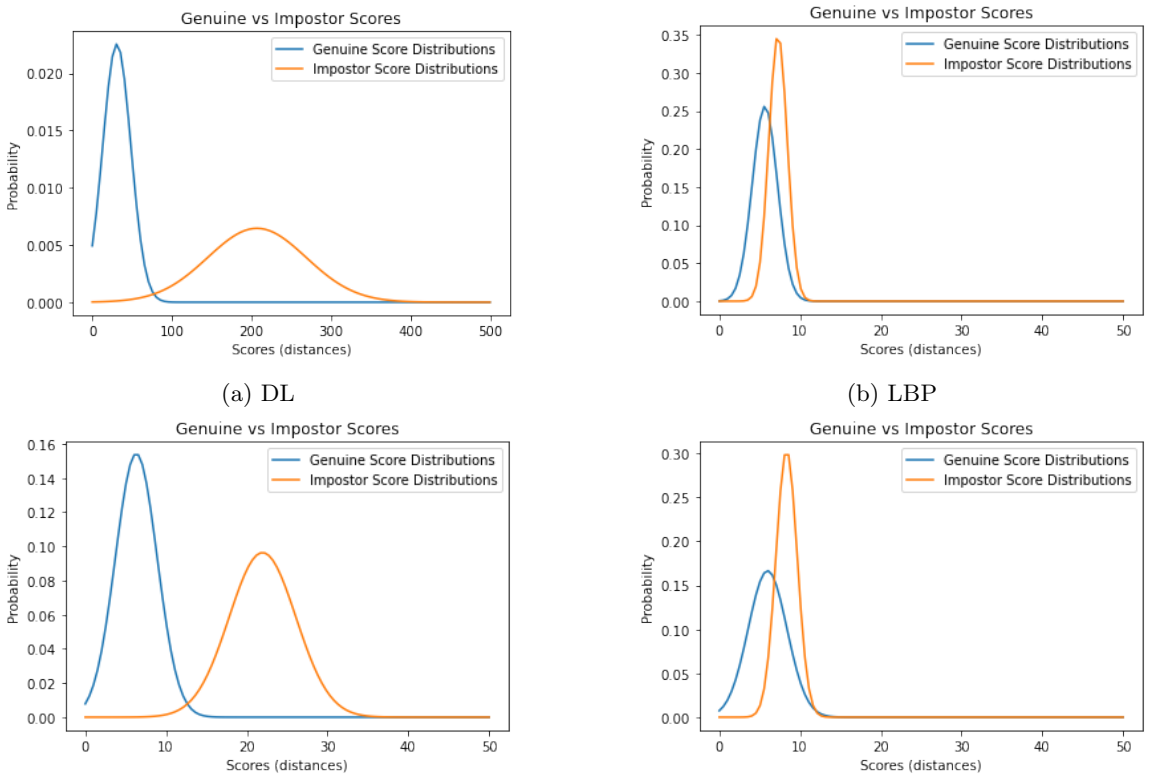
In Figure 1, we can see the F1 scores for different thresholds and for the 4 different methods. F1 scores are a great metric in determining how well our system classifies the data and it can be interpreted as the model's balanced ability to both capture positive cases (recall) and be accurate with the cases it does capture (precision).

We can see that for the DL and LDA methods perform very good as they are almost able to act as a perfect classifier reaching a maximum score of 95% and 96% respectively, for optimal thresholds 0.14 and 0.24 respectively. The other two methods, PCA and LBP, did not perform as well, reaching a maximum score of 52% and 46% respectively, for optimal thresholds 0.27 and 0.16 respectively. This shows that our system works as a great classifier with the first two methods.

Next, we look at the Acccuracy Scores in Figure 7, we can see that the methods reach and almost perfect accuracy with DL and LDA taking the lead reaching a maximum of 100% for thresholds of 0.14 and 0.24 respectively. Whereas PCA and LBP reach a somewhat lower maximum for thresholds 0.24 and 0.14 respectively. This shows that our classifier is very accurate and has correct predictions.

5.2 Genuine and Impostor Scores

Figure 2 shows us the genuine and impostor score distributions for the 4 different methods. The area under each curve integrates to 1, meaning our metric takes into account all of the scores we calculated. As we can see, in all examples, the blue Genuine Score Distributions sit closely towards the lower end of the scores, meaning that the distances between individuals with genuine scores are



(a) DL

(b) LBP

(c) LDA

(d) PCA

Figure 2: Graph showing different Genuine and Impostor score distributions for 4 methods.

smaller, which makes sense as their features are more similar. The Impostor scores are generally more spread out (larger deviation) in better classifiers. We can see that the overlapping region between the genuine and impostor scores in LBP and PCA is much larger than in the other two methods. This is due to the distance scores not being different enough between classes, hence also the smaller scores in the LBP, LDA, and PCA methods in general - but also specifically in LBP and PCA, we can see that the overlap where false rejects and false accepts lie is big, meaning there are many of those and our classifiers are bad. We can clearly see that our DL and LDA classifiers perform the best from this metric, with DL having a larger distinction between similar images (lower score) and different images (higher score).

5.3 EER, PR Curve, AUC, and average precision

Figure 3 shows us the different metrics we have calculated. Although the Area Under Curve for the PR Curve or ROC curve gives us a general idea of how well our classifier is doing, it is not perfect since it is a summary measure, this is why we also take into account the other graphs we have plotted. Looking only at the average precision and AUC, we can see that the DL and LDA methods dominate in comparison to the PCA and LBP methods. This distinction can clearly be seen in the graphs as well. E.g. the precision-recall curve shows us that our PCA and LBP classifiers fail when it comes to having the ability to identify only the relevant data points (precision) and also fail when

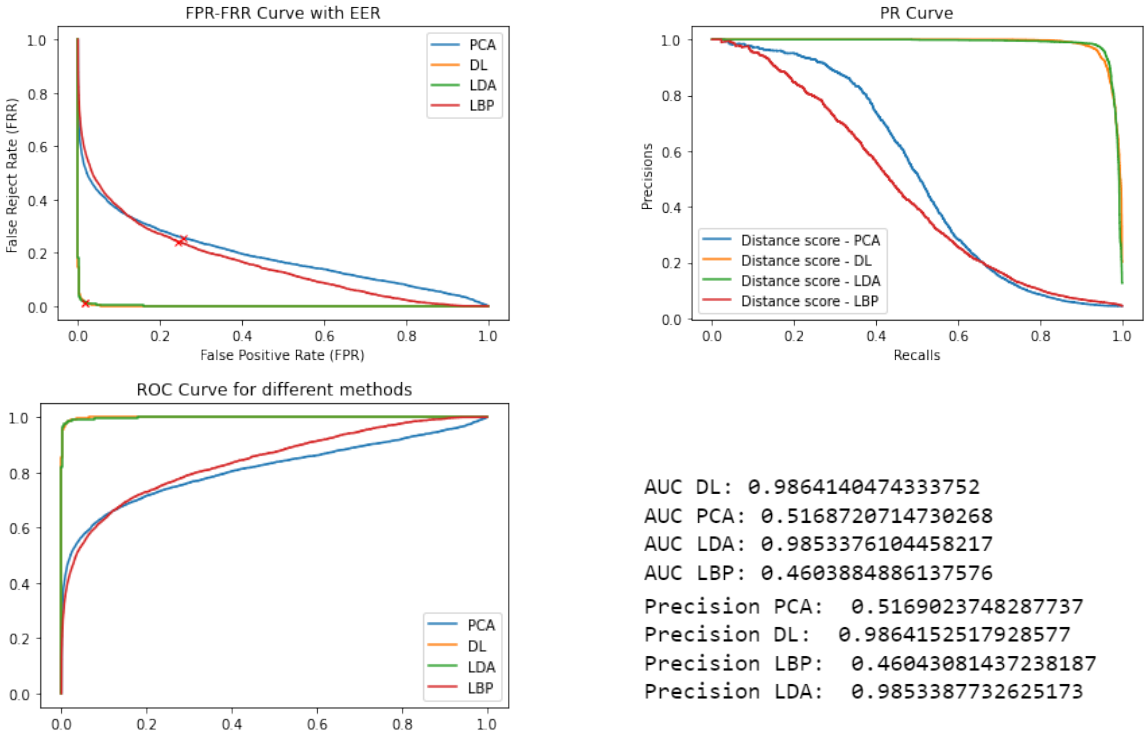


Figure 3: Graphs showing the full-on verification assessment.

it comes to having the ability to find all the relevant cases within the dataset (recall). The closer the knee is to (1.0, 1.0) the better the classifier is since it means that it identified the relevant data points and found all the relevant cases.

From the FPR-FRR Curve with EER, we can see also that our Equal Error Rate point lies close to (0.0, 0.0), meaning that there are no false rejects and no false positives, which we also saw in the distribution graph earlier. The PCA and LBP methods fail to do this.

The receiver operating characteristic curve shows us a similar result, where the sensitivity of the system is maximized and the probability of a false alarm is minimized in our two best methods.

5.4 CMC Curve

Lastly, when we are validating the systems in an identification scenario, we often plot the Cumulative Matching Curve, we can see this in Figure 4. In this multi-shot scenario, we choose to calculate the probability of a certain rank by taking one image from each class and dividing it by the total number of images in that class. I.e. for image 1, we would divide the rank-1 performance by 20, as there are 21 total images in class 1. After doing this for all images, we get a constantly increasing CMC Curve that shows us the quality of our classification system. We can see that already at rank-20 our system performs almost a perfect classification using the DL and LDA methods, while the LBP and PCA methods never reach a high enough classification (the probability of classifying stagnates around 80%). There are other methods that could have been used for multi-shot scenarios, but I chose this as it seems the most logical for this sort of metric.

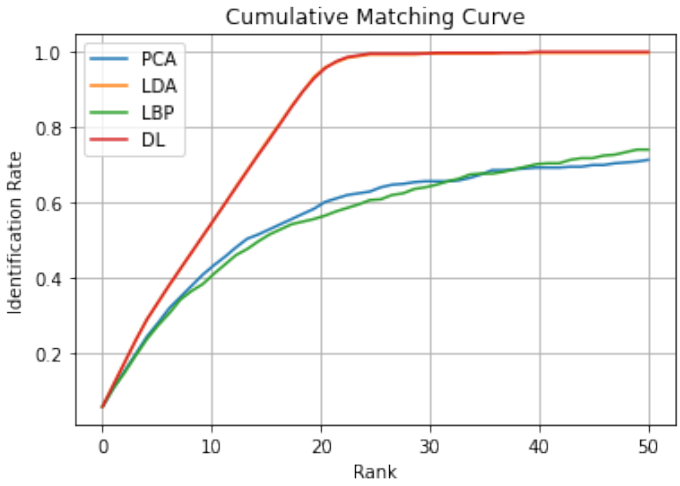


Figure 4: The CMC Curve for 4 different methods.

Lastly, due to the nature of this CMC Curve, the rank-1 performance doesn't give us a distinction between the methods used, i.e. all four methods classify the image with the shortest distance (smallest matching score) as the image from that same class. So the rank-1 performance for all four methods (since only 26 images were chosen for the 26 classes) is 0.059 or 26/440.

6 Optional Tasks

6.1 Optional task 1 - evaluating two other datasets

The four methods for feature extraction were examined on other datasets, both smaller and bigger. The AT&T Facedatabase, also referred to as Olivetti faces, contains ten different images of each of 40 distinct subjects. It is generally considered a fairly easy database with all pictures having a homogeneous background.

We can see in figure 5 that the comparing two methods that usually give very different results (LDA against PCA), we are given a clear distinction where LDA is much better in the F1 score and Accuracy score than PCA. We can also see that in comparison to the Caltech dataset F1 score shown in Figure 1, our system generally performs similarly.

On the other hand, a much harder facial dataset is The Labeled Faces in the Wild (lfw). This dataset is a collection of more than 13000 JPEG pictures of famous people collected over the internet. For testing purposes, we limit the number of classes to 7, or rather each class has to have at least 70 images of the person. So the number of samples is 1288, 3 times larger than our initial dataset. Here we can see from the figure that PCA completely fails to grasp the large number of calculations needed while LDA still shows a fairly good classification.

6.2 Optional task 2 - comparing two different face detectors

For this task, we choose to compare two face detectors that are different than in the original code. The original code uses the HAAR-cascade method, explained previously.

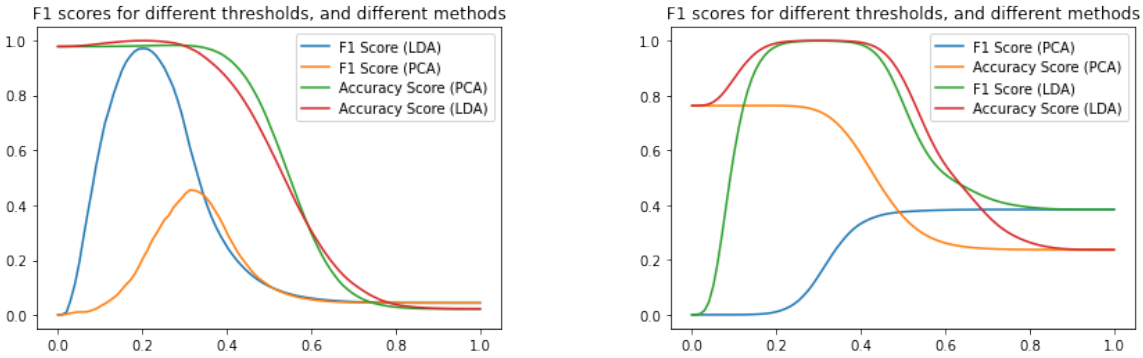


Figure 5: The F1 and Accuracy Scores for two different methods for the Olivetti dataset.

We choose to compare a more traditional approach of HOG and a newer approach of CNN's with MMOD. Both of which are available through the dlib library.

Histogram of Oriented Gradients is our first method. The idea behind HOG is to extract features into a vector, and feed it into a classification algorithm like a Support Vector Machine for example that will assess whether a face (or any object you train it to recognize actually) is present in a region or not.

The second method uses a Convolutional Neural Network and implements Max-Margin Object Detection which enhances the ability of the neural network to "see" faces.

We can see from Figure 6 that, although the CNN face detector is non-deterministic, we can see that it sometimes fails in correctly finding the borders of a face, but the HOG method finds the wrong borders more often. We use image 1 to illustrate this where the black border is the HOG method and the white border is the CNN-MMOD method.

We can then compare this to the ground truth that can be found in ImageData.mat and see how far off we are. This is done only for image 1 but can easily be extended to cover all 440 images from our dataset. For the evaluation we use Intersection over Union, which is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. For the first image, this turns out to be 0,61 for the HOG face detector while it is a high 0,93 for the MMOD face detector. This shows how well the CNN-based face detectors work.

6.3 Optional task 4 - evaluating a different loss function in the Siamese network

We can see from Figure 7 that there is a distinct difference between the loss function used in the original code - Contrastive Loss - and the loss function that I tested - Binary Cross Entropy Loss. The Contrastive Loss Function performs better on both accounts and improves the classifier drastically from around 62% F1 Score at the BCE Loss peak to around 83% for the Contrastive loss.

It's important to note that other loss functions could have an even better score than Contrastive Loss such as the Triplet Loss function which tries to move distance between the anchor image vector and the negative image vector further away and the distance between the anchor image vector and the positive image vector closer. Meaning that similar images will have a shorter distance.

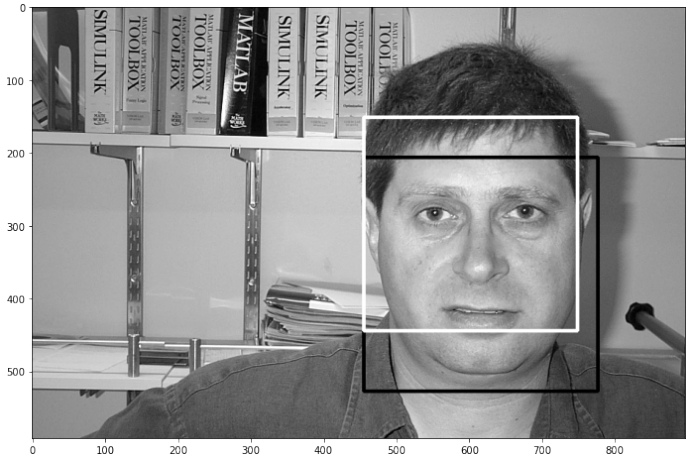


Figure 6: The bounding boxes between the HOG (black) and CNN-MMOD (white) methods.

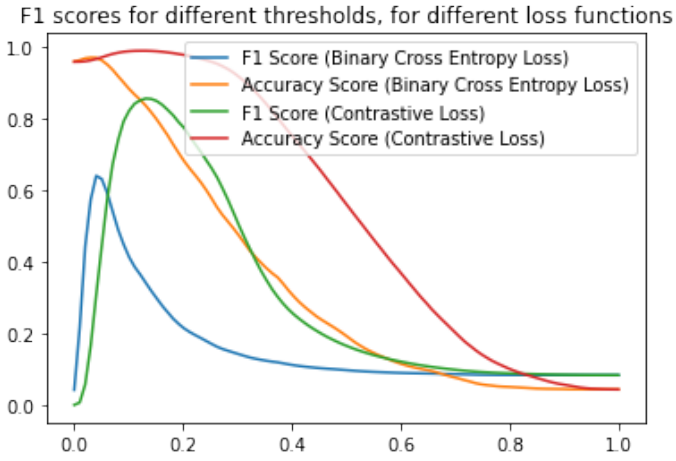


Figure 7: The F1 and Accuracy Scores for two different loss functions