

Biometrics System Concepts - Assignment 3 - KU Leuven

Deniz Soysal - r0875700

May 28, 2022

1 Introduction

In this assignment, we will implement techniques for Facial Recognition, in a verification and a identification scenario. The dataset we will use is the CALTECH Faces dataset. This assignment is divided into 2 parts : in the first part, we will compare 4 techniques : Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Local Binary Patterns (LBP) and Deep Learning (DL) for Facial Recognition. We will compare them for verification and for identification. In the second part, we will first implement and compare face detection methods, and implement a different deep learning model.

The additional tasks chosen are therefore : "2. Implement 2 different face detectors and compare all techniques to the ground truth bounding boxes provided in CalTechFacesDirs/ImageData.mat. Look up the literature for methods to compare different face detectors. (1pt.)" and "6. Implement a different deep learning model (2pt.)"

2 Mandatory tasks : Comparison of PCA, LDA, LBP and DL

We consider the CALTECH Faces dataset. The dataset is composed of 445 images, corresponding to 26 people. The size of the images is 592x896 pixels. Note that the dataset is unbalanced, as depicted in Figure 1 : some person has more images than others. We will interpret the effect of this distribution later in the report.

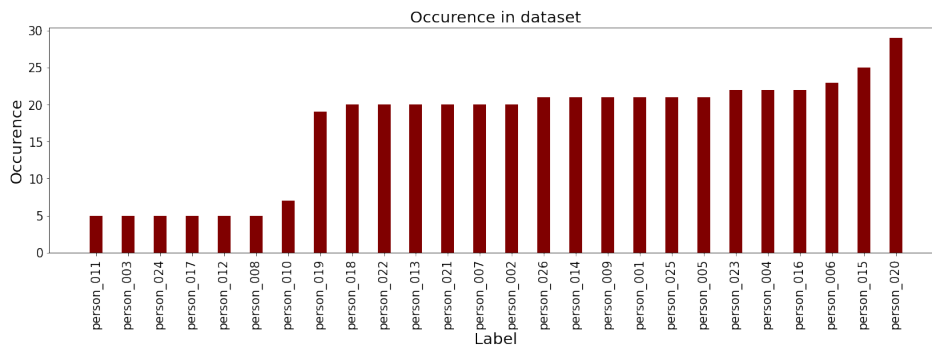


Figure 1: Similarity Matrix for PCA

To detect the faces, we use a HAAR detector. The detected faces are of size 47x47 pixels. We use PCA, LDA, LBP and DL to extract features from the images. Before comparing the results of the techniques, let's define briefly the idea of the different techniques.

PCA The idea of PCA is to extract the direction of maximum variance of a distribution, by computing the eigen vectors of the covariance matrix of the distribution. The largest eigen value correspond to the first principal component. PCA is often used to do dimensionality reduction, but it is also used in Face Recognition to extract the principal components of faces, called *eigen faces*. These eigen faces will represent facial variation, and a linear

combination of these will represent a face. The first principal component (the first eigen face) will capture global features, while more lower eigen faces will capture local/detailed features. The drawbacks of PCA is that it does not distinguish between shape and appearance. Moreover, PCA does not take into account class information (e.g. that two images belong to the same class).

LDA LDA, compared to PCA, takes into account class information. The idea of LDA is to find the direction that separates the best two classes. So, we find a projected space in such a way that the interclass variation is maximized and the intraclass variation is minimized. While LDA, a supervised technique, find the most discriminating features, PCA, a unsupervised technique, find the most expressive features. Usually, on bigger balanced dataset, LDA performs better.

LBP Instead of looking to global features, we can also consider the image locally. We can divide the face onto regions, and for each of these region, we can extract features. One way of doing so is Local Binary Patterns or LBP, which compare for each point, the value of the point to those of the neighboring points.

DL Finally, Deep Learning can also be used to extract the features. The advantage of DL is that the network extract itself the features that maximize the user-defined objective function. Convolutional Neural Networks have demonstrated great performance in extracting the features from images. Moreover, in Face Recognition, Siameze networks are often used, which allows to build embeddings of images that maximize the distance between embeddings of different classes and minimize the distance between embeddings of the same class.

2.1 Question 1 : Compute distance-based pair-wise matching scores

Given the distance metrics and the feature representation for each of the technique, we compute a similarity matrix between the images (so the similarity matrix is of size number of images x number of images). The result for PCA is shown at Figure 2. As we can see, we have 0 values in the diagonal of the matrix. This is normal because in the diagonal of the matrix, we compute the distance¹ between the same image. However, we see that the distance between image 0 and image 1 is 5.397361, even if these images are from the same person, i.e. Person 001. This shows that we have **intra**class variability. The similarity matrix for the other techniques are displayed in the notebook.

	0	1	2	3	4	5	6	7	8	9	...	430	431	432	433	434
0	0.000000	0.394814	0.529357	0.243951	0.289165	0.377613	0.457520	0.483310	0.423979	0.462058	...	0.602729	0.565200	0.491742	0.471834	0.482004
1	0.413937	0.000000	0.492699	0.329549	0.374593	0.497905	0.443969	0.461151	0.343246	0.497455	...	0.649151	0.518039	0.439006	0.470836	0.467805
2	0.520474	0.462052	0.000000	0.416498	0.583197	0.571650	0.468009	0.599183	0.403717	0.665968	...	0.670245	0.555389	0.491023	0.580533	0.515317
3	0.263386	0.339367	0.457355	0.000000	0.358408	0.468218	0.445246	0.422727	0.396288	0.441049	...	0.644986	0.512789	0.471615	0.457328	0.451303
4	0.284986	0.352124	0.584578	0.327164	0.000000	0.327385	0.446513	0.440657	0.389465	0.416757	...	0.599412	0.574885	0.523934	0.519014	0.540293
...
435	0.450568	0.508217	0.676909	0.432403	0.494681	0.508784	0.591688	0.592345	0.577620	0.591638	...	0.614343	0.514387	0.471766	0.474892	0.406321
436	0.373496	0.475795	0.611872	0.387776	0.423288	0.464032	0.511406	0.515377	0.515330	0.481526	...	0.616658	0.463739	0.423604	0.448008	0.376601
437	0.475879	0.436333	0.579166	0.423658	0.497093	0.528696	0.513628	0.553362	0.511782	0.563599	...	0.704875	0.409891	0.267597	0.249651	0.245313
438	0.558996	0.482728	0.568845	0.480278	0.579525	0.579874	0.482522	0.606154	0.527957	0.612743	...	0.715398	0.381877	0.328457	0.318934	0.277220
439	0.488374	0.401567	0.545275	0.413426	0.502505	0.532563	0.580130	0.561910	0.520572	0.621927	...	0.670301	0.364390	0.270928	0.419825	0.220135

440 rows × 440 columns

Figure 2: Similarity Matrix for PCA

2.2 Question 2 : Compute F1 and accuracy scores for variable thresholds

After having constructed the similarity matrices, we compute the F1 and accuracy scores for different thresholds on the distance values between the images. The values are averaged over all classes. Remember that **accuracy** is

¹Note that formally the matrix is a *distance* matrix, but $\text{similarity} = 1/\text{distance}$

simply the ratio between correct predictions and all predictions. In biometric systems, using accuracy as a metric is not advised : indeed, it does not take into account the unbalance of the dataset. Remember the class distribution, shown at Figure 1. We are in a verification scenario, so as an example, consider that we want to verify if the input image is from person_011. By looking at Figure 1, we can see that we have **only** 5 images for person_011 in a dataset of 440 images. Therefore, for this particular case, a system that simply always predict *image is not person_011* will have a $395/440 = 0.8977$ success rate ! This can be seen on the Figure 3 : by highering the threshold, the system will always reject all input samples and the accuracy reaches a certain limit, corresponding to the average accuracy over all classes when all input images are rejected. On the other hand, **F1-score** is a metric that combines precision and recall. Precision computes the ratio of correct positive predictions over all positive predictions ; Recall computes the ratio of correct positive prediction over all positive samples in the dataset. F1-score is a harmonic mean of precision and recall : both of them needs to be high to have a high F1-Score. Now, we can see, at Figure 3, that for high thresholds, the F1-score is equal to 0. Therefore, when rejecting all input samples, the F1-score is equal to 0, which shows that F1-score is a more robust metric than accuracy. As we can see at Figure 3 and Table 1, all the methods have similar maximum accuracies values. On the other hand, DL and LDA are the best performing methods in terms of F1-score while PCA and LBP are performing poorly according to this metric.

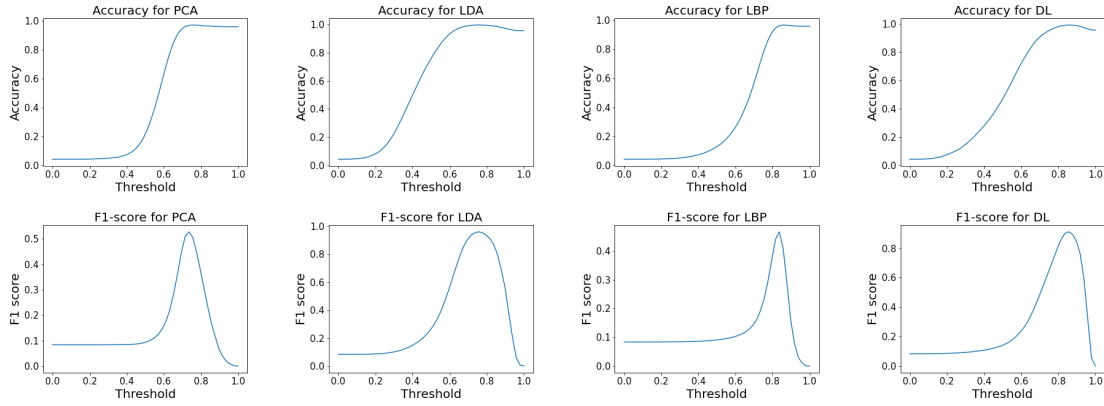


Figure 3: Accuracies and F1-scores for each method

Method	Opt Thresh fl	Max fl	Opt Thresh acc	Max acc
PCA	0.735	0.528	0.755	0.9690
LDA	0.755	0.96	0.755	0.997
LBP	0.837	0.466	0.857	0.965
DL	0.857	0.911	0.857	0.993

Table 1: Optimal thresholds for maximum distance-based f1-score and accuracy

2.3 Question 3 : Plot genuine and impostor scores

At Figure 3 and Table 1, we have seen that DL and LDA were performing much better in terms of distance-based F1-score compared to the other two techniques. To view this from a different point of view, let's plot the genuine and impostor scores distributions. The genuine distribution is the probability distribution of the similarity scores between images of the same class, while the impostor distribution is the probability distribution of similarity scores between images of different classes. Based on the given similarity scores and a user-defined threshold η , the system predict if the input is genuine or an impostor. In a *good* system, the overlap between these distributions is small. Indeed, if the distribution were completely separated, choosing a threshold η that perfectly separates the distribution would be possible. However, in real dataset, the distributions are often overlapping, as it is in our case, depicted

at Figure 4. This leads to a non-zero False Rejection Rate (correspond to area of the genuine distribution below the threshold η) and a non-zero False Acceptance Rate (correspond to area of the impostor distribution over the threshold η). At Figure 4, we can observe that the overlapping areas for LDA and DL are much smaller compared to PCA and LBP. This means that LDA and DL will allow to have a system with a lower FAR and a lower FRR rate than PCA and LBP. These results are coherent with the results at Figure 3 and Table 1. We see clearly that **LDA**, by taking into account class information, is able to find a projected space in such a way that the interclass variation is maximized and the intraclass variation is minimized, compared to PCA. We also observe that **DL**, by minimizing the *contrastive loss*, is able to reach a similar performance.

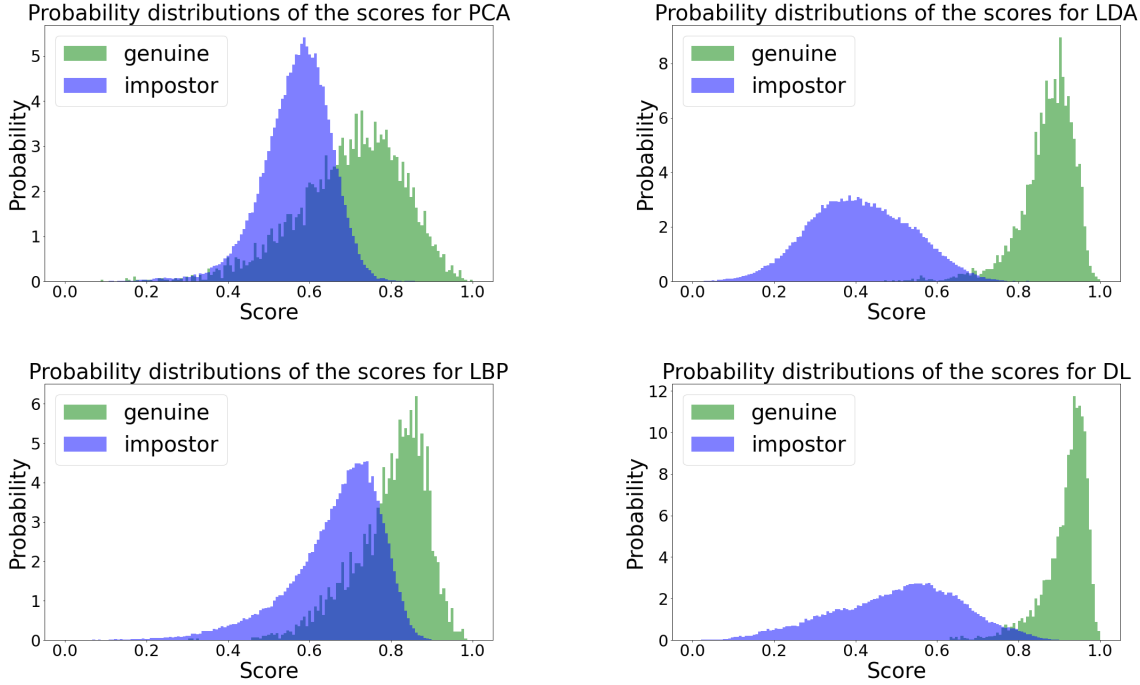


Figure 4: Genuine and impostor distributions for each method

2.4 Question 4 : Perform a full-on verification assessment based on the scores obtained. Interpret the results.

Besides looking at F1-scores and the genuine-impostor distributions, other ways to benchmark a verification system is the ROC curve, the Precision-Recall curve and the EER.

The **ROC** curve plots the True Acceptance Rate, TAR (or $1 - \text{FRR}$) against the False Acceptance Rate, FAR. One desires a ROC curve that increase quickly. Indeed, this will allow to choose an operating point, by selecting a threshold η , with a high TAR and a low FRR. The ROC curve for each method is shown at Figure 5, with the area under curve presented at Table 2. We can clearly see again that LDA and DL outperforms PCA and LBP. Another way to assess the performance of the system is to look at the **Precision-Recall** curve, or PR curve, which plots the Precision against the Recall. As we have discussed when analyzing the F1-score, precision compute the ratio of correct positive predictions over all positive predictions while recall compute the ratio of correct positive prediction over all positive samples in the dataset. One desires to have the highest precision and the highest recall possible, therefore a curve that stays up for a long range of precision (which will allow to select an operating point with a high recall and a high precision). At Figure 6a, we can again see that LDA and DL outperforms the other two methods, while LDA is slightly better than DL. The Average Precision values are 0.517 for PCA, 0.985 for LDA, 0.46 for LBP and 0.968 for DL. Finally, the **Equal Error Rate (EER)** is the point where $\text{FAR} = \text{FRR}$. A lower

Method	ROC AUC
PCA	0.811
LDA	0.998
LBP	0.840
DL	0.997

Table 2: ROC AUC for each method

Method	FAR = FRR	Threshold
PCA	0.256	0.623
LDA	0.016	0.670
LBP	0.243	0.746
DL	0.023	0.780

Table 3: EER for each method and corresponding threshold

EER means a more performing system at the particular point where $FAR = FRR$. But is it really a good operating point ? In fact, it depends on the application : We might want to have a really secure system (so selecting a high threshold to have a low FAR), or a more convenient system (so selecting a low threshold to have a low FRR). In practice, EER is often used as operating point where the cost of a False Acceptance and the cost of a False Rejection is similar. At Figure 6b and Table 3, we can observe the EER for each method. Again, LDA and DL outperforms the other two methods by allowing a lower EER. LDA is the one that allows the lower EER.

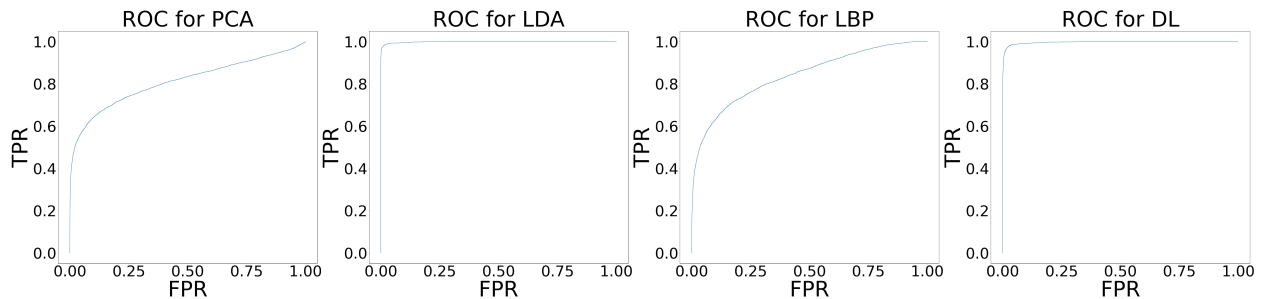
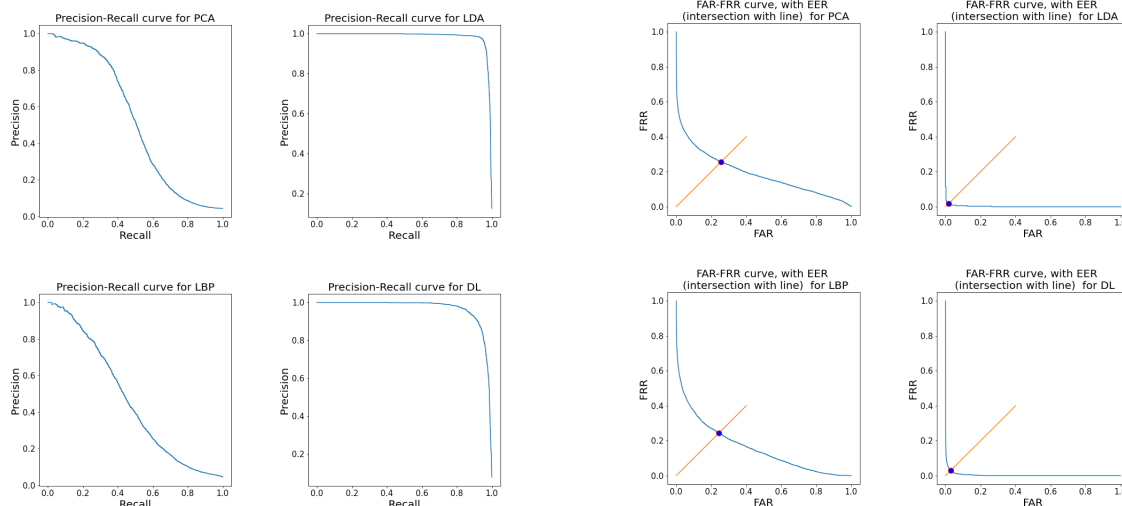


Figure 5: ROC curve for each method



(a) Precision-Recall curve

(b) FRR-FAR curve and corresponding EER for each method

Figure 6: Precision-Recall curve and FAR-FRR curve for each method

2.5 Question 5 : Validate the systems in an identification scenario.

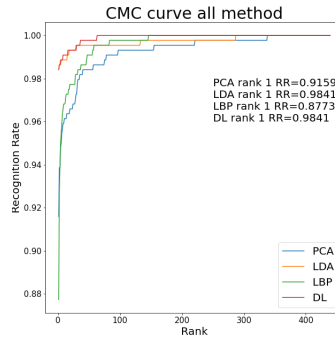


Figure 7: CMC curves for each method

The **CMC curve** plots the probability that a correct identification is returned with the top "t" ranked matching scores. So, for example, if we have a probability of 85% at a rank of 5, this means that for 85% of the cases, we will find the correct identity in the top 5 predictions. To compute the CMC curve in our dataset, we use two **for loops** on the similarity matrix : for each identity "i", we first sort the other identities based on the similarity scores. Then, for an increasing rank "j", we look if the identity "i" is in the "j" highest similarity scores. If so, the identification is successful at this rank. If not, it is unsuccessful. The resulting CMC curves and the Rank 1 recognition rate for each method are depicted at Figure 7. One wants the highest rank-1 recognition rate possible. We can observe that the highest rank-1 recognition rate is reached following the DL and the LDA method. Moreover, the DL method has a higher recognition rate at each rank compared to all other methods.

3 Additional tasks

3.1 Implement 2 different face detectors and compare all techniques to the ground truth bounding boxes

MTCNN The first technique we will implement is Multi-task Cascaded Convolutional Networks, or MTCNN. Authors² have developed this technique based on the idea that face detection and face alignment are two correlated process. If the face is detected at a particular position, the way we are going to align it depends on the face detection. The authors have exploited this idea, by defining a cascade architecture with 3 CNNs that predict face and landmark positions in a coarse-to-fine manner. Before inputting the image into the 3 CNNs, the image is resized into different scales using image pyramids. Then, the image goes through 3 stages : The **first stage** is the Proposal network, where the network is Fully Convolutional (without dense layers). The purpose of this stage is to detect faces in a coarse manner: it has a very low threshold, so it will give several candidate windows (areas where a face could potentially be detected). The False Positive Rate is high after this stage. The **second stage** is the Refine Network, where the network is a Convolutional Neural Network (with dense layers). We will feed to this network all the candidate windows. In this network, we do refinement by reducing the number of candidate, merging overlapping candidate, etc. The **third stage** is the Refine Network, where we will perform more refinement and finally output the bounding box, the facial landmark localization, etc. At the end, we output 5 facial landmark positions. At **each stage**, NMS (Non-Maximum Suppression) is used to merge highly overlapped candidate window.

Dlib : HOG + SVM The other face detector we will implement is from the **dlib library**. The detector proposed by dlib is a histogram of oriented gradients (HOG) and linear SVM based face detection system. Basically, HOG will extract the features from the image, while SVM will classify them. At the end, we will be able to find which features correspond to the face, and from that determine the area where the face is located.

²See paper "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional, by Kaipeng Zhang & all"

3.1.1 The results

After implementing the MTCNN detector and the dlib detector, we have compared the two techniques. In fact, in our dataset, the faces are pretty easy to detect : all the pictures are frontal images, with high resolution, no occlusion, ... So, comparing the techniques based on how many faces is detected will be difficult, because both techniques detect the faces in those conditions with ease. However, it is important to note that we use here the function `dlib.get_frontal_face_detector` which is a HOG + linear SVM based techniques. It is fast and efficient but by nature of how the Histogram of Oriented Gradients (HOG) descriptor works, it is not invariant to changes in rotation and viewing angle. This is a drawback of this technique³, but in our dataset, we expect dlib to work pretty well because the variance in rotation, viewing angle etc are limited in our dataset. Finally, we have decided to compare the MTCNN detector with the dlib detector based on two metrics : 1) Time taken to detect a single face and 2) Intersection over Union (IoU) between bounding boxes for the faces detected by the method and ground truth bounding boxes. The idea of IoU is presented at Figure 8. If the bounding boxes are perfectly overlapping, the area of overlap will be equal to the area of union, and therefore $IoU = 1$. In other cases, $IoU < 1$.

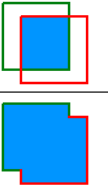
$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$


Figure 8: IoU between two bounding box (from A Survey on Performance Metrics for Object-Detection Algorithms, by Rafael Padilla)

The results of the comparison are presented in Table 4. As expected, we can see that MTCNN has a higher IoU, but is 12,25 times slower than dlib to detect one face. Even if the performance of dlib is lower, the difference is limited. This can be explained because, as stated before, the faces in the dataset are easy to detect. On a more challenging dataset, we expect the difference in terms of IoU between dlib and MTCNN to be bigger.

Method	IoU	Time taken per face [s]
MTCNN	0.52	0.525
dlib (HOG + SVM)	0.43	0.045

Table 4: IoU and time taken to detect one face for MTCNN and dlib

At Figure 9, we can observe the ground truth, the dlib and the mtcnn bounding boxes for an image. We can see that the ground truth consider the face as a whole : the hair, ears etc are also inside the bounding box. Dlib only consider a smaller portion of the face and cut out the hair, the ears and a portion of the chin. MTCNN is closer to the ground truth.

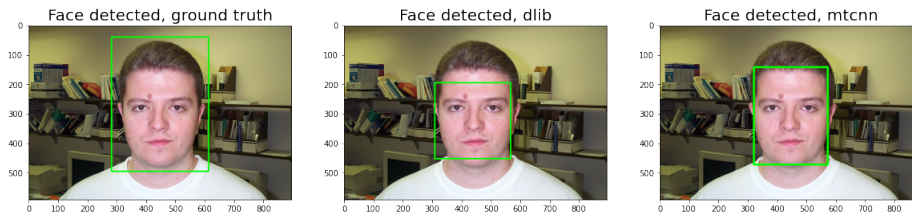


Figure 9: Faces detection bounding boxees :ground truth, using dlib and using MTCNN

³Note that Dlib propose also a more robust detector, via `dlib.cnn_face_detection_model_v1`, but this technique has not been studied here

3.2 Implement a different deep learning model

Now, we will use FaceNet to create an embedding of the image. We will use a pretrained version of FaceNet on the MS-Celeb-1M dataset, by Hiroki Tanai. The model we will use, with pretrained weights, is available at his GitHub page⁴. We will not fine tune the model, because the model has already been trained on a face dataset of 1 000 000 images. What we will do however is put a classification step on top of that, using SVM or KNN, and **fit** these classifiers to our dataset. We will use the faces detected with MTCNN of the previous section, as its has proven high performance. Note that we will only consider the **identification scenario**.

FaceNet The basic idea of FaceNet is, based on input images, build an embedding of these images by maximizing the distance, in the feature space, between embeddings belonging to different classes. To do so, FaceNet use convolutional neural networks with Triplet Loss. The idea is to define three convolutional neural networks with shared weights and use as inputs, an anchor image for network A, positive images (all images of the same class as the anchor image) for network B and negative images (all images of different classes as the anchor image) for network C. Then, using triplet loss, we will train the networks to maximize the distance, in the feature space, of images belonging to different classes (maximize distance between anchor and negative images) and minimize the distance, in the feature space, of images belonging to the same class (minimize distance between anchor and positive images).

Classification step After having obtained an embedding of the image with FaceNet, we will classify the embedding using KNN and SVM. We use grid search, with 5-split cross validation to tune the classifiers. The parameters obtained are $C=1$, kernel=rbf, $\gamma = 1$ for **SVM**, and $k=3$ for **KNN**. Best performance is obtained using the SVM.



Figure 10: Pipeline of face recognition with MTCNN, FaceNet and SVM/KNN

The full pipeline of the system is presented at Figure 10. At Figure 11, we compare our system in an identification scenario with the Deep Learning model of the first part of the assignment. We observe that we are able to reach a higher rank-1 recognition rate, and that the recognition rate is higher for all the ranks.

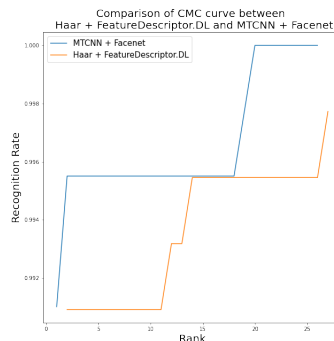


Figure 11: Comparison of CMC curve between the DL model of the first part of the assignment and our model

⁴<https://github.com/nyoki-mtl/keras-facenet>