SAPIENZA
UNIVERSITÀ DI ROMA

FACE RECOGNITION AI BASED

Course: Biometric Systems
Professor: Mari De Marsico
Student: Anxhelo Diko
Matricola: 1867167

Face Recognition

A **facial recognition system** is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from a given image with faces within a database. Recent works on this field are based on Artificial Intelligence. One of the most remarkable works on this problem is FaceNet from Google DeepMind labs that can achieve a very high accuracy rate. FaceNet is an End-to-End system so at most it does an image resize and nothing else, then input it to the AI model to extract features. In this project another approach that uses preprocessing on the face images is presented. TransferLearning is used to make use of the FaceNet model (impossible to train from scratch a model of this size and the amount of data it requires) and make a small fine tuning on some layers to adapt it to the preprocessed images. The tools used to make this project possible are: Python3.7, dllib (for face alignment), OpenCV (for preprocessing), faceNet pretrained weights and Tensorflow 2.0.

Dataset

The used dataset for this homework is labeled faces in the wild LFW which is a benchmark for recognition systems. It comes with already defined train test separation. It can be found in the following link http://vis-www.cs.umass.edu/lfw/.

Preprocessing

Face Recognition is a process that may happen everywhere, even in environments where the people may think that they are not being recognized. The recognition process nowadays is done automatically. To make it possible it is necessary to capture images of the peoples (faces) and feed them to the system and compare it to a database or a gallery. This process of recognition is highly dependent on the quality of the image that the system takes as input which is actually captured from the cameras. The chances of correct recognition depends on how good these images are. Normally the images may vary in light, noise, whether the person is looking at the camera or not, etc.. Since there are different scenarios we think that applying preprocessing may help to somehow improve the recognition rate. The preprocessing is always applied before the recognition as in the following image:

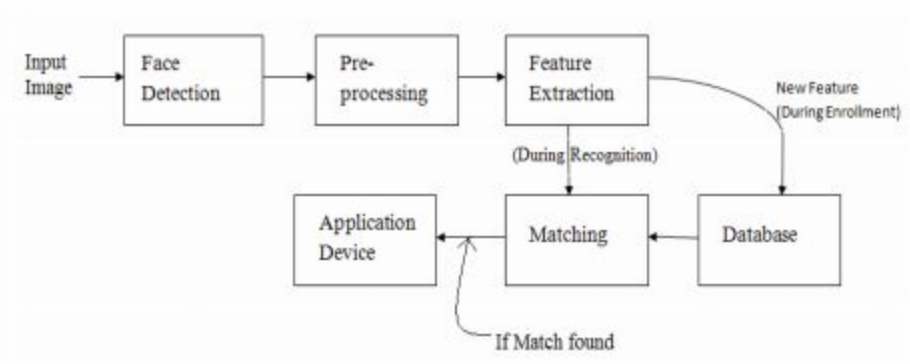


Fig. 1: Face Recognition workflow.

The preprocessing process that we try to implement is like following:

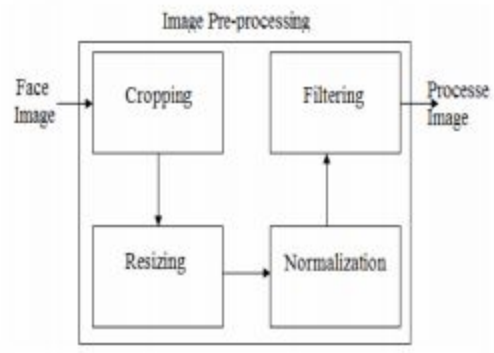


Fig. 2: Preprocessing

This is the preprocessing cycle that we took as reference but also gave our contribution for adding extra features to improve the images.

Face Cropping

Face cropping is an important task to achieve a high recognition rate. Cropping can be done using various face detection techniques. Face detection involves detecting a face from an image using complete image (image based approach) or by detecting one or more features from the image (Feature based approach) such as nose, eyes, lips etc. We are going to use a feature based approach called CascadeClassifier from OpenCV library that uses the Viola and Jones algorithm for object detection but in this case is applied to faces.

```
def crop_faces(image_array):
    """
    input: array of input images
    output: new resized images with only the wanted face
```

```
    """
    new_array = []
    for i in range(len(image_array)):
        img = image_array[i]
        gray_image = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        faces = cv.face_cascade.detectMultiScale(gray_image, 1.1, 4)
        for (x, y, w, h) in faces:
            cv.rectangle(img, (x-10, y-10), (x+w+10, y+h+10), (255, 0, 0),
2)
            crop_img = img[y-10:y+h+10, x-10:x+w+10]
            res_img = cv.resize(crop_img, (220,220),interpolation =
cv.INTER_CUBIC)
            new_array.append(res_img)

    new_array = np.array(new_array)
    return new_array
```

The above piece of code shows the use of openCV and CascadeClassifier to detect faces, crop them and produce images of size 220x220.

Image Normalization

Illumination variation is one of the important challenges in face recognition. Image with uncontrolled lighting conditions contains non-uniform contrast, i.e. the distribution of intensity/gray levels is not alike. To make these levels equal or almost equal I use histogram equalization technique. Again with the use of OpenCV as shown in the following piece of code.

```
def histogram_equalization(image):
    """
    input: image
    output: equalized image

    """
    #First change color space from rgb to HSV (cuz rgb is made of channels
and hist. eq. is focused on values intensity)
    hsv = cv.cvtColor(image, cv.COLOR_BGR2HSV)
    #we split the channels of hsv to work only with the value channel which
represents the intensity that we want to equalize
    H, S, V = cv.split(hsv)
```

```
    V = cv.equalizeHist(V)
    image = cv.merge([H, S, V])
    image = cv.cvtColor(image, cv.COLOR_HSV2BGR)
    return image
```

Image De-noising

Images often by default have Gaussian noise due to illumination variations. To somehow reduce the noise we use smoothing filters like the Gaussian filter to blur the image or denoise it.

```
def smooth_images(image_dataset):
    """
    input: image array/dataset
    output: image array/dataset with denoised or blured images
    """
    new_array = []
    for i in image_dataset:
        image = i.copy()
        new_array.append(cv.GaussianBlur(image,(5,5),0))

    new_array = np.array(new_array)
    return new_array
```

Extra (not in the paper): Face alignment

Face alignment is used to help in the face extraction (what we tried to do with face cropping). When we cropped the face the image still was not as good as we wanted to, so we implemented this face alignment using the dlib library to extract images that contains only the face of the person, dlib is based on extracting 5 face landmarks (there is also another version with 64 landmarks) needed to do the alignment.

```
def detect_and_align(image_path):
    """
    Input: The path to the image
    Output: An image resized to 224 pixels (vgg input shape is 224 224 3)
with the face cropped and aligned
    detail: The dlib algorithm for face detection was accurate 99.97% in
our dataset
```

```
    """
    img = cv.imread(image_path)
    dets = DETECTOR__(img, 1)
    faces = dlib.full_object_detections()

    if len(dets) != 0:
        for detection in dets:
            faces.append(SHAPE__(img, detection))
        images = dlib.get_face_chips(img, faces,size=160, padding=0.1)

        return images[0]
    else:
        print(image_path + 'no face \n')
        return('no image')
```

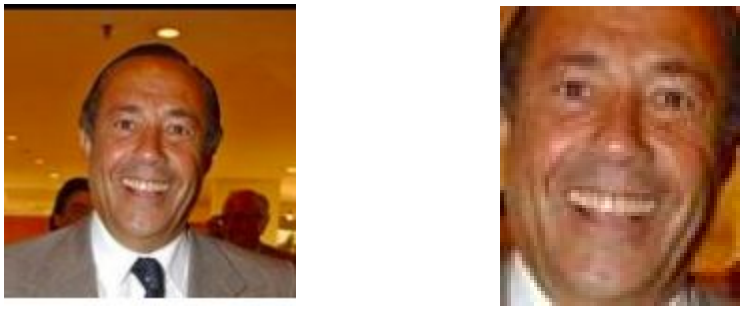


Fig. 3: Image before and after crop using alignment

Feature Extraction

For building feature recognition systems there are two main approaches: 1) Build a classifier 2) Build a feature extractor and use the features to recognize. In this work we went for the feature extractor approach and more concrete we used faceNet through transfer learning. This system outputs features of the images after it takes an image as input.

Transfer Learning

Transfer learning is a machine learning area that tries to mimic how humans use their experience from different domains to process information in a specific domain. Talking about neural networks

this means that you can use a model trained or pre trained for a certain task and then use it again for the same or for similar tasks. We exploited this characteristic of NN to use the FaceNet system built by google DeepMind. Also we tried to fine tune some layers of the network by feeding it preprocessed images but for the sake of reality that fine tuning did not affect the performance of the model at all so it is fair not to take the credits for the quality of the model on this work. The reason behind it is because the model, the original one, is trained in millions of images and for sure during this training it has seen images of all kinds and the modest dataset that we used to fine tune it did not produce any new information that it has not seen before.

FaceNet

FaceNet is an artificial intelligence system created by google for face recognition introducing a novel approach in the field of contrastive learning. The architecture used in FaceNet is called Inception_Resnet_v2 which is a combination of inception and resnet architecture. It is very huge and to train it you need a lot of computation power. Just for information, it took 40 days to Google to properly train this model. Now this model as we said uses a feature extraction approach, very important in artificial intelligence models is the loss, it defines how the model learns, or based on what criteria it improves itself. FaceNet uses a loss called Triplet Loss, which takes the feature vectors produced by the network and gives the difference that exists between images of the same person (positive pair) and images of different persons (negative pair), having this the model tries to minimize this loss. In other words it tries to produce such features that the difference between images of the same person is very small compared to the difference between images of two different persons.

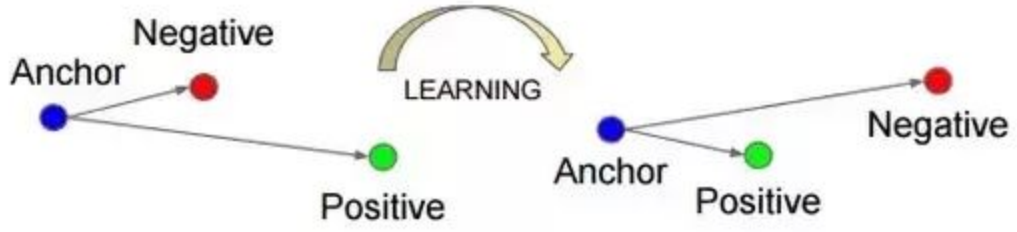


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

Fig. 4: Triplet loss

Evaluation

Evaluation is a long and very important process for a biometric system so we are going to give more importance to this section. It starts with choosing the metric and goes up to how we define if a model is good and reliable or not.

Metric

When you deal with feature embeddings the metrics can be of two types: distance or similarity. In this project I decided to go with a distance and more concrete Euclidean Distance. The reason behind this metric is because FaceNet is trained in such a way that features of the same entity have small distance (close to each other) meanwhile features of different entities have a bigger distance (not close to each other). The Euclidean distance does exactly what we want, measures how close two vectors are in the Euclidean space.

```
def findEuclideanDistance(anchor, paired):
    """
    input: 2 images, the anchor and the paired image
    output: the euclidean distance between the two images
    details: The 2 images are not the actual images but the features
predicted from CNN model
    """
    distance = anchor - paired
    distance = np.sum(np.multiply(distance, distance))
    euclidean_distance = np.sqrt(distance)
    return euclidean_distance
```

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

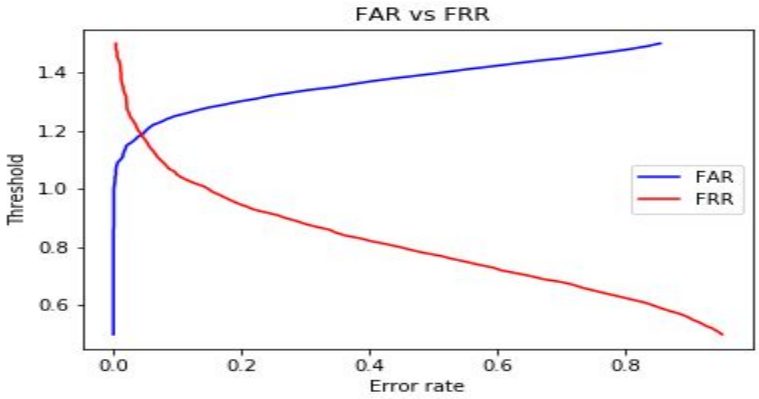
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Fig. 5: Euclidean Distance

Threshold

After we set up the metric we have to decide how we define what is a good threshold for the distance so we can separate positive pairs from negative pairs? To select a proper distance I

used a line search iterative algorithm that tried 150 different thresholds to get the one that produces FRR = FAR to have a balanced system even though in practice the decision is based on what the goal is, so it is based on the application.



As seen from the picture the FAR = FRR is reached where the threshold is 1.19 in verification for identification we are going to see different things.

Verification

Verification is the evaluation process where someone declares an identity and you have to check if the claimed identity is correct or not so is it an imposter or not. In this situation an important thing is the way to organize the data to better evaluate the model and is important that the amount of data is considerably big enough to have reliable results. In my case, I am using labeled faces in the wild dataset and the separation is done in pairs because it is used as a benchmark dataset for face recognition systems. So the separation is provided by the creators of the dataset and there are more than 1500 pairs of data, genuine pairs and fake pairs.
The process of verification is done by taking each pair provided, extracting the features using FaceNet and finding the distance. After finding the distance it is compared with the settled threshold and if it is lower than threshold the pairs are classified as genuine so the claimed identity is true otherwise the claimed identity is false and we have to do with an impostor. Some useful and important statistics to calculate during verification are FAR, FRR, GAR, GRR. Some statistics extracted from the system are as following:

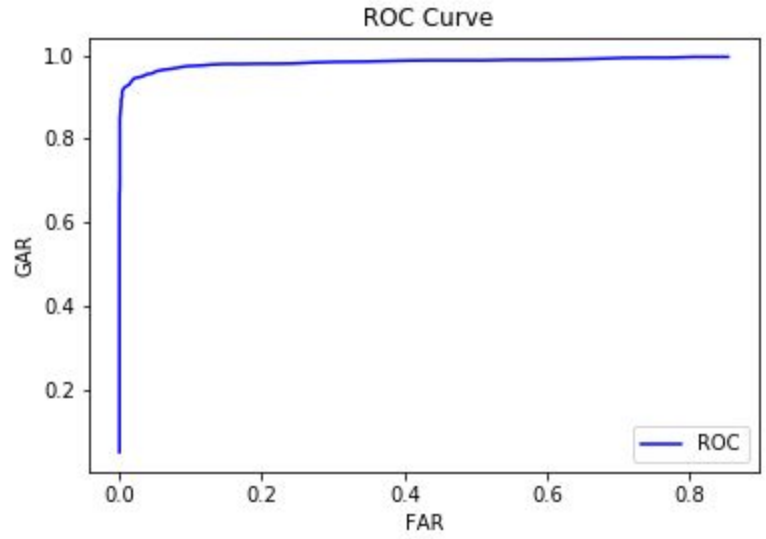


Fig. 7: ROC curve

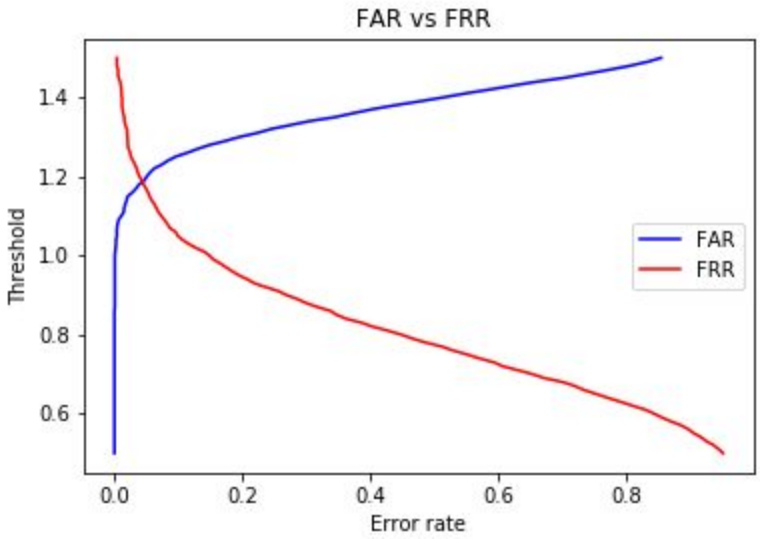


Fig. 8: FAR vs FRR for 150 thresholds taken in consideration.

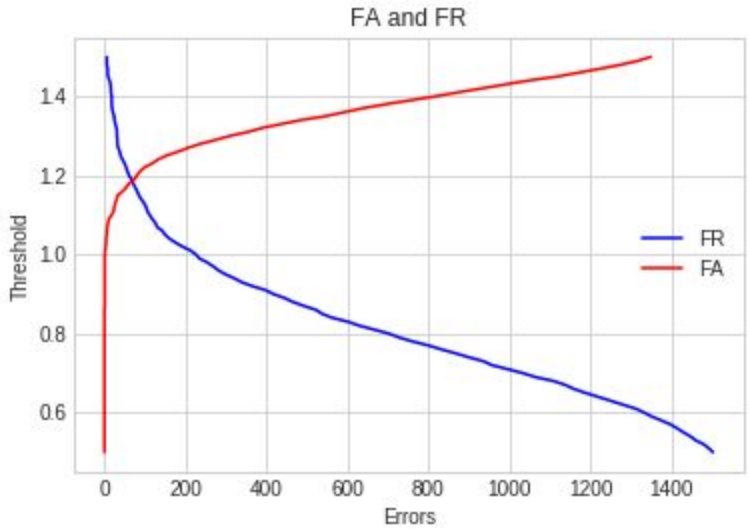


Fig. 9: False Acceptance and Rejection.

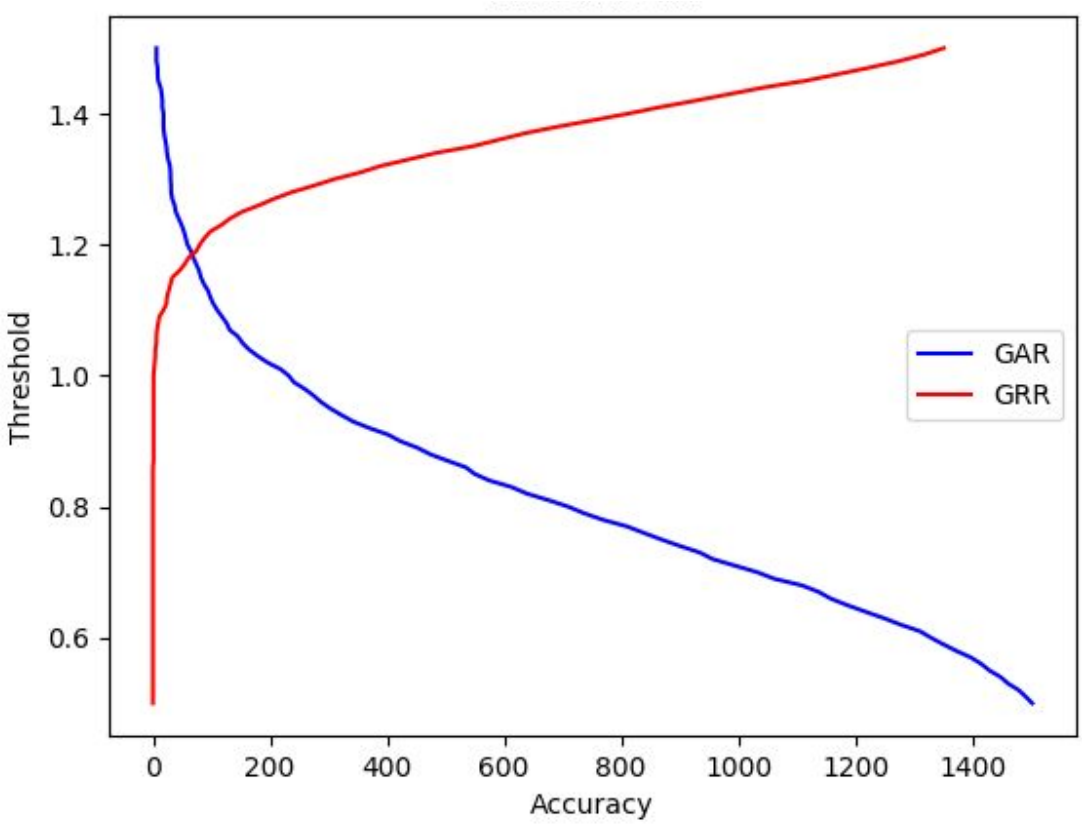
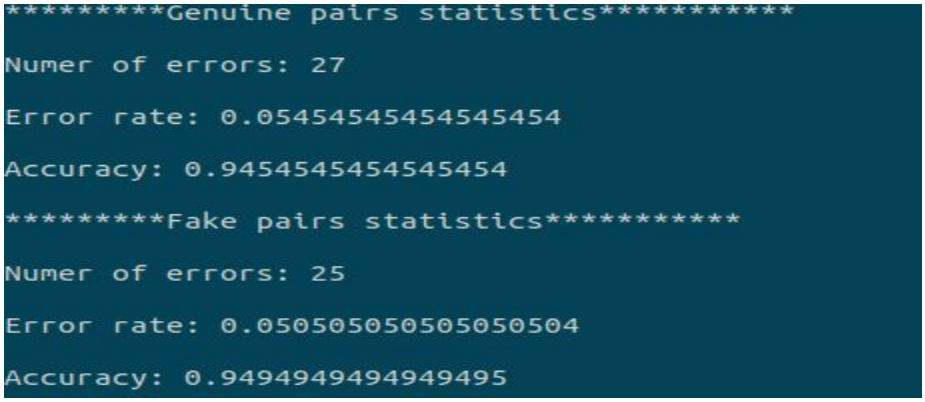


Fig. 10: Genuine Acceptance and Genuine Rejection Rate

So as we can see from the pictures that the threshold which gives us the equilibrium FRR=FAR is 1.19. FAR is ZERO for a threshold = 0.86 and with this threshold the FRR is 33.8%. FRR is ZERO for a threshold = 1.66 and with this threshold the FAR is 99.6%. When the threshold is 1.19 we have error rate for the

genuine pairs 0.052 (5.2%) which means accuracy of 94.8%, for fake pairs error rate of 0.0505 (5.05%) and accuracy of 94.95%. From the values we can see that this is a very equilibrated result in terms of false acceptance and false rejections.(Note: All evaluations are done on test pairs, or test set).



Identification

Identification is another part of recognition which is very important. Identification brings us in a different situation, so here the person does not claim any identity and we have to make a check in the gallery to see if he matches any person from the gallery or not. So basically an important step here is the creation of a gallery. In this section I used only the test set again. In the test set we have 1980 images, so almost 2000. The gallery contains 351 unique elements then for proceeding with identification test I have 807 out of gallery probes and 688 in gallery probes. In identification each probe should be compared with all the probes that are in the gallery and then it is to decide whether the person is identified (correctly or wrong) or it is rejected based on the lowest distance we find among all the members of the gallery. In identification, for evaluation i proceeded with finding top5 distances, the 5 lowest distance and then evaluating the performance on how often the system finds the correct identity in top1, top2, top3, top4 and top5 smallest distance. The Top1 statistics were also used for finding the best threshold for identification which was different from the one used in verification, as shown on the following picture here the best threshold is 0.925.

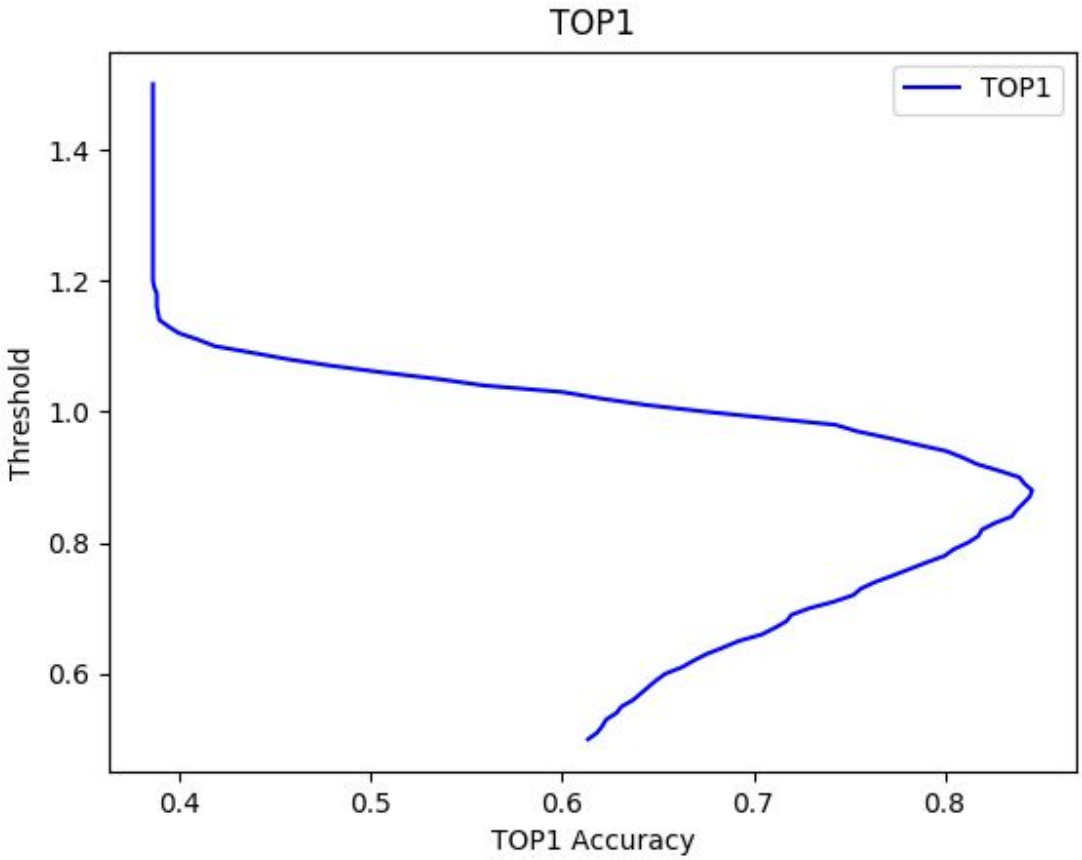


Fig 11: Top1 accuracy for different threshold

```
SAMPLES:  1485
CORRECT IDENTIFICATION TOP 1:  515
CORRECT IDENTIFICATION TOP 2:  522
CORRECT IDENTIFICATION TOP 3:  522
CORRECT IDENTIFICATION TOP 4:  522
CORRECT IDENTIFICATION TOP 5:  522
CORRECT REJECTION:  687
WRONG OUT OF GALLERY IDENTIFICATION:  120
WORNG REJECTIONS 49
TOP 1 ACCURACY: 0.8094276094276094
TOP 2 ACCURACY: 0.8141414141414142
TOP 3 ACCURACY: 0.8141414141414142
TOP 4 ACCURACY: 0.8141414141414142
TOP 5 ACCURACY: 0.8141414141414142
```

Fig.12 : Identification Statistics