



## Project Report 6.1 for The Third year- Project «Face Recognition – Android -App»



ENSTA Bretagne  
2 rue François Verny  
29806 Brest Cedex 9, France

Project Tutor:

AbedELMalek TOUMI prof-  
Ensta Bretagne.  
[abdelmalek.toumi@ensta-bretagne.fr](mailto:abdelmalek.toumi@ensta-bretagne.fr)

Student:

TANIOS Georges  
[Georges.tanios@ensta-bretagne.org](mailto:Georges.tanios@ensta-bretagne.org)  
SPID  
Système logiciel et sécurité  
CI 2020



## Acknowledgement

I have taken efforts in this project. However, it would not have been possible without the kind support and help of my project supervisor “**Abedelmalek TOUMI**”. I would like to extend my sincere thanks to him.

I am really fortunate that I had this kind of experience in artificial intelligence in general and facial recognition especially.

In this project at Ensta Bretagne during my last year of education, I was mentored by the head of **SOYA** section at Ensta Bretagne “**Abedelmalek TOUMI**”. He has provided a huge amount of his precious time and effort for me. I feel really lucky to be able to work under his supervision.

Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

## Abstract

As part of my software and security engineer studies and diploma. An end of study project is mandatory to achieve my studies in this field. And put into practice the knowledge acquired during our studies at Ensta Bretagne. For this reason, we have chosen to set up a project called: Face Recognition – Android.

This project aims to develop an automated system that would be present in classes. Allowing the professor in the class to know the attends in the class, by using his android smartphone camera. That in this part, my system detects in real-time and recognizes each person that appears on the screen.

To accomplish this project there are four main parts: Dataset Generator, Trainer, Face Detector and Face Recognition.

First, the Dataset creator or generator takes images for each person and properly divide them, to organize the dataset. Then the trainer is to train our dataset and generate a file with yaml<sup>1</sup> extension. Yaml or yml is a human-readable data serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted [1]. Then this yaml file is used in the face recognition application.

The Face detector program's main idea is to detect the frontal face and draw a rectangle on the face detected by the camera. Generally, in any facial recognition system, the face detection program is used. Indeed, I'm using the library Open CV<sup>2</sup>.

After finishing these four parts. We arrive to the desired goal, where the system could be the identification of the persons present in the examination room. The teacher will know exactly who is present or absent. This project will be used also in companies to recognize clients.

---

<sup>1</sup> Yaml Ain't Markup Language

<sup>2</sup> OpenCV is a library of programming functions mainly aimed at real-time computer vision

## Table of Contents

Acknowledgement .....	3
Abstract .....	4
Introduction.....	7
I. Different Algorithm for Face Recognition .....	9
1. PCA.....	9
2. LDA .....	9
3. LBP .....	10
4. Haar Cascade .....	11
5. YOLO .....	12
II. Software Used.....	13
1. Python .....	13
2. Android Studio.....	14
3. OpenCV .....	14
Open CV Installation: .....	14
Characteristics of OpenCV: .....	15
4. SQLite Studio.....	16
III. Face detection .....	16
IV. Dataset Generator and Trainer .....	18
1. Data set Generator.....	18
Importance of data set.....	19
Data set Generator script.....	19
2. Training Dataset.....	22
Trainer Script .....	23
V. Face Recognition .....	25
Face Recognition Script.....	26
VI. Integration python-for-android .....	28
VII. Problem Encountered.....	29
Conclusion .....	30
bibliography .....	31

## Table of figures

Figure 1-performance gray value of local binary pattern (LBP) operator .....	10
Figure 2- LBP histogram for each block and the feature histogram.....	10
Figure 3- Haar Features .....	11
Figure 4- yolo algorithm .....	13
Figure 5- Face detection.....	16
Figure 6-insert id and name .....	20
Figure 7-insert a person in the dataset .....	21
Figure 8- dataset repository .....	22
Figure 9- data base for the dataset .....	22
Figure 10- repository.....	23
Figure 11- Face Recognition.....	28

## Introduction

People have always had the ability to recognize and distinguish two different faces, while computers have begun to show the same ability not very long ago. According to [2]. Face recognition is an easy task for humans. By the mid-1960s, researchers began the work on the recognition of human faces using the computer. Lately, robust facial recognition systems are becoming more useful, being helpful in various fields such as terrorism and crime-fighting and different user authentication in real or virtual spaces for better security.

There are several unique physical characteristics for an individual, which explains the diversity of systems applying biometrics, let us quote some: The fingerprint [3], The dynamics of signatures, iris, retina, voice and face recognition, text recognition [4]. The interest of applications using biometrics can be summarized in three classes: facilitating the mode of life, avoid fraud and control the population.

Experiments have shown, that even one to three-day old babies are able to distinguish between known faces. So how hard could it be for a computer? It turns out we know little about human recognition to date. Are inner features (eyes, nose, mouth) or outer features (head shape, hairline) used for successful face recognition? How do we analyze an image and how does the brain encode it? It was shown by **David Hubel** and **Torsten Wiesel**, that our brain has specialized nerve cells responding to specific local features of a scene, such as lines, edges, angles or movement. Since we don't see the world as scattered pieces, our visual cortex must somehow combine the different sources of information into useful patterns. Automatic face recognition is all about extracting those meaningful features from an image, putting them into a useful representation and performing classification on them [5].

Also, facial and form recognition could help marketers in having a better impact on their clients, because for example in facial recognition the marketers will know their customers by entering to store. Their face will be detected by the camera and recognize it from the database and will know what he bought, last time he comes to the store. Perhaps with these smart technologies, the loyalty card will become obsolete.

Face detection and recognition, sequence different algorithms over the past few years. Each algorithm has its own advantages and disadvantages.

Nowadays face recognition systems are implements in a platform on smartphones. The quality of the smartphone's camera enables us to capture high-quality pictures at a high resolution, so we can perform different types of recognition on these images.

In this report, I will introduce the main ideas of the facial recognition system. And the target of using it, after that I give some explanation of different algorithms that has been used in this decade. By talking on their result percentage and their difference. In which I talked about my face detection and face recognition algorithm. Indeed, for the facial recognition system, I must use a data set to recognize the faces in the real-time screen record.



## I. Different Algorithm for Face Recognition

In this field or new technology, we can find multiple algorithms that can be used for this type of project. Each algorithm has its own advantages and disadvantages. Every algorithm changes in precision and percentage results.

We List some popular algorithms used in this decade for face recognition:

### 1. PCA

PCA<sup>3</sup> is principal component analysis. In the PCA algorithm, we are using eigenface. Eigenfaces is a method that is useful for face recognition and detection by determining the variance of faces in a collection of face images and use those variances to encode and decode a face in a machine learning way without the full information reducing computation and space complexity [6]. In this algorithm first step is to let the detected frontal face image be a matrix with a dimensional of  $M \times N$ . Then we prepare the training faces by converting each image to  $(M \times N) \times 1$  image vector. In the next step, we are going to find the average face vector which is the sum of all image's vectors / number of images. Then subtract the average face vector from every image vector, so we put  $(\text{imageVectors} - \text{averageFaceVector})$  in a matrix of  $A [ (M \times N) \times I ]$ . Where "I" is the number of images. After finishing this, we compute the covariance matrix of above matrix

" $C = A * \text{transpose}(A)$ ". We find the **eigenvectors** and **eigenvalues** from C matrix. As the size of C is huge, so it will take a long time to compute the eigenvectors, so we will calculate the eigenvectors of  $A^T * A$ , which will make the matrix with  $I \times I$  dimension. Then we multiply the precedent eigenvectors with A to get the eigenvectors of  $A * A^T$ . The last step will be to take only K eigenvectors (corresponding to the K largest eigenvalues) Eigenfaces with low eigenvalues can be omitted.

### 2. LDA

It has been demonstrated that the **Linear Discriminant Analysis (LDA)** approach outperforms the Principal Component Analysis (PCA) approach to face recognition tasks. Due to the high dimensionality of image space, many LDA based approaches, however,

---

<sup>3</sup> Principal Component Analysis

first use the PCA to project an image into a lower dimensional space or so-called face space and then perform the LDA to maximize the discriminatory [7].

### 3. LBP

**Local Binary Pattern (LBP)** is a popular face recognition algorithm. Although it sounds like a very simple task for us, it has proven to be a complex task for a computer, as it has many variables that can impair the accuracy of the methods, for example: illumination variation, low resolution, occlusion, quality of image, amongst other [8].

In this algorithm, we have to examine the image before using it, like the illumination of the noise contrast, resolution, equalization and normalizing (can follow the steps in this reference for training the image [9]). Then we apply the LBP algorithm. This algorithm works with the eight neighbors of a pixel, using the value of this center pixel. If a neighbor pixel has a higher gray value than the center pixel, then this pixel in the middle gets one as value. And if there is not a higher gray value from the neighbors, the center pixel gets zero as a binary value (figure 1).

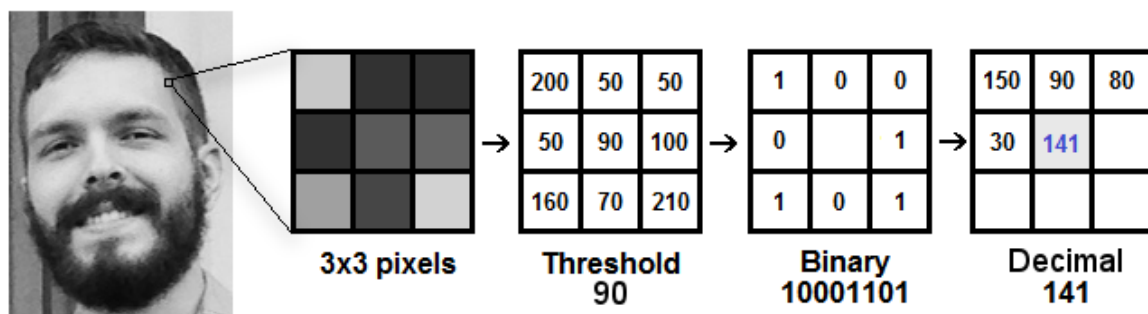


Figure 1-performance gray value of local binary pattern (LBP) operator

The objective of **LBP** is to calculate the local binary pattern for each pixel of an input image. Finally, the histogram is calculated to find out the similarities between the detected image and the image in the input “dataset” (figure 2).

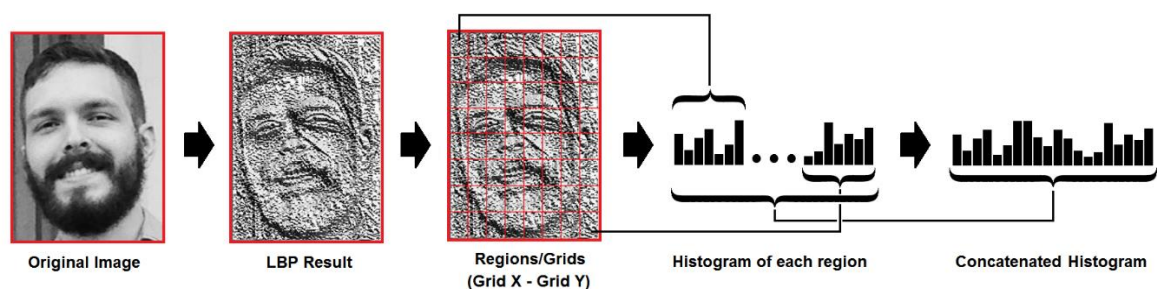


Figure 2- LBP histogram for each block and the feature histogram

## 4. Haar Cascade

**Haar Cascade** classifier is based on the **Haar Wavelet technique** [10] to examine and analyze pixels in the image into squares by function. This uses “integral image” concepts to compute “features” (figure 3) detected. This Haar Cascade uses the Ada-boost algorithm, which selects a small number of important features. The Haar Cascade classifier uses **voila Jones** algorithm in face detection to detect if there is in the image a face or non-face. And Haar Cascade algorithm was created by Paul Viola and Michael Jones.

And as we can see in figure 3 that there are two types of features “edge features” and “line features”. The first one is like square 1 and 2 in figure 3, which detects the edges quite effectively. Or the second one represents the square 2 and 3 in the figure below, which detects the lines quite effectively.

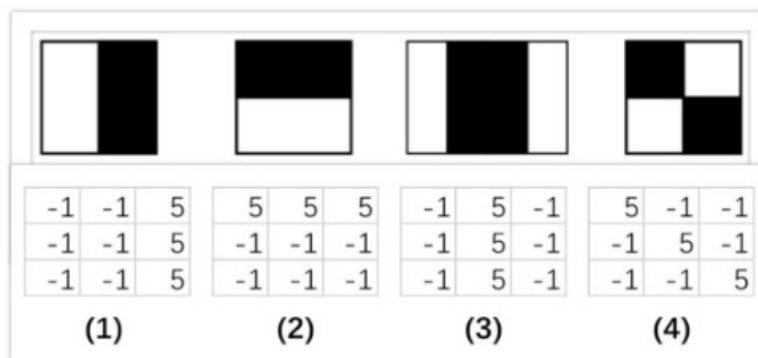
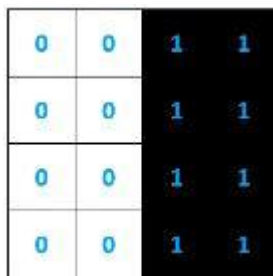


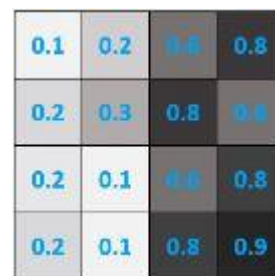
Figure 3- Haar Features

For this algorithm, the input image will be divided into pixels and we make a default matrix, for example, the first one in figure 3. And we give “0” for the white section and “1” for the black section:



Ideal **Haar-feature**

Pixel intensities



these are the real values

detected on an image

0: white || 1: black

**Viola-jones algorithm** [11] will compare how close the real scenario to the ideal case.

And how we can see in the real case the white section values are not zero because we're dealing with gray scale images, so much the value is higher much the pixel is darker.

Now we must sum up the white pixel intensities and the sum of the black pixel intensities, then calculate the average of each section. And finally, get the difference between these two sums.

→ Dark =  $(0.6+0.8+0.8+0.6+0.6+0.8+0.8+0.9)/n$ . (n: is the number of black pixels)

→ White =  $(0.1+0.2+0.2+0.3+0.2+0.2+0.1)/n$ . (n: is the number of white pixels)

→ delta = Dark - White =  $0.74 - 0.18 = 0.56$  (for the real image)

→ delta' = 1 (for ideal Haar feature)

Finally, in comparison, the delta value is closer to "1", the more likely we have found a Haar feature of the image! It's a match!

## 5. YOLO

YOLO (You Only Look Once) real-time object detection algorithm. Yolo algorithm is a clever convolutional neural network (CNN) for doing object detection in real time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities [12]. At the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression, it then outputs recognized objects together

with the bounding boxes.

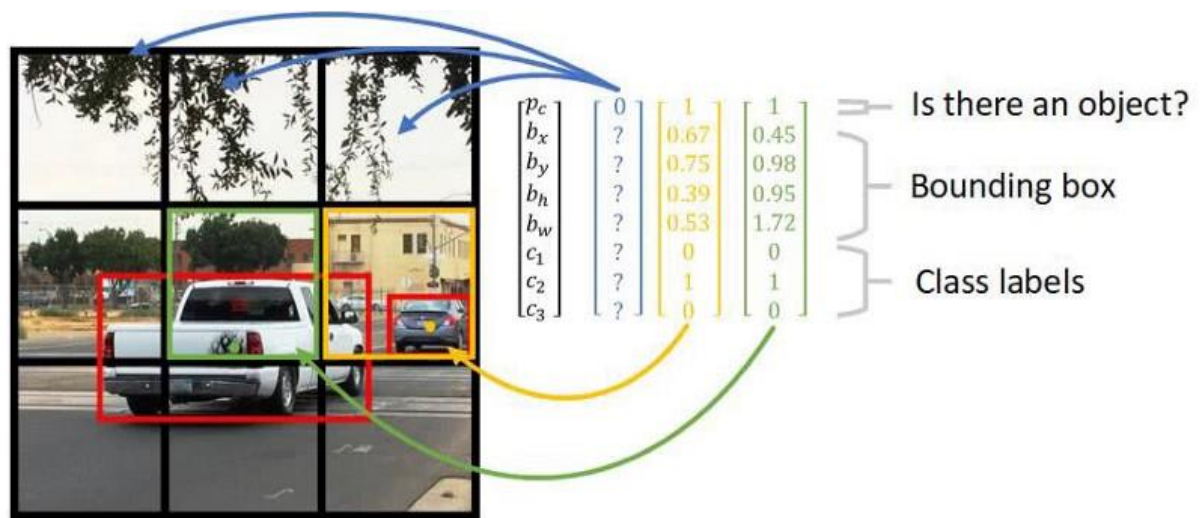


Figure 4- yolo algorithm

As we can see in the image that we make in each region 2 prediction bounds, and if there is an object the “pc” in the matrix will be equal to “1” if not it will be “0”. After detecting the object where the  $p_c = 1$ , we pass to see which class in our matrix is equal to 1. The class who has the value equal one is the object detected in the image input.

## II. Software Used

To accomplish a face detection and recognition system, we need technologies to do so. Some of these technologies or software are: Python, Android Studio, Open Cv.

### 1. Python

Python [13] is the popular programming language in the world. Python is a powerful language, both easy to learn and rich in possibilities. From the moment you install it on your computer, you have many features built into this language. Also, there are called libraries that help the developer to work on projects. There are different versions of python. (version 2, version 3).

To install python in our operating system, first, we enter this link: [Python.org](https://www.python.org/). Then we choose the version and the operating system where we will download this python file. After that we can install is normally on our machine.

## 2. Android Studio

Android Studio is a development environment for developing Android mobile applications. It is based on IntelliJ IDEA and uses the Gradle production engine. It can be downloaded under Windows operating systems, macOS. To install it we must download the file .exe of Android Studio then install it on windows for example:

- Check if there is a java version before launching the .exe file.
- Checking for the SDK.

Normally in android studio we use Java as a programming language. But there are new tools, that allow coding with another language in **Android Studio**, like python for example.

## 3. OpenCV

**OpenCV**<sup>4</sup> is an open-source library for computer vision and machine learning. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has more than 2500 optimized algorithm, these algorithms can be used to detect and recognize faces, identify an object, classify human action in videos etc. [14].

OpenCV has more than 47 thousand people of the user community and the number download in around 18 million this day. This library is used by huge companies like Google, Microsoft, Intel, IBM etc.

### Open CV Installation:

To install OpenCV library we need a version of python. I used python 2.7 because it is already installed in my computer, but you can use whatever version you want of python.

- Enter to : <https://sourceforge.net/projects/opencvlibrary/files/>. And choose the OpenCV Windows version.
- then choose the recent version after choosing “opencv-win”.
- Download the exe file

---

<sup>4</sup>Open Source Computer Vision Library

- Install it.
- Finally, it's done you have an open cv folder which contains 2 folders “build” and “source”.

### **Characteristics of OpenCV:**

- Image of data manipulation (distribution, outputs, copying, creation, conversion).
- Image and video, I / O (file and input camera in function, image / file of video output).
- The manipulation of matrices and vectors and routines of linear algebra (eigenvalues products, solvers, SVD).
- Various dynamic data structures (lists, queues, sets, trees, graphs).
- Basic image processing (filtering, edge detection, angle detection, sampling and interpolation, color conversion, operations morphological, histograms, pyramids of images).
- Structural analysis (related components, contour processing, transformed from distance, different times, corresponding model, Hough transform, polygonal approximation, line fitting, ellipse fitting, triangulation of Delaunay).
- Camera calibration (and calibration test patterns, calibration, estimation of the fundamental matrix, homography estimation, stereo correspondence).
- Analysis of movement (optical flow, movement segmentation, monitoring).
- Object recognition (Eigen-methods, HMM).

After that, we can use OpenCV in our python code. We must go to “build” folder in OpenCV → python → 2.7 → x86. Here there is a file called “cv2.pyd”. We copy this file and go to Python27 in C: Python27/Lib/site-packages and paste the cv2 here. In order to complete the configuration, we need also numpy<sup>5</sup> for python: we enter to c:/python27/Scripts using the Cmd. Then we type this command: *pip install numpy*.

---

<sup>5</sup> NumPy is a library for the Python programming language



## 4. SQLite Studio

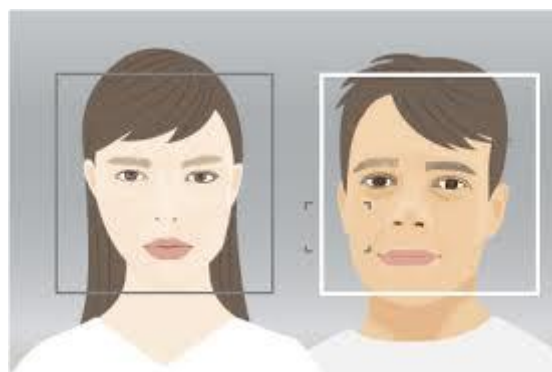
**SQLite Studio** is a user interface of SQLite databases. SQLite is normally relational database management contained in C library. SQLite uses a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. This means that one can, for example, insert a string into a column defined as an integer. SQLite will attempt to convert data between formats where appropriate, the string "123" into an integer in this case, but does not guarantee such conversions, and will store the data as-is if such a conversion fails [15].

And we can download the SQLite Studio via this link: <https://sqlitestudio.pl/index.rvt>

## III. Face detection

**Face detection** is a computer technology being used in a variety of applications that identifies human faces in digital images. **Face detection** also refers to the psychological process by which humans locate and attend to faces in a visual scene [16]. Face detection is a process in which algorithms are developed and trained to properly locate faces or objects

In this part I will talk about the importance of face detection and what are the procedures that help us detect the face of a person, using the Open CV library. So that's when the program detects a face. It draws a rectangle around this detected face.



*Figure 5- Face detection*

This technology is widely used in social media applications like Snapchat when they put a filter on the face by adding some shapes, for example, the virtual wear of dog face masks using a fancy filter. In this project, I implemented a system which detects faces in real time video records that helps us in the dataset generator and for the facial recognition system.



Face detection uses a classifier to identify if there is a face in the image or not. There is a lot of classifier, but in this project, I'm using Haar Cascade classifier with OpenCV, where this classifier or algorithm is explained in section four, that this classifier in result selects a small number of important features, and compare the real feature extracted from an input image to an ideal feature. This classifier needs a training image, so in my case I'm using the training data from the Open CV library. the file is called: "haarcascade\_frontalface\_default.xml". We can find this file in: *C:/OpenCV/sources\data\haarcascades*. Then we copy this file to the repository location when we're coding our system.

```
import numpy as np
import cv2

detector= cv2.CascadeClassifier('haarcascade_frontalface_default.xml');
cap = cv2.VideoCapture(0);

while(True):
    ret, img = cap.read();
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = detector.detectMultiScale(gray, 1.3, 5);
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255), 2)
    cv2.imshow('Face',img);
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break;

cap.release()
cv2.destroyAllWindows()
```

As we can see in this part of the code. We are importing cv2 to use it for the Cascade classifier. In Cascade classifier we put in parameter the path of the training file for the frontal face, which will help us to detect the faces, as we explained above in the section “**Haar cascade classifier**”. And this file can be in xml or yml extension. In the second line the videoCapture, is capturing video in real time. Then in a while loop we read the image in the video and convert it to gray with the function “*cvtColor*”. Perhaps the gray information within a face is treated as an important feature. The eyebrows, the lips, and the sides of the nose are generally darker than all over the facial region.

Indeed, the Cascade Classifier uses the frontal face xml file to detect the scale of faces and finally, we draw the rectangle around the faces detected in the video by considering the height and the width of the face. In conclusion here we're testing every pixel in the image

“the real feature” to the xml file who contains the ideal haar feature. After this comparison, if this is a face or faces, then we make a loop to detect every face in the input video and draw the rectangle around these faces.

In my project, I’m using the real-time face detection, which is a detection of a face in a series of images from video capturing devices. Real-time face detection is a far simpler process than detecting a face in a static image. This is because unlike most of our surrounding Department of ECE Page 28 environment, people are continually moving.

And as we can see in my code I added a quit button when it is clicked the camera close when calling “*cap.release()*” then we destroy all windows opened. This action is made when we click on the letter ‘**Q**’ from the keyboard.

## IV. Dataset Generator and Trainer

In this section of this report, I would explain how to generate the dataset for this project, and how this dataset is trained [17]. The **Dataset** is a collection of related, discrete items of related data that may be accessed individually or in combination or managed as a whole entity. A data set is organized by some data or structure like (name, last name, images, ages ...). Which’s mean that the data set is equivalent to a database. The term **dataset** is originated with **IBM**, it means a file that contains multiple related and organized data.

The **training data set** is the one used to train an algorithm to understand how to apply concepts such as neural networks, to learn and produce results. It includes both input data and the expected output. This training step becomes after the data set creation then we train the data set generated.

### 1. Data set Generator

In our case, we need the data set that stores the images with the name and other information on the person in the image. Now we can find a lot of algorithms and platforms to help us in the data set generation thing. And these days most datasets are damaged, so we must prepare our data very carefully. The data preparation is an important step to create your suitable dataset for your machine learning project. Certain values in your dataset can be complex and breaking them down into parts will help

capture more specific relationships. This process is the opposite of data reduction, since you must add new attributes based on existing ones.

### Importance of data set

In any Artificial intelligence project, we need a data set, no matter how you know in this section, if your data set is not good enough, your entire artificial intelligence project will fail. In conclusion, the most of your data set is good the most of your AI project has a great performance.

### Data set Generator script

To begin with my own **dataset generator**, we're going to use the face detector file. Where it helps to detect the face in real-time, the difference is that here we're going to capture few samples of one person from the live video frame and assign an ID to it and it will save those samples in a folder which we have had to create it before running the dataset generator file. And we will create the folder in the same location where we're typing our script .py and we name this folder "dataset". This folder will stock all the face images created by the dataset generator.

By saving the images in dataset folder. We have to follow the naming convention for the sample images, to make sure that they don't mix up with other person's samples:

For example, if the user id is 2 and its 10th sample from the sample list then the file name will be:

`User.[ID].[SampleNumber].jpg`

We're using this naming format because, we can easily get which user's face it is from its file name, while loading the image for training the recognizer.

`User.2.10.jpg`

In my script, I start with getting the id from the shell as input and initialize a counter variable to store the sample number.

```
Id=raw_input('enter your id: ')
sampleNum=0
```

Now let Start the main loop, we will take 20 samples from the video feed ans will save it in the **dataset** folder that we created previously.

```

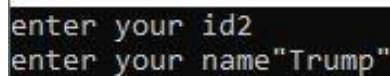
while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)

        #incrementing sample number
        sampleNum=sampleNum+1
        #saving the captured face in the dataset folder
        cv2.imwrite("dataSet/user."+Id + '.' + str(sampleNum) + ".jpg",
            gray[y:y+h,x:x+w]) #

        cv2.imshow('frame',img)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

```

We wrote the above code to detect face that I explained in section three “face detection”, so in this code there is some modification comparing to the face detection code, to make the dataset generator for our face recognizer program. We added these two lines there to get the sample number and save the face in jpg format with our naming convention. And in my code, I’m saving the information of the person detected while generating the dataset like his name and id figure6.



```

enter your id2
enter your name"Trump"

```

Figure 6-insert id and name

After compiling the dataset generator code as we can see in figure 6, we must enter the information (id, name) for each person. And those data will be stocked in a database using sqlite3 <sup>6</sup>(SQLite Studio). And look at this function below, where it insert or update the information in the database for the person who we add in the dataset.

```

def insertOrUpdate(Id, Name):
    conn = sqlite3.connect("FaceBase.db")
    cmd = "SELECT * FROM people WHERE ID="+str(Id)
    cursor = conn.execute(cmd)
    isRecordExist = 0
    for row in cursor:
        isRecordExist = 1
    if(isRecordExist == 1):
        cmd = "UPDATE people SET Name="+str(Name)+" WHERE ID="+str(Id)
    else:

```

<sup>6</sup> SQLite is a relational database management system

```

cmd = "INSERT INTO people(ID, Name)
Values (" + str(Id) + ", " + str(Name) + ") "
conn.execute(cmd)
conn.commit()
conn.close()

```

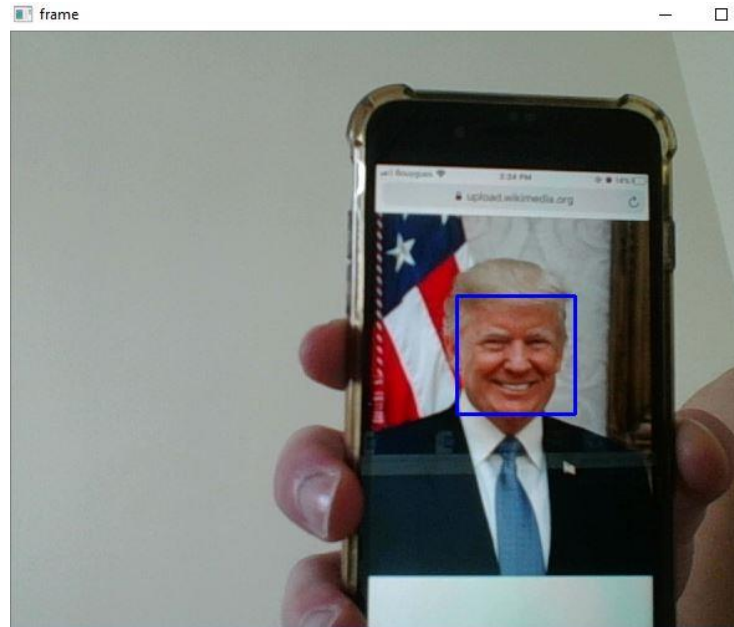


Figure 7-insert a person in the dataset

In figure7 we can see that we are adding pictures for trump. As explained in the code, we captured the face, it's this "gray[y:y+h,x:x+w]" as explained previously. Where x, y is the top left coordinate of the face rectangle and h, w is the height and the width of the face in terms of pixels

but this code will take samples vary rapidly like 20 samples in a second. Perhaps we don't want that, we want to capture faces from different angles and for that, it needs to be slow. For that, we need to increase the delay between the frames, and we need to break the loop after it took 20 samples, so we change at the end of the loop.

```

cv2.imshow('frame',img)
#wait for 100 milliseconds
if cv2.waitKey(100) & 0xFF == ord('q'):
    break
# break if the sample number is morethan 20
elif sampleNum>20:
    break

```

There we go, now it will wait for 100 milliseconds between frames which will give you time to move your face to get a different angle and it will close after taking 20 samples

Now our main loop is done, we just have to release the camera and close the windows

```
cam.release()
cv2.destroyAllWindows()
```

Finally, after compiling the dataset generator. We can find in the database a new row that contains the information entered, and the repository “dataset” already created contains the 20 samples of trump face figure8.



Figure 8- dataset repository

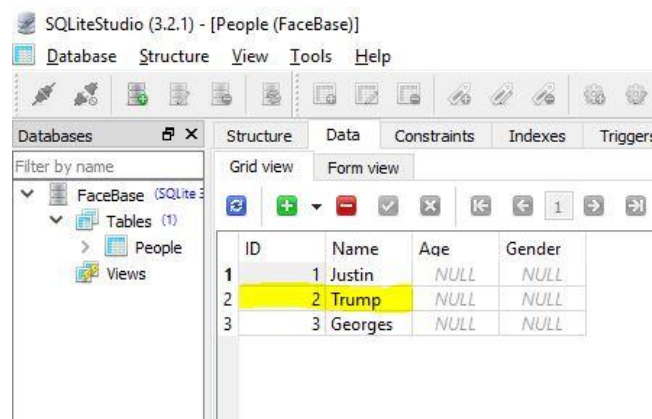


Figure 9- data base for the dataset

## 2. Training Dataset

To accomplish the facial recognition system, we need to train a face recognizer. To do so we going to use our dataset. In the section of dataset generator, I explained how to create a dataset for our facial recognition system, then we are going to use this dataset to train the face recognizer using OpenCV.

First, we create a python "trainer.py" file in the same folder where we saved our scripts. Then we create a repository called "trainer" in the same folder where we work in the project. This is the folder in which we will save our recognition module after training.

In the main directory we will see:

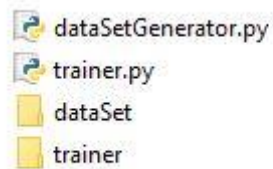


Figure 10- repository

In this part of my project we need a new library in our script "Pillow". We can install this library by going the scripts for python. Open your **cmd** and go to "cd :/python27/Scripts". Then in the scripts we can install the pillow library by the following command: "*pip install pillow*".

### Trainer Script

After installing the pillow library, we need in our code to import the cv2 Library for OpenCV. And the **os**<sup>7</sup> is needed, because we are going to use some privileges to have access to folder where the images are saved. And to import the pillow we have to call it "PIL".

```
import cv2,os
import numpy as np
from PIL import Image
```

Here we are creating the face recognizer predefined from OpenCV that use the Haar and the viola jones algorithm.

```
recognizer = cv2.createLBPHFaceRecognizer()
detector= cv2.CascadeClassifier("haarcascade_frontalface_default.xml");
```

Indeed, we are going to create a function which will grab the training images from the dataset folder and will also get the corresponding Ids from its file name, and the files name has the format **user.id.numsimple**.

---

<sup>7</sup> Operating system

This is the function where we are going to grab all the needed information. And this function needs an argument that will be the path of the dataset folder.

```
def getImagesLabels (path) :
```

Now the main objective of this function is firstly to load the training images from the dataset folder. Then capture the faces and Id from the training images and put those faces and ids in a List of Ids and Faces and return it.

To load the images, we need to create the paths of the image

```
imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
```

this will get the path of each image in the folder.

```
FacesData=[]  
Ids=[]
```

Now we will loop the images using the image path and will load those images and Ids, we will add that in the created list “FacesData” and “Ids”.

```
for imagePath in imagePaths:  
    #load the images and convert it to gray scale  
    pilImage=Image.open(imagePath).convert('L')  
    #Now we convert the PIL image in numpy array  
    imageNumpy=np.array(pilImage,'uint8')  
    #getting the Id from the image  
    Id=int(os.path.split(imagePath)[-1].split(".")[1])  
    # we get the face from the training image sample  
    faces=detector.detectMultiScale(imageNumpy)  
    # if there is a face then we append the face and the id in the  
    lists  
    for (x,y,w,h) in faces:  
        FacesData.append(imageNumpy[y:y+h,x:x+w])  
        Ids.append(Id)
```

In the above code, we used “Image.open(imagePath).convert(‘L’)” is loading the image and converting it to gray scale, but now it’s a PIL image we need to convert it to numpy array.

To convert the gray scale image to numpy array we use

“imageNumpy=np.array(pilImage,’uint8’)”. Now we are going to extract the Id of the image by splitting the image path and extract the first from the last part, which is “-1” in python. This “-1” index gives us the name of the image file. Then to get the needed value “Id”. We need to split the path one more time “user.Id.Numsample” using “.”, which



make : index0:user, index1:Id and index2:Numsample. As we can see, if we need the Id, we will choose the first index (index1).

```
Id=int(os.path.split(imagePath)[-1].split(".")[1])
```

Now we are using the detector to extract the faces and append them in the face Samples list with the Id. which looks like:

```
for (x,y,w,h) in faces:
    FacesData.append(imageNumpy[y:y+h,x:x+w])
    Ids.append(Id)
```

To oblige our program to not take every file in the dataset folder we put a condition in the for loop

```
for imagePath in imagePaths:
    # Updates in Code
    # don't take the file if it's not an image
    if os.path.split(imagePath)[-1].split(".")[-1]!='jpg':
        continue
    # load the images and convert it to gray scale
    pilImage=Image.open(imagePath).convert('L')
```

And in the main of our script we call this method by passing in argument the dataset folder:

```
faces,Ids = getImagesLabels('dataSet')
recognizer.train(faces, np.array(Ids))
recognizer.save('trainer/trainer.yml')
```

Finally, we got the training file “trainer.yml” in the folder trainer, that we will use in facial recognition system.

## V. Face Recognition

In this section, we will explain method of face recognition. To begin, in the training section, we trained our images in the dataset and created for each image a concatenated histogram. As we have the images in grayscale, and each histogram (not the concatenated) will contain only 256 position (0 – 255) representing the darkness intensity of each pixel. Then in the trainer

we concatenate each histogram to create a bigger one, so the final histogram represents the characteristics of the image in the dataset. In conclusion, this what the trainer has done.

In this step our dataset is already trained, and each histogram created represent an image in our dataset. Here in the facial recognition system, we take an input image from a real-time video record, then we remake the same steps done in the trainer and create the histogram for this image and then we try to find with this algorithm the image in the dataset that matches to the input image. In this step we just need to compare the histogram for the image in the dataset with the input image and return the closest histogram. To compare the histogram, there are many methods that can be used like “Euclidean distance, absolute value, etc.”. At the end this algorithm return the Id of the closest histogram matches to the input histogram(image). And with the Id of the image we can know the name and info concerning the input image by extracting them using this Id.

### Face Recognition Script

First, we define the same variables of the trainer, which are the Cascade classifier and the LBPHFaceRecognizer. And in our facial recognition script, we load also the trained file “trainer.yml”. This trained file we will using it later in our script:

```
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
recognizer = cv2.createLBPHFaceRecognizer()
recognizer.load('trainer//trainner.yml')
```

After that in our script, in a while loop, we detect a face. Then we predict this detected face with our trained file, that contains the characteristics of faces stored in the dataset. As we can see that we have a conf and Id variables that is returned from the function “recognizer.predict”. conf contains the rate of recognition, which is the Euclidean distance between the input histogram image and the histogram of the image in our dataset and the Id contains the id of detected face that might be one of faces in our dataset.

```
while True:
    ret, im =cam.read()
    gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    faces=faceCascade.detectMultiScale(gray, 1.2,5)
    for(x,y,w,h) in faces:
        cv2.rectangle(im, (x,y) , (x+w,y+h) , (225,0,0) ,2)
        # Prediction de photo detector
        Id, conf = recognizer.predict(gray[y:y+h,x:x+w])
```

```
        if(conf<50):
            profile = getProfile(Id)
            if(profile!=None):
                #afficher le nom du personne detecter "profile[1]"
                cv2.cv.PutText(cv2.cv.fromarray(im),str(profile[1]),
(x,y+h),font, 255)

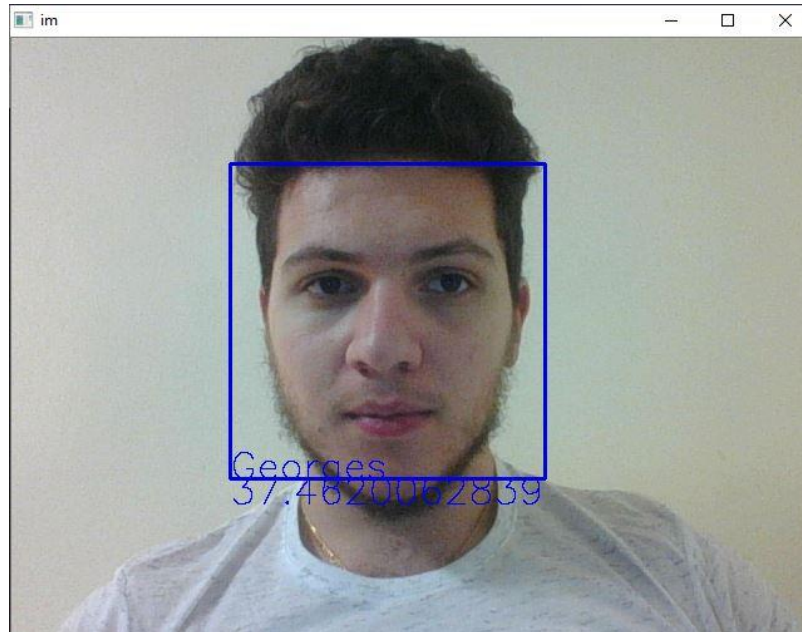
                #taux de reconnaissance
                cv2.cv.PutText(cv2.cv.fromarray(im),str(conf),
(x,y+h+20),font, 255)
            else:
                cv2.cv.PutText(cv2.cv.fromarray(im),"Unknown", (x,y+h),font,
255)

cv2.imshow('im',im)
if cv2.waitKey(10) & 0xFF==ord('q'):
    break
```

As we can see in the code above. To get the name of the person detected, we're using a function "getProfile", it returns the person (Name, Id) from the database using the id returned with "recognizer.predict".

```
def getProfile(Id):
    conn = sqlite3.connect("FaceBase.db")
    cmd = "SELECT * FROM People WHERE ID="+str(Id)
    cursor = conn.execute(cmd)
    profile=None
    for row in cursor:
        profile=row
    conn.close
    return profile
```

As I explained previously in **dataset generator** section, where I've added to my database and to my dataset, images and info of Trump, Obama and mine. So, will test our script on my own photo to see if he recognizes my face or not. And the confidence value that we print on under the name of the detected face, is the distance between the tow histograms (input image and image in the dataset). As much as the confidence has a lower value, as much as the recognizer is better.



*Figure 11- Face Recognition*

As we can see in the figure11 that my facial recognition system, recognize that it is me in front of the screen with a confidence value of 37,46. Which is good as a rate of prediction for a camera like this, with a standard quality and resolution.

## VI. Integration python-for-android

Python has proven itself as a highly capable language — approachable for newcomers, but powerful in the hands of experts. Why shouldn't you be able to use Python everywhere that you need to tell a computer to do something[18]? Modern computing doesn't happen in an 80x25 console window. It happens on phones, tablets, and desktop machines with rich user interfaces. Shouldn't you be able to use Python in all those locations and exploit the unique capabilities of those platforms.

And lot of tools can be used to run our python code scripts on android, for example “**BeeWare**” is a collection of tools for building a native user interface. This is what **BeeWare** provides. Tools to help you write Python code with a rich, native user interface; and the libraries and support code necessary to get that code running on iOS, Android, macOS, Linux, Windows, tvOS, and more.

First, we must download git the Version control system [git-scm.org](https://git-scm.org).

Setting up a virtual environment:

```
C:\...>md beeware-tutorial
C:\...>cd beeware-tutorial
C:\...>py -m venv beeware-venv
C:\...>beeware-venv\Scripts\activate.bat
```

After creating the virtual environment, we are ready to begin:

- 1- Installing the BeeWare tools:

```
(beeware-venv)C:\...>python -m pip install --pre beeware
```

- 2- Starting our first BeeWare project

```
(beeware-venv) C:\...>briefcase new helloworld
```

- 3- Running the app in developer mode

```
(beeware-venv) C:\...>cd helloworld
(beeware-venv) C:\...>briefcase dev
```

- 4- Here we can develop our python code normally. See all the tutorial

<https://docs.beeware.org/en/latest/tutorial>

## VII. Problem Encountered

- OpenCV support a few programming languages like namely C/C++, Python and Java for Android. Even OpenCV offer more than 3000 algorithms.
- Not being able to retrieve the correct for the corresponding person. To solve this problem if I used the same id, when trying to add a new face to my dataset it will erase the oldest images and the data in the database and replace it with the newest ones.
- Camera has not a high quality and resolution which is reduce our recognizing rate.

## Conclusion

The computational models, which were implemented in this project, were chosen after extensive research, and the successful testing results. Even the bad quality used for our system. Given the need to use access control applications, face recognition has emerged as an active area of research, spanning disciplines such as image processing, model identification.

the recognition of individuals remains a complex and not perfectly resolved problem, despite all the work carried out in recent years. Several problems fall to this task of identification and each of them is nontrivial. Many real conditions affect the performance of a system, however automatic face detection greatly influences the performance of the identification module.

face recognition is part of the biometry which is undoubtedly an area of the future. Over the next few decades, more and more people will carry out increased surveillance and a student attendance detection.

A face detection and recognition system would certainly speed up the process of checking student attendance in comparison to other biometrics authentication methods and in the right circumstances it would be able to match their accuracy. Nowadays there are a wide variety of software, whether it is a Face API like Microsoft's or a library like OpenCV, that makes face detection and recognition accessible and reliable and is constantly improving. Each software imposes various restrictions, such as the limited number of calls you can make to Microsoft's Face API. However, using more than one software can reduce these restrictions and lead to better results[19].

Finally, in this project we arrived in a good result with facial recognition system, where we recognize different faces in real-time video record. Then after the recognitions of these persons. We store the detected persons who are in the class and give them the authorization for doing the exam or whatever we want to give them.

## Bibliography

- [1] « YAML », *Wikipedia*. 27-janv-2020.
- [2] « What is Facial Recognition? - Definition from WhatIs.com », *SearchEnterpriseAI*. [En ligne]. Disponible sur: <https://searchenterpriseai.techtarget.com/definition/facial-recognition>. [Consulté le: 19-févr-2020].
- [3] « Fingerprint — Fingerprint 0.3 documentation ». [En ligne]. Disponible sur: <https://fingerprint.readthedocs.io/en/latest/>. [Consulté le: 21-févr-2020].
- [4] C. Wolf et J.-M. Jolion, « Model based text detection in images and videos: a learning approach », p. 24.
- [5] « Face Recognition with OpenCV — OpenCV 2.4.13.7 documentation ». [En ligne]. Disponible sur: [https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec\\_tutorial.html](https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html). [Consulté le: 08-févr-2020].
- [6] « Eigenfaces: Recovering Humans from Ghosts - Towards Data Science ». [En ligne]. Disponible sur: <https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184>. [Consulté le: 22-févr-2020].
- [7] « An Efficient LDA Algorithm for Face Recognition | Semantic Scholar ». [En ligne]. Disponible sur: <https://www.semanticscholar.org/paper/An-Efficient-LDA-Algorithm-for-Face-Recognition-Yang-Ye/13b8e93e463286a4b0275ecbb599569bbb7bf70f#citing-papers>. [Consulté le: 23-févr-2020].
- [8] « Face Recognition: Understanding LBPH Algorithm - Towards Data Science ». [En ligne]. Disponible sur: <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>. [Consulté le: 23-févr-2020].
- [9] « An improved face recognition algorithm and its application in attendance management system - ScienceDirect ». [En ligne]. Disponible sur: <https://www.sciencedirect.com/science/article/pii/S2590005619300141>. [Consulté le: 23-févr-2020].
- [10] « Haar wavelet », *Wikipedia*. 06-janv-2020.
- [11] A. Parande, « Understanding and Implementing the Viola-Jones Image Classification Algorithm », *Medium*, 27-janv-2019. [En ligne]. Disponible sur: <https://medium.com/datadriveninvestor/understanding-and-implementing-the-viola-jones-image-classification-algorithm-85621f7fe20b>. [Consulté le: 25-févr-2020].
- [12] O.-O. D. Science, « Overview of the YOLO Object Detection Algorithm », *Medium*, 25-sept-2018. [En ligne]. Disponible sur: <https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0>. [Consulté le: 25-févr-2020].
- [13] « Welcome to Python.org », *Python.org*. [En ligne]. Disponible sur: <https://www.python.org/>. [Consulté le: 25-févr-2020].
- [14] « About ». [En ligne]. Disponible sur: <https://opencv.org/about/>. [Consulté le: 26-févr-2020].
- [15] « SQLite », *Wikipedia*. 02-févr-2020.
- [16] « Face detection », *Wikipedia*. 10-févr-2020.
- [17] « What is data set? - Definition from WhatIs.com », *WhatIs.com*. [En ligne]. Disponible sur: <https://whatIs.techtarget.com/definition/data-set>. [Consulté le: 28-févr-2020].
- [18] U. Farooq, « Tools to run Python on Android », *Medium*, 28-mars-2019. [En ligne]. Disponible sur: [https://medium.com/@umerfarooq\\_26378/tools-to-run-python-on-android-9060663972b4](https://medium.com/@umerfarooq_26378/tools-to-run-python-on-android-9060663972b4). [Consulté le: 05-mars-2020].
- [19] A. D. Dinca, « Face Detection & Recognition Report », p. 7.