Report for Data Science Project d

REPORT TITLE:

Design and develop a biometric 2D faces captured in environments not controlled

Under the supervision of:
[Nefissa KHIARI]

Under the supervision of:
Driss mohamed Ahmed
Barhoumi mohamed Aymen

Table of Contents

Tables

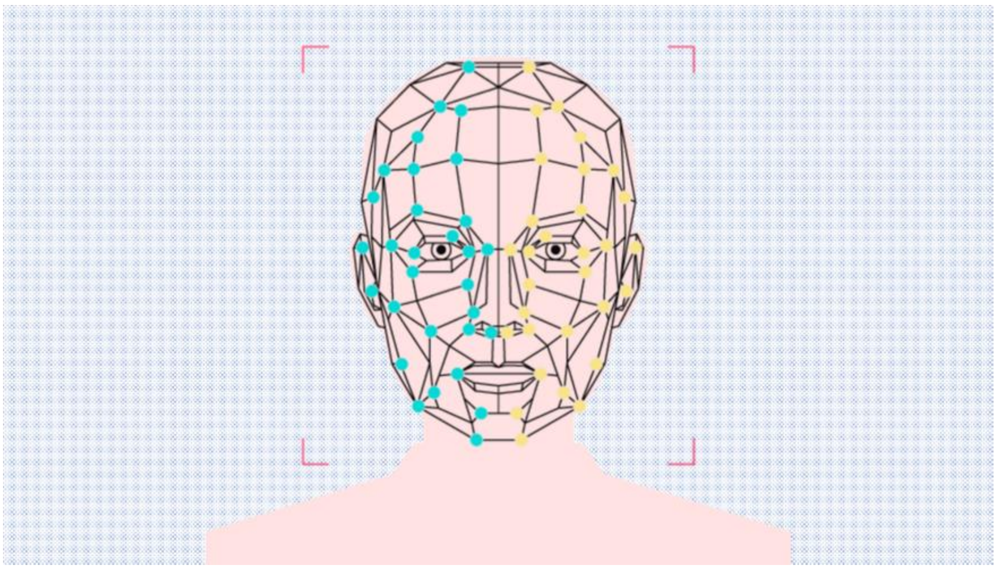
Table of Figures

SUMMARY

Thanks to the quick advance in technology, especially in computer science and electronics.

Nowadays, facial detection and recognition is becoming the second most largely deployed biometric authentication method at the world level in terms of market quota right after fingerprints. Each day more and more manufacturers are including face recognition in their products, such as Apple with its Face-ID technology, the banks with the implementation of eKYC solutions for the onboarding process.

Contrary to the main aim of research in face detection and recognition has been given to the improvement of the performance at the verification and identification tasks, the security vulnerabilities of face detection and recognition systems have been much less studied in the past, and only over the recent few years, some attention has been given to detecting different types of attacks consists of detecting whether a biometric trait comes from a living person or it is a fake.



INTRODUCTION

Facial recognition and detection technology is a set of algorithms that work together to identify people in a video or a static image. This technology has existed for decades, but it has become much more prevalent and innovative in recent years.

One such innovation is the integration of artificial intelligence (AI) within facial recognition and detection systems. Intelligent, AI-based software can instantaneously search databases of faces and compare them to one or multiple faces that are detected in a scene. In an instant, you can get highly accurate results – typically, systems deliver 99.5% accuracy rates on public standard data sets.

AI face recognition and detection software has the following advantages:

- Real-time identification;
- Anti-spoofing measures;
- Lessened racial or gender bias due to model training across millions of faces;
- Can be used across multiple cameras.

I. BUSINESS UNDERSTANDING & DATA SCIENCE OBJECTIVES

1 Introduction

For the first stage of our project we start with finding a deeper understanding of the subject we will work on.

First, we start by establishing the business objectives we have to set for our project. Next, we look at the data science objectives that we need to fulfill by the end of our project.

2 Project Context

Detecting faces in pictures can be complicated due to the variability of factors such as pose, expression, position and orientation, skin color and pixel values, the presence of glasses or facial hair, and differences in camera gain, lighting conditions and image resolution.

Recent years have brought advances in face detection using deep learning, which presents the advantage of significantly outperforming traditional computer vision methods.

3 IBM Master Plan Methodology

To ensure an efficient work flow we chose to use the IBM Master Plan Methodology. It helps the organization of documents by using a single up-to-date repository of product and service information as well as in taking strategic business initiatives.

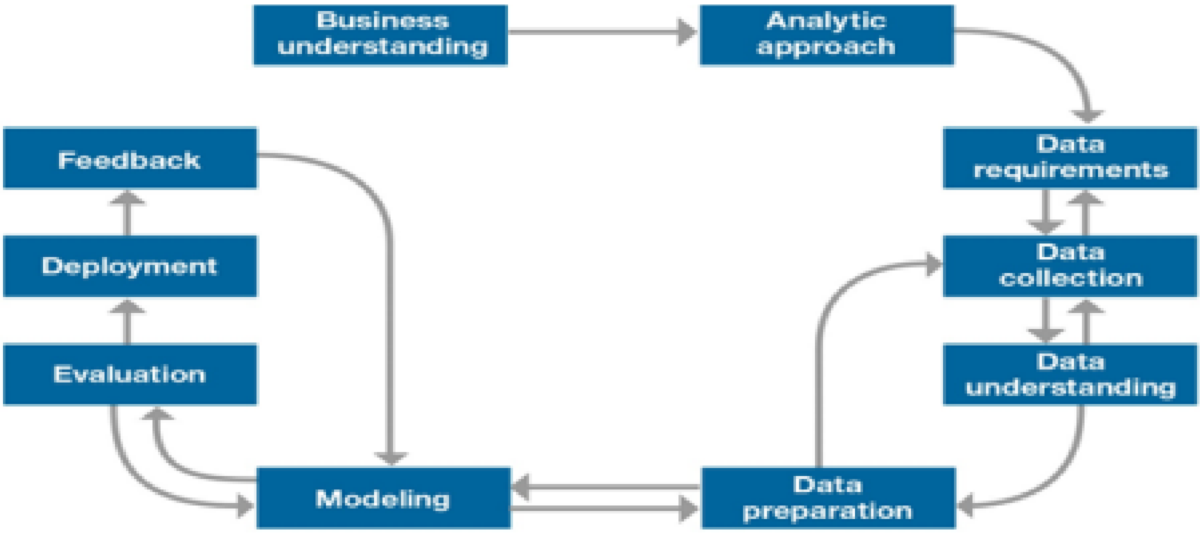


Figure 1 : IBM Master Plan Methodology

4 Business Objectives

For the business understanding we chose to follow the S.M.A.R.T methodology. It gives us the most important points we have to respect in order for our project to achieve its goals from a business perspective.

Figure 2 : S.M.A.R.T Methodology

5 Data Science Objectives

From a Data Science perspective we have a few objectives we hope to achieve through this project mainly :

- Adapt the current deep learning techniques to our subject.
- Practice new deep learning techniques such as transfer learning.
- Fully automate the analytical process for face detection and recognition.

6 Conclusion

After fully understanding the context of our project, establishing our objectives and the necessities for our work we can move on to the next steps.

II. DATA COLLECTION & PREPARATION

1 Introduction

In the second step of our project we will move on to the data related parts and shed light on the steps we took to accomplish this part.

2 Data Collection

Internal Data

The main data we collected was provided by a group of Tufts researchers across the Schools of Arts & Sciences and Engineering who have created an image database using multiple modalities. Incorporating photograph images, thermal images, near infrared images, recorded video, computerized facial sketches, and 3D images, the team has amassed a collection of more than 10,000 images from 113 individual volunteers. The volunteers represent a diverse set of ethnicities, genders, and countries of origin.

In this project , we used a part of this database which is 2Gb size and it's composed of 5 images with different emotions from 113 subjects.

3 Data Understanding

Figure 3: Database samples

The data we are going to use consists of 5 images from each subject in different emotions and with sunglasses .

In the first step, we are going to use different algorithms , in order to detect the face of the subject in each image.

Then we will evaluate the algorithms by analysing its output.

After that, we are going to use the best algorithm to extract the subject faces.

In the second step we will apply face recognition algorithms on the new dataset. In order to evaluate algorithms , we are going to use different methods and comparison lists .

4 Data Preparation

Our goal in this part is to make the data ready to use for different face detection algorithms.

We used Google collab which is a Python development environment that runs in the browser using Google Cloud.

We uploaded our data in google drive so that all the group members can use it at the same time.

Then getting access to any image of the data will be through its drive link:

```
imgPath = [ ]
#+str(j+1)+'/'
for j in range(113):
  i = str(j+1)
  for f in os.listdir(projet_path+i):

    objectPaths = os.path.join(projet_path+str(j+1)+'/' ,f)
    #print(f)
    print(objectPaths)
    imgPath.append(objectPaths)
```

```
/content/drive/MyDrive/faces_set/1/Sujet1-1.jpg
/content/drive/MyDrive/faces_set/1/Sujet1-2.jpg
/content/drive/MyDrive/faces_set/1/Sujet1-3.jpg
/content/drive/MyDrive/faces_set/1/Sujet1-4.jpg
/content/drive/MyDrive/faces_set/1/Sujet1-5.jpg
/content/drive/MyDrive/faces_set/2/Sujet2-1.jpg
/content/drive/MyDrive/faces_set/2/Sujet2-2.jpg
/content/drive/MyDrive/faces_set/2/Sujet2-3.jpg
/content/drive/MyDrive/faces_set/2/Sujet2-4.jpg
/content/drive/MyDrive/faces_set/2/Sujet2-5.jpg
/content/drive/MyDrive/faces_set/3/Sujet3-1.jpg
/content/drive/MyDrive/faces_set/3/Sujet3-2.jpg
/content/drive/MyDrive/faces_set/3/Sujet3-3.jpg
/content/drive/MyDrive/faces_set/3/Sujet3-4.jpg
/content/drive/MyDrive/faces_set/3/Sujet3-5.jpg
/content/drive/MyDrive/faces_set/4/Sujet4-1.jpg
/content/drive/MyDrive/faces_set/4/Sujet4-2.jpg
/content/drive/MyDrive/faces_set/4/Sujet4-3.jpg
/content/drive/MyDrive/faces_set/4/Sujet4-4.jpg
/content/drive/MyDrive/faces_set/4/Sujet4-5.jpg
/content/drive/MyDrive/faces_set/5/Sujet5-1.jpg
/content/drive/MyDrive/faces_set/5/Sujet5-2.jpg
```

Figure 4: Database image links

5 Conclusion

Finally, after refining and cleaning our data we can move on to the next step.

III. MODELING & EVALUATION

1. Introduction

The next step in our project is Modeling. We are going to use all the preprocessed data we have collected, and we are going to make models utilizing different algorithms and techniques, such as Cascade classifier, HAAR, HOG, MTCNN for face detection , and transfer learning for face recognition.

2. Models

2.1 Face Detection :

2.1.1 Cascade classifier

In an image, most of the image is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions.

For this they introduced the concept of **Cascade of Classifiers**. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one. (Normally the first few stages will contain very few features). If a window fails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region. How is that plan!

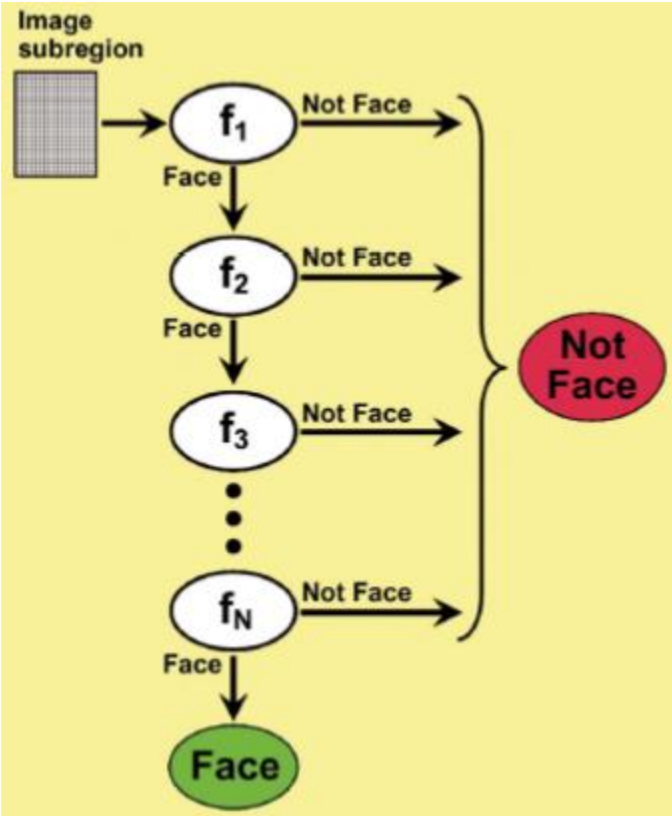


Figure 5: Cascade classifier architecture

After applying this algorithm and regulazing its hyperparameters, we get a rectangle around each subject face.

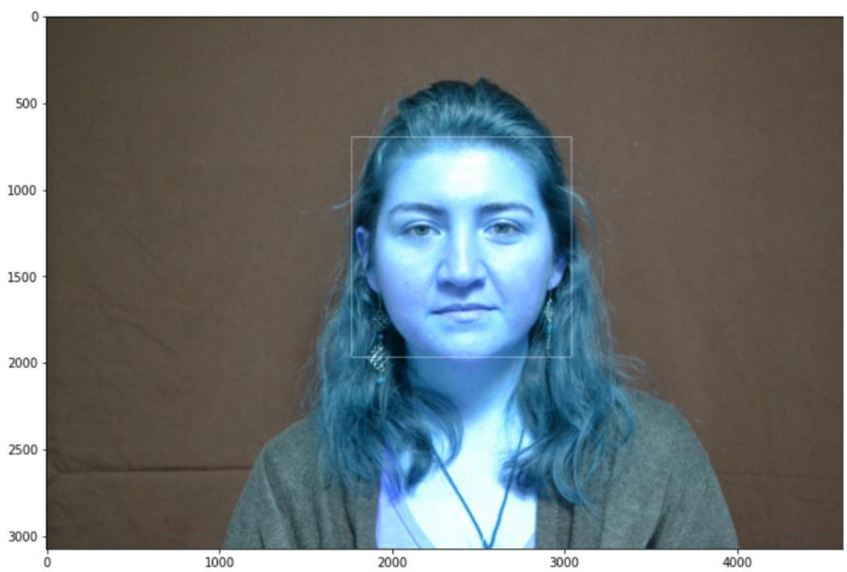


Figure 6: Cascade classifier result on an image

After that , To evaluate our models we generated a .txt file which contains **4 features** :

x , y -> the coordinates of the rectangle point.

w, h -> weight and height of the rectangle.

```
  gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
  faces = face_cascade.detectMultiScale(gray,1.2,5,minSize=(500, 500))
  for face in faces:
    print(str(face)+"  "+path+"\n")
    f.write(str(face)+"  "+path+"\n")
f.close()
```

```
[1581  401 1467 1467]  /content/drive/MyDrive/faces_set/1/Sujet1-1.jpg
[1593  429 1433 1433]  /content/drive/MyDrive/faces_set/1/Sujet1-2.jpg
[1629  450 1411 1411]  /content/drive/MyDrive/faces_set/1/Sujet1-3.jpg
[1630  400 1466 1466]  /content/drive/MyDrive/faces_set/1/Sujet1-4.jpg
[1468  419 1593 1593]  /content/drive/MyDrive/faces_set/1/Sujet1-5.jpg
[1537  722 1541 1541]  /content/drive/MyDrive/faces_set/2/Sujet2-1.jpg
[1615  735 1419 1419]  /content/drive/MyDrive/faces_set/2/Sujet2-2.jpg
[1527  663 1616 1616]  /content/drive/MyDrive/faces_set/2/Sujet2-3.jpg
[1555  695 1625 1625]  /content/drive/MyDrive/faces_set/2/Sujet2-4.jpg
[1428  719 1657 1657]  /content/drive/MyDrive/faces_set/2/Sujet2-5.jpg
[1501  526 1549 1549]  /content/drive/MyDrive/faces_set/3/Sujet3-1.jpg
```

Figure 7: Cascade classifier features results

Hyper parameters tuning:

faces = face_cascade.detectMultiScale(gray,1.2,**minNeighbors**= 5,**minSize**=(500, 500))

1/ **minNeighbors :** This parameter will affect the quality of the detected faces: higher value results in less detections but with higher quality. We're using 5 in the code.

2/**minSize:**In order to detect only one face in the image we updated the value minSize which is the Minimum possible object size. Objects smaller than that are ignored.

2.1.2 Histogram of Oriented Gradients (HOG)

One of the most popular algorithms for face detection is offered by Dlib and uses a concept called Histogram of Oriented Gradients (HOG). This is an implementation of the original paper by Dalal and Triggs.

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

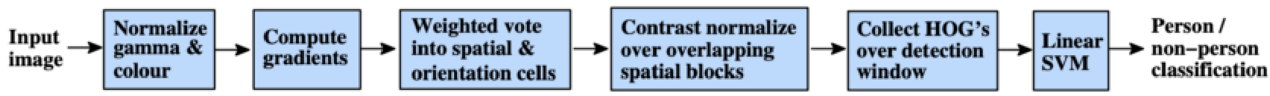


Figure 8 : HOG architecture

After applying this algorithm and regulazing its hyperparameters, we get a rectangle around each subject face.

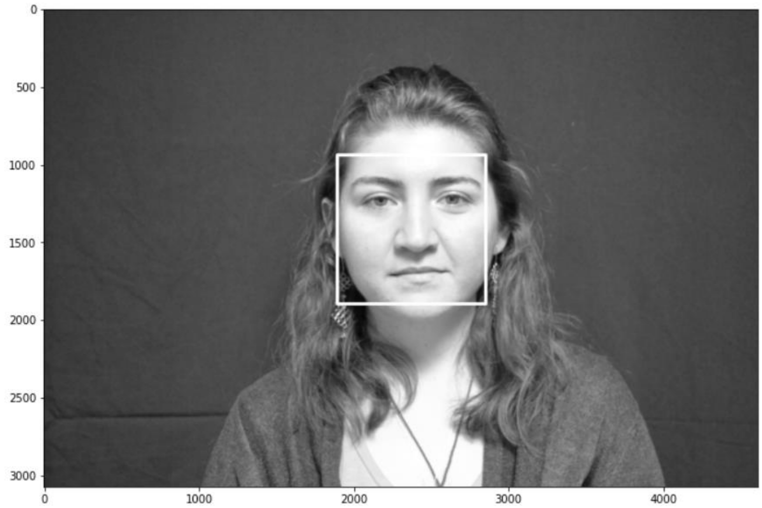


Figure 9: HOG result on an image

After that , To evaluate the algorithm just like the previous one we generated a .txt file which contains **4 features** :

x , y -> the coordinates of the rectangle point.

w, h -> weight and height of the rectangle.

Figure 10: HOG features results

2.1.3 Histogram of Oriented Gradients (MTCNN)

Multi-task Cascaded Convolutional Networks (MTCNN) is a framework developed as a solution for both face detection and face alignment. The process consists of three stages of convolutional networks that are able to recognize faces and landmark locations such as eyes, nose, and mouth.

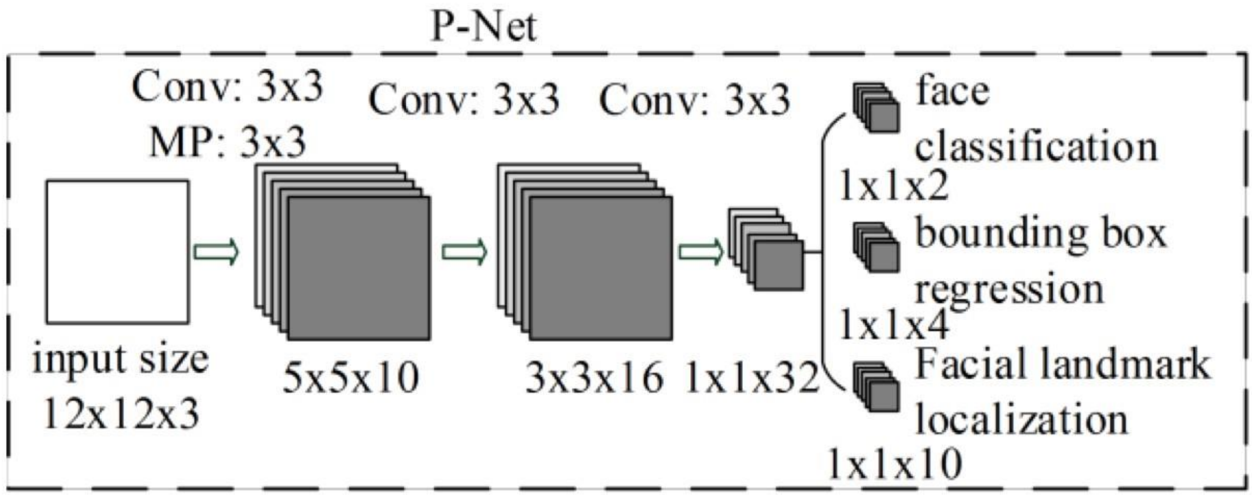


Figure 11: MTCNN architecture

We applied the same treatment for this algorithm as same as the previous ones, we also tried another version of cascade (HAAR) in order to evaluate all the 4 algorithm outputs (4 .txt files)

2.1.5 Model Comparison

Since we have 4 txt files where there are 4 features with the corresponding image link , we used Microsoft Excel to analyse and compare our algorithms.

We mentioned that all the algorithms can not detect all the images as we can see in the figure above , also we noticed that the surface of the rectangle generated by each algorithm is different.

So that , our choice was based on the number of non-detected faces and the surface of the rectangle .

We made two graphics in order to compare our algorithms :

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Algorithmes:	Cascade classifier:	x	y	w	h	sujet:	surface:	HOG :	x	y	w	h	sujet:
		1581	401	1467	1467	Sujet1-1.jpg	2152089		1763	741	1150	1150	Sujet1-1.jpg
		1593	429	1433	1433	Sujet1-2.jpg	2053489		1763	741	1150	1150	Sujet1-2.jpg
		1629	450	1411	1411	Sujet1-3.jpg	1990921		1763	741	1150	1150	Sujet1-3.jpg
		1630	400	1466	1466	Sujet1-4.jpg	2149156		0	0	0	0	FACE NOT DETECTED
		1468	419	1593	1593	Sujet1-5.jpg	2537649		1635	741	1150	1150	Sujet1-5.jpg
		1537	722	1541	1541	Sujet2-1.jpg	2374681		1656	1043	1380	1380	Sujet2-1.jpg
		1615	735	1419	1419	Sujet2-2.jpg	2013561		1656	890	1380	1380	Sujet2-2.jpg
		1527	663	1616	1616	Sujet2-3.jpg	2611456		1656	890	1380	1380	Sujet2-3.jpg
		1555	695	1625	1625	Sujet2-4.jpg	2640625		1656	1043	1380	1380	Sujet2-4.jpg
		1428	719	1657	1657	Sujet2-5.jpg	2745649		1436	884	1656	1656	Sujet2-5.jpg
		1501	526	1549	1549	Sujet3-1.jpg	2399401		1788	830	959	959	Sujet3-1.jpg
		1490	511	1543	1543	Sujet3-2.jpg	2380849		1635	741	1150	1150	Sujet3-2.jpg
		1520	536	1451	1451	Sujet3-3.jpg	2105401		1635	741	1150	1150	Sujet3-3.jpg
		1504	494	1479	1479	Sujet3-4.jpg	2187441		1635	741	1150	1150	Sujet3-4.jpg
		1498	502	1674	1674	Sujet3-5.jpg	2802276		1656	736	1380	1381	Sujet3-5.jpg
		1677	434	1436	1436	Sujet4-1.jpg	2062096		1763	741	1150	1150	Sujet4-1.jpg
		1697	394	1493	1493	Sujet4-2.jpg	2229049		1891	613	1150	1151	Sujet4-2.jpg
		1688	408	1480	1480	Sujet4-3.jpg	2190400		1891	741	1150	1150	Sujet4-3.jpg
		1711	387	1464	1464	Sujet4-4.jpg	2143296		1891	613	1150	1151	Sujet4-4.jpg
		1654	388	1633	1633	Sujet4-5.jpg	2666689		1891	741	1150	1150	Sujet4-5.jpg
		1690	560	1405	1405	Sujet5-1.jpg	1974025		1656	736	1380	1381	Sujet5-1.jpg
		1712	576	1378	1378	Sujet5-2.jpg	1898884		1656	736	1380	1381	Sujet5-2.jpg
		1679	510	1451	1451	Sujet5-3.jpg	2105401		1656	583	1380	1380	Sujet5-3.jpg
		1666	473	1459	1459	Sujet5-4.jpg	2128681		1763	741	1150	1150	Sujet5-4.jpg
		1594	550	1702	1702	Sujet5-5.jpg	2896804		1809	890	1381	1380	Sujet5-5.jpg

P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB
HAAR:	x	y	w	h	sujet :	surface:	MTCNN:	x	y	w	h	sujet :
	1628	428	1386	1386	Sujet1-1.jpg	1920996		1702	432	1133	1436	Sujet1-1.jpg
	1593	421	1440	1440	Sujet1-2.jpg	2073600		1692	419	1150	1464	Sujet1-2.jpg
	1608	435	1437	1437	Sujet1-3.jpg	2064969		1718	534	1083	1348	Sujet1-3.jpg
	1637	406	1454	1454	Sujet1-4.jpg	2114116		1779	462	1086	1487	Sujet1-4.jpg
	1478	421	1583	1583	Sujet1-5.jpg	2505889		1700	552	1080	1291	Sujet1-5.jpg
	1519	698	1593	1593	Sujet2-1.jpg	2537649		1695	779	1258	1516	Sujet2-1.jpg
	1553	688	1502	1502	Sujet2-2.jpg	2256004		0	0	0	0	FACE NOT DETECTED
	1523	662	1600	1600	Sujet2-3.jpg	2560000		1728	752	1263	1537	Sujet2-3.jpg
	1529	673	1637	1637	Sujet2-4.jpg	2679769		1726	856	1295	1600	Sujet2-4.jpg
	1417	708	1665	1665	Sujet2-5.jpg	2772225		3139	2412	96	123	Sujet2-5.jpg
	1536	545	1479	1479	Sujet3-1.jpg	2187441		1655	612	1171	1404	Sujet3-1.jpg
	1523	525	1480	1480	Sujet3-2.jpg	2190400		1644	589	1187	1459	Sujet3-2.jpg
	1529	525	1441	1441	Sujet3-3.jpg	2076481		1640	615	1118	1326	Sujet3-3.jpg
	1488	476	1495	1495	Sujet3-4.jpg	2235025		1601	529	1224	1521	Sujet3-4.jpg
	1461	473	1722	1722	Sujet3-5.jpg	2965284		1660	561	1203	1403	Sujet3-5.jpg
	1688	433	1418	1418	Sujet4-1.jpg	2010724		1802	559	1124	1275	Sujet4-1.jpg
	1711	401	1472	1472	Sujet4-2.jpg	2166784		1848	537	1110	1290	Sujet4-2.jpg
	1695	402	1466	1466	Sujet4-3.jpg	2149156		1841	565	1110	1260	Sujet4-3.jpg
	1673	349	1547	1547	Sujet4-4.jpg	2393209		1806	470	1223	1593	Sujet4-4.jpg
	1651	404	1606	1606	Sujet4-5.jpg	2579236		1849	518	1160	1311	Sujet4-5.jpg
	1698	557	1408	1408	Sujet5-1.jpg	1982464		1859	655	1088	1370	Sujet5-1.jpg
	1698	568	1378	1378	Sujet5-2.jpg	1898884		1847	635	1086	1352	Sujet5-2.jpg
	1679	508	1461	1461	Sujet5-3.jpg	2134521		1876	684	1080	1332	Sujet5-3.jpg

Figure 12: algorithm comparaison

Figure 13:   Number of non-detected faces by algorithm

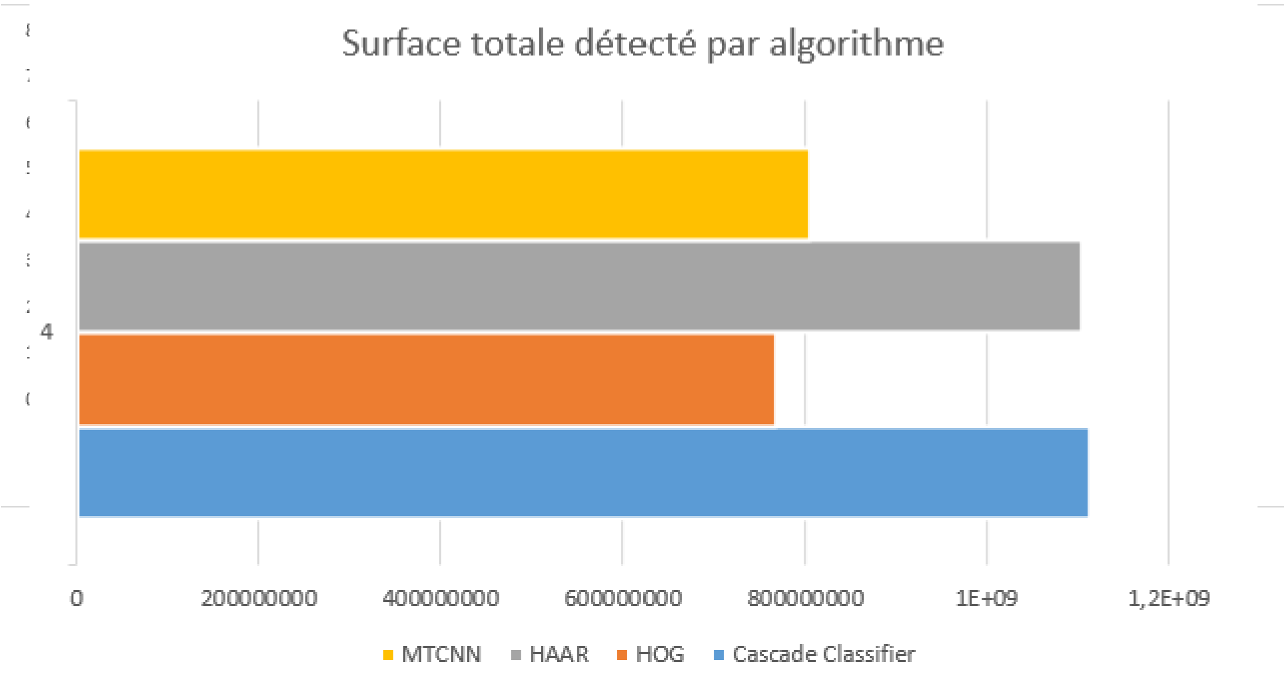


Figure 14:   Sum of surfaces by algorithm

Thanks to both graphics in the figures above, we considered that the best algorithm from all the ones we tried, is HOG.

2.2 Face Recognition

After performing the detection step, we now go to the next step which is recognition. Facial recognition is a method of identification or verification of the identity of a person using their face. There are various algorithms that can do facial recognition, but their accuracy may vary. We have did our research and read many articles on this topic and we found some algorithms, famous for dealing with this kind of subject.

2.2.1 Data Splitting

We have created a data block containing the data set x representing the path of the image and y its label (the class) First place, we have verified that the path to the cropped images is correct and that it shown pictures for this purpose. Then we started by working on our database detected images, that is, where the face appears cleanly. Second, we have creates a data frame containing the data set x: representing the path of the image and y: its label (the class).

	image	label
0	/content/ColabNotebooks/MyDrive/faces_set/Imag...	Sujet1
1	/content/ColabNotebooks/MyDrive/faces_set/Imag...	Sujet1
2	/content/ColabNotebooks/MyDrive/faces_set/Imag...	Sujet1
3	/content/ColabNotebooks/MyDrive/faces_set/Imag...	Sujet1
4	/content/ColabNotebooks/MyDrive/faces_set/Imag...	Sujet1
...	...	...
555	/content/ColabNotebooks/MyDrive/faces_set/Imag...	Sujet113
556	/content/ColabNotebooks/MyDrive/faces_set/Imag...	Sujet113
557	/content/ColabNotebooks/MyDrive/faces_set/Imag...	Sujet113
558	/content/ColabNotebooks/MyDrive/faces_set/Imag...	Sujet113
559	/content/ColabNotebooks/MyDrive/faces_set/Imag...	Sujet113

560 rows × 2 columns

Figure 15: Transformation of the database into a data frame

In this part we will show how we distribute our data, have better results and have a model that allows us to do the tests by category, we will use the "stratify" parameter which allows us to distribute our data according to categories.

Our 560 images were distributed over 336 for the training and 224 for the test (60/40).

```
Found 336 validated image filenames belonging to 112 classes.
Found 224 validated image filenames belonging to 112 classes.
```

Figure 16 : Data Splitting

2.2.2 VGG FACE

Face recognition is a computer vision task of identifying and verifying a person based on a photograph of their face.

We have chosen a VGG neural network, in particular the VGGface model based on Resnet-50 developed by researchers at the Visual Geometry Group in Oxford. The pre-trained open source model offers much better performance than “shallow” functionality reduction techniques such as PCA, LDA, SIFT...

2.2.2.1 The Architecture

Among the deep learning methods for facial feature extractions, VGGFace outperforms Facebook's DeepFace and Carnegie Mellon University's OpenFace, while being much lighter than Google's FaceNet,
It has 22 layers and 37 deep units.(25.6 million parameters versus 140 million parameters).

Figure 17 : VGG face layers

Then we will visualize the VGG-Face architure to be understood clear

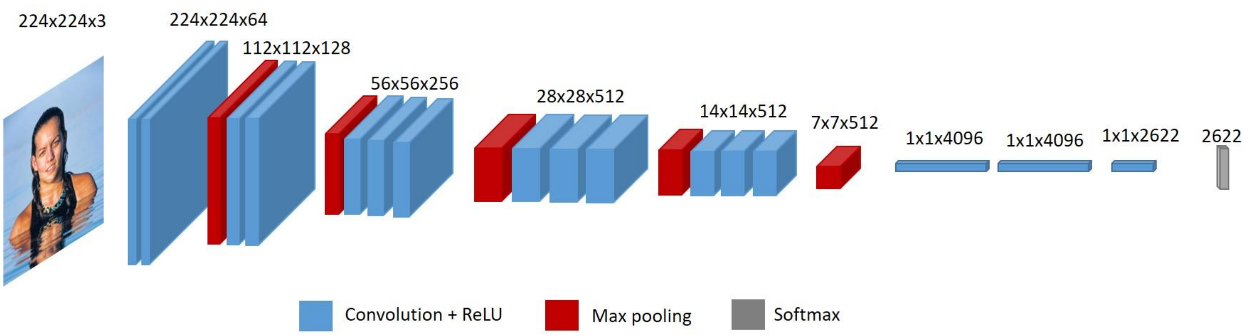


Figure 18: VGG face architecture

2.2.2.2 The implementation

So after understanding our model , we will show how we construct our algorithm

➢ The first step is to Define VGG_FACE_MODEL architecture

```
# Load VGG Face model weights
model.load_weights('vgg_face_weights.h5')
```

```
# Softmax regressor to classify images based on encoding
classifier_model=Sequential()
classifier_model.add(Dense(units=100,input_dim=x_train.shape[1],kernel_initializer='glorot_uniform'))
classifier_model.add(BatchNormalization())
classifier_model.add(Activation('tanh'))
classifier_model.add(Dropout(0.3))
classifier_model.add(Dense(units=10,kernel_initializer='glorot_uniform'))
classifier_model.add(BatchNormalization())
classifier_model.add(Activation('tanh'))
classifier_model.add(Dropout(0.2))
classifier_model.add(Dense(units=6,kernel_initializer='he_uniform'))
classifier_model.add(Activation('softmax'))
classifier_model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(),optimizer='nadam',metrics=['accuracy'])
```

Figure 19: Implementing VGG face model

➢ The next step is training our model and calculating the train score and the test score:

```
classifier_model.fit(x_train,y_train,epochs=32,validation_data=(x_test,y_test))
```

```
WARNING:tensorflow:From C:\Users\esprit\.conda\envs\PythonGPU\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.

Epoch 1/32
32/32 [==============================] - 44s 1s/step - loss: 2.7346 - accuracy: 0.2812 - val_loss: 3.4655 - val_accuracy: 0.0316
Epoch 2/32
32/32 [==============================] - 32s 989ms/step - loss: 1.2618 - accuracy: 0.5921 - val_loss: 4.0272 - val_accuracy: 0.0632
Epoch 3/32
32/32 [==============================] - 32s 990ms/step - loss: 0.8278 - accuracy: 0.7248 - val_loss: 4.7607 - val_accuracy: 0.0632
Epoch 4/32
32/32 [==============================] - 32s 990ms/step - loss: 0.6298 - accuracy: 0.7851 - val_loss: 5.3658 - val_accuracy: 0.0632
Epoch 5/32
32/32 [==============================] - 32s 990ms/step - loss: 0.4295 - accuracy: 0.8475 - val_loss: 4.8659 - val_accuracy: 0.0632
Epoch 6/32
32/32 [==============================] - 32s 989ms/step - loss: 0.3171 - accuracy: 0.8901 - val_loss: 4.4620 - val_accuracy: 0.0632
Epoch 7/32
32/32 [==============================] - 32s 990ms/step - loss: 0.2544 - accuracy: 0.9010 - val_loss: 3.6745 - val_accuracy: 0.0632
Epoch 8/32
32/32 [==============================] - 32s 988ms/step - loss: 0.2116 - accuracy: 0.9257 - val_loss: 5.4014 - val_accuracy: 0.0632
```

```
32/32 [==============================] - 32s 989ms/step - loss: 0.0820 - accuracy: 0.9733 - val_loss: 0.2056 - val_accuracy: 0.9328
Epoch 20/32
32/32 [==============================] - 32s 990ms/step - loss: 0.0667 - accuracy: 0.9792 - val_loss: 0.0483 - val_accuracy: 0.9684
Epoch 21/32
32/32 [==============================] - 32s 988ms/step - loss: 0.0364 - accuracy: 0.9911 - val_loss: 0.0693 - val_accuracy: 0.9407
Epoch 22/32
32/32 [==============================] - 32s 987ms/step - loss: 0.0356 - accuracy: 0.9881 - val_loss: 0.0440 - val_accuracy: 0.9605
Epoch 23/32
32/32 [==============================] - 32s 988ms/step - loss: 0.0185 - accuracy: 0.9960 - val_loss: 0.0910 - val_accuracy: 0.9486
Epoch 24/32
32/32 [==============================] - 32s 990ms/step - loss: 0.0226 - accuracy: 0.9941 - val_loss: 0.0439 - val_accuracy: 0.9447
Epoch 25/32
32/32 [==============================] - 32s 988ms/step - loss: 0.0182 - accuracy: 0.9950 - val_loss: 0.2009 - val_accuracy: 0.9526
Epoch 26/32
32/32 [==============================] - 32s 988ms/step - loss: 0.0489 - accuracy: 0.9881 - val_loss: 0.3510 - val_accuracy: 0.9526
Epoch 27/32
32/32 [==============================] - 32s 987ms/step - loss: 0.0437 - accuracy: 0.9822 - val_loss: 0.2109 - val_accuracy: 0.9407
Epoch 28/32
32/32 [==============================] - 32s 989ms/step - loss: 0.0644 - accuracy: 0.9772 - val_loss: 0.1580 - val_accuracy: 0.9526
Epoch 29/32
32/32 [==============================] - 32s 989ms/step - loss: 0.0587 - accuracy: 0.9802 - val_loss: 0.0420 - val_accuracy: 0.9723
Epoch 30/32
32/32 [==============================] - 32s 988ms/step - loss: 0.0316 - accuracy: 0.9911 - val_loss: 0.0491 - val_accuracy: 0.9526
Epoch 31/32
32/32 [==============================] - 32s 988ms/step - loss: 0.0290 - accuracy: 0.9911 - val_loss: 0.0584 - val_accuracy: 0.9447
Epoch 32/32
32/32 [==============================] - 32s 988ms/step - loss: 0.0217 - accuracy: 0.9921 - val_loss: 0.0885 - val_accuracy: 0.9605
```

Figure 20 : VGG face model result

⇒ we obtained Accuracy 0.9921

- for this model 219 images among 224 are well predicted

```
print("les images bien predict= ",len(test_df)-modele_rel
les images bien predict= 219
```

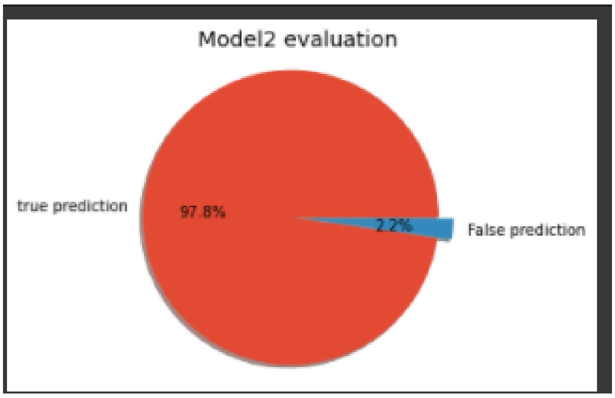


Figure 21 : VGG face model result

2.2.3 CNN (Alex Net)

At present, the typical architecture of a neural network is divided into the following categories: LeNet5, AlexNet, ZF Net, GooLeNet, and VGGNet, the following will be Alexnet architecture for a detailed analysis. Alexnet is a CNN classic structure that existed long ago, and it is mainly used in image classification . AlexNet, which employed an 8-layer CNN, won the ImageNet Large Scale Visual Recognition Challenge 2012 by a phenomenally large margin. This network showed, for the first time, that the features obtained by learning can transcend manually-designed features, breaking the previous paradigm in computer vision.

2.2.3.1 The Architecture

The architecture consists of 5 Convolutional layers, with the 1st, 2nd and 5th having Max-Pooling layers for proper feature extraction. The Max-Pooling layers are overlapped having strides of 2 with filter size 3×3. This resulted in decreasing the top-1 and top-5 error rates by 0.4% and 0.3% respectively in comparison to non-overlapped Max-Pooling layers. They are followed by 2 fully-connected layers (each with dropout) and a softmax layer at the end for predictions.

The figure below shows the architecture of AlexNet with all the layers defined.

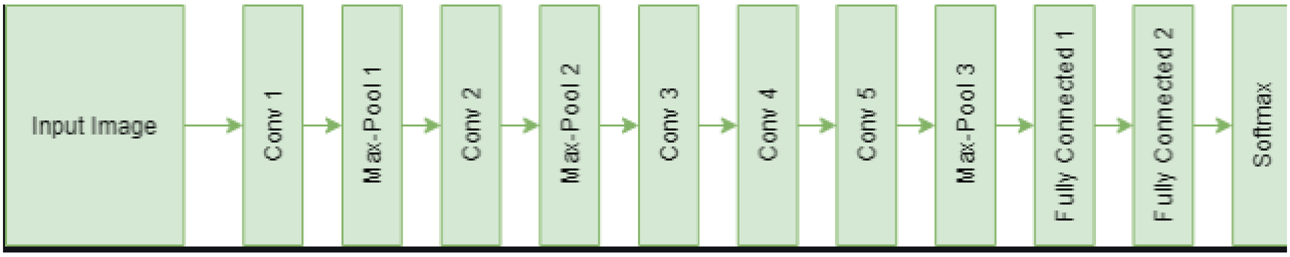
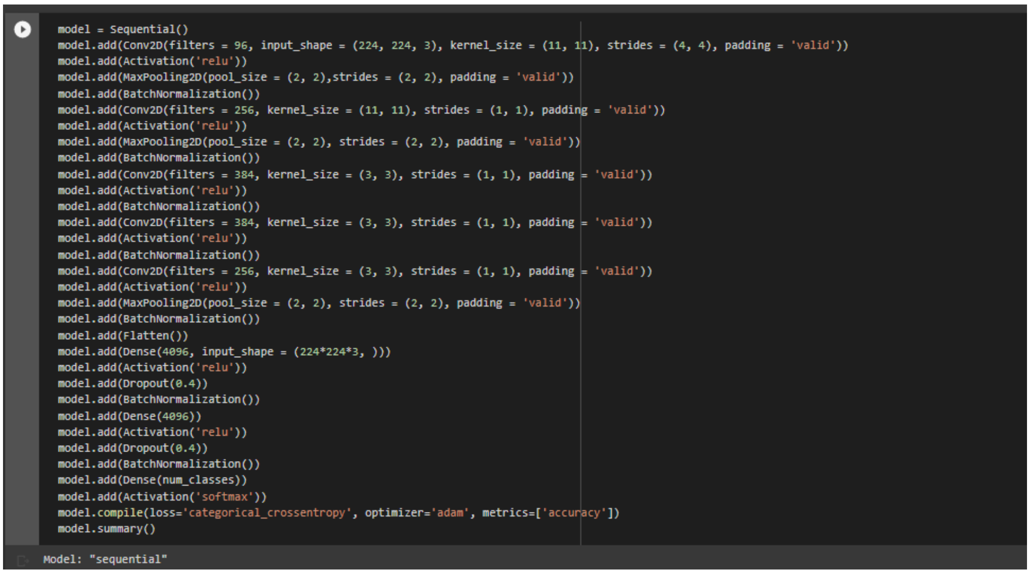


Figure 22: CNN (Alex net) architecture

2.2.3.2 Implementing the model

- ➢ The first step is to Define CNN (AlexNet Model)architecture



```
model = Sequential()
model.add(Conv2D(filters = 96, input_shape = (224, 224, 3), kernel_size = (11, 11), strides = (4, 4), padding = 'valid'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2),strides = (2, 2), padding = 'valid'))
model.add(BatchNormalization())
model.add(Conv2D(filters = 256, kernel_size = (11, 11), strides = (1, 1), padding = 'valid'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding = 'valid'))
model.add(BatchNormalization())
model.add(Conv2D(filters = 384, kernel_size = (3, 3), strides = (1, 1), padding = 'valid'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Conv2D(filters = 384, kernel_size = (3, 3), strides = (1, 1), padding = 'valid'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Conv2D(filters = 256, kernel_size = (3, 3), strides = (1, 1), padding = 'valid'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding = 'valid'))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dense(4096, input_shape = (224*224*3, )))
model.add(Activation('relu'))
model.add(Dropout(0.4))
model.add(BatchNormalization())
model.add(Dense(4096))
model.add(Activation('relu'))
model.add(Dropout(0.4))
model.add(BatchNormalization())
model.add(Dense(num_classes))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()

Model: "sequential"
```

Figure 23: Implementing Alex net model

- ➢ The next step is training our model and calculating the train score and the test score:

```
        traingenerator,
        epochs=30,
        validation_data = testgenerator,
        validation_steps= len(testgenerator),
        steps_per_epoch = len(traingenerator)
    )
```

```
Epoch 1/15
33/33 [==============================] - 0s 7ms/step - loss: 0.6634 - accuracy: 0.6496 - val_loss: 0.4754 - val_accuracy: 0.8333
Epoch 2/15
33/33 [==============================] - 0s 2ms/step - loss: 0.4567 - accuracy: 0.8247 - val_loss: 0.3734 - val_accuracy: 0.8902
Epoch 3/15
33/33 [==============================] - 0s 3ms/step - loss: 0.3665 - accuracy: 0.8809 - val_loss: 0.3163 - val_accuracy: 0.9053
Epoch 4/15
33/33 [==============================] - 0s 2ms/step - loss: 0.3209 - accuracy: 0.8837 - val_loss: 0.2797 - val_accuracy: 0.9091
Epoch 5/15
33/33 [==============================] - 0s 3ms/step - loss: 0.2687 - accuracy: 0.9153 - val_loss: 0.2542 - val_accuracy: 0.9129
Epoch 6/15
33/33 [==============================] - 0s 2ms/step - loss: 0.2527 - accuracy: 0.9166 - val_loss: 0.2360 - val_accuracy: 0.9280
Epoch 7/15
33/33 [==============================] - 0s 2ms/step - loss: 0.2321 - accuracy: 0.9173 - val_loss: 0.2222 - val_accuracy: 0.9242
Epoch 8/15
33/33 [==============================] - 0s 2ms/step - loss: 0.2178 - accuracy: 0.9239 - val_loss: 0.2115 - val_accuracy: 0.9205
Epoch 9/15
33/33 [==============================] - 0s 2ms/step - loss: 0.1998 - accuracy: 0.9234 - val_loss: 0.2033 - val_accuracy: 0.9318
Epoch 10/15
33/33 [==============================] - 0s 3ms/step - loss: 0.1838 - accuracy: 0.9436 - val_loss: 0.1970 - val_accuracy: 0.9356
Epoch 11/15
33/33 [==============================] - 0s 3ms/step - loss: 0.1658 - accuracy: 0.9526 - val_loss: 0.1918 - val_accuracy: 0.9356
Epoch 12/15
33/33 [==============================] - 0s 2ms/step - loss: 0.1567 - accuracy: 0.9590 - val_loss: 0.1875 - val_accuracy: 0.9356
Epoch 13/15
33/33 [==============================] - 0s 3ms/step - loss: 0.1595 - accuracy: 0.9594 - val_loss: 0.1840 - val_accuracy: 0.9356
Epoch 14/15
33/33 [==============================] - 0s 2ms/step - loss: 0.1597 - accuracy: 0.9576 - val_loss: 0.1809 - val_accuracy: 0.9356
Epoch 15/15
33/33 [==============================] - 0s 3ms/step - loss: 0.1383 - accuracy: 0.9689 - val_loss: 0.1787 - val_accuracy: 0.9356
```

Figure 24: CNN model result

⇒ we obtained Accuracy 0.96.

- for this model 214 images among 224 are well predicted

```
[9]  print('les image bien detetcte',len(test_df)-modele_reliability)

     les image bien detetcte 214
```

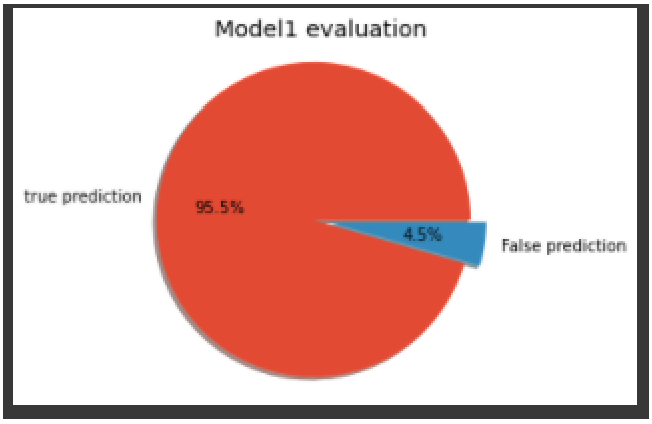


Figure 25 : CNN model result

2.2.4 Model Comparison

For the next part we move to the comparison of our pre-trained models and choose the best one .

model name	Accuracy	True Predict
VGG Face	0.98	219
CNN (AlexNet)	0.96	214

Table 1 : Transfer Learning Models comparison

2.2.6 Conclusion

We are going to conclude the modeling part by comparing the performance of the different models we implemented throughout this phase.

Based on the accuracy we had, the best model is clearly the VGG-Face

2.3 Comparison Lists

The main purpose of this part is to compare and identify the comparison lists images that we have inter and intra, that is to say compare the images with others in the same folder (the same subject) and compare these images with others that belong to different paths.

We started with the total list of test images we had (112 * 2 = 224). Next we have calculated all the possible combinations (without repetition) which can take place in comparing these images there (24976) and we displayed them. This list represents all lists of comparison (intra and inter).

```python
list_comp1 = [(a, b) for idx, a in enumerate(List_img) for b in List_img[idx + 1:]]
list_comp1 #list of all possible pairs of the images
```

```
[('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet65/Sujet65-4.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet11/Sujet11-4.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet18/Sujet18-1.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet63/Sujet63-4.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet38/Sujet38-4.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet85/Sujet85-3.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet13/Sujet13-5.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet97/Sujet97-2.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet106/Sujet106-4.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet86/Sujet86-3.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet57/Sujet57-1.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet98/Sujet98-2.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet83/Sujet83-4.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet36/Sujet36-1.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet110/Sujet110-1.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
  '/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet113/Sujet113-3.jpg'),
 ('/content/ColabNotebooks/MyDrive/faces_set/Images_crop/Sujet87/Sujet87-5.jpg',
```

Figure 26: The possible combinations of images

Then we decided to collect the results of the different pairs of images found on this list in a data frame of two columns, the first containing the path of the first image and the second column containing the path to the second image.

	image1	image2
0	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...
1	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...
2	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...
3	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...
4	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...
...	...	...
24971	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...
24972	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...
24973	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...
24974	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...
24975	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...

24976 rows × 2 columns

Figure 27: Inter and intra comparison list

2.3.1 Inter comparison list:

Now that we have the comparison list gathering all the images we must extract the comparison list of the inter images, i.e. the comparison of each
image with images that do not belong to the same folder (class) as it.
To do this, we created a new data frame by browsing the one we already have everything
keeping only those which had different subjects (classes). After finalizing
In this step, we applied our model to the entire resulting data frame. In fact, we will compare the predicted class for image 1 with the predicted class for image 2. since we only kept the images of different people.This certainty will help us later to calculate the reliability rate of this algorithm, by calculating the number of valid comparison

2.3.1.1 CNN:

	image1	image2	predicted_person1	predicted_person2
0	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	12	10
1	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	103	108
2	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	108	105
3	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	105	25
4	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	25	100
...	...	...	...	...
632	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	65	70
633	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	70	55
634	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	55	5
635	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	5	1
636	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	1	108

637 rows × 4 columns

Figure 28 : Inter comparison list with CNN

```
[ ]   modele_reliability2=0
      for index,row in df_compare_finale.iterrows():
        if row['predicted_person1']==row['predicted_person2']:
          modele_reliability2=modele_reliability2+1
```

```
[18]  print("les nombres des  erreur dans les comparaison interclasse = ",modele_reliability2)

      les nombres des  erreur dans les comparaison interclasse =  10
```

Figure 29 : Test inter comparaison

- using this “CNN” model we found only 10 false comparison

2.3.1.2 VGG :

	image1	image2	predicted_person1	predicted_person2
0	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	12	10
1	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	103	108
2	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	108	105
3	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	105	25
4	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	25	100
...	...	...	...	...
632	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	65	70
633	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	70	55
634	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	55	5
635	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	5	1
636	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	1	108

637 rows × 4 columns

Figure 30 : Inter comparison list with VGG_Face

```
print("les nombres des  erreur dans les comparaison interclasse = ",modele_reliability2_vgg)
les nombres des  erreur dans les comparaison interclasse =  5
```

Figure 31 : Test inter comparaison

- using “VGG” model we found only 5 false comparison

2.3.2 Intra comparison list:

This part is reserved for the intra comparison list, i.e. the comparison of an image with a folder that matches the same subject (class) and check if our models shows us whether it is the same person or not. To achieve this result, what we have thought to do is exactly the same as what we did in the inter part. For better explain, we will need the same starting data frame containing the list of all possible comparisons but the difference is that we will keep only the pairs or the comparison is made on the same subject. Like that, we will have subsequently
all the intra list that we need and we re-apply our models which in this case we will logically display the same predicted class for the 2 images since we are doing our studies on people similar.
Subsequently in the evaluation part, we will calculate the total of the erroneous vAlues.

2.3.2.1 VGG :

```
df_comparaison_extract_inter-vgg
```

	image1	image2	predicted_person1	predicted_person2
0	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	12	12
1	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	103	103
2	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	108	108
3	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	105	105
4	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	25	25
...	...	...	...	...
107	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	8	8
108	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	21	21
109	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	36	36
110	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	94	94
111	/content/ColabNotebooks/MyDrive/faces_set/Imag...	/content/ColabNotebooks/MyDrive/faces_set/Imag...	40	40

112 rows × 4 columns

Figure 32 : intra comparison list with VGG_Face

```
[ ]  print("les nombres des  erreur dans les comparaison intra = ",modele_reliability_inter_vgg)

     les nombres des  erreur dans les comparaison intra =  5
```

- using “VGG” model we found only 5 false intraclass comparison

2.3.2.1 CNN :

```
print("les nombres des  erreur dans les comparaison intra classe = ",modele_reliability_intra)

les nombres des  erreur dans les comparaison intra classe =  8
```

- Using “CNN” model we found only 8 false intraclass comparison

3. Evaluation and comparison of results:

At this point, we will assess the value of our models in achieving the operational goals that started the data mining process. We will look for the reasons why the model would not be satisfactory for the client. In our case, after trying different models and different methods, and after comparing your results with those found for our comrades with the same dataset we came to the conclusion that 'Facenet' and 'Openface' are the best, since by displaying the number of errors found (inter and intra) in the test of each algorithm no error is found for the latter two proving their efficiency, except for a few errors have been found for 'VGG' and 'CNN' but the number remains very small if not negligible. As a conclusion, the 4 algorithms used gave extraordinary results which proves the success of our data preparation phase and also the phase of detection of the different faces.The figures below perfectly illustrate everything we have just said, the premier couple 34.1 and 34.2 are diagrams describing the number of images recognized on the set of images in the list. The following figures show the total number of errors made by each algorithm

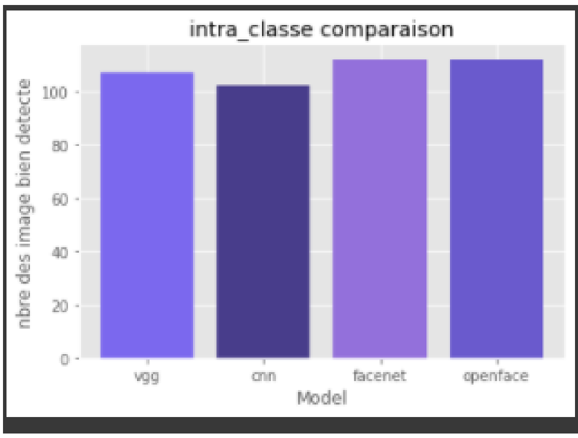


Figure 33.1: Comparaison intern

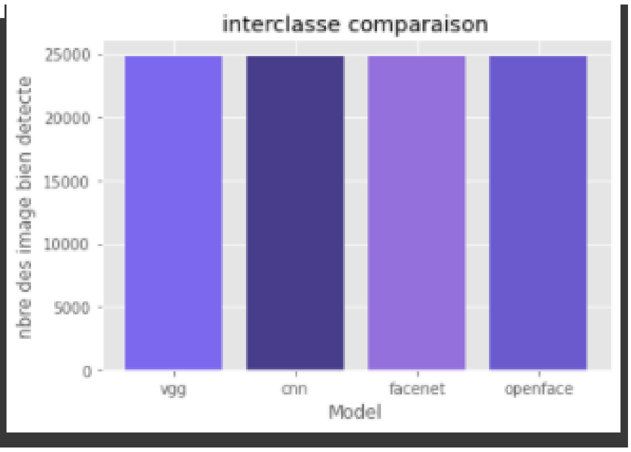


Figure 33.2 : Comparaison Intra

	vgg	cnn	facenet	openface
nbre de comparasion inter_classe	24863	24863	24863	24863
nbre des erreur	5	10	0	0

Figure 34.1: Comparaison intern DF

	vgg	cnn	facenet	openface
nbre de comparasion intra_classe	112	112	112	112
nbre des erreur	5	8	0	0

Figure 34.2: Comparaison intra DF

V.Conclusion

The problem that we handled in this project was trying to detect and recognize faces using Data Science tools and technologies.
Following the IBM Master Plan Methodology, we managed to choose the best models in order to get the accurate predictions.

Using multiple Machine Learning and Deep Learning techniques such as the Transfer Learning and the CNN, we succeeded in creating good predictive models that will help to detect face from an image , then extract the face and finally can recognize it through a dataset of faces.

Our next step in this project is deploying our application using each of these frameworks : Django or flask .