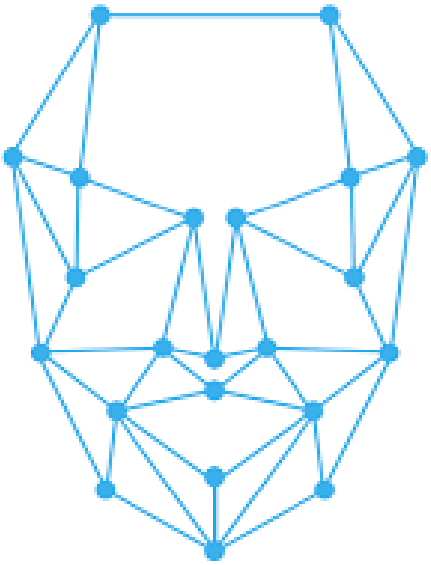
Data Science project report



Genical Project 2 : Facial Recognition

Elaborated by :

Fosso Steve - Motcheho Romuald

Bouaziz Mohamed Anis - Bagbag Khalil

Ghedass Hela - Arous Souhir

Harketi Yassine

Supervised by :

Mrs Dorra Trabelsi - Mrs Khedher Oussayma

2020 - 2021

Abstract

This report has been prepared by students at ESPRIT University under a partnership with the society Genical.

The project fits into academic purposes and focuses on Artificial Intelligence especially Facial Recognition. A 4-months project implemented from 21 January to 20 May 2021 which has 4 main features : Face detection, facial attributes detection, face recognition and facial emotions detection.

This report provides an overview of the project and presents in detail the different stages that allowed its creation.

Acknowledgement

We would like to express our special thanks to our teacher Mrs Dorra Trabelsi as well as Mrs Oussayma Khedher who supervised the realization of the project and gave us the opportunity to be part of a very rewarding partnership. We are indebted to them for their patience, continuous support and enthusiasm.

We would also like to express our sincere gratitude to all the Genical team for the time they dedicated to us, the resources made available to us and most importantly, for their trust.

Table of contents

General introduction

Face detection is the process of finding a face in an image. If you've ever used a camera that detects a face and draws a box around it to auto-focus, you've seen this technology in action. On its own, it isn't nefarious, face detection only focuses on finding a face, not the identity behind it.

Facial recognition is a way of identifying or confirming an individual's identity using their face. This process is used for verification, such as in a security feature on a newer smartphone, or for identification, which attempts to answer the question "Who is in this picture?".

Analysis of face attributes is the step that maps faces, often by measuring the distance between the eyes, the shape of the chin, the distance between the nose and mouth, then converts that into a string of numbers or points, often called a "faceprint". Moreover, face attributes detection can be used to determine the color of the hair, the color of the skin, the age, if a person wears a mask or not, ect.

Emotion detection uses AI techniques to detect the mood of an individual. It's the task of recognizing a person's emotional state from facial expressions, for example, anger, happiness, confusion, sadness.

As part of a partnership with the society Genical , we are working on developing a multi purpose face API that detects, recognizes, and analyzes human faces, attributes and emotions in images and videos in order to improve working and learning conditions in the educational field.

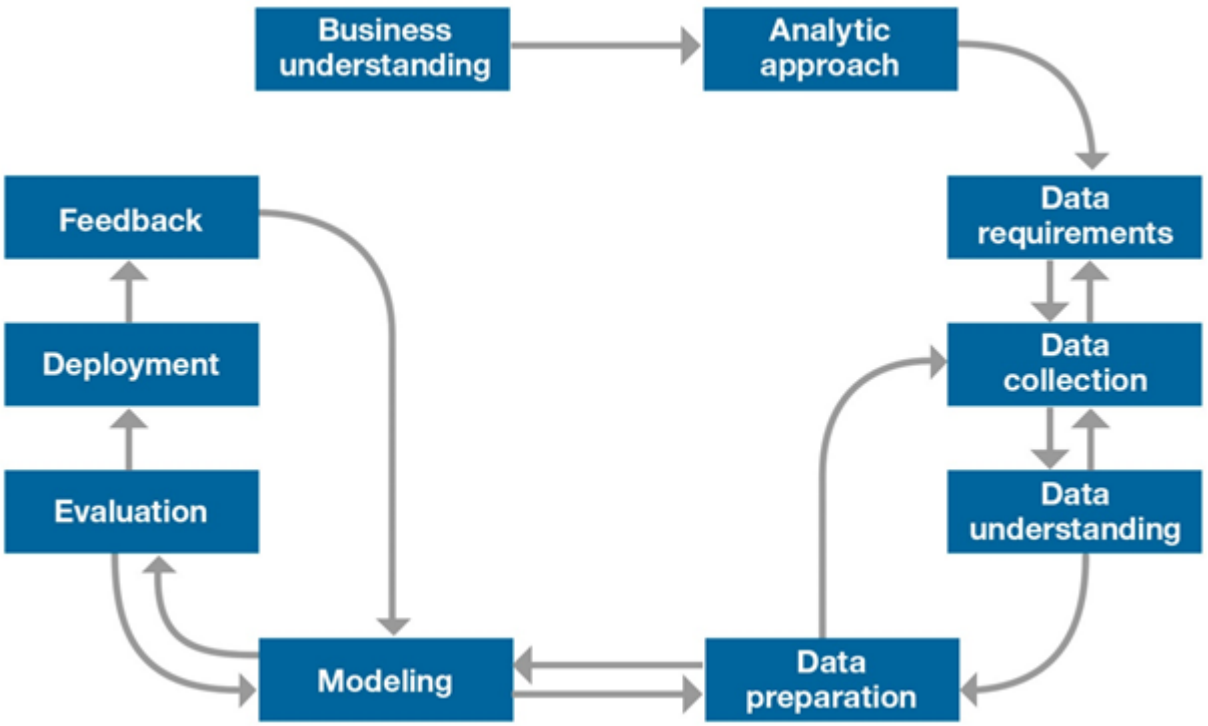
Business strategy

To identify the key steps we will take to reach our business goals, we need a business-oriented strategy.

It's a very important step in order to limit the huge number of existing algorithms and therefore, understand and answer the client's needs.

To define it, a methodology is a set of methods, rules and postulates employed by a discipline which is, in our case, Data Science.

We chose the **IBM Master Plan** that is based on a problem-solving logic and it's schemed as follows :



Chapter I : Business understanding

1. Business objectives

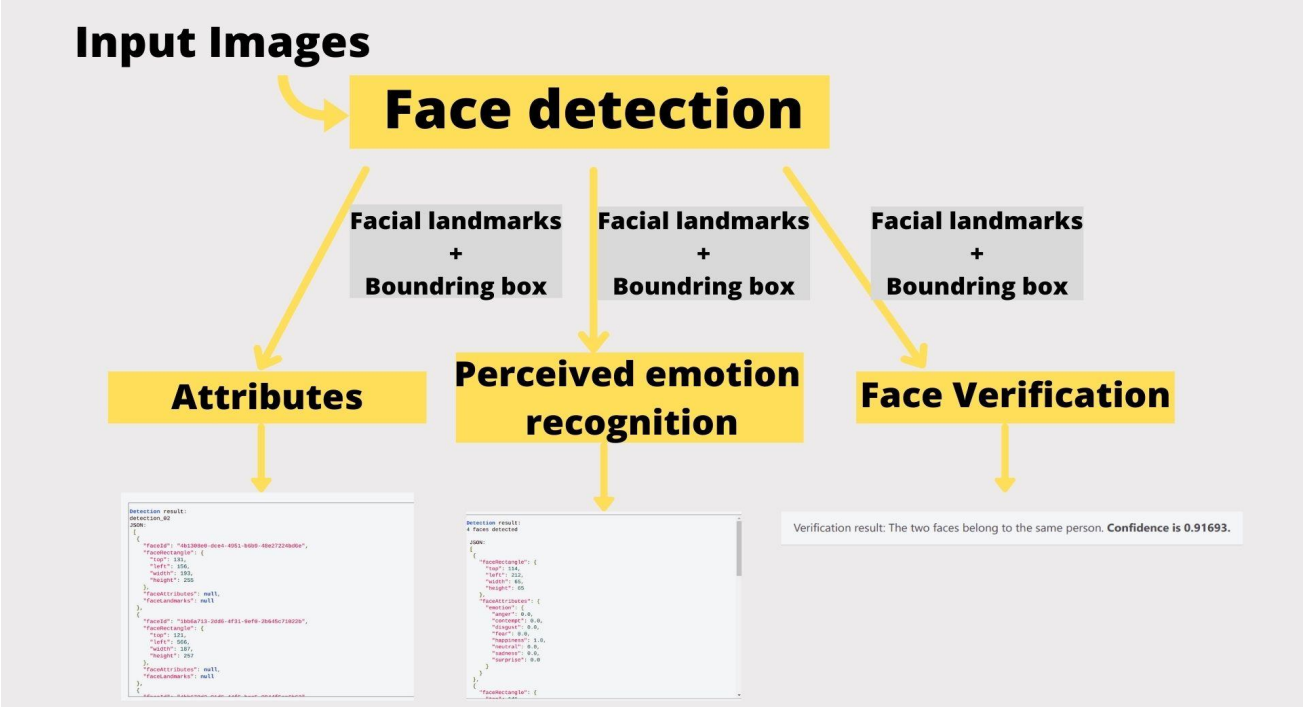
- Facial detection to control comings and goings in schools
- Facial recognition in order to optimize functions and improve safety in the educational field. For example, we can use it to detect unwanted persons, check the students absence, verify the students identity before an exam or before giving access to a private platform or course, ect.
- Facial attributes detection to identify the person in a better way and with more precision. For example, giving access to a person may depend on some characteristics
- Mask detection to for health reasons, especially at this time with the coronavirus problem, we must ensure that sanitary measures are respected within our educational establishments
- Emotions detection to improve safety and pedagogy by analyzing student attendance, detecting excessive stress, uneasiness, anger, cries, rage, ect. First step before any improvement is to know what needs to be improved.

2. Data Science objectives

- First of all we need to identify the suitable technologies for our business objectives.
- Second, our goal is to train and deploy fast and efficient Deep Learning models, for that we will use pretrained models and transfer learning.
- Last but not least we will Implement the API in a demo web application that is going to be a Django web app.

3. Analytic approach

As a first step we have to build a model for Face detection using a dataset, after that the output which is a facial landmarks and a Bound boxes will be used for the other models, for the attributes and emotional recognition models the output is a json file and for the Face verification it will be a confidence score .



Chapter II : Data collecting and understanding

1. Data requirements

Our Data should have both low and high quality images for specific use and detection accuracy.
The number of images should be considerable due to face tracking in specific videos and also face authentication at least 5 to 10 images.
For specific images we should also have different kinds of versions especially for emotion detection and face verification.
We will also need : images of people wearing a mask and others of people not wearing a mask.
We will manage that the amount of images in each class will be approximately near to each other .

2. Data collection

2.1. Internal data

2.1.1. Facial attributes detection

It's a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations.

The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including

This dataset is great for training and testing models for face detection, particularly for recognising facial attributes such as finding people with brown hair, are smiling, or wearing glasses.

[Link:https://www.kaggle.com/jessicali9530/celeba-dataset?select=list_bbox_celeba.csv](https://www.kaggle.com/jessicali9530/celeba-dataset?select=list_bbox_celeba.csv)

2.1.2. Mask detection

The dataset contains approximately 1370 images.

it is created by Prajna Bhandar containing images of human faces, it's an artificial but still real-world applicable dataset.

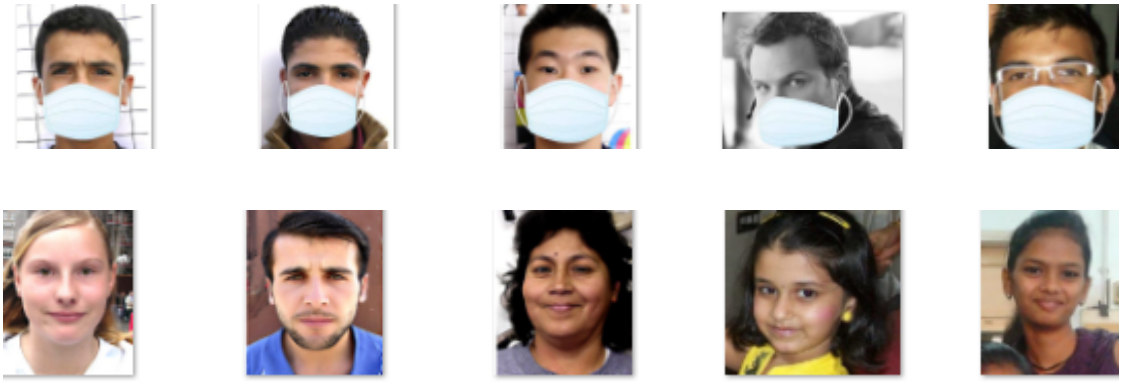


Figure: overview of the dataset

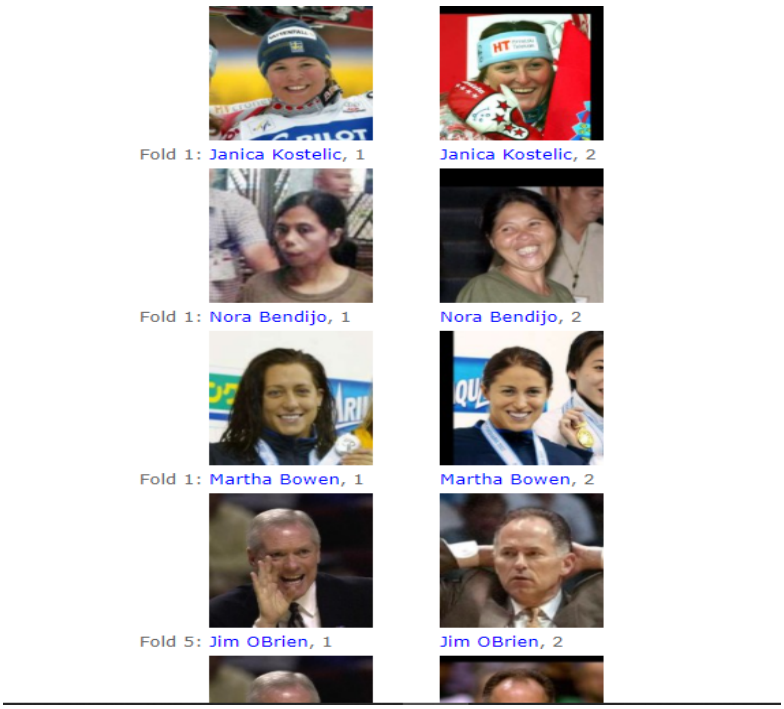
2.1.3. Emotion detection

Fer2013 contains approximately 30,000 facial RGB images of different expressions with size restricted to 48×48, and the main labels of it can be divided into 7 types: 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral. The Disgust expression has the minimal number of images – 600, while other labels have nearly 5,000 samples each.
Link to upload the dataset :
https://paperswithcode.com/sota/facial-expression-recognition-on-fer2013

Figure: Overview of the images in the data set

2.1.4. Face verification

LFWcrop is a cropped version of the Labeled Faces in the Wild (LFW) dataset, keeping only the center portion of each image. It was created due to concern about the misuse of the original LFW dataset, where face matching accuracy can be unrealistically boosted through the use of background parts of images.
Link:https://conradsanderson.id.au/lfwcrop



2.2. External data

2.2.1. Mask detection

We decided to manually add other data :

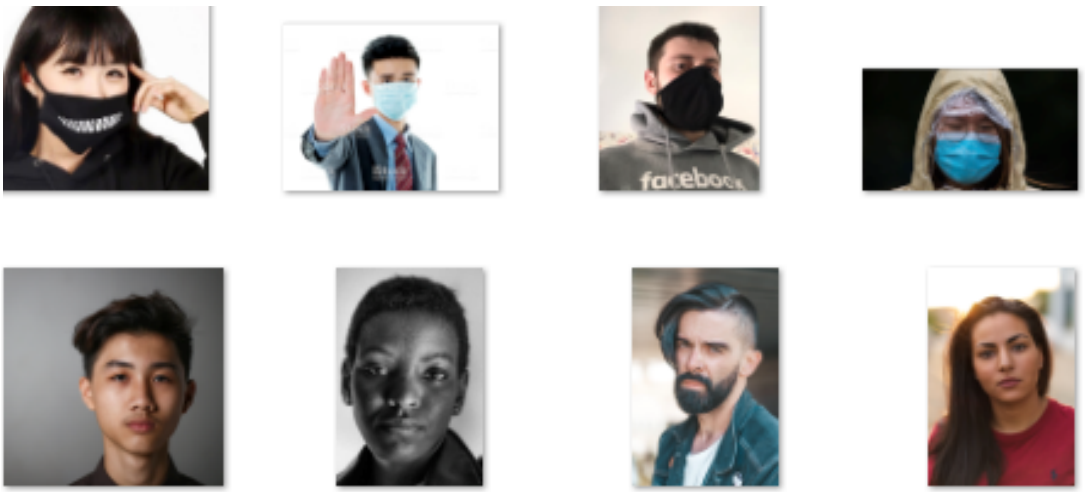


Figure: Data added manually

2.2.2. Emotion detection

In order to reduce the gap between the data numbers of the emotions we decided to scrape images from google, process them and add them to our initial database.
Using an image downloader chrome plugin we managed to obtain at the end a number of 4177 additional images.

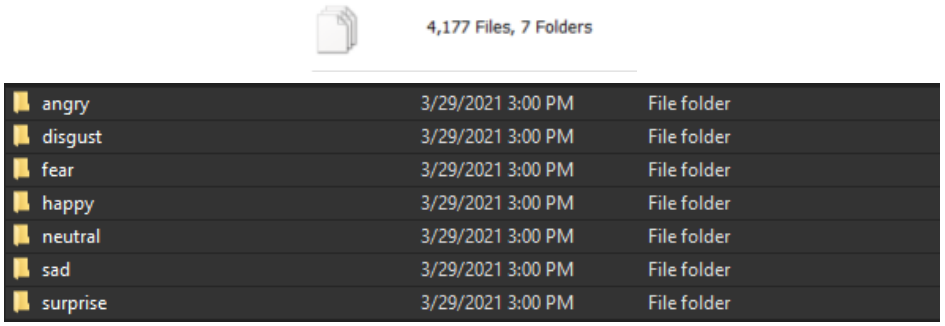


Figure: Image folder overview after scraping

Cleaning and prepare external data

But our newly obtained data is :

-messy
-might not contain faces
-might contain 2 or more faces
-is RGB and not grayscale
-is not 48*48
-is files and not pixels

We need to clean it and convert it to its appropriate form to then add it to fer2013.After detecting and cropping faces each face needs to be turned into grayscale and resized to its proper size 48*48.

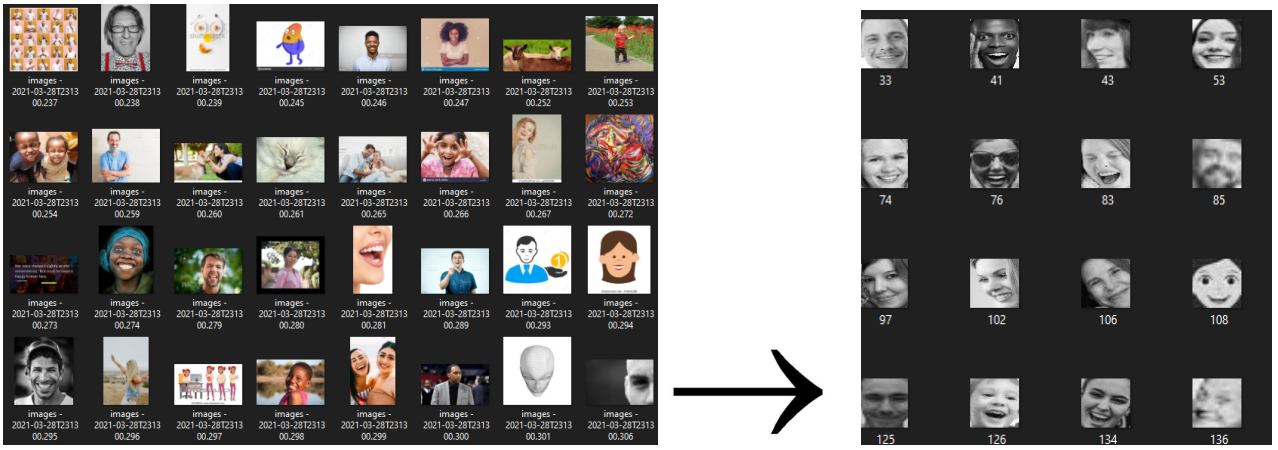


Figure: Data cleaning of the images after scraping

Then each image is transformed into a string of pixels , labeled then added to our original dataset which is now ready for the normalization process .

After all the necessary transformations we added the photos to our initial database and here is the result after adding the 4177 photos.

Figure: Overview of our dataset after adding the new pictures

3. Data understanding and visualization

3.1. Facial attributes detection

3.1.1. Familiarization with the dataset

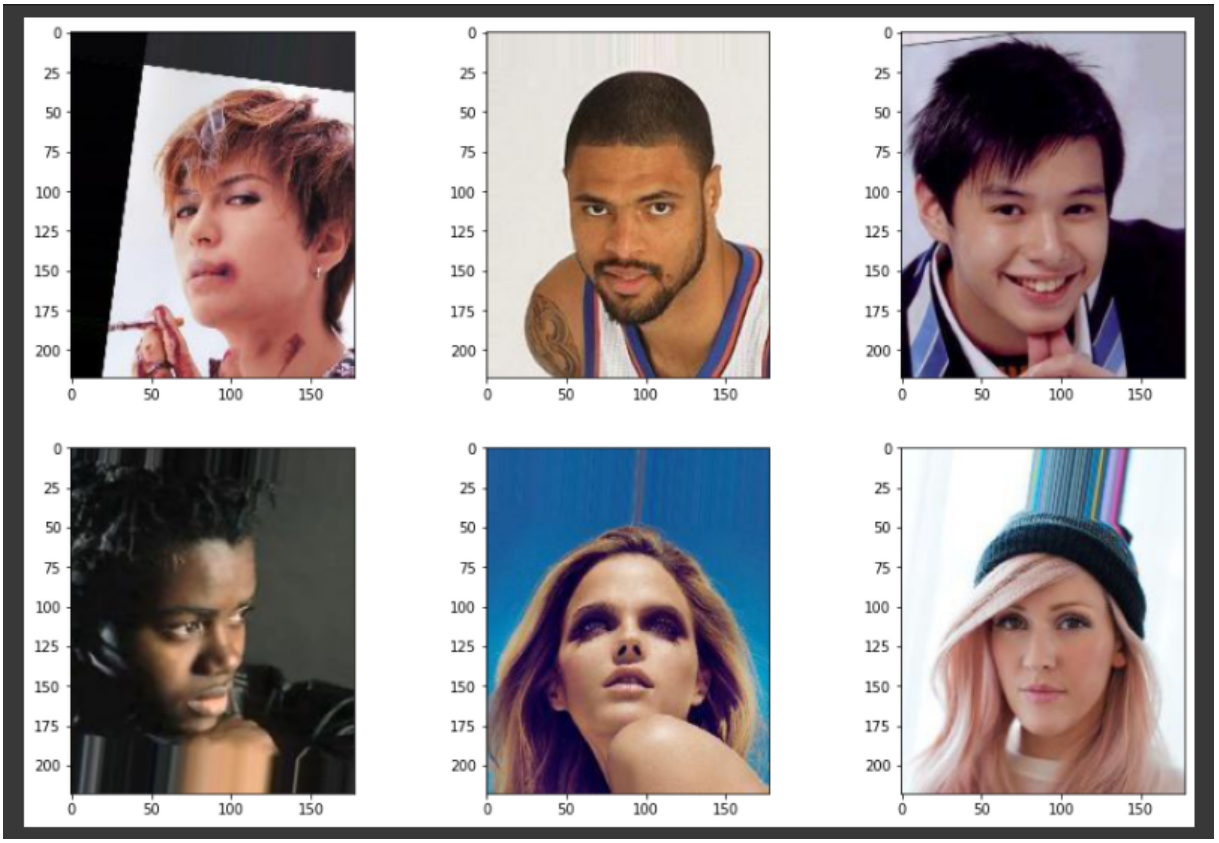
A nice, wide, and diversified dataset to work with is the **CelebA dataset**.

It is a large-scale face attributes dataset with more than 200K celebrity images, covering a large amount of variations, each with 40 attribute annotations.

CelebA Facial Attributes:

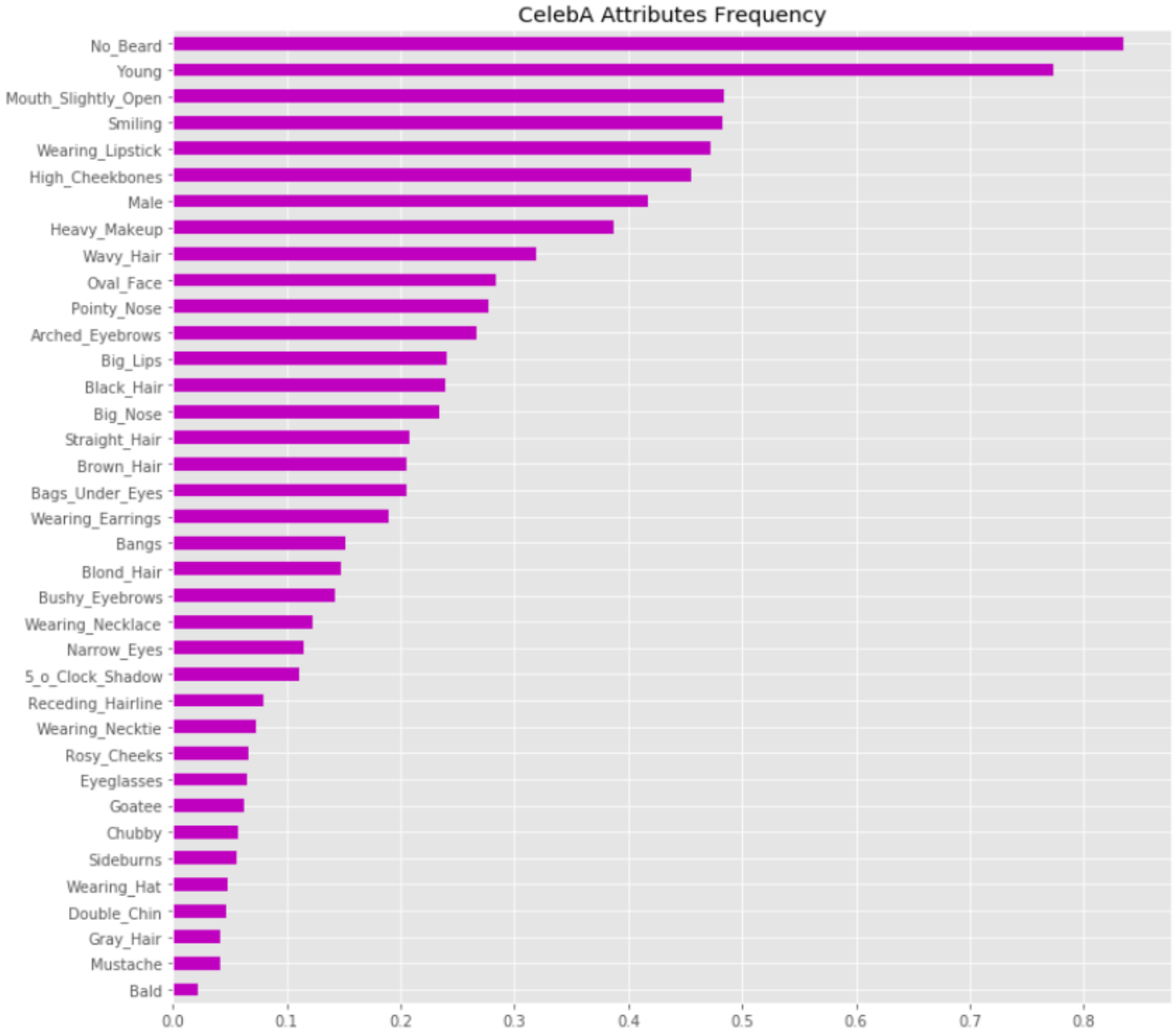
0: 5_o_Clock_Shadow
1: Arched_Eyebrows
2: Attractive
3: Bags_Under_Eyes
4: Bald
5: Bangs
6: Big_Lips
7: Big_Nose
8: Black_Hair
9: Blond_Hair
10: Blurry
11: Brown_Hair
12: Bushy_Eyebrows
13: Chubby
14: Double_Chin
15: Eyeglasses
16: Goatee
17: Gray_Hair
18: Heavy_Makeup
19: High_Cheekbones
20: Male
21: Mouth_Slightly_Open
22: Mustache
23: Narrow_Eyes
24: No_Beard
25: Oval_Face
26: Pale_Skin
27: Pointy_Nose
28: Receding_Hairline
29: Rosy_Cheeks
30: Sideburns
31: Smiling
32: Straight_Hair
33: Wavy_Hair
34: Wearing_Earrings
35: Wearing_Hat
36: Wearing_Lipstick
37: Wearing_Necklace
38: Wearing_Necktie
39: Young

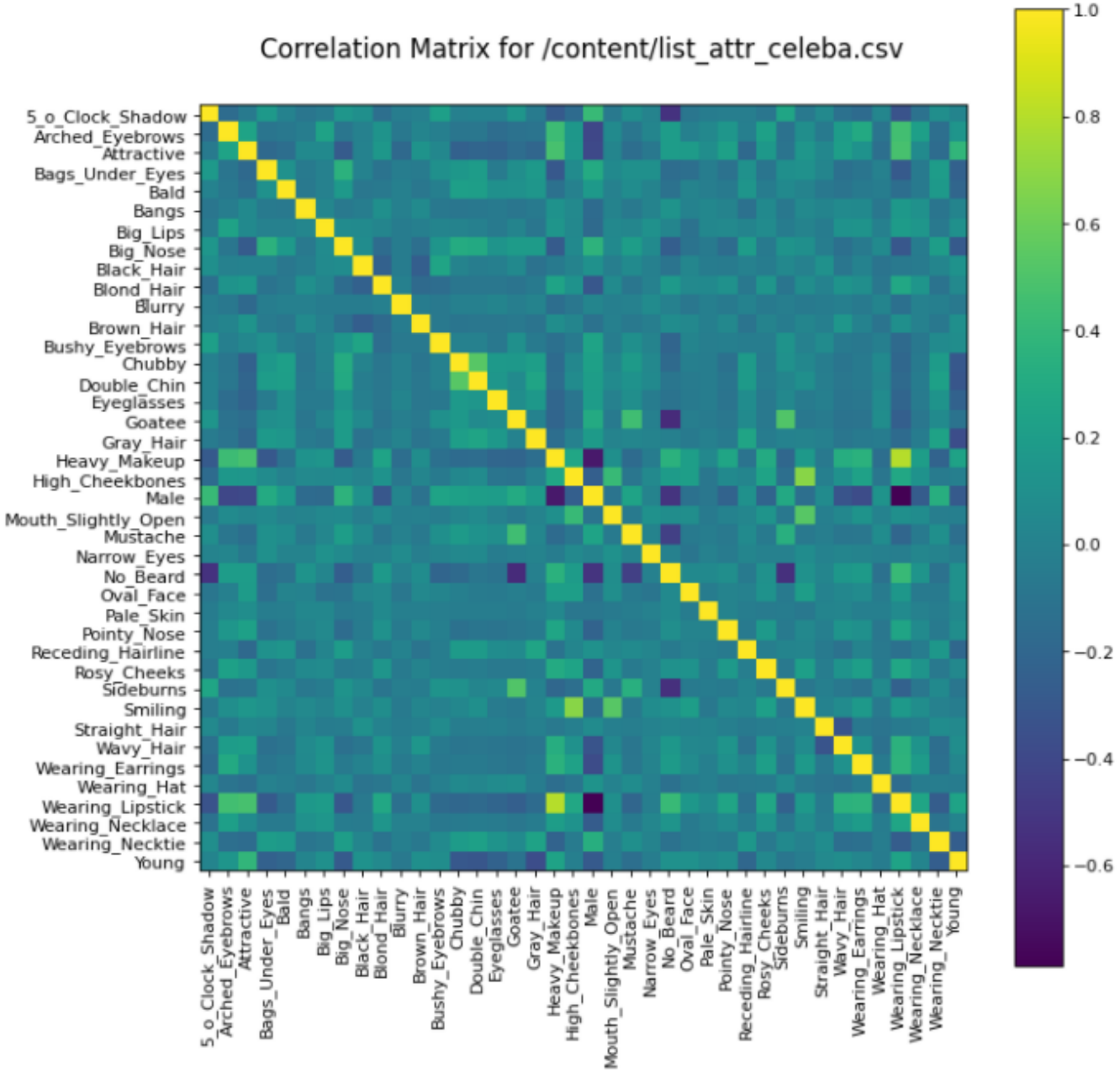
3.1.2. Data Visualization



Data Sample

Here, we can clearly observe a data imbalance problem. The frequency of the facial attributes vary a lot, rather than being roughly equal to each other. There are rare attributes (Bald, Mustache, Double_Chin, etc) with a frequency below 10%, and a couple of very common attributes (No_Beard, Young) with a frequency above 70%.


CelebA Attributes Frequency
No_Beard
Young
Mouth_Slightly_Open
Smiling
Wearing_Lipstick
High_Cheekbones
Male
Heavy_Makeup
Wavy_Hair
Oval_Face
Pointy_Nose
Arched_Eyebrows
Big_Lips
Black_Hair
Big_Nose
Straight_Hair
Brown_Hair
Bags_Under_Eyes
Wearing_Earrings
Bangs
Blond_Hair
Bushy_Eyebrows
Wearing_Necklace
Narrow_Eyes
5_o_Clock_Shadow
Receding_Hairline
Wearing_Necktie
Rosy_Cheeks
Eyeglasses
Goatee
Chubby
Sideburns
Wearing_Hat
Double_Chin
Gray_Hair
Mustache
Bald
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


Correlation Matrix for /content/list_attr_celeba.csv
5_o_Clock_Shadow
Arched_Eyebrows
Attractive
Bags_Under_Eyes
Bald
Bangs
Big_Lips
Big_Nose
Black_Hair
Blond_Hair
Blurry
Brown_Hair
Bushy_Eyebrows
Chubby
Double_Chin
Eyeglasses
Goatee
Gray_Hair
Heavy_Makeup
High_Cheekbones
Male
Mouth_Slightly_Open
Mustache
Narrow_Eyes
No_Beard
Oval_Face
Pale_Skin
Pointy_Nose
Receding_Hairline
Rosy_Cheeks
Sideburns
Smiling
Straight_Hair
Wavy_Hair
Wearing_Earrings
Wearing_Hat
Wearing_Lipstick
Wearing_Necklace
Wearing_Necktie
Young
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

3.2. Mask detection

The data(images) is divided into three folders in Prajna ‘s work :

train , test and validation folders, each folder contains folders of people wearing masks and other not ,we decided to share the data only in two folders:with_mask and without_mask and divide the data (train and test) later and choose the validation in the notebook.

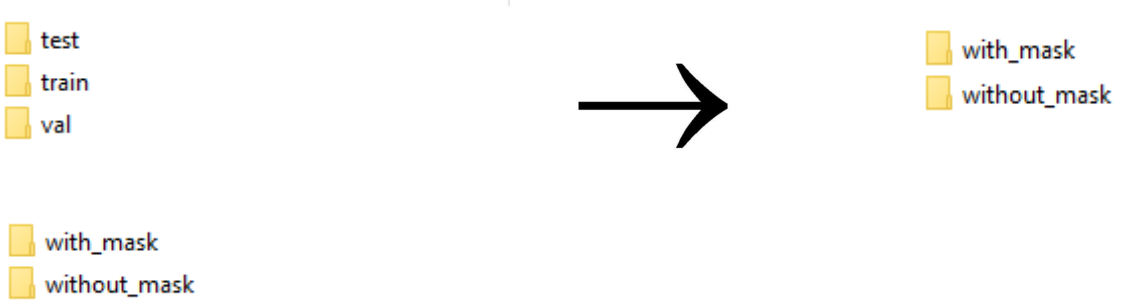


Figure: Folders overview

As we receive images from the dataset we will need to label the data :

0 : A person is wearing a mask
1 : A person is not wearing a mask

To do that we needed to do some code treatment:we load images from each folder and we labeled them with 0 or one 1.

To deal with data:

```
my_data1=load_imgs_from_folder('with_mask')
 my_data2=load_imgs_from_folder('without_mask')
```

We concatenate data:

```
my_data=my_data1+my_data2
```

To deal with labels:

```
my_label1=fill_label('with_mask')
my_label2=fill_label('without_mask')
```

We concatenate labels :

```
my_label=my_label1+my_label2
```

This is the result :

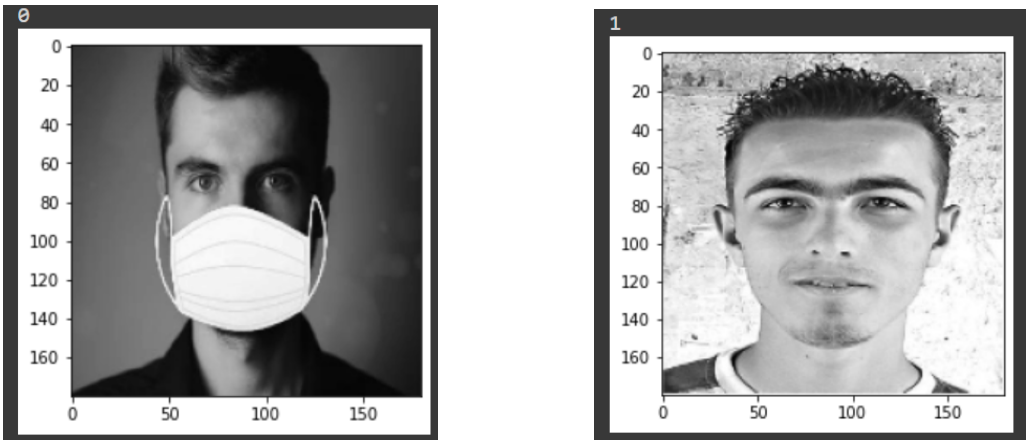


Figure: Labeled data

Our data is well balanced as the amount of data in each class(mask/No mask) is approximately the same :

```
print(len(my_data1))
```

```
735
```

```
print(len(my_data2))
```

```
767
```

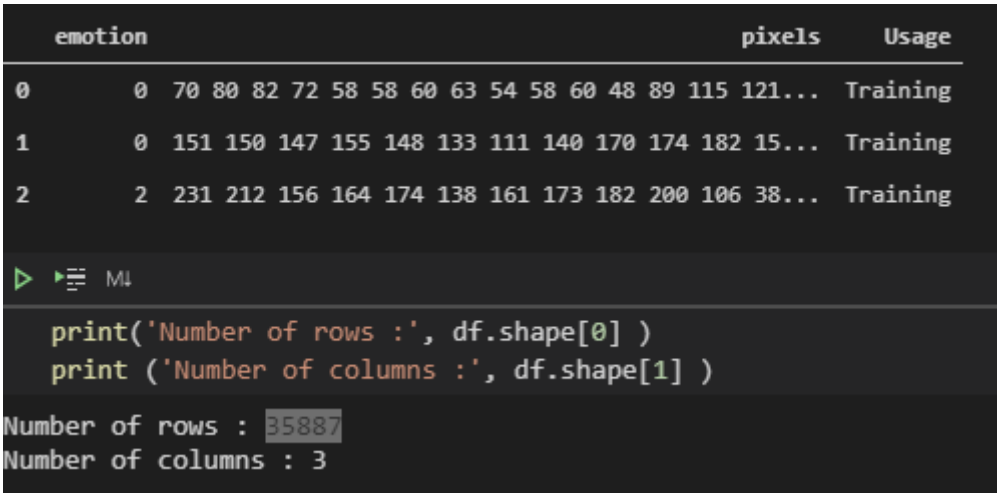
3.3. Emotion recognition

3.3.1. Familiarization with the dataset

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories:
-0=Angry,-1=Disgust ,-2=Fear ,- 3=Happy ,-4=Sad ,-5=Surprise ,-6=Neutral

3.3.2. Composition of the dataset

The dataset contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order. Also it contains only the "pixels" column and our task is to predict the emotion column.



	emotion	pixels	Usage
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training
1	0	151 150 147 155 148 133 111 140 170 174 182 15...	Training
2	2	231 212 156 164 174 138 161 173 182 200 106 38...	Training

```
print('Number of rows :', df.shape[0] )
print ('Number of columns :', df.shape[1] )
```

```
Number of rows : 35887
Number of columns : 3
```

Figure: Shape of our data

3.3.3. Data Visualization

The data is distributed in a heterogeneous way since for the disgusted emotion we have only 547 images while for the happy emotion we have more than 8000 images.

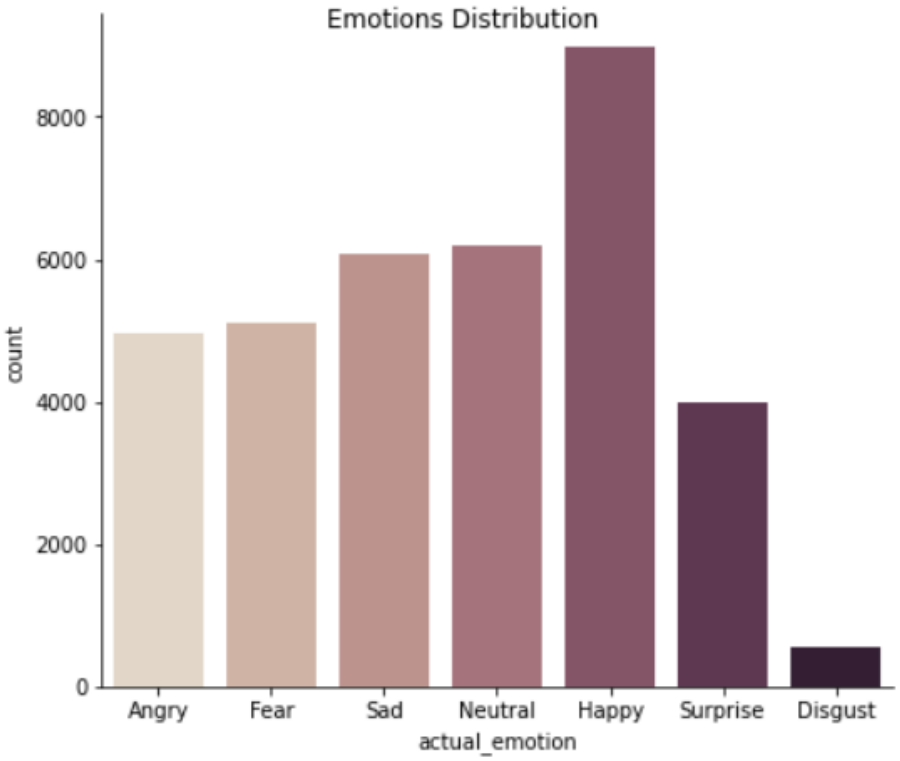


Figure: Emotion distribution in the dataset

3.4. Face verification

3.4.1. Familiarization with the dataset

The LFW dataset contains **13233** images of **5749** people and among those people **1680** with two or more images.

Each image has a 150x150 size in rgb.

Chapter III : Data preparation

1. Facial attributes detection

1.1. Split Data and Data Augmentation

Partition file is read to get which image belongs to which set (train, validation or test)

```
eval_file = open('/content/drive/MyDrive/CelebAa/Eval/list_eval_partition.txt', 'r').read().splitlines() # Read the partition file
eval_file = list(filter(None, eval_file)) # remove any empty cell in the list
```

```
# Train set is taken
eval_file_train = [eline for eline in eval_file if eline[-1] == '0']
eval_file_train = [eline.split()[0] for eline in eval_file_train]


# Training data is written into the text file.
with open('/content/drive/MyDrive/CelebAa/TRAIN_data.txt', 'w') as f:
    for item in labels_file[:int(eval_file_train[-1].split(".")[0])]:
        f.write("%s\n" % ("img_align_celeba/" + item))
```

```
# Validation set is taken
eval_file_validation = [eline for eline in eval_file if eline[-1] == '1']
eval_file_validation = [eline.split()[0] for eline in eval_file_validation]

# Validation data is written into the text file.
with open('/content/drive/MyDrive/CelebAa/VALIDATION_data.txt', 'w') as f:
    for item in labels_file[int(eval_file_validation[0].split(".")[0])-1:int(eval_file_validation[-1].split(".")[0])]:
        f.write("%s\n" % ("img_align_celeba/" + item))
```

```
# Test set is taken
eval_file_test = [eline for eline in eval_file if eline[-1] == '2']
eval_file_test = [eline.split()[0] for eline in eval_file_test]

# Test set is written into the text file
with open('/content/drive/MyDrive/CelebAa/TEST_data.txt', 'w') as f:
    for item in labels_file[int(eval_file_test[0].split(".")[0])-1:]:
        f.write("%s\n" % ("img_align_celeba/" + item))
```

```
: f_train[:5] # To see that the train set is read successfully.
```

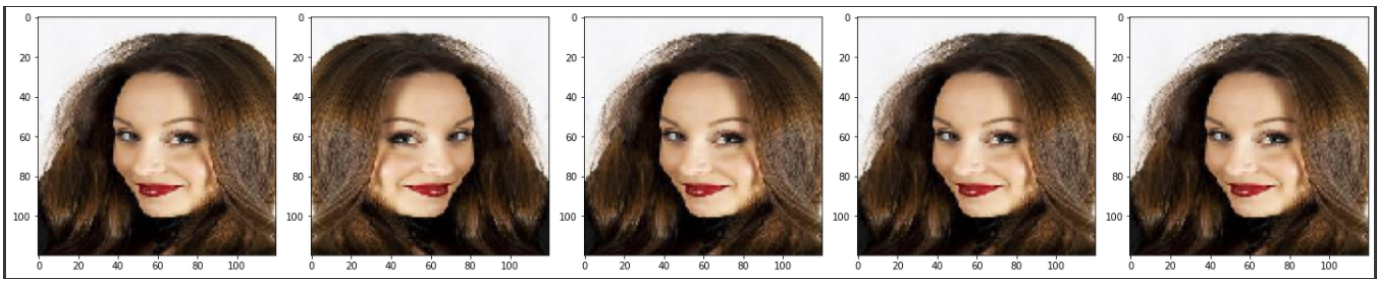
```
: ['img_align_celeba/000001.jpg 0 1 0 0 0 0 0 1 0',
   'img_align_celeba/000002.jpg 0 1 0 0 0 0 0 0 0',
   'img_align_celeba/000003.jpg 1 0 0 0 0 0 0 0 1',
   'img_align_celeba/000004.jpg 0 0 0 0 0 0 0 1 0',
   'img_align_celeba/000005.jpg 0 0 0 0 0 0 0 1 0']
```

```
: f_validation[:5] # To see that the validation set is read successfully.
```

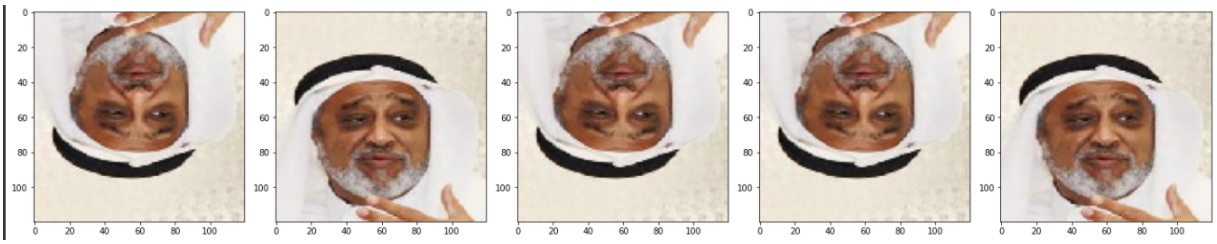
```
: ['img_align_celeba/162771.jpg 0 1 0 1 0 0 0 1 0',
   'img_align_celeba/162772.jpg 1 1 1 0 0 0 0 0 0',
   'img_align_celeba/162773.jpg 0 0 0 0 0 0 0 1 0',
   'img_align_celeba/162774.jpg 1 1 1 0 0 0 0 1 0',
   'img_align_celeba/162775.jpg 0 0 0 0 0 0 0 0 0']
```

```
: f_test[:5] # # To see that the test set is read successfully.
```

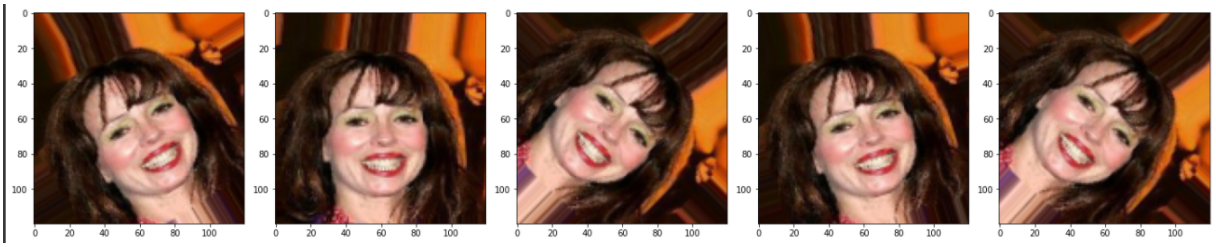
```
: ['img_align_celeba/182638.jpg 0 1 0 0 0 1 0 0 0 0',
   'img_align_celeba/182639.jpg 0 0 0 0 0 0 0 0 0 0',
   'img_align_celeba/182640.jpg 0 1 0 0 0 0 0 1 0',
   'img_align_celeba/182641.jpg 0 1 0 0 0 0 0 1 0',
   'img_align_celeba/182642.jpg 0 1 0 0 0 0 0 0 0']
```



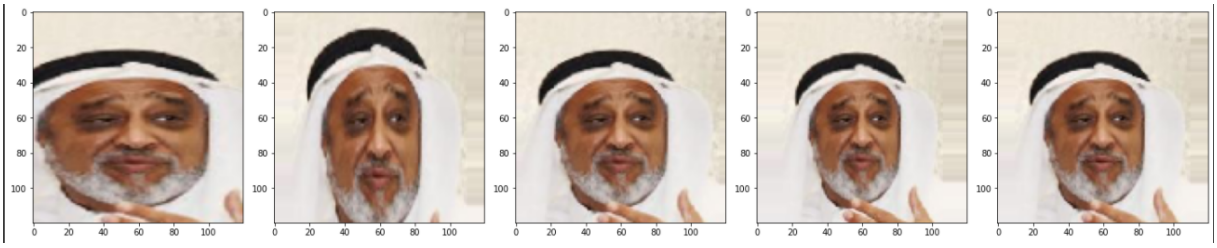
Flip



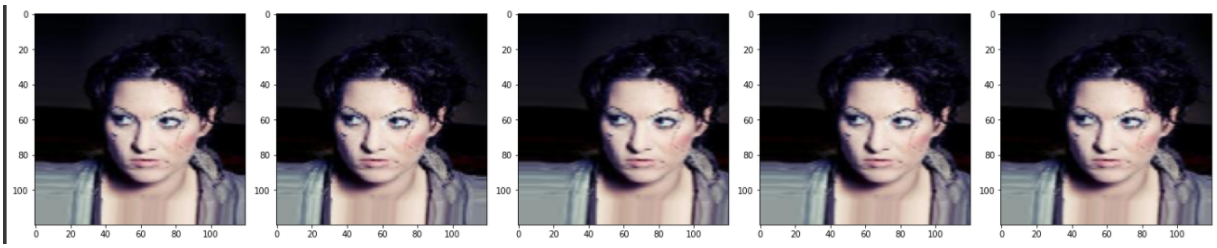
Flip



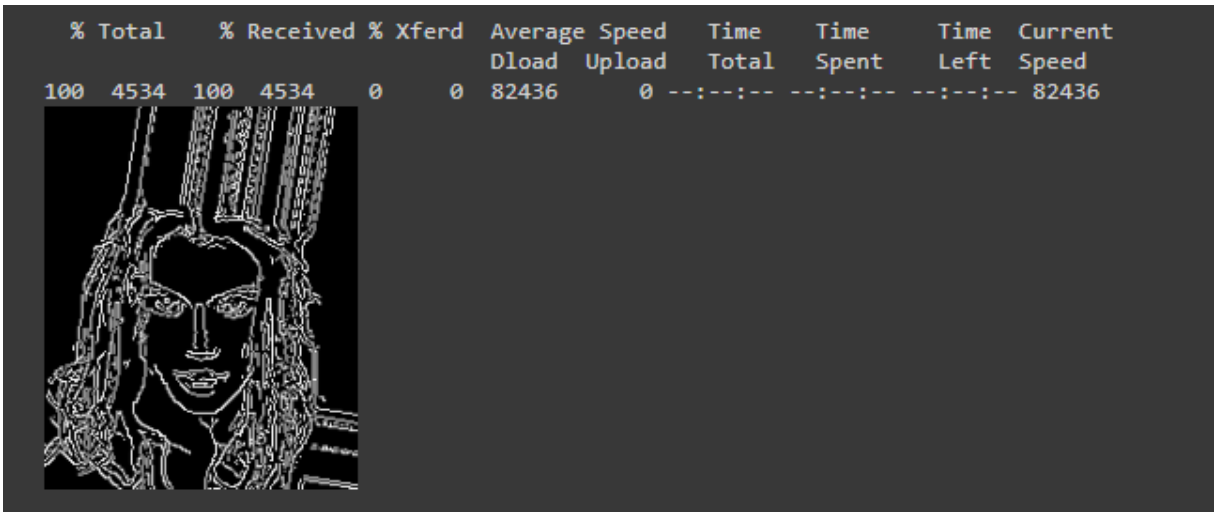
Rotation



Zoom



Rescale



Edge detection

2. Mask detection

2.1. Split Data, Normalization, Encoding, Reshaping

First we applied a certain number of action, with the help of opencv library, on our images such as :

- ➔ Resizing images
- ➔ Transform images from RGB(3 colors channels) to grayscale

Reshaping

After that, we needed to prepare our data before moving to modeling , so we reshaped our data because keras models expect images in a certain shape.

```
my_data=my_data.reshape(my_data.shape[0],180,180,1)
```

Encoding

Then we categorized our labels with the help of keras.

```
num_categories = 2
my_new_label = keras.utils.to_categorical(my_label, num_categories)
```

Split

We split the dataset in 60% training 20% for testing and validation.

```
x_train,x_test,y_train,y_test=train_test_split(my_data,my_new_label,test_size=0.2)
```

Normalization

Deep learning models give good result dealing with numbers between 0 and 1 so we are going to normalize our data by dividing pixels of images by 255.

```
x_train=x_train/255
x_test=x_test/255
```

3. Emotion recognition

3.1. Split Data, Normalization and Data Augmentation

In this part the data normalization is to change the values in our data into a common scale because for some classification models , features with different ranges could seriously affect its output and accuracy .

We did that using the ImgeDataGenerator that makes the normalization and the data augmentation .

```
# using ImageDataGenerator to normalize data and for data augmentation
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
validation_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)

# Spliting Data
xtrain, xtest,ytrain,ytest = train_test_split(faces, emotions,test_size=0.2,shuffle=True)

train_gen = train_datagen.flow(xtrain, ytrain, batch_size)
validation_gen = validation_datagen.flow(xtest, ytest, batch_size)
```

4. Face verification

4.1. Face extraction

We cut the faces of our images with the MTCNN model to focus on the useful part for face verification and recognition.

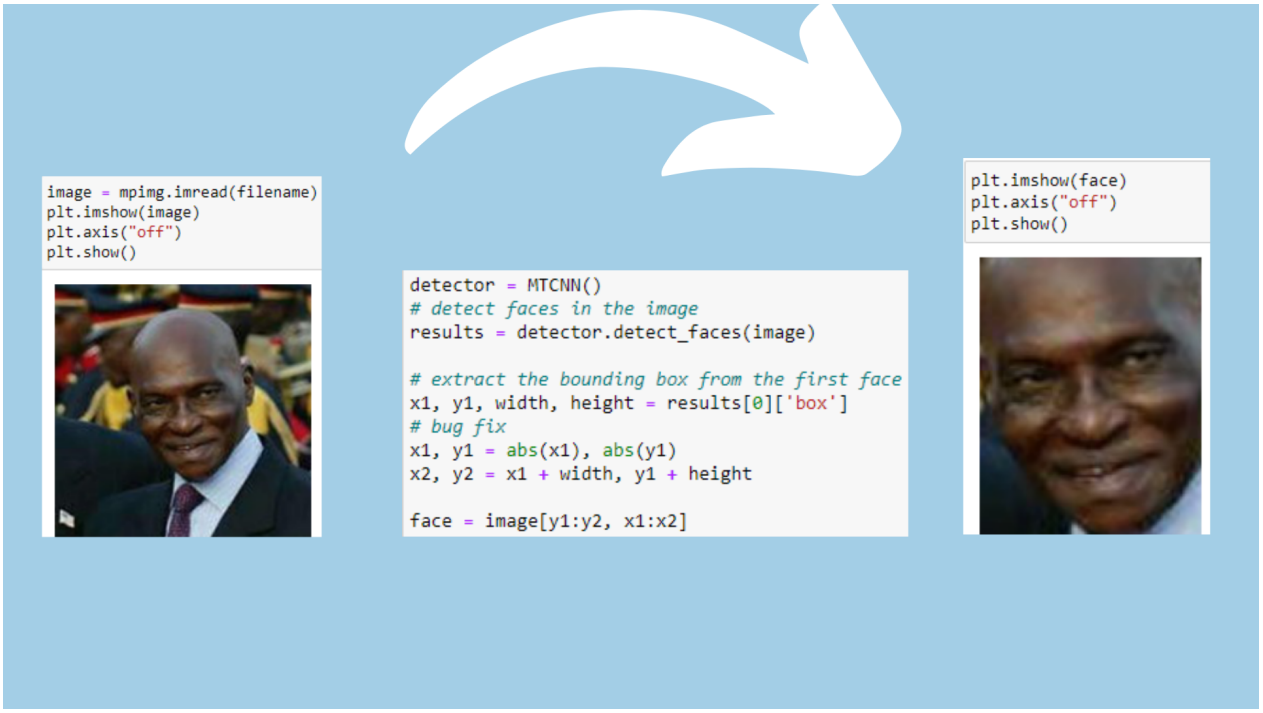


Figure : using MTCNN to crop faces

Chapter IV : Modeling and evaluation

1. Facial attributes detection

In this model, we trained the last hidden layer of **VGG16** from scratch.

Therefore, we removed one hidden layer + output layer.

The layers we added:

- ➔ Dense layer with 4096 units → The last hidden layer we trained from scratch.
- ➔ Dropout layer with rate 0.5 → To prevent overfitting.
- ➔ Dense layer with 10 units and sigmoid activation function. → 10 attributes, multilabel classification

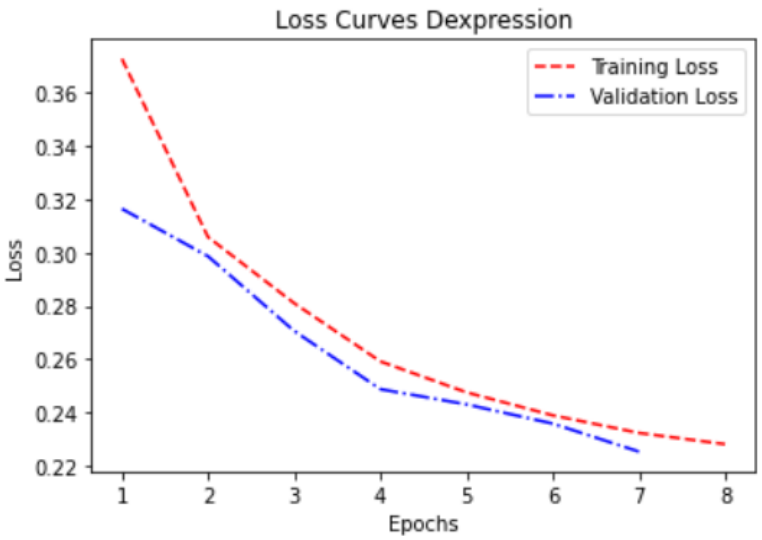
Our problem is a multilabel classification problem because we assign multiple targets (attributes) to each image!

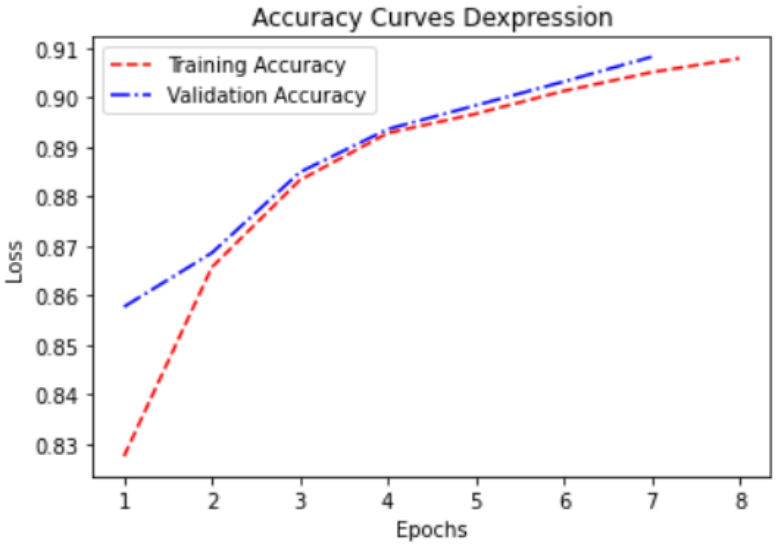
Therefore, sigmoid activation function is used instead of softmax activation function.

```
# Remove last two layers (one hidden layers + Output layer)
VGGmodel.layers.pop()
VGGmodel.layers.pop()
```

```
# Add new layers
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='sigmoid'))
```

```
=================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
```





Testing on the test data to see the binary accuracy :

```
Final test accuracy: 89.805128430809
```

Accuracy per attribute :

```
getPerAcc(f_test)

Calculating the accuracy of each attribute.
This may take about 10 minutes...

Accuracy of each attribute:
male/female: 0.8979060214407374
smile/not: 0.8293758140466887
mustache-beard/not: 0.8810740406772869
bangs/not: 0.9268109407874963
eyeglasses/not: 0.9466486324015629
hat/not: 0.9577697625488428
blonde/not: 0.8926460274521592
pale skin: 0.9579200480913737
attractive/not: 0.7353972547840898
blurry/not: 0.9494038673479611

Average Accuracy: 0.8974952409578197
```

2. Mask detection

2.1. Architecture

Our model is designed to detect if a person wears or doesn't wear a mask , it is a classification model.

Our model is based on convolutional neural network architecture:

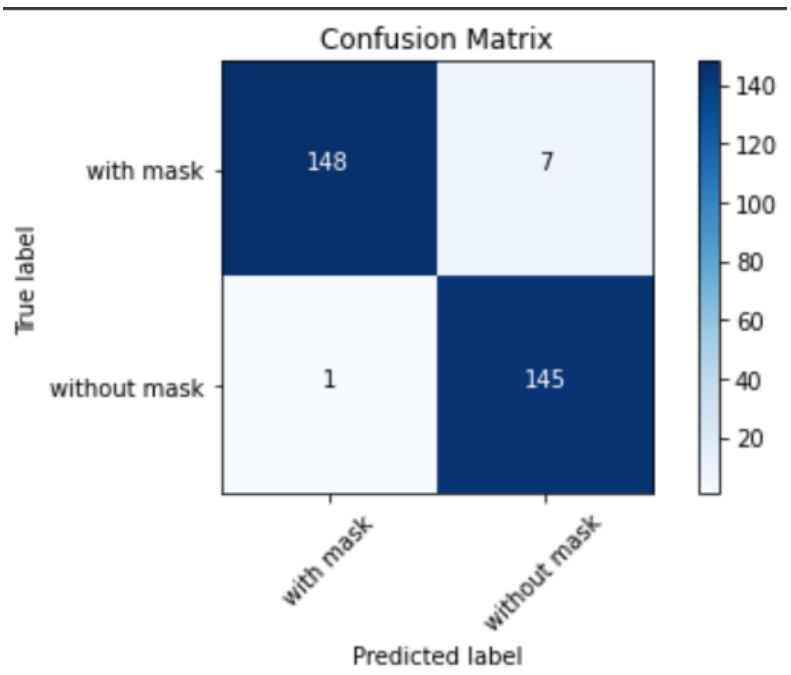
We start by the input image , we apply filters to generate feature maps,then we apply non linearity(often Relu),after that comes the pooling,max or average then for the classification part we apply the flatten , fully connected layer and softmax.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 174, 174, 300)     15000
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 29, 29, 300)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 25, 25, 200)       1500200
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 200)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 70)          126070
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 70)          0
_________________________________________________________________
dropout (Dropout)            (None, 2, 2, 70)          0
_________________________________________________________________
flatten (Flatten)            (None, 280)               0
_________________________________________________________________
dense (Dense)                (None, 128)               35968
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258
=================================================================
Total params: 1,677,496
Trainable params: 1,677,496
Non-trainable params: 0
```

Figure: Summary of the model

2.2. Evaluation

After plotting the confusion matrix we noticed that our model gave good results,
because the false negatives and false positives are much lesser compared to true positives and true negatives.



We trained our model on a certain number of epochs ,and we tracked some important values such as : loss, accuracy, val_loss, val_accuracy and we plotted some curves :

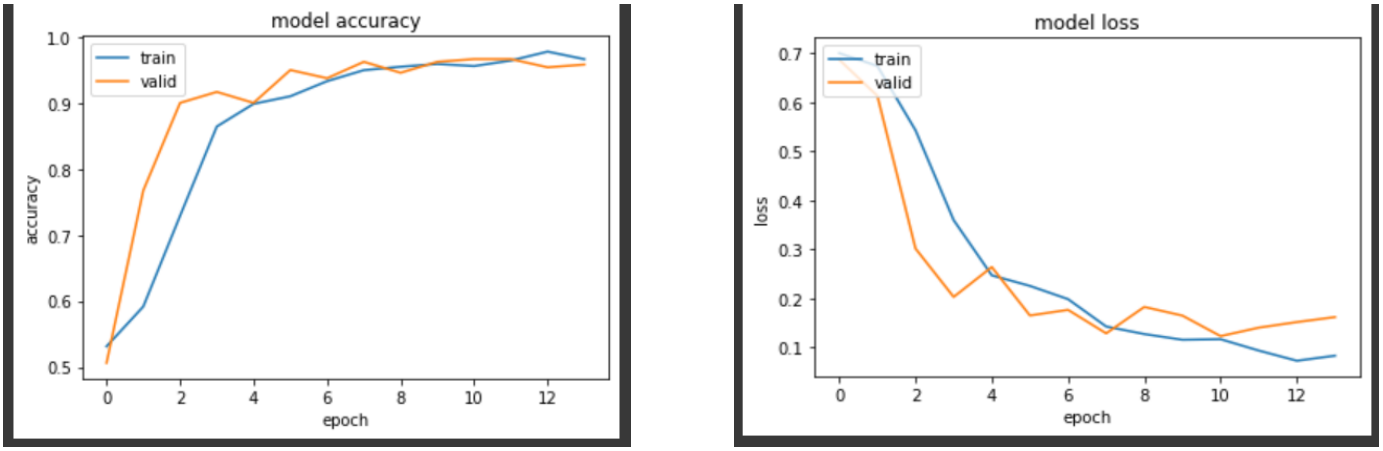


Figure: Loss and accuracy curves

We can notice that both the training and validation curves are near to each other in the last epochs , it's a sign that our model does not overfit.

Accuracy is an important metric that help us to evaluate our model, but it is not the unique one , for that we added other metrics to have extra information on our model such as : Precision, Recall, f1-score

Classification Report

	precision	recall	f1-score	support
WITHMASK	0.99	0.95	0.97	155
WITHOUTMASK	0.95	0.99	0.97	146
accuracy			0.97	301
macro avg	0.97	0.97	0.97	301
weighted avg	0.97	0.97	0.97	301

Figure: Classification report

3. Emotion recognition

3.1. Xception: Deep Learning with Depth Wise Separable Convolutions

The Xception architecture has 36 convolutional layers forming the feature extraction base of the network followed by a logistic regression layer.

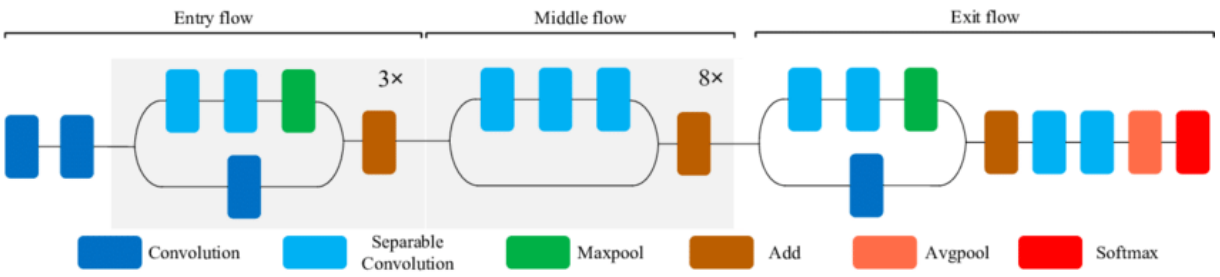


Figure: Architecture of the Xception model

The model had a total of 3.969.639 trainable parameters and 20.336 non trainable ones.

```
============
Total params: 3,989,975
Trainable params: 3,969,639
Non-trainable params: 20,336
```

Figure: Number of parameters of the Xception model

When trained on our FER13 dataset , the Xception model gave an accuracy of 0.64 which is good given the fact that human accuracy on emotion detection is roughly 0.65

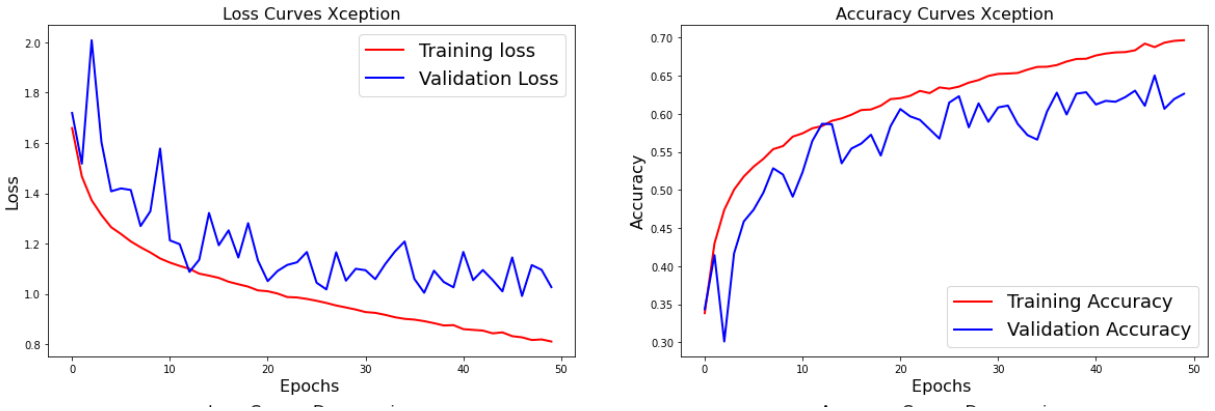


Figure: Loss and accuracy curves of the Xception model

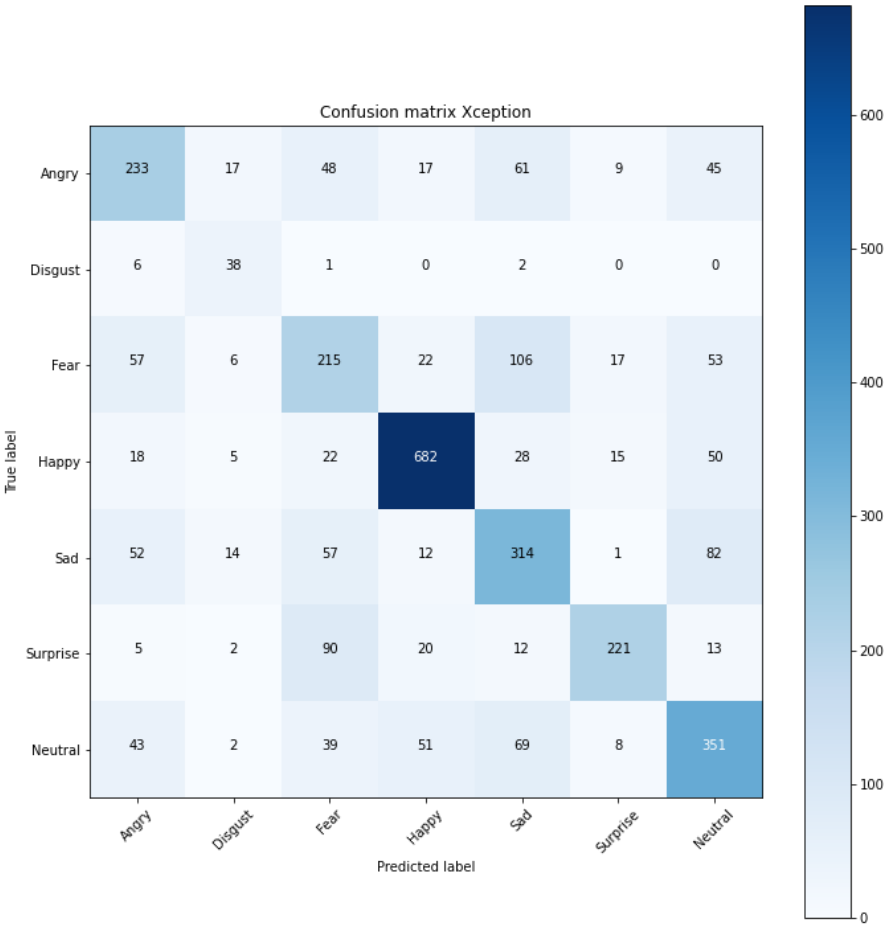


Figure: Confusion matrix of Xception model

	precision	recall	f1-score	support
0	0.563	0.542	0.552	430
1	0.452	0.809	0.580	47
2	0.456	0.452	0.454	476
3	0.848	0.832	0.840	820
4	0.530	0.590	0.559	532
5	0.815	0.609	0.697	363
6	0.591	0.623	0.607	563
accuracy			0.636	3231
macro avg	0.608	0.637	0.613	3231
weighted avg	0.646	0.636	0.638	3231

Figure: Classification report of Xception model

3.2. DeXpression : Deep Convolutional Neural Network for Expression Recognition

The proposed deep Convolutional Neural Network architecture consists of four parts. The first part automatically preprocesses the data. This begins with Convolution 1, which applies 64 different filters. The next layer is Pooling 1, which down-samples the images and then they are normalized by LRN 1. The next steps are the two FeatEx (Parallel Feature Extraction Block) blocks,They are the core of the proposed architecture a. The features extracted by theses blocks are forwarded to a fully connected layer, which uses them to classify the input into the different emotions. The described architecture is compact, which makes it not only fast to train, but also suitable for real-time applications.

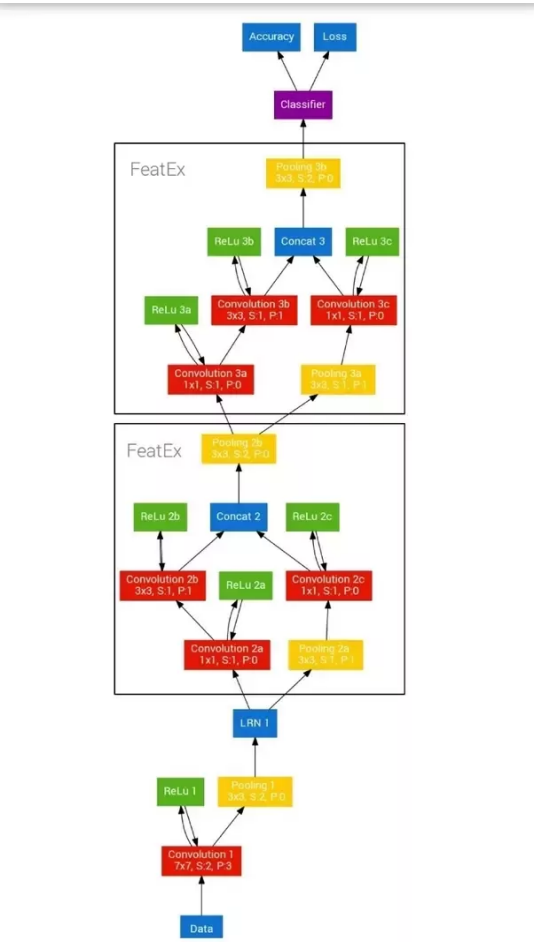


Figure: Architecture of DeXception model

Layer	Output Size
Data	224 × 224
Convolution 1	64 × 112 × 112
Pooling 1	64 × 56 × 56
LRN 1	64 × 56 × 56
Convolution 2a	96 × 56 × 56
Convolution 2b	208 × 56 × 56
Pooling 2a	64 × 56 × 56
Convolution 2c	64 × 56 × 56
Concat 2	272 × 56 × 56
Pooling 2b	272 × 28 × 28
Convolution 3a	96 × 28 × 28
Convolution 3b	208 × 28 × 28
Pooling 3a	272 × 28 × 28
Convolution 3c	64 × 28 × 28
Concat 3	272 × 28 × 28
Pooling 3b	282 × 14 × 14
Classifier	11 × 1 × 1

```
Total params: 693,991
Trainable params: 693,671
Non-trainable params: 320
```

3.3. CNN inspired by Goodfellow I.J

3.3.1. Introduction

A simple architecture is fast to train and easy to implement. An effective architecture achieves good accuracy on the test data. CNN architectures are black boxes to us. VGGNet, AlexNet and Inception are well-known CNN architectures. These architectures have strongly influenced CNN model designs for new datasets. Almost all CNN models known to achieve high accuracy on facial expression recognition problem are influenced by these architectures. This work tries to overcome this limitation by using FER-2013 dataset as starting point to design new CNN models. In this work, the effect of CNN parameters namely kernel size and number of filters on the classification accuracy is investigated using FER-2013 dataset.

3.3.2. Architecture

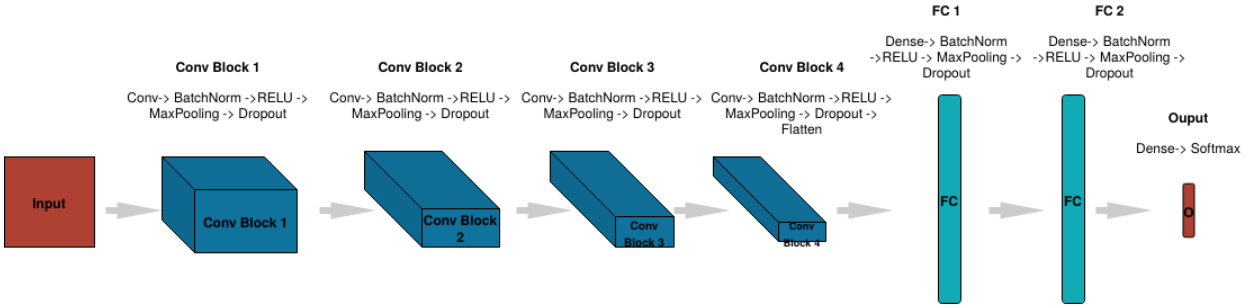


Figure: Architecture of the CNN model

As mentioned in the figure above, the model is composed of an input layer and 4 hidden layers which every layer is also composed of a convolution,batch normalization function and activation function (Relu) and a maxpoling function and finally a dropout layer.

We find also two dense layers with a structure like this : Dense → BAtchNorm → RELU → MaxPolling → DropOut and finally an output layer composed by a dense layer and a softmax function which is used for multi-class classification.

3.3.3. Loss-Accuracy curves

After training our model on 50 epochs we finally obtained an accuracy of 0.668 for the training part and a loss fe 0.38 for the training and validating part.

Our model has no overfitting, which is very common in deep learning models.

We can see all the results belows on the two graphs :

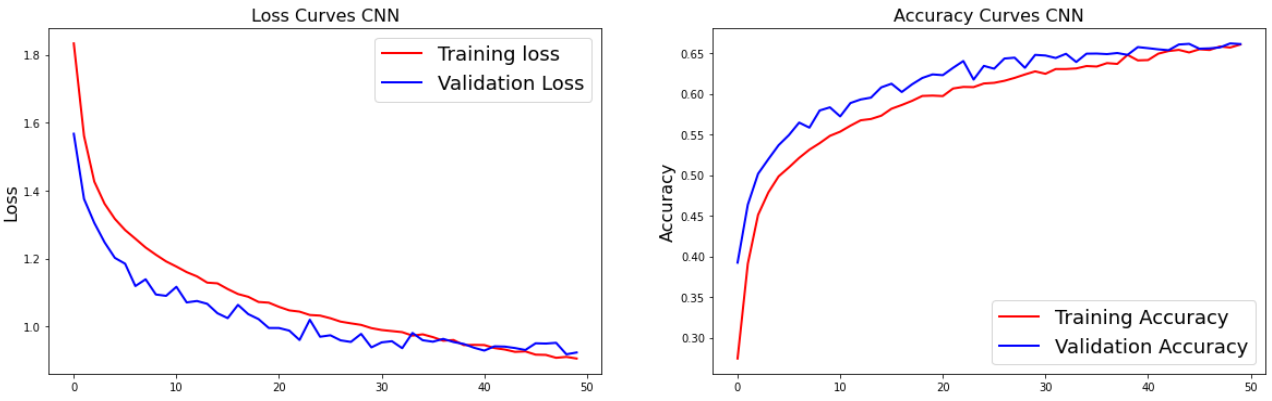


Figure: Loss and accuracy curves of the CNN model

3.3.4. Confusion Matrix and classification report

After that we obtained the confusion matrix with 693 of true positive values for the 'happy' emotion which is the highest prediction of our model.But only 45 true positive values for the emotion 'disgust' and for the others emotions all the true positive values are between approximately 200 and 400.

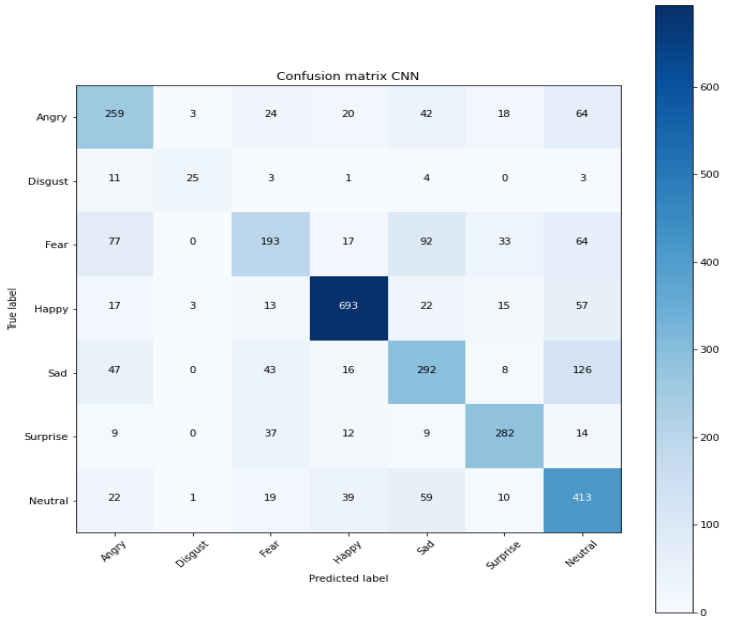


Figure: Confusion Matrix of the CNN model

For more detail we obtained using the classification report we onìbtained a precision of 0.86 for the happy emotion and for the other result all the precision values are between 0.55 and 0.78 .

	precision	recall	f1-score	support
0	0.586	0.602	0.594	430
1	0.781	0.532	0.633	47
2	0.581	0.405	0.478	476
3	0.868	0.845	0.857	820
4	0.562	0.549	0.555	532
5	0.770	0.777	0.774	363
6	0.557	0.734	0.633	563
accuracy			0.668	3231
macro avg	0.672	0.635	0.646	3231
weighted avg	0.672	0.668	0.665	3231

Figure:Classification Report of the CNN model

3.4. Conclusion

We should note that human accuracy for emotion detection is 65% , so overall all our models gave a reasonably good accuracy. We finally chose the **CNN model** with the best validation accuracy which is 0.68 .

4. Face verification

We treated face recognition as a classification problem with facenet and cosine similarity .

4.1. Presentation of the Facenet

Facenet is a pretrained model from google used in features extraction from different faces.

Advantages:

- ➔ State-of-the-art face recognition performance using only 128-bytes per face
- ➔ Minimal alignment required on the input dataset
- ➔ Understand the error cases and improve the model further.
- ➔ Reduce the model size and computational requirements.
- ➔ Classification accuracy achieved is 95.12% on Youtube DB and **99.63% accuracy on LFW**

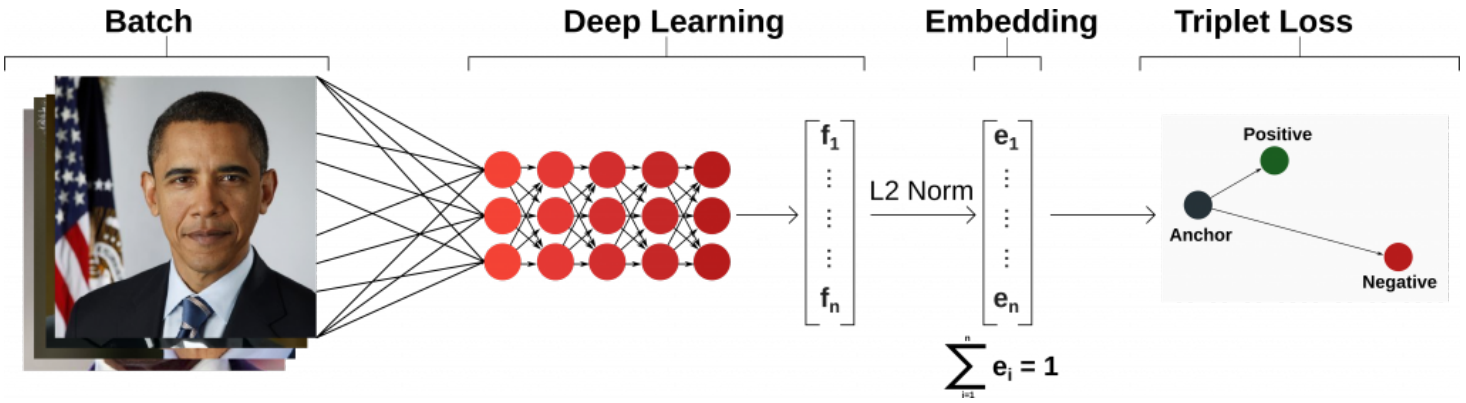


Figure : google facenet approach

4.2. Architecture of the Facenet

Total params : 22,808,144 Trainable params : 22,779,312

Non-trainable params : 28,832 22 convolutional layers

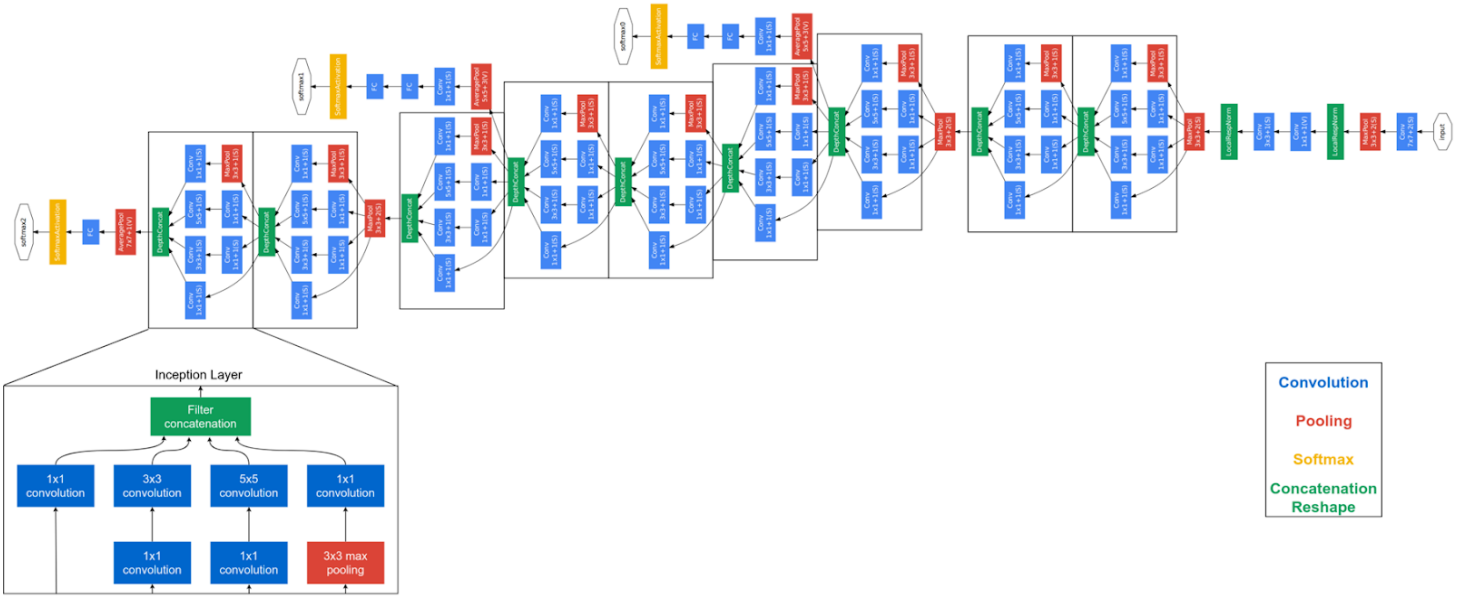


Figure : Architecture of the facenet

4.3. Face embedding with Facenet

While using the pretrained model facenet, we managed to extract the features from each person and save the model for further use.

Face embedding is 128 numbers representing a face.

```
# convert each face in the train set to an embedding
x_train_e = list()
for face_pixels in x_train:
    embedding = get_embedding(model, face_pixels)
    x_train_e.append(embedding)

x_train_e = np.asarray(x_train_e)
print(x_train_e.shape)

# convert each face in the test set to an embedding
x_test_e = list()
for face_pixels in x_test:
    embedding = get_embedding(model, face_pixels)
    x_test_e.append(embedding)
x_test_e = np.asarray(x_test_e)
print(x_test_e.shape)
# save arrays to one file in compressed format
np.savez_compressed('faces-embeddings.npz', x_train_e, y_train, x_test_e, y_test)
```

Figure : getting embeddings from faces

4.4. Cosine Similarity

The 128 numbers that the facenet has extracted perfectly represent a face and in a space of 128 dimensions the embeddings of each person will point in the same direction.
So the cosine between two embeddings of the same person must be too small so we can use the cosine of two embeddings for facial verification or recognition .
In practice for two given images , after obtaining the embeddings we calculate the cosine between these two embeddings and if the value is below a threshold of 0.7 we conclude that the two images belong to the same person.

In this figure we calculate the cosine between images of the same person.

```
plt.figure(figsize = (20,6))
for i in range(1,7):
    #calcul du cosinus
    alpha = np.sum((x_test[25]*x_test[24+i]))/(np.linalg.norm(x_test[25])*np.linalg.norm(x_test[24+i]))
    alpha = (np.arccos(alpha)*180)/np.pi

    ax = plt.subplot(1,6,i)

    plt.imshow(x_test_face[24+i])
    ax.set_title("le cos est = {:.3f}".format(alpha))
    plt.axis("off")
```

```
C:\Users\MSI\.conda\envs\pi-face-recognition\lib\site-packages\ipykernel_launcher.py:5: RuntimeWarning: invalid value encount
ed in arccos
  """
```

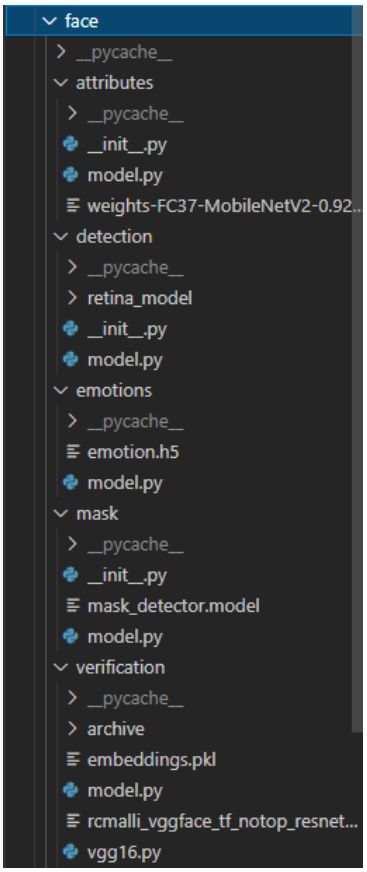
Chapter V : Deployment

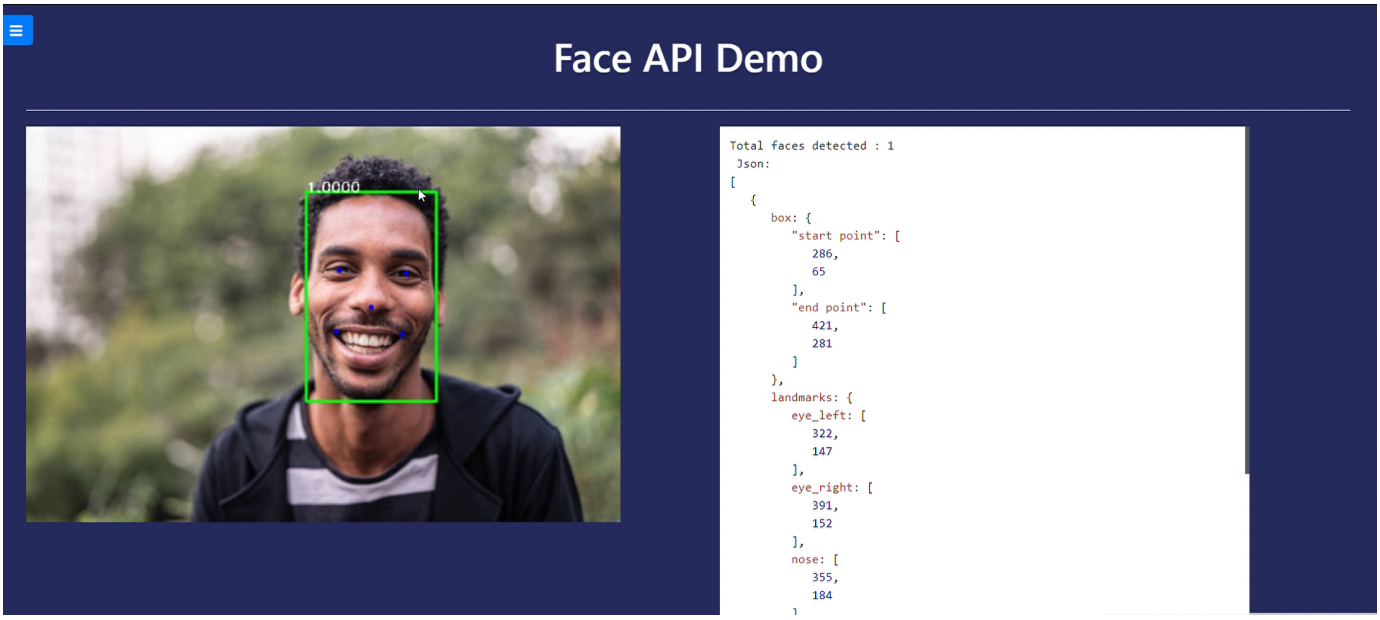
In this step, we are going to deploy our models within web-based applications in order to provide the services we have promised to give and make it easier for our users to actually use the app. first we saved our best models .

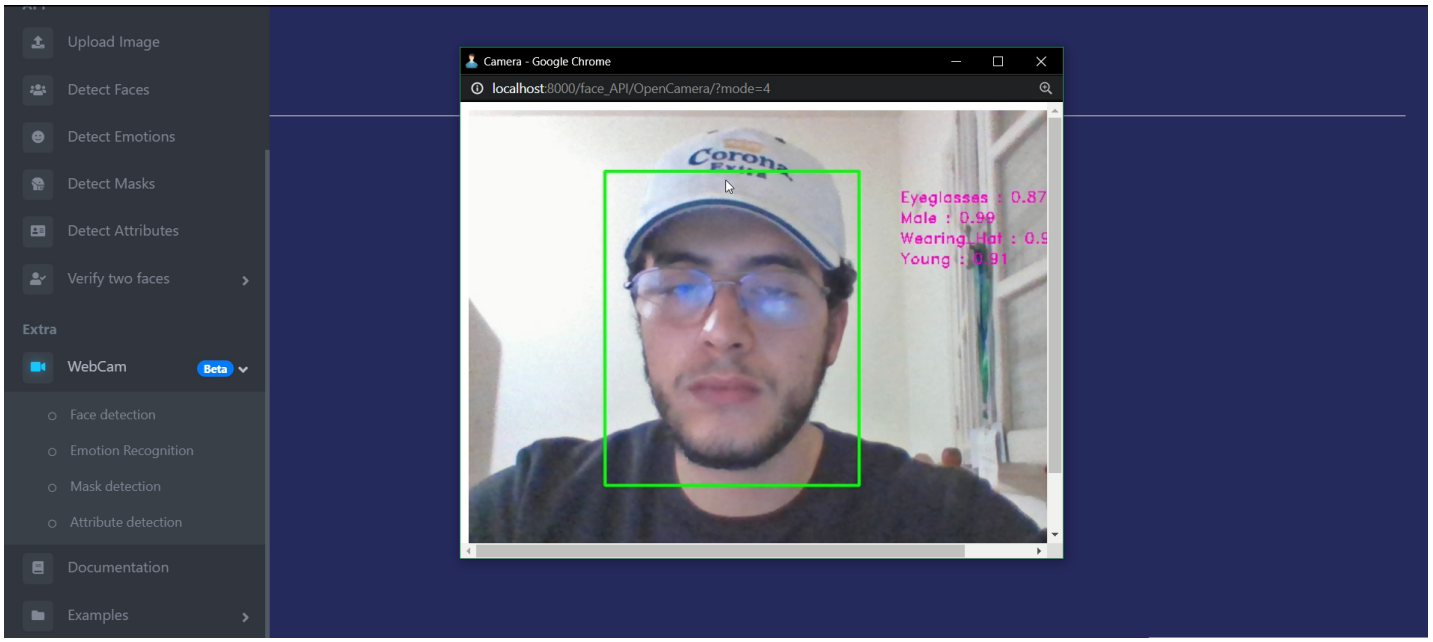
2-1 DJANGO Framework

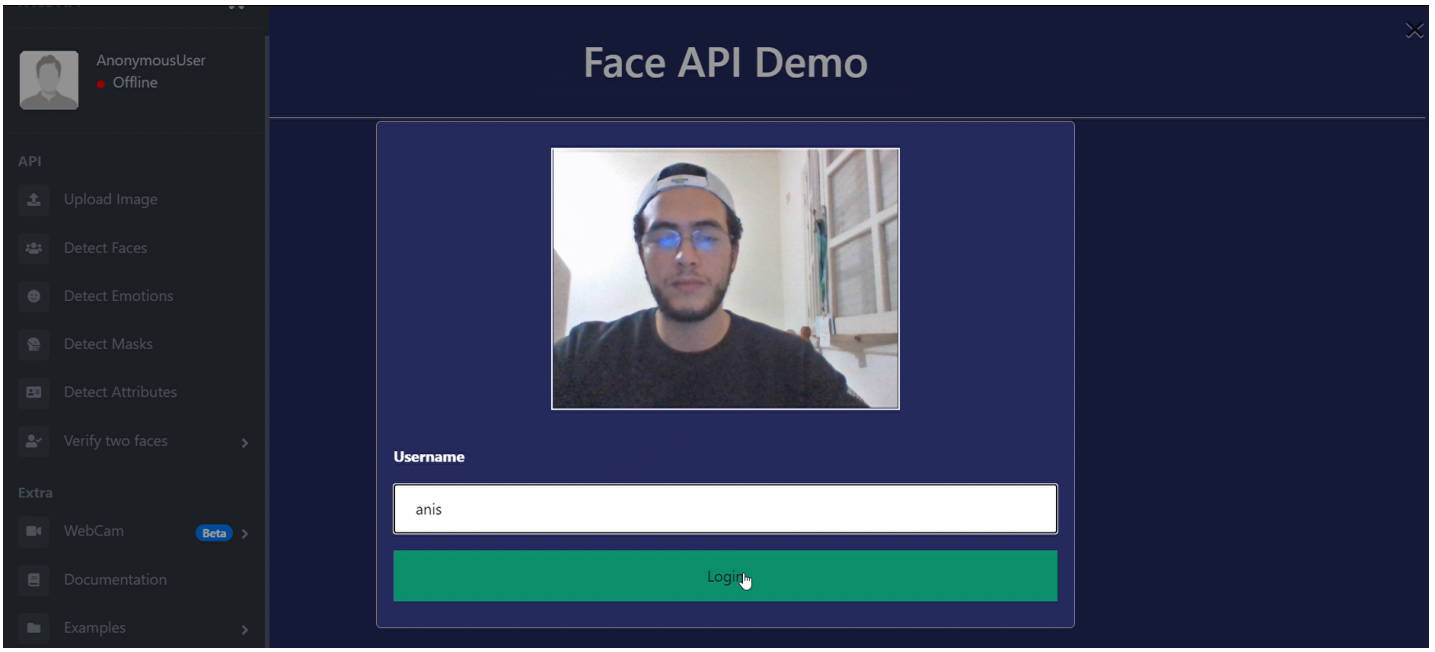
Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design,due to the work done with python programming language and the use of multiple python machine learning libraries such as keras,tensorflow,sklearn and many other for scalability and rapid integration the choice of Django as the web framework was systematic.

API Architecture :






Upload Image
Detect Faces
Detect Emotions
Detect Masks
Detect Attributes
Verify two faces
Extra
WebCam
Beta
Face detection
Emotion Recognition
Mask detection
Attribute detection
Documentation
Examples
Camera - Google Chrome
localhost:8000/face_API/OpenCamera/?mode=4
Eyeglasses : 0.87
Male : 0.99
Young : 0.91


AnonymousUser
Offline
API
Upload Image
Detect Faces
Detect Emotions
Detect Masks
Detect Attributes
Verify two faces
Extra
WebCam
Beta
Documentation
Examples
Face API Demo
Username
anis
Login

Conclusion

Facial recognition is one of the revolutionary technologies of our time, its many advantages make it almost essential in terms of security and identification of individuals. Therefore, we want to improve the conditions of our educational system by including facial recognition in educational establishments. Accordingly, our project consists of a multi purpose face API that detects, recognizes, and analyzes human faces, attributes and emotions in images and videos and through this report we show how all the objectives were achieved successfully while developing how we did it and by exposing the steps we went through.

References

https://www.kaggle.com/jessicali9530/celeba-dataset?select=list_bbox_celeba.csv

https://paperswithcode.com/sota/facial-expression-recognition-on-fer2013

https://conradsanderson.id.au/lfwcrop/

https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data/

https://towardsdatascience.com/real-time-multi-facial-attribute-detection-using-transfer-learning-and-haar-cascades-with-fastai-47ff59e36df0

https://www.kaggle.com/ky2019/starter-celebfaces-attributes-celeba-b5421ae1-e

https://arxiv.org/pdf/1610.02357.pdf

https://arxiv.org/pdf/1509.05371.pdf

https://arxiv.org/pdf/1307.0414.pdf