1.INTRODUCTION

Facial recognition is a way of identifying or confirming an individual's identity using their face. Facial recognition systems can be used to identify people in photos, videos, or in real-time.

Facial recognition is a category of biometric security. Other forms of biometric software include voice recognition, fingerprint recognition, and eye retina or iris recognition. The technology is mostly used for security and law enforcement, though there is increasing interest in other areas of use.

Many people are familiar with face recognition technology through the FaceID used to unlock iPhones (however, this is only one application of face recognition). Typically, facial recognition does not rely on a massive database of photos to determine an individual's identity — it simply identifies and recognizes one person as the sole owner of the device, while limiting access to others.

In this project by using this facial recognition concept we build a model that takes the pictures of the users face with which the user wanted to be locked and unlocked. Then the model will be trained on this data i.e., images that are captured by the system. And it is ready to us

2. MODULES

2.1 OpenCV

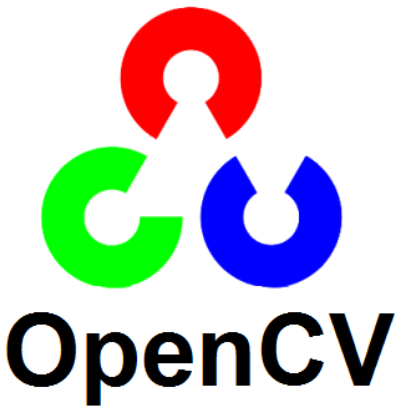
OpenCV is an open-source Python library, which used to understand the content of the digital image. The CV is the abbreviation form of computer vision. It extracts the description from the real-time image or digital image, which may be an object, a text description, and so on.

We can perform many tasks using the OpenCV library such as face detection, face recognition, blob detection, edge-detection, image filter, template matching, and etc. To work with the OpenCV, we need to install it in our Python environment.

When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.

By using OpenCV we can do the following:

1. Reading an Image
2. Extracting the RGB values of a pixel
3. Extracting the Region of Interest (ROI)
4. Resizing the Image
5. Rotating the Image
6. Drawing a Rectangle
7. Displaying text



- **Installation of OpenCV**

  We install OpenCV by using command prompt using pip command. Below are the steps to install OpenCV using command prompt.

  1. Open the command prompt.
  2. Now type the below command.

     pip install opencv-contrib-python --upgrade

  3. Or we can install it without extra module by following command

     pip install opencv-python

- **Importing OpenCV module**

  By using the below line we import this OpenCV module into our program.

  import cv2

  After executing this line the cv2 module will be imported into our program.

- **Applications**

  OpenCV's application areas include

  1. 2D and 3D feature toolkits
  2. Egomotion estimation
  3. Facial recognition system
  4. Gesture recognition
  5. Human–computer interaction (HCI)
  6. Mobile robotics
  7. Motion understanding
  8. Object detection
  9. Segmentation and recognition
  10. Stereopsis stereo vision: depth perception from 2 cameras
  11. Structure from motion (SFM)
  12. Motion tracking
  13. Augmented reality

2.2 Numpy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy.

- **Installation of Numpy module**

  We install OpenCV by using command prompt using pip command. Below are the steps to install OpenCV using command prompt.

  1. Open the Command Prompt
  2. Enter the below command and hit enter

     pip install numpy

- **Importing Numpy module**

  By using the below line in our code we can easily import the numpy package into out program

  import numpy

  And another way to inport is:

  import numpy as np

  By using this where ever we use numpy we can replace it with np.

- **Applications**

  Some of the applications of numpy are:

  1. An alternative for lists and arrays in Python
  2. NumPy maintains minimal memory
  3. Using NumPy for multi-dimensional arrays
  4. Mathematical operations with NumPy
  5. Shape Manipulations

2.3 OS Module

Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.

The OS comes under Python's standard utility modules. This module offers a portable way of using operating system dependent functionality.

- **Installing OS Module**
  In python os module is already available. There will be no need to install. When we install python it automatically includes into the system.

- **Importing OS Module**

  By using the below line in the code we can import the os module into the code.

  import os

- **Applications**

These are some of the applications of the OS Module:

1. To create a directory.
2. To list out files and directories.
3. To delete files and directories.
4. To locate the files.
5. To handle the current working directory.
6. To move files from one location to another location.

3. WORKING

3.1 Loading the Haar Cascade Frontal Face xml file

The project starts with importing the two important modules Cv2 and Numpy

import cv2

import numpy as np

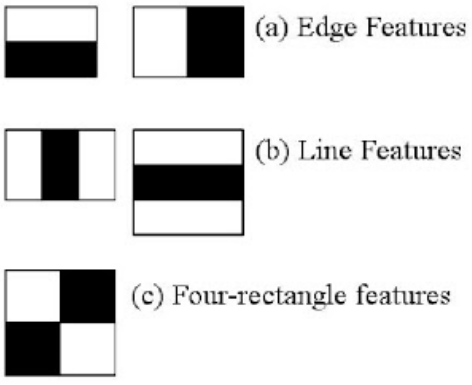
Then continues with loading haarcascade file i.e., **haarcascade_frontalface_default.xml** by using the following line

cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')

Haar Cascade Object Detection

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

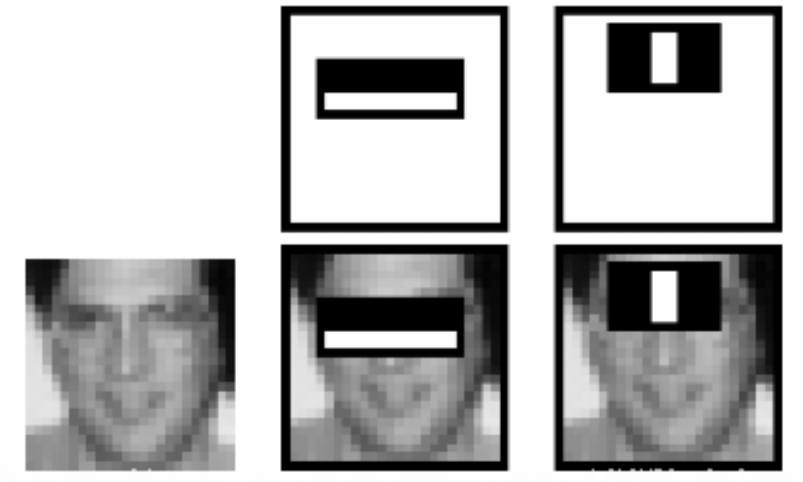
Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.



3.1.1 Haar Classifier Features

Now, all possible sizes and locations of each kernel are used to calculate lots of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, we need to find the sum of the pixels under white and black rectangles. To solve this, they introduced the integral image. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels. Nice, isn't it? It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. The top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applied to cheeks or any other place is irrelevant. So how do we select the best features out of 160000+ features? It is achieved by **Adaboost**.



3.1.2 Features selecting on images

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that most accurately classify the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then the same process is done. New error rates are calculated. Also new weights. The process is continued until the required accuracy or error rate is achieved or the required number of features are found).

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that most accurately classify the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then the same process is done. New error rates are calculated. Also new weights. The process is continued until the required accuracy or error rate is achieved or the required number of features are found).

The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

So now you take an image. Take each 24x24 window. Apply 6000 features to it. Check if it is face or not. Wow.. Isn't it a little inefficient and time consuming? Yes, it is. The authors have a good solution for that.

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 many fewer 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!

The authors' detector had 6000+ features with 38 stages with 1, 10, 25, 25 and 50 features in the first five stages. (The two features in the above image are actually obtained as the best two features from Adaboost). According to the authors, on average 10 features out of 6000+ are evaluated per sub-window.

So, this is about the haar feature based cascade classifier.

3.2 Capturing Images and Storing them in a directory

Now we on the camera using cv2 module. Cv2 has an method i.e., VideoCapture(). By this we create a object of camera. It requires an argument i.e., which camera we have to select i.e., inbuild one or external one. 0 for the inbuild and other for the external camera.

cap = cv2.VideoCapture(0)

Now start capturing the images. Check the each captured image has face or not by using user-defined function **face_extractor(img)** which takes the parameter of captured image to check. First it will change the color of the image.Then finds the face. If it finds the face then it will crop the face in the image and returns. If no face is found then it returns None.

```
def face_extractor(img):
    # Function detects faces and returns the cropped face
    # If no face detected, it returns the input image
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray, 1.3, 5)
    if faces is ():
        return None
    # Crop all faces found
    for (x,y,w,h) in faces:
        cropped_face = img[y:y+h, x:x+w]
    return cropped_face
```

And now if the above function returns none no operation will be performed. If it returns the face then the returned image color is changed from BGR to Gray. After that the image will be saved in the specified directory with the name as count variable value. And also the count of the sample will also be updated on the window itself i.e., the value of the count variable. Initially count value is 0.

This will continue until we press Enter key or 100 images were taken.

After the we release the camera object and destroy all the windows and print **Colection of samples are done.**

This is all about the collection of samples from the user to train the on that data.

```
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if face_extractor(frame) is not None:
        count += 1
        face = cv2.resize(face_extractor(frame), (200, 200))
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        file_name_path = './faces/user/' + str(count) + '.jpg'
        cv2.imwrite(file_name_path, face)
        cv2.putText(face, str(count), (50, 50),
cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
        cv2.imshow('Face Cropper', face)
    else:
        print("Face not found")
        pass
    if cv2.waitKey(1) == 13 or count == 100: #13 is the Enter Key
        break
cap.release()
cv2.destroyAllWindows()
print("Colection of samples are done")
```

3.3 Traing the Model

After successful completion of collecting samples, the next step is to train our model to get ready to detect the face of the user according to the samples.

The training of model starts with importing of the modules.

1. Cv2 Module
2. Numpy Module
3. Os Module

```
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join
```

Now we store the entire images into a list. We load those images into a list named **onlyfiles** using os module by specifying the path of those images.
Now by iterating in the **onlyfiles** list we append those data into the **Training_Data** list and file names into the **labels** list.
After that we convert the labels list into the numpy array.

Now we initialize facial recognizer like:

```
model = cv2.face.LBPHFaceRecognizer_create()
```

And by using model reference variable we train the data like we have a **train()** method which takes the parameters **Training_Data** and **labels.**
After successful completion of the traing we print **Model training is completed.**

```
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join
# Get the training data we previously made
data_path = './faces/user/'
onlyfiles = [f for f in listdir(data_path) if isfile(join(data_path, f))]
# Create arrays for training data and labels
Training_Data, Labels = [], []
# Open training images in our datapath
# Create a numpy array for training data
for i, files in enumerate(onlyfiles):
    image_path = data_path + onlyfiles[i]
    images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    Training_Data.append(np.asarray(images, dtype=np.uint8))
    Labels.append(i)
# Create a numpy array for both training data and labels
Labels = np.asarray(Labels, dtype=np.int32)
# Initialize facial recognizer
model = cv2.face.LBPHFaceRecognizer_create()
# NOTE: For OpenCV 3.0 use cv2.face.createLBPHFaceRecognizer()
# Let's train our model
model.train(np.asarray(Training_Data), np.asarray(Labels))
print("Model trained sucessefully")
```

3.4 Recognizing the face

Now this is the main part of this system. After training of model on the data now we test the model.

We start this by importing the modules:

1. Cv2
2. Numpy

import cv2

import numpy as np

After importing the modules here we load the Haarcascade frontal face xml file i.e., **haarcascade_frontalface_default.xml** using cv2 module and store it in a variable **face_classifier.**

face_classifier = cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')

And now we create a VideoCapture object using cv2 module

cap = cv2.VideoCapture(0)

After that we capture or read the image from the webcam using read() method. And now we pass that image to **face_detector(img)** user-defined function. In that function we convert image to the geayscale and try to detect the face. If face is found it will crop the face and return that cropped image and we store that data in the **image**, **face** variables.

Now we do prediction using the model variable which we have at the time of training of the data. There is a method names **predict() which** predicts and returns the output. And that output we store in **results** variable i.e., an list.

We check the condition that results[1] must be less than 500, then only we calculate the confidence. If it is less than 500 then, we find the confidence by using below formula and also display it on the image.

confidence = int(100 * (1 - (results[1])/400))

display_string = str(confidence) + '% Confident it is User'

If the calculated **confidence is greater than 75%,** then it will **unlock**. And user the see the unlock text on the screen too.

If the calculated **confidence is less than or equals to 75%** then it remains **Locked**.

If any error occurs in this then it remains locked. And display No face found. This entire process will continue until we press the enter button in keyboard. After that we release webcam and destroy all the windows.

```
import cv2

import numpy as np

face_classifier = cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')

def face_detector(img, size=0.5):

    # Convert image to grayscale

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    faces = face_classifier.detectMultiScale(gray, 1.3, 5)

    if faces is ():

        return img, []

    for (x,y,w,h) in faces:

        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,255),2)

        roi = img[y:y+h, x:x+w]

        roi = cv2.resize(roi, (200, 200))

    return img, roi
```

```
# Open Webcam
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    image, face = face_detector(frame)


    try:
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        # Pass face to prediction model
        # "results" comprises of a tuple containing the label and the confidence value
        results = model.predict(face)
        if results[1] < 500:
            confidence = int( 100 * (1 - (results[1])/400) )
            display_string = str(confidence) + '% Confident it is User'
        cv2.putText(image, display_string, (100, 120), cv2.FONT_HERSHEY_COMPLEX, 1, (255,120,150), 2)
        if confidence > 75:
            cv2.putText(image, "Unlocked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
            cv2.imshow('Face Recognition', image )
```

```
        else:
            cv2.putText(image, "Locked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0,0,255), 2)
            cv2.imshow('Face Recognition', image )
    except:
        cv2.putText(image, "No Face Found", (220, 120) , cv2.FONT_HERSHEY_COMPLEX, 1, (0,0,255), 2)
        cv2.putText(image, "Locked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0,0,255), 2)
        cv2.imshow('Face Recognition', image )
        pass
    if cv2.waitKey(1) == 13: #13 is the Enter Key
        break
cap.release()
cv2.destroyAllWindows()
```

4. IMPLEMENTATION

4.1 CODING

4.1.1 Collecting Samples

```
import cv2

import numpy as np

# Load HAAR face classifier

face_classifier = cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')

def face_extractor(img):



    # Function detects faces and returns the cropped face

    # If no face detected, it returns the input image

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    faces = face_classifier.detectMultiScale(gray, 1.3, 5)

    if faces == ():

        return None

    # Crop all faces found

    for (x,y,w,h) in faces:

        cropped_face = img[y:y+h, x:x+w]
```

```
    return cropped_face
# Initialize Webcam
cap = cv2.VideoCapture(0)
count = 0

# Collect 100 samples of your face from webcam input
while True:
    ret, frame = cap.read()
    if face_extractor(frame) is not None:
        count += 1
        face = cv2.resize(face_extractor(frame), (200, 200))
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        # Save file in specified directory with unique name
        file_name_path = './faces/user/' + str(count) + '.jpg'
        cv2.imwrite(file_name_path, face)
        # Put count on images and display live count
        cv2.putText(face, str(count), (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
        cv2.imshow('Face Cropper', face)
    else:
```

```
        print("Face not found")
        pass
    if cv2.waitKey(1) == 13 or count == 100: #13 is the Enter Key
        break
cap.release()
cv2.destroyAllWindows()
print("Colection of samples are done")
```

4.1.2 Training the Model

```
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join
# Get the training data we previously made
data_path = './faces/user/'
onlyfiles = [f for f in listdir(data_path) if isfile(join(data_path, f))]
# Create arrays for training data and labels
Training_Data, Labels = [], []
# Open training images in our datapath
```

```
# Create a numpy array for training data
for i, files in enumerate(onlyfiles):
    image_path = data_path + onlyfiles[i]
    images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    Training_Data.append(np.asarray(images, dtype=np.uint8))
    Labels.append(i)
# Create a numpy array for both training data and labels
Labels = np.asarray(Labels, dtype=np.int32)
# Initialize facial recognizer
model = cv2.face.LBPHFaceRecognizer_create()
# NOTE: For OpenCV 3.0 use cv2.face.createLBPHFaceRecognizer()
# Let's train our model
model.train(np.asarray(Training_Data), np.asarray(Labels))
print("Model trained sucessefully")
```

4.1.3 Recognizing the Face

```
import cv2

import numpy as np

face_classifier = cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')

def face_detector(img, size=0.5):

    # Convert image to grayscale

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    faces = face_classifier.detectMultiScale(gray, 1.3, 5)

    if faces is ():

        return img, []

    for (x,y,w,h) in faces:

        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,255),2)

        roi = img[y:y+h, x:x+w]

        roi = cv2.resize(roi, (200, 200))

    return img, roi

# Open Webcam

cap = cv2.VideoCapture(0)

while True:

    ret, frame = cap.read()
```

```
    image, face = face_detector(frame)
    try:
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        # Pass face to prediction model
        # "results" comprises of a tuple containing the label and the confidence value
        results = model.predict(face)
        if results[1] < 500:
            confidence = int( 100 * (1 - (results[1])/400) )
            display_string = str(confidence) + '% Confident it is User'
        cv2.putText(image, display_string, (100, 120), cv2.FONT_HERSHEY_COMPLEX, 1, (255,120,150), 2)
        if confidence > 75:
            cv2.putText(image, "Unlocked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
            cv2.imshow('Face Recognition', image )
        else:
            cv2.putText(image, "Locked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0,0,255), 2)
            cv2.imshow('Face Recognition', image )
```

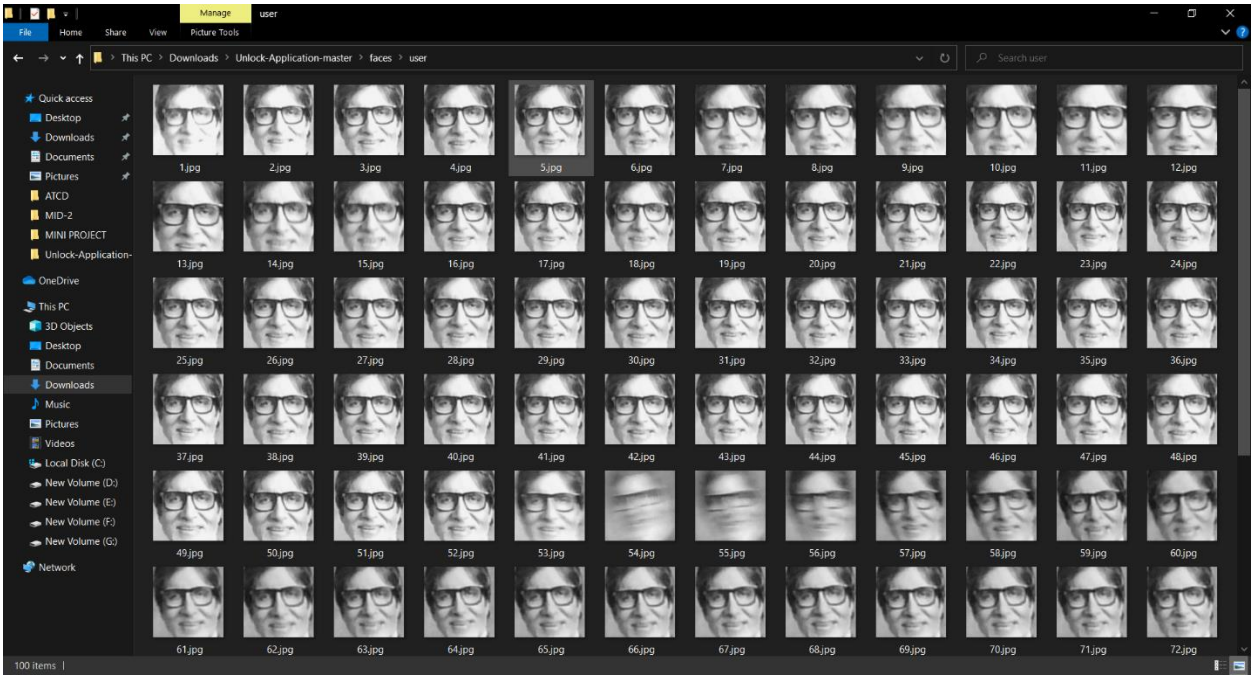
```
    except:
        cv2.putText(image, "No Face Found", (220, 120) , cv2.FONT_HERSHEY_COMPLEX, 1, (0,0,255), 2)
        cv2.putText(image, "Locked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0,0,255), 2)
        cv2.imshow('Face Recognition', image )
        pass
    if cv2.waitKey(1) == 13: #13 is the Enter Key
        break
cap.release()
cv2.destroyAllWindows()
```

5. RESULT

5.1 Collecting samples

5.1.1 Snapshot while collecting samples

5.1.2 Folder after collecting samples



5.2 Training of Model

5.2.1 Trained the module

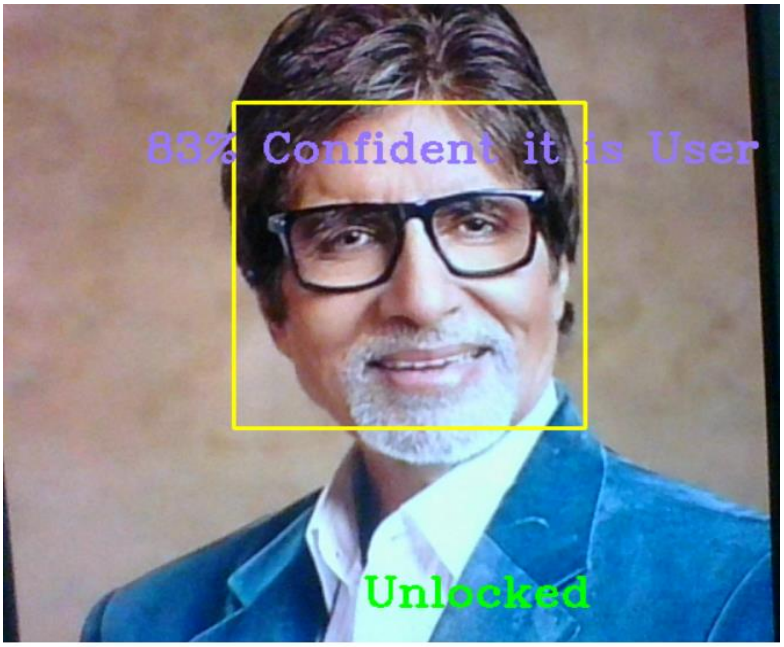
Step 2 - Train Model ¶

In [5]:
```
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join

# Get the training data we previously made
data_path = './faces/user/'
onlyfiles = [f for f in listdir(data_path) if isfile(join(data_path, f))]

# Create arrays for training data and labels
Training_Data, Labels = [], []

# Open training images in our datapath
# Create a numpy array for training data
for i, files in enumerate(onlyfiles):
    image_path = data_path + onlyfiles[i]
    images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    Training_Data.append(np.asarray(images, dtype=np.uint8))
    Labels.append(i)

# Create a numpy array for both training data and labels
Labels = np.asarray(Labels, dtype=np.int32)

# Initialize facial recognizer
model = cv2.face.LBPHFaceRecognizer_create()

# NOTE: For OpenCV 3.0 use cv2.face.createLBPHFaceRecognizer()

# Let's train our model
model.train(np.asarray(Training_Data), np.asarray(Labels))
print("Model trained sucessefully")
```

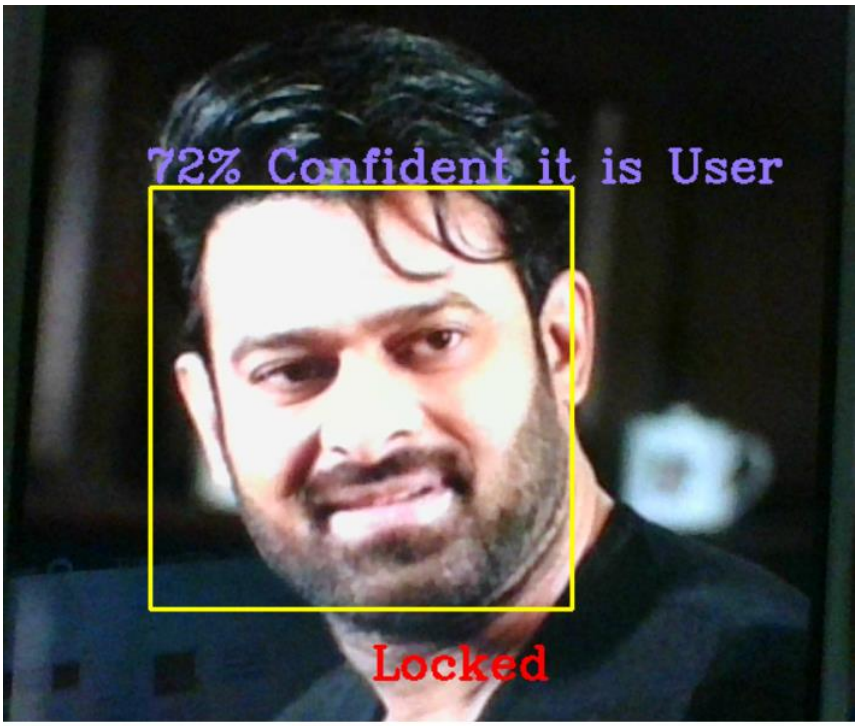
```
Model trained sucessefully
```

5.3 Recognizing the Face

5.3.1 Unlocking with trained face



5.3.2 Trying to unlock with different face



LINK FOR THE CODE

GitHub - Sadhik0905/FACE-UNLOCK-USING-PYTHON

6. CONCLUSION

Face recognition is an emerging technology that can provide many benefits. Face recognition can save resources and time, and even generate new income streams, for companies that implement it right.

There are many applications of this system:

1. Social Media
2. ID Verification
3. Face ID
4. Mobile Lock
5. To Identify the thief in video footages
6. Access Control
7. Security
8. Image Database Investigation
9. Surveillance
10. Immigration Checkpoints to enforce smarter board control

7. REFERENCES

- https://opencv.org/
- https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/#:~:text=OpenCV%2DPython%20is%20a%20library,method%20returns%20an%20empty%20matrix.
- https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/
- https://www.geeksforgeeks.org/os-module-python-examples/?ref=gcse
- https://www.geeksforgeeks.org/opencv-python-program-face-detection/
- https://www.geeksforgeeks.org/face-detection-using-python-and-opencv-with-webcam/?ref=lbp