

# A Neuron Model of Facial Recognition and Detection of an Authorized Entity Using Machine Learning System

J. K. Adedeji, M. O. Oyekanmi

**Abstract**—This paper has critically examined the use of Machine Learning procedures in curbing unauthorized access into valuable areas of an organization. The use of passwords, pin codes, user's identification in recent times has been partially successful in curbing crimes involving identities, hence the need for the design of a system which incorporates biometric characteristics such as DNA and pattern recognition of variations in facial expressions. The facial model used is the OpenCV library which is based on the use of certain physiological features, the Raspberry Pi 3 module is used to compile the OpenCV library, which extracts and stores the detected faces into the datasets directory through the use of camera. The model is trained with 50 epoch run in the database and recognized by the Local Binary Pattern Histogram (LBPH) recognizer contained in the OpenCV. The training algorithm used by the neural network is back propagation coded using python algorithmic language with 200 epoch runs to identify specific resemblance in the exclusive OR (XOR) output neurons. The research however confirmed that physiological parameters are better effective measures to curb crimes relating to identities.

**Keywords**—Biometric characters, facial recognition, neural network, OpenCV.

## I. INTRODUCTION

IN recent times, there is a need to maintain global security of information, most especially in the developing countries, where un-identifiable attacks are rampant, and every organization or individual needs to be security conscious in this environment. There is a need to improve their existing security systems by some measures of proper identification of individuals entering into a public places involving group of people. The present state of security in Nigeria demands a complete surveillance, most especially in the areas of providing adequate security to curb pipeline vandals, for railway protection, international airports' surveillance, monitoring abandoned government project areas which hide criminals, and the colleges and schools in remote and outskirts part of the nation, as well as protecting the Niger Delta Oil rich regions against militancy, and banks, among many others. The outcries for adequate security and protections of our petroleum pipelines against vandals in recent times are on the

high side, creating a nightmare for stakeholders. In most of these identification crimes, the criminals were taking advantage of hacking the information from commercial or academic access control systems or improper identities. In some recent researches, a Biometric capturing technology was introduced for identifying individuals using Biometric characteristics such as fingerprints, eye, face, DNA or other unique characters such as the voice, signatures, etc.

The biometric process is a method of identifying individual electronically by considering certain similarities and variation factors, it was concluded that it is the most stable approach to using conventional ID and passwords methods etc. [1]. The most common and familiar crimes nowadays are crimes relating to credit card, Internet fraud, computer break-in by hackers, or security breaches in a company, in shops, and in government buildings. The use of a face recognition system by organizations or individuals will decrease crimes of credit card, internet frauds, computer break-in by hackers [2]. Some recent researches provided face recognition as a three stages task; acquisition, normalization and recognition, these three components are explained thus: acquisition means the detection and tracking of images similar to a target images in patches, while normalization is the arrangement into segments suitable for storage in the database. The last stage is the recognition which means selected images are identified on one-to-one unique arrangement to form a recognition model [3]. The problems often encountered with the conventional methods such as Pin Codes, passwords, users' ID etc., in identifying items, people have been taking care of by the Biometric technology methods, such as fingerprint, DNA features, eye shapes etc., which makes it unique and easier for identification.

## II. CONCEPTUAL FRAMEWORK AND MODELING

Though there is a problem with recognizing faces when the pose of the face is different, in particular, there is a limit on face rotations in depth, which include left and right and up and down rotations. Face recognition itself is difficult because it is a fine discrimination task among similar objects, once we can find faces, which are quite similar. Adding pose variation naturally makes the problem more difficult. This is because the appearance of a person's face changes under rotation since the face has a complex 3D structure [6]. The research also combined the neural network architecture using certain biometric characteristics; both the Raspberry pi model and neural algorithm are coded in python language to make

J. K. Adedeji is with the, Department of Physics and Electronics, Adekunle Ajasin University Akungba Akoko Ondo State Nigeria (corresponding author, phone: 08133666144, e-mail: adedejikunle2@gmail.com., jelili.adedeji@aaua.edu.ng).

M. O. Oyekanmi is with the Department of Physics and Electronics, Adekunle Ajasin University Akungba Akoko Ondo State Nigeria (phone: 07012842774 e-mail: oyetunjiyekanmi@gmail.com).

comparison easy.

The method adopted in this research is a closed-loop module that includes the computation and fusion of three different visual cues: motion, color and face appearance models. This research focuses on personal identification within its framework and uses Raspberry Pi with OpenCV compiled on it to build the face recognition system.

### III. RASPBERRY PI CONFIGURATION

The Raspberry Pi that was used for this research is a credit card-sized computer originally designed for education, inspired by the 1981 BBC Micro. This research took the advantages of its small size and accessible price and the flexibility with other electronics in projects that require more than a basic microcontroller (such as Arduino devices).



Fig. 1 Picture of a Raspberry Pi 3 Model B

The Raspberry Pi model is a small electronic chip that runs

on Linux computer on a low power consumption level with great abilities and functionalities.

The Raspberry Pi is open hardware, with the exception of the primary chip on the Raspberry Pi, the Broadcom SoC (System on a Chip), which runs many of the main components of the board—CPU, graphics, memory, the USB controller. The advantages of this module is that many of the projects made with it are open and well-documented as well and can be modified to suite a special purpose.

The two commercially available Raspberry Pi models are nomenclature as A and B which are designed for Linux operating system.

For the purpose of this project the SOC (System on Chip) used is the Raspberry Pi 3 Model B and the Linux operating system installed on it is Raspbian.

### IV. COMPONENTS AND HARDWARE REQUIREMENTS

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

### V. HARDWARE COMPONENTS AND CONNECTION

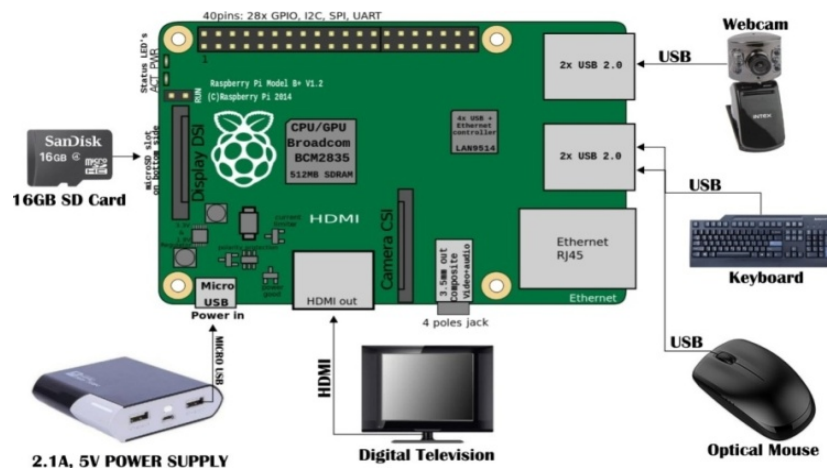


Fig.2 Hardware Components Used

### VI. EXPLANATION OF VARIOUS HARDWARE COMPONENTS

#### A. Power Supply

This is an electrical device that supplies 2.1A electrical power to the Raspberry Pi. The research used a new age 6000 MAH power bank as a supply power to our Raspberry Pi 3 due to the unstable state of electricity in the country.

#### B. Digital Television

This is an output device to monitor the capturing process in the data base of the Raspberry Pi.

#### C. HDMI (High Definition Multimedia Interface)

This is a proprietary audio/video interface for transmitting

uncompressed video data and compressed or uncompressed digital audio data from an HDMI-compliant source device, such as a display controller, to a compatible computer monitor, video projector, digital television, or digital audio device.



Fig. 3 HDMI Cable

*D. Keyboard*

This is a typewriter-style device which uses an arrangement of buttons or keys which is part of the main input devices for the Raspberry Pi.



Fig. 4 Memory Card Reader

*E. Memory Card Reader*

This is a device for accessing the data on a memory card such as the Secure Digital (SD) card used for this project.

*F. The Mouse*

This is a hand-held input device that detects two-dimensional motion relative to a surface. It serves as an input device for the Raspberry Pi.

*G. Secure Digital (SD) Card*

This is a non-volatile memory card format developed by the SD Card Association (SDA) for use in portable devices.

*H. Webcam*

This is a video camera that feeds or streams its image in real time to or through the Raspberry Pi.

*I. The Etcher*

This is a software tool used to burn disc images to SD cards and USB drives, safely and easily.

*J. Numpy*

This is a library for the Python programming language, containing the complex functions, and dimensional array of matrices in high level languages.

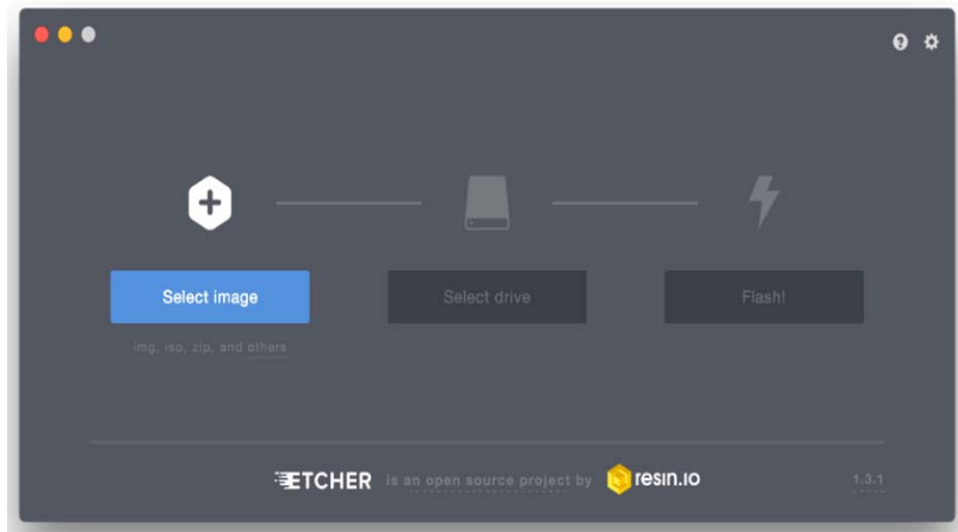


Fig. 5 Etcher Software Dialog Box

VII. METHODOLOGY

*A. Neural Architecture*

The Architecture below represents the internal arrangements of how the images in the database are coded and supplied into the neural network for detecting the faces and checking the resemblance. There are many biometric characteristics that can be used by the neurons to perform the recognition exercise, but in this research only two important ones are considered (i.e. shape of the face in terms of dimensionality and the color). The dotted lines showed that they are many, but only two are used by the neurons in this research, the last neuron is always the bias which is coded

with digit 1 in the algorithm.

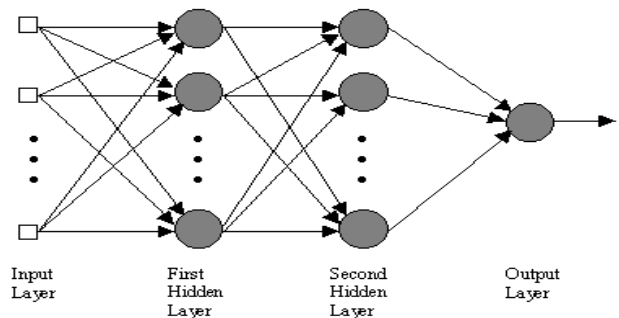


Fig. 6 Neural Configuration [7]

This research has chosen a multilayer perceptron, which is a type of supervised network, since it requires a desired output in order to learn and recognize a pattern. The objective of using it is to find a model that gives an XOR output that matched input at each layer at the run time to produce a required pattern. Within the first hidden layer, they get summed, and then processed by a nonlinear activation function, Fig. 5. When the data is supplied into the neural network as they occur naturally at the input stage through the hidden layers to the output, they are used to multiply the weights at each stage and then summed up into a sigmoid function to obtain a threshold value and produce the neural network output [5].

### VIII. SIGMOID FUNCTION

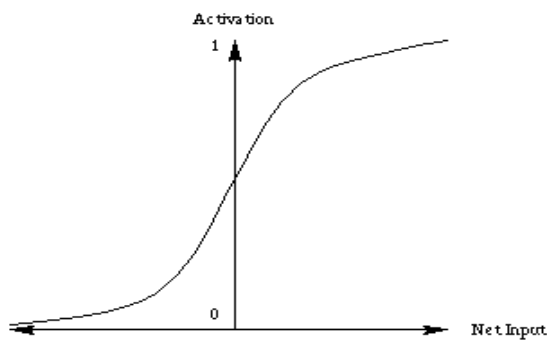


Fig. 7 Sigmoid Activation Function

#### A. Algorithm Design

The neural network used for this design is of three input layers configuration with the bias neuron attached to all the four hidden neurons in the layer preceding the output neuron. The third input neuron which is the bias are always coded with digit 1, while the output codes expected is the XOR pattern to make the classification easy (i.e. 0110).

The neurons are coded with a python algorithmic language and the trained with the back propagation algorithm configuration with 200epoch runs to ensure that the errors are maintained at the barest minimum and good results obtained in all the cases.

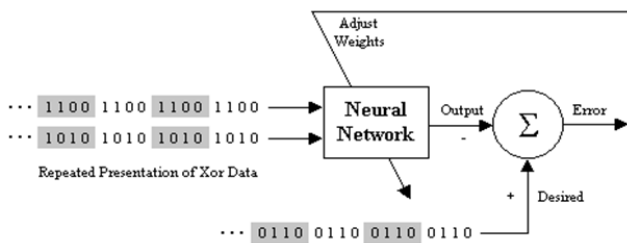


Fig. 8 The Training Process with Back Propagation [5]

The Neural network model of the XOR data is repeatedly presented to the neural network. With each presentation, the error between the network inputs, the hidden layers and the desired output which is the threshold value when it has been activated through the sigmoid function. The computed values

are then fed back to the neural network for proper adjustments [7].

#### B. Neural Training and Error Correction Function

The transfer function used is the sigmoid function of the form below to ensure the non-linear XOR outputs is obtained. The basic function of the sigmoid  $F(x) = \frac{1}{1 + e^{-x}}$  and its derivative  $F'(x) = f(x)(1-f(x))$  is used to adjust the weights. The algorithm knows what output is correct when the error is getting under a threshold value.

The error of output neuron j after the activation of the network on the n-th training  $(\chi(n), d(n))$  is  $e_j(n) = d_j(n) - y_j(n)$ ; and pattern error is the sum of the squared errors of the output neurons:

$$E(n) = \frac{1}{2} \sum e_j^2(n)$$

j = output node.

Total mean error is calculated as the average of the network errors of the training algorithm as.  $E_{AV} = \frac{1}{n} \sum_{n=1}^n E(n)$ .

the back propagation weight update rule that is used for the research is based on the gradient descent approach which is given as;

$$w_{ji} = w_{ji} + \Delta w_{ji} \quad \text{where}$$

$$\Delta w_{ji} = -\xi \frac{\partial E}{\partial w_{ji}}, \quad \xi > 0, \quad \text{where } \xi \text{ is the gradient}$$

multiplicity coefficient or correction factor, the weights are updated using the function:  $v_j = \sum_{i=0 \dots m} w_{ji} y_i$  For the j<sup>th</sup>

input neuron; applying the chain rule, we have,  $\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial v_j} \cdot \frac{\partial v_j}{\partial w_{ji}}$ . But the error signal of j can be defined

$$\text{as: } \delta_{ji} = -\frac{\partial E}{\partial v_j}, \quad \text{therefore } \frac{\partial v_j}{\partial w_{ji}} = y_i, \quad \text{and can be}$$

substituted above as:

$$\Delta w_{ji} = \xi \delta_j y_i$$

The output weights can be updated using the weight change  $\Delta w_{ji}$  to update the error signal  $\delta_j$  of neuron j depending on the condition on whether j is an output or a hidden neuron.

#### C. Training the Algorithm with LBPH

In achieving our objectives, firstly, we need to train the algorithm, to do this, there is need to use a dataset with the facial images of the people we want to recognize. Also there is a need to set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and gives an output. The images of the same person must have a one-to-one correspondence with the person's ID [8].

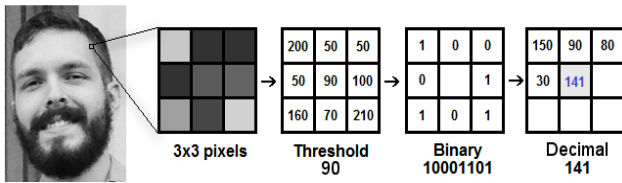


Fig. 9 LBPH Operation Procedure [8]

#### D. Coding Method and Assignment

The methods adopted in this research for the two efficient input neurons used are the Shape of the face in terms of the dimensionality and distances between its members, and the color of the face. These two biometric characteristics are classified as:

The shape of the face (normal, under influence, masked), the normal character means that the images are taken when the person is under normal expression of the face, this is given a higher weight in the neural architecture as the threshold value to be detected by the neuron when it searches for resemblance in the database. The term under the influence is used to denote certain facial expressions in a particular mood of the person, and these include; Anger, Happiness, Surprising behaviors. The condition of facial expressions creates a problem because of its ability to change the facial personal details such as the distance between the eyebrows and the iris in case of anger or surprising behaviors or change in size of mouth in case of

happiness. This classification is given a lower percentage for the neurons not to mistakenly access these biometric characters under the influence of certain expressions.

The third classification under shape of the face is when the person is masked or used a makeup, this can also change the lower eyebrows, therefore the neurons are guided not to pick this as a resemblance, and a lower weight is attached to this for the neurons to avoid during face recognition.

The color input neurons (fair, gray, dark), these are the classifications to be used by the neurons in other to perform facial recognition in terms of the color of the images in database, for the purpose of clear identification by neurons in this research, gray color has been given a higher weights, which is same color used in the LBPH, while other colors are given very low weights to guide the neurons during the training for resemblance.

The selection procedures are as follows: The coding procedures as they are fed into the algorithm can be selected according to the following rules,  $S_1^{x_1}, S_2^{x_1}, C_1^{x_2}, 1$  in the sequence one, or  $S^{x_1}, S_1^{x_2}, C_2^{x_2}, 1$  in sequence two and these selections can be viewed from the table below. It is worth noting that the last digit in all the cases is the bias, which is always digit 1. This is the way digits have been supplied to the python program written for the neurons training.

TABLE I  
 LOCATIONS AND CODING SHEET FOR THE IMAGES

	A	B	C	D	E	F					
21	1011	30	1011	15	1011	35	1001	20	1011	1	1001
22	0011	31	10011	16	1001	36	1011	7	1011	2	1001
23	0011	32	1011	17	0101	38	1011	10	1001	4	1011
24	1001	33	1001	26	1011	39	1001	5	1001	6	1001
25	1001	34	1001	43	1001	40	1011	9	1011	42	1001
27	1011	3	1011	45	1011	41	1011	8	1011	44	1001
28	0001	11	1011	47	1011	48	1001	13	1011	18	1011
29	1011	12	0011	37	1011	19	1011	14	0111	46	1001

#### E. Performing the Face Recognition

There are four basic functions performed by the algorithm as follows; (i) capturing, i.e. the process of identifying a physical or behavioral, where a template is captured by the system, (ii) extraction; when a unique data is collected from the sample to create a template, (iii) comparison; the process of comparing the new sample and the template, (iv) matching: a decision tool which decides if the features collected from the new sample is of good resemblance. The algorithm analyses the relative position, size, color and shape of the eye relative to the mouth or jaw and check whether it matches the template supplied during the training period [4]. In this process, the tracking of images is done to know if it is higher than the threshold value in the neural network. In cases of characteristics under certain influences of expressions, the neurons are trained to output value far away from the threshold value; in this case the LBPH algorithm traps the images in the grid and binary value equivalents in order to perform a facial recognition exercise.

## IX. RESULTS AND DISCUSSION

### A. The Face Recognition Procedures

1. Datasets stage
2. Training stage
3. Recognition stage

### B. Datasets Stage

This project uses the LBPH algorithm in OpenCV to perform face recognition. In other to use this algorithm, there is a need to create a set of training data with pictures of faces that will be recognized by the system. The database of contained large number of images which have been trained using the command line:

“haarcascade\_frontalface\_default.xml” which is a cascade function trained from a lot of negative images). It is then used to detect similar objects in other images.

To generate images of the people who will be recognized by the system, the “face\_datasets.py” script command line was

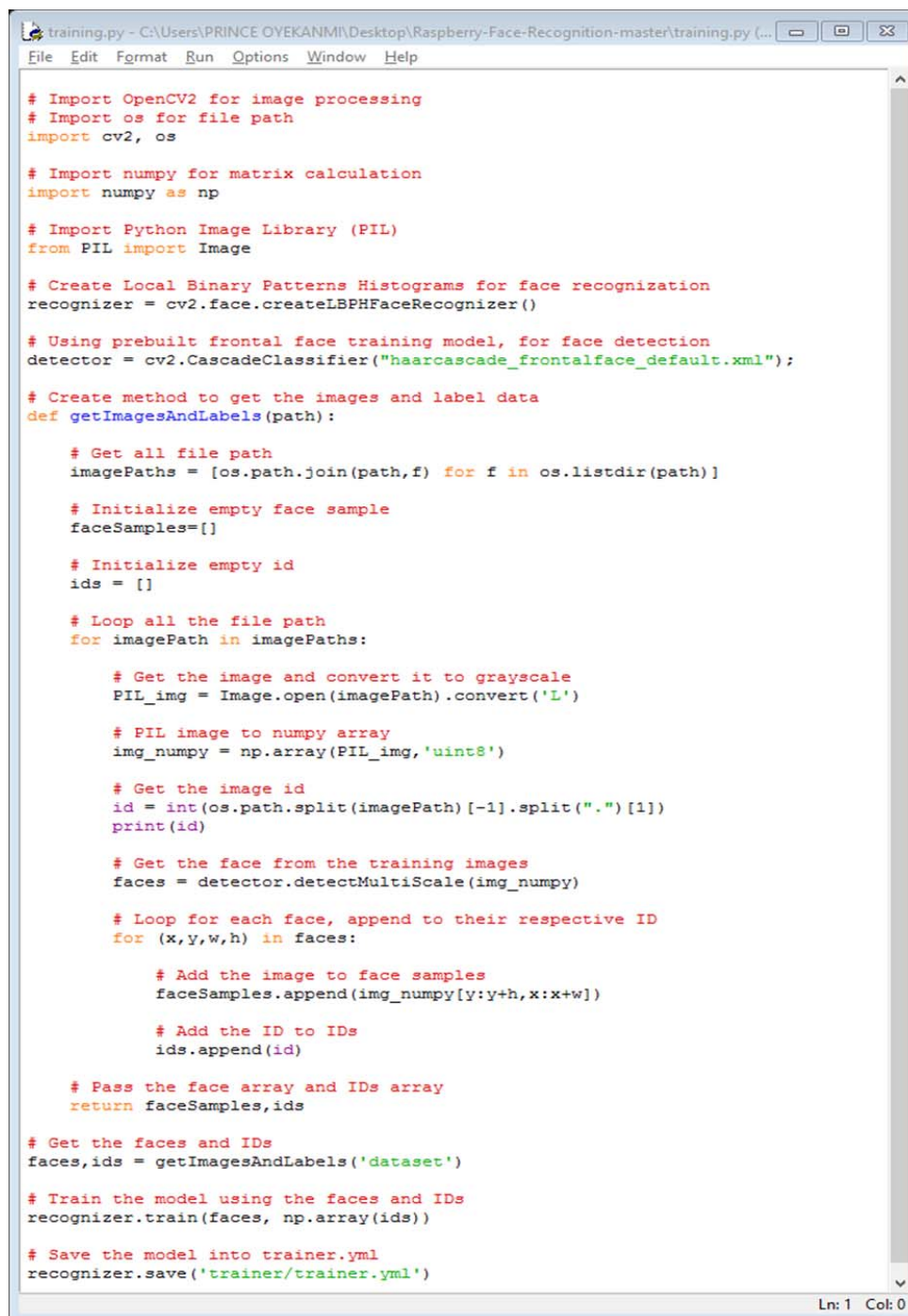
run. This script takes pictures with the camera connected to the Raspberry Pi and writes them to the datasets directory (which will be created by the script if it does not exist). With the box hardware assembled and powered up, which was connected to the Raspberry Pi in a terminal session and navigated to the directory in the project software and executed the following command to run the script below:

```
Python face_datasets.py -face cascades/
```

haarcascade\_frontalface\_default.xml

After waiting a few moments for the python script to load, a dialog box popped up with a video live stream of the environment detected by the camera, and then detects a face. After detecting a face from the live stream, the face is extracted from the environment and saved in the datasets directory.

The algorithm for the datasets is shown in Fig. 10:



```
training.py - C:\Users\PRINCE OYEKANMI\Desktop\Raspberry-Face-Recognition-master\training.py (...)
```

```
File Edit Format Run Options Window Help
```

```
# Import OpenCV2 for image processing
# Import os for file path
import cv2, os

# Import numpy for matrix calculation
import numpy as np

# Import Python Image Library (PIL)
from PIL import Image

# Create Local Binary Patterns Histograms for face recognition
recognizer = cv2.face.createLBPHFaceRecognizer()

# Using prebuilt frontal face training model, for face detection
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");

# Create method to get the images and label data
def getImagesAndLabels(path) :

    # Get all file path
    imagePath = [os.path.join(path,f) for f in os.listdir(path)]

    # Initialize empty face sample
    faceSamples=[]

    # Initialize empty id
    ids = []

    # Loop all the file path
    for imagePath in imagePath:

        # Get the image and convert it to grayscale
        PIL_img = Image.open(imagePath).convert('L')

        # PIL image to numpy array
        img_numpy = np.array(PIL_img,'uint8')

        # Get the image id
        id = int(os.path.splitext(imagePath)[-1].split(".")[1])
        print(id)

        # Get the face from the training images
        faces = detector.detectMultiScale(img_numpy)

        # Loop for each face, append to their respective ID
        for (x,y,w,h) in faces:

            # Add the image to face samples
            faceSamples.append(img_numpy[y:y+h,x:x+w])

            # Add the ID to IDs
            ids.append(id)

        # Pass the face array and IDs array
        return faceSamples,ids

# Get the faces and IDs
faces,ids = getImagesAndLabels('dataset')

# Train the model using the faces and IDs
recognizer.train(faces, np.array(ids))

# Save the model into trainer.yml
recognizer.save('trainer/trainer.yml')
```

Ln: 1 Col: 0

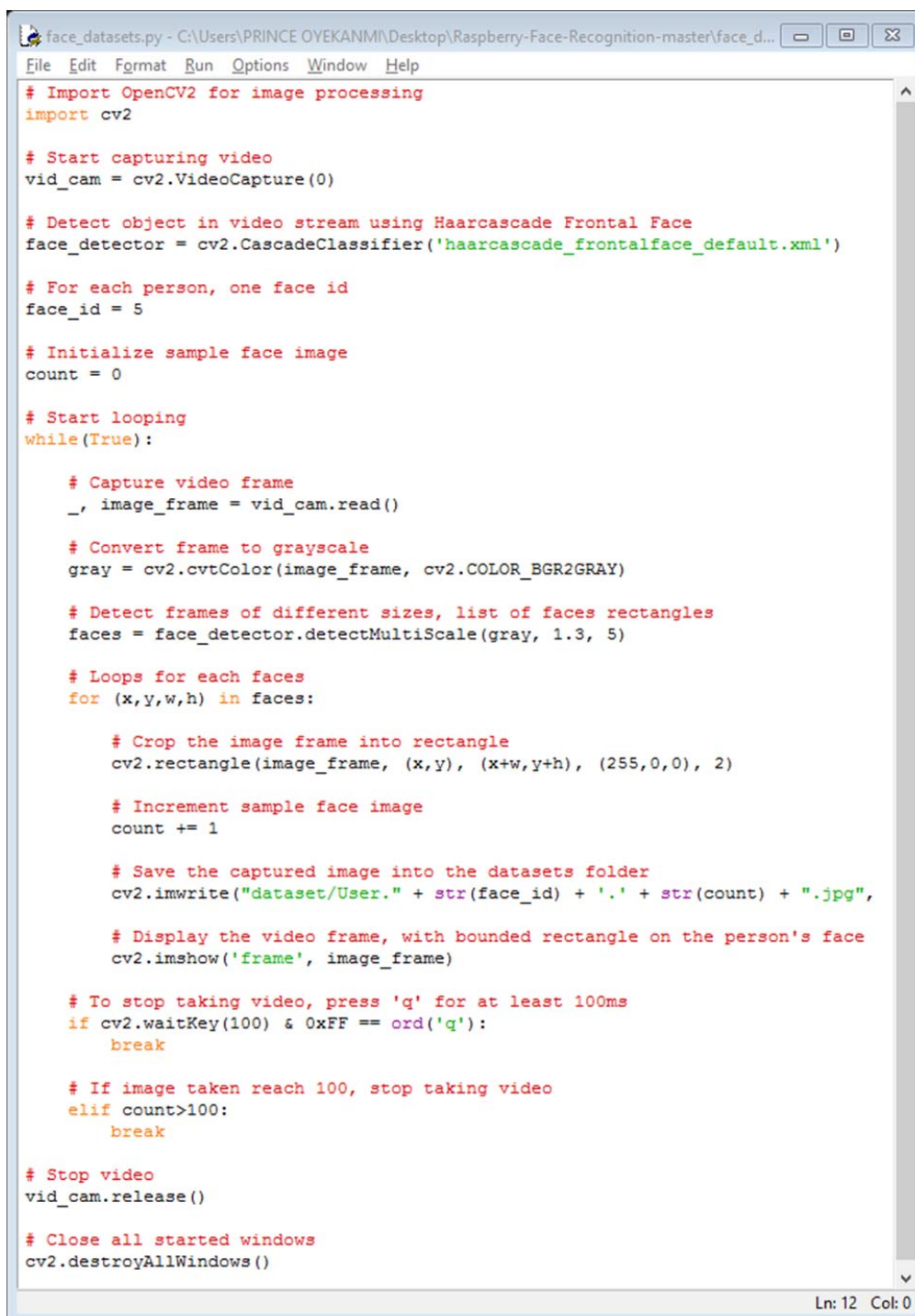
Fig. 10 Face Dataset Algorithm

### C. Training Stage

After detecting and extracting a number of faces for the datasets, the next step is to run the trainer. The Python script to perform the training is Python training.py –face cascades/haarcascade\_frontalface\_default.xml, by connecting the Raspberry Pi in a terminal session and navigating to the directory with the project software and executing the command line.

Thereafter, running the script, the training begins by displaying the numbers of users consecutively till the end of the dataset. The training time depends on the number of dataset images to be trained.

Once the training is complete; the training data would be stored in the trainer directory as ‘trainer.yml’, as can be seen in the Fig. 12. The algorithm for training the images in the dataset is shown in Fig. 11:

A screenshot of a Python script titled 'face\_datasets.py' in a code editor. The script is designed to capture video frames, detect faces using a Haar cascade classifier, crop the faces, and save them to a dataset folder. It includes comments in red and code in black. The script starts with importing cv2, then sets up video capture. It enters a while loop that captures frames, converts them to grayscale, detects faces, loops through each face to crop and save it, and finally releases the video capture and closes windows. The script ends with 'Ln: 12 Col: 0' at the bottom right.

```
face_datasets.py - C:\Users\PRINCE OYEKANMI\Desktop\Raspberry-Face-Recognition-master\face_d...
File Edit Format Run Options Window Help
# Import OpenCV2 for image processing
import cv2

# Start capturing video
vid_cam = cv2.VideoCapture(0)

# Detect object in video stream using Haarcascade Frontal Face
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# For each person, one face id
face_id = 5

# Initialize sample face image
count = 0

# Start looping
while(True):

    # Capture video frame
    _, image_frame = vid_cam.read()

    # Convert frame to grayscale
    gray = cv2.cvtColor(image_frame, cv2.COLOR_BGR2GRAY)

    # Detect frames of different sizes, list of faces rectangles
    faces = face_detector.detectMultiScale(gray, 1.3, 5)

    # Loops for each faces
    for (x,y,w,h) in faces:

        # Crop the image frame into rectangle
        cv2.rectangle(image_frame, (x,y), (x+w,y+h), (255,0,0), 2)

        # Increment sample face image
        count += 1

        # Save the captured image into the datasets folder
        cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg",

        # Display the video frame, with bounded rectangle on the person's face
        cv2.imshow('frame', image_frame)

        # To stop taking video, press 'q' for at least 100ms
        if cv2.waitKey(100) & 0xFF == ord('q'):
            break

        # If image taken reach 100, stop taking video
        elif count>100:
            break

# Stop video
vid_cam.release()

# Close all started windows
cv2.destroyAllWindows()

Ln: 12 Col: 0
```

Fig. 11 Dataset Training Algorithm

#### D. Recognition Stage

In order to recognize people, their faces must have been extracted by the 'datasets algorithm' and also trained using the 'training algorithm' (which uses the LBPH).

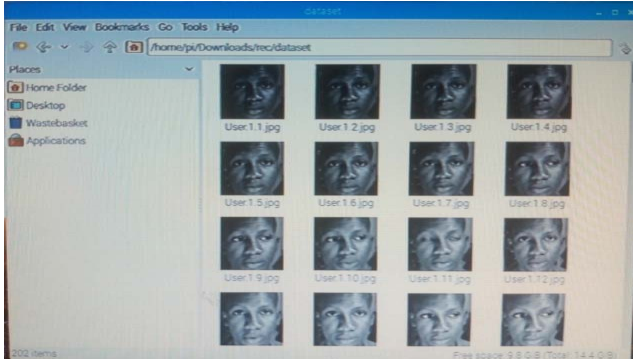


Fig. 12 Dataset Dialog Box

The box hardware assembled and powered up, was connected to the Raspberry Pi in a terminal session and navigated to the directory which contains the project software, datasets sub-director, and the trainer sub-directory, then executed the following command line to run the script below:

Python face\_recognition.py -face cascades/haarcascade\_frontalface\_default.xml

After waiting a few moments for the script to load, a dialog box pops up with a video live stream of the environment detected by the camera, detects a face. The extracted face is processed and compared with the database and the training file. If the extracted face is recognized, it displays the name (which has been inputted in the recognizer algorithm of the user), as in Figs. 13 (a) and (b).



(a) (b)

Fig. 13 Dialog Box of two Recognized Faces

```
face_recognition.py - C:\Users\PRINCE OYEKANMI\Desktop\Raspberry-Face-Recognition-master\fac...
File Edit Format Run Options Window Help
# Import OpenCV2 for image processing
import cv2

# Import numpy for matrices calculations
import numpy as np

# Create Local Binary Patterns Histograms for face recognition
recognizer = cv2.face.createLBPHFaceRecognizer()

# Load the trained mode
recognizer.load('trainer/trainer.yml')

# Load prebuilt model for Frontal Face
cascadePath = "haarcascade_frontalface_default.xml"

# Create classifier from prebuilt model
faceCascade = cv2.CascadeClassifier(cascadePath);

# Set the font style
font = cv2.FONT_HERSHEY_SIMPLEX

# Initialize and start the video frame capture
cam = cv2.VideoCapture(0)

# Loop
while True:
    # Read the video frame
    ret, im = cam.read()

    # Convert the captured frame into grayscale
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

    # Get all face from the video frame
    faces = faceCascade.detectMultiScale(gray, 1.2, 5)

    # For each face in faces
    for (x,y,w,h) in faces:

        # Create rectangle around the face
        cv2.rectangle(im, (x-20,y-20), (x+w+20,y+h+20), (0,255,0), 4)

        # Recognize the face belongs to which ID
        Id = recognizer.predict(gray[y:y+h,x:x+w])

        # Check the ID if exist
        if (Id == 1):
            Id = "Musa"
        # If not exist, then it is Unknown
        elif (Id == 2):
            Id = "Uche"
        else:
            print(Id)
            Id = "Unknown"

        # Put text describe who is in the picture
        cv2.rectangle(im, (x-22,y-90), (x+w+22, y-22), (0,255,0), -1)
        cv2.putText(im, str(Id), (x,y-40), font, 2, (255,255,255), 3)

        # Display the video frame with the bounded rectangle
        cv2.imshow('im', im)

        # If 'q' is pressed, close program
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

# Stop the camera
cam.release()

# Close all windows
cv2.destroyAllWindows()
```

Fig. 14 Face Recognition Algorithm



#### X.FACE RECOGNITION BY THE NEURAL NETWORK

The research made a critical comparison using the algorithm configured with Raspberry Pi 3 model and the machine learning algorithm written in python. The output from the LBPH which was made of gray color is an indication that the neurons are intelligent in detecting the faces of the resemblance images. The results of the 200epoch runs showed that the neurons are effective in the Face recognition exercise. The neurons after the training actually predicted an XOR output as expected from Raspberry Pi 3 model.

#### ACKNOWLEDGMENT

The efforts of some of some senior colleagues in the Department of Physics and Electronics Adekunle Ajasin University cannot be overlooked, with qualitative and reasonable advices coupled with immense contributions to the success of the research. Big thanks to all.

#### REFERENCES

- [1] Pulli, Kari, Baksheev, Anatoly, Korniyakov, Kirill; Eruhimov, Victor, (2012) "Realtime Computer Vision with OpenCV" (1 April, 2012), pp. 40-56.
- [2] C. E. Geoffrey, (2012) "Automatic Access Control System using Face Recognition", Unpublished Thesis Faculty of Electrical Engineering, University of Teknologi Malaysia.
- [3] B. M. Stephen, E. S. Sanjay and R. W. John, (2008) "Face Recognition Technology and Applications", Cambridge University Press.
- [4] Phillips, P. J. (2000) the methodology for face-recognition algorithms. IEEE Transactions on PAMI, 22(10), 1090-1104.
- [5] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986)."Learning internal representations by error propagation."In Rumelhart, D. E., McClelland, J. L. (Eds.), Parallel distributed processing: Explorations in the microstructure of cognition\_Vol. 1. Cambridge, MA. MIT Press.
- [6] Goldstein, A. J., Harmon, L.D., &Lesk, A.B (1971), "Identification of Human Faces Proceeding of the IEEE" pp. 748-760.
- [7] Floyd, Thomas L. (2003) Digital fundamentals. (8th ed.).New Jersey: Prentice Hall.
- [8] T. Ojala, M. Pietikanen, and D. Harwood (1996), "A Comparative Study of Texture Measures with classification based on Feature Distributions, Pattern Recognition" Vol. 29, pp. 51-59.