Educate Elevate Enlighten

JSS Mahavidyapeetha

JSS Science And Technology University

(Established Under JSS Science and Technology University Act No. 43 of 2013)

(Formerly Known as SJCE)

DATA COMMUNICATION

Intrusion Detection Systems

Submitted to:

Mr. Manju N

Assistant professor

Dept. of Information Science and Engineering

JSS Science and Technology University, Mysuru

Submitted by:

Sl.No.	USN	NAME
1.	01JST18IS019	Karthik H S
2.	01JST18IS036	Ranjan M B
3.	01JST18IS057	Chandrakanth
4.	01JST18IS060	Hruthik Gowda K N

ABSTRACT:

Face recognition has been a fundamental part of the latest security technology with its applications ranging from smartphones to self-driving cars to prevent intruders from misuse of data and its applications. Convolutional neural network(CNN) has been exceedingly opted over other network types for facial recognition due to its high accuracy. The CNN follows a hierarchical model that works on building a network, like a funnel, and finally gives out a fully-connected layer where all the neurons are interconnected and the output is processed.

Our project distinguishes intruders and authorized users using signature-based detection. It is built using dlib's state-of-the-art face recognition library, which in turn is built using CNN. The model used has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark and is implemented using Python 3.9.

PROBLEM STATEMENT:

To differentiate between authorized users and intruders using convolutional neural network algorithm of the deep learning domain by implementing face recognition.

INTRODUCTION:

With an emerging development in the field of Computer Networks and Networking Technology, it is necessary to optimize the Intrusion Detection Systems(IDS) that detect and avoid cyber-attacks. The network attack detection methodologies should be more intelligent and efficient than before to prevent growing hacker technology. A software or a device that monitors networks or a system for any kind of malicious activity and typically reports it to the administrator or management system is known as Intrusion Detection Systems(IDS). But the traditional intrusion detection algorithms work on the principles of Data Mining association techniques. However, they have various drawbacks to fully extract the characteristic information of intruders and they have poor timeliness and poor generation capacity. In this synopsis, we propose an intrusion detection model that uses Convolutional Neural Networks(CNN) for face recognition.

Convolutional Neural Network(CNN) is a class of deep neural networks that are mainly applied in the various fields of image classification and in analyzing visual imagery. CNN uses the multilayer perceptron and contains one or more convolutional layers that can be either pooled or entirely connected. These Convolutional layers can generate feature maps that can record any region of an image that is ultimately broken into multiple rectangles and it can be sent further nonlinear processing. Hence CNN is advantageous as compared to traditional feed-forward neural networks which can support linear processing but they do not support non-linear processing. CNN can detect the important features of an image without any human supervision, so it can detect the face of a person more efficiently than any other neural network models. This makes CNN use face recognition to build trained models like AlexNet.

Face recognition technology plays a vital role in biometric authentication and network security. Here we implement a deep learning algorithm that uses CNN to achieve face recognition. It is a multi-layered network model that has been trained to detect malicious activities by performing image classification and detection techniques. It can produce an accuracy of 98.5% by using 2500 trained parameters of images in a class. So it has a higher accuracy rate as compared to machine learning algorithms that use portable security cameras to implement face recognition.

AIM: To implement Intrusion Detection System with the help of Face Recognition.

DOMAIN:

This model uses a class of deep neural networks named Convolutional Neural Networks of deep learning domain. A face recognition is a biometric information process, it’s applicability is easier and the working range is larger than others, i.e., fingerprint, signature and iris scanning. This CNN uses the multilayer perceptron and contains one or more convolutional layers that can be either pooled or entirely connected. So CNN can detect important features of an image without any Human supervision. Convolutional Neural Networks is also advantageous over traditional feed-forward neural networks which can support only linear processing.

APPLICATIONS:

Face recognition has a wide range of applications in the fields of cyber security, biometrics, forensic science and in various fields of business, science and technology. It is used in face unlocking mechanisms in mobile phones. It has been applied in Smarter advertisements and to identify the people or criminals on social media. It can be used to find missing persons, to protect the law enforcements and to aid forensic investigators

CHALLENGES:

Although the CNN model is quite popular and comparably has an advantage over other models for face recognition it has its drawbacks and challenges.

1) CNN does not encode the position and orientation of the object:

Convolution neural network algorithms fail to detect different orientations and positions of the image captured. That is, CNN primarily looks at an oval shape(for the face structure), two eyes, a nose, and a mouth to detect whether it's a face or not But fails to capture the spatial relationships between these components. The algorithm basically captures the edges, gradients, and dense areas and converts it into matrices, and operates accordingly.

2) Inefficiency to be spatially invariant to images:

Although the CNN based face recognition models have good efficiency in determining the previously recognized face, it fails in associating and identifying the same face when the object is tilted or orientated at a different angle. For CNN, it's extremely hard as there isn't any built-in understanding of 3D space.

3) The exploitation of colors and degradation in performance:

The CNN model shows degradation in performance when exposed to images which are blurry, noisy, and low- resolution based which are quite disadvantaged as it resembles a part of a real-life scenario. With the current growth in technology, better innovative algorithms can be developed to overcome this drawback.

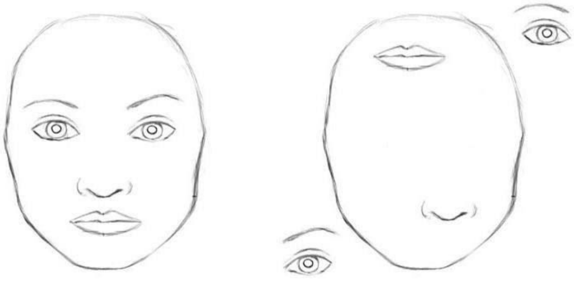


Fig1: To a CNN, both pictures are similar, since they both contain similar elements

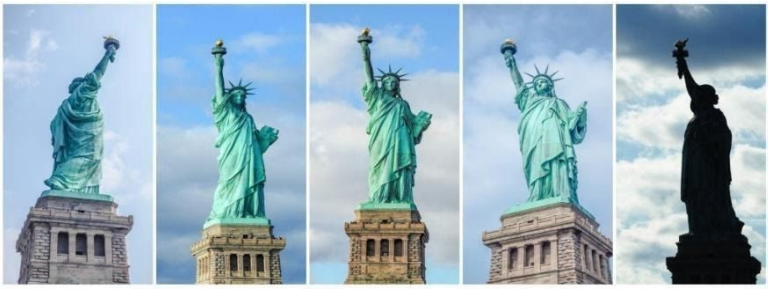


Fig 2: Your brain can easily recognize this is the same object, even though all photos are taken from different angles. CNNs do not have this capability.

LITERATURE SURVEY:

Giuseppe Amato et al. proved that Deep Learning and Convolutional Neural Network (CNN) approaches have been proposed that highly improved the performance in executing the face recognition task. Yihan Xiao et al. used convolutional neural network (CNN) because as compared to other deep
learning algorithms, the advantage of CNN is that it shares the same convolutional kernels. Adam Geitgey et al proposed that deep learning can be useful to build a face recognition system in python using openCV and dlib(a state-of- the-art face recognition built with CNN). They used a step-by-step procedure to explain different deep learning algorithms. Cheng Xing proposed that CNN–IDS, an intrusion detection algorithm based on the typical CNN model Lenet-5. By improving its network structure and applying batch normalization (BN) optimization, this algorithm can be effectively used to detect network intrusion data.
MusabCoúkun et al. implied that The prominent features of the proposed algorithm(CNN) is that it employs the batch normalization for the outputs of the first and final convolutional layers and that makes the network reach higher accuracy rates.

IMPLEMENTATION:

FUNCTIONAL REQUIREMENT:

System RequirementPhase

The purpose of this phase is to determine the project's main goal and how the system will function. All possible requirements of the system to be developed are captured in this phase.

AnalysisPhase

In this phase analysis of the user's requirement is carried out. This is to determine the scope of the users.
Thing to be cogitated are:

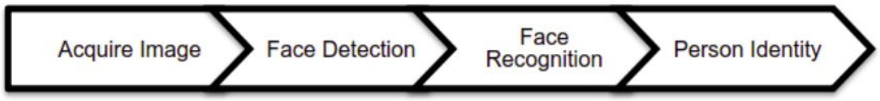
Scope of users

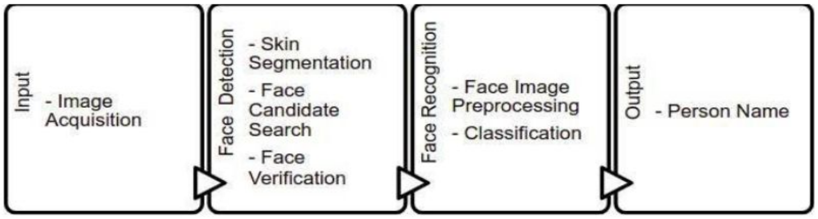
Purpose of the system
Suitable equipment’s (camera, laptop etc.)

The overall purpose of the analysis phase is to define the project goals that have been determined in the requirements phase into defined functions and operation of the intended system.

Design Phase

This is the plan of how the system will look like and how it works. It describes the desired features and operations in detail and may include screen layouts, process diagrams, pseudocode and other documentation. A sample of the project is developed in this phase. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design(what are the interfaces going to look like)anddatadesign (what data will be required). During these phases, the software's overall structure is defined and the logical system of the product is developed in this phase. It also helps in specifying hardware and system requirements and also helps in defining overall system architecture.





Implementation and Unit Testing Phase

This phase is considered to be the longest phase of the software development life cycle. This is so because this is where the code is created and work is divided into small programs that are referred to as units. This phase includes unit testing whereby units will be tested individually for their functionality before the whole system. Unit testing mainly verifies if the modules also known as units meet project specifications.

Testing Phase

This is the main testing phase in the SDLC, as the project is divided into small modules in the previous phase then the modules will be integrated together to test the system as whole. This is to make sure that the modules work together as intended by the developer (as in the specifications) and required by users. It also checks for bugs, errors and ensures the system is able to work in the intended platform. After ensuring that the product had solved the problem the system is then delivered to the customers

Maintenance / Operation Phase

Not all problems can be seen directly, but they occur with time and as other problems they need to be solved. Usually these kinds of problems come in picture after the practical use of the system they are never found throughout the development life cycle. This phase of the Waterfall Model is considered to be very long, it never ends. The changes that occur after the product is handed to the users must not affect the main operation of the system, so a system must be developed in a way that it will adapt to change.

Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated.

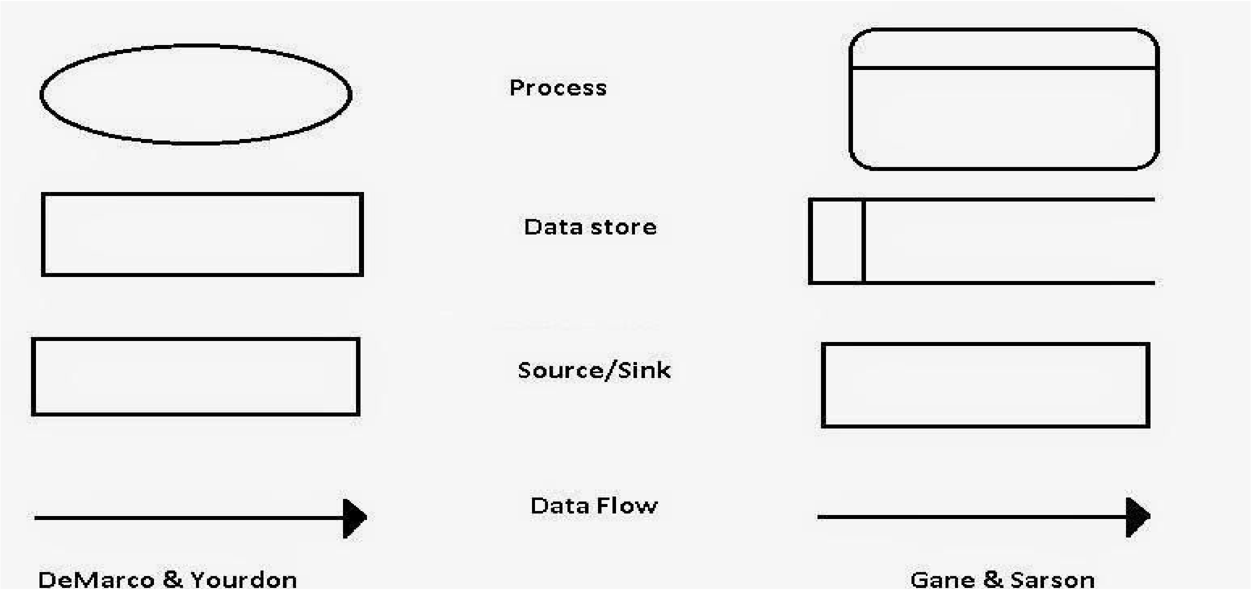
DFDs represent the following:

1. External devices sending andreceivingdata
2. Processes that change that data
3. Dataflow Themselves
4. Data Storage Locations

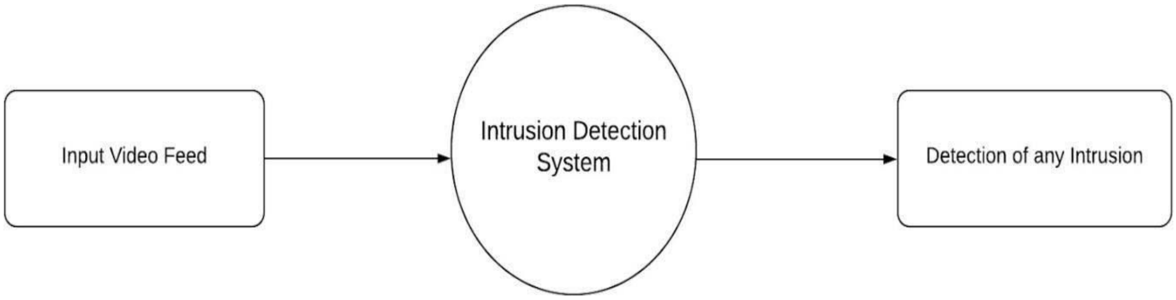
DFDs are the method of choice over technical descriptions for three principal reasons:

1. DFDs are easier to understand by technical and non technical audiences.
2. DFDs can provide a high level system overview, complete with boundaries and connections to other systems.

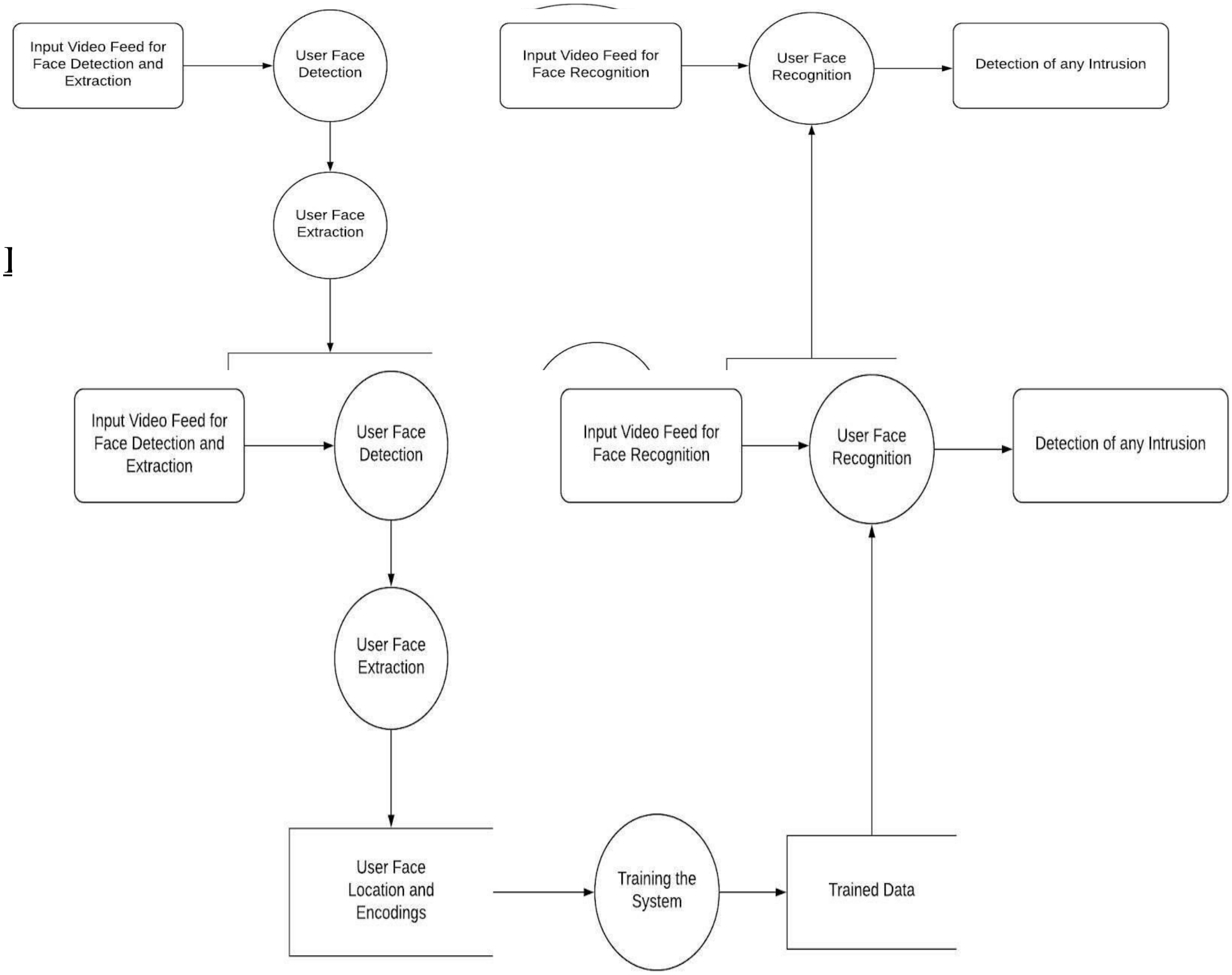
DFDs can provide a detailed representation of system components



LEVEL 0



LEVEL1


Input Video Feed for Face Detection and Extraction
User Face Detection
User Face Extraction
Input Video Feed for Face Recognition
User Face Recognition
Detection of any Intrusion
Input Video Feed for Face Detection and Extraction
User Face Detection
User Face Extraction
Input Video Feed for Face Recognition
User Face Recognition
Detection of any Intrusion
User Face Location and Encodings
Training the System
Trained Data

Running the program

Step 1: Training the model

1) To train the model with your face images,run

python capture.py

2) Now to capture your face press C key.

(If no face detected you will be prompted on the terminal / cmd).

3) Now, you will prompt to enter your name, on the terminal /cmd.

(If the image name is already present / exists, you will prompt to enter another name or overwrite the existing entry of the image).

4) After this, training the model for your image gets completed.

Step 2: Detection of any Intrusion.

I. For detecting any intrusion, run the script.

   1. **Pythonscript.py**

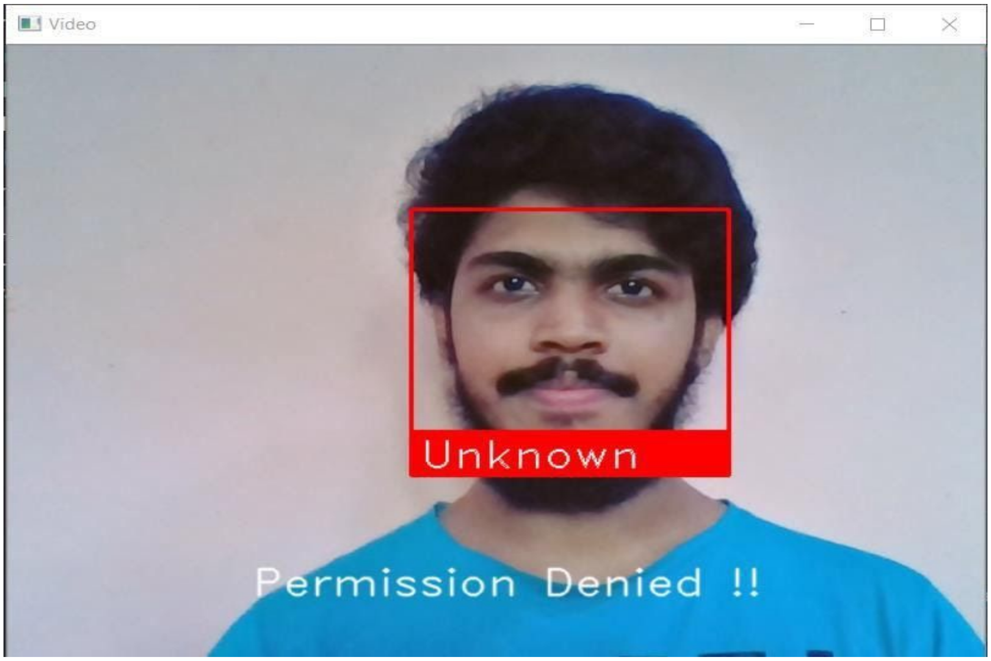
II. The image window will display the person's name, if that face exists in the database, and the system will prompt **Permission Granted !!**message.

III. Else if the face does not exist in the database, the image window will display $_{\text{Unknown}}$ with the face, and will prompt **Permission Denied !!**message.

   a. To exit the process press **q** key.

RESULT AND DISCUSSION:

After the process of running the Project (ie, python script.py)

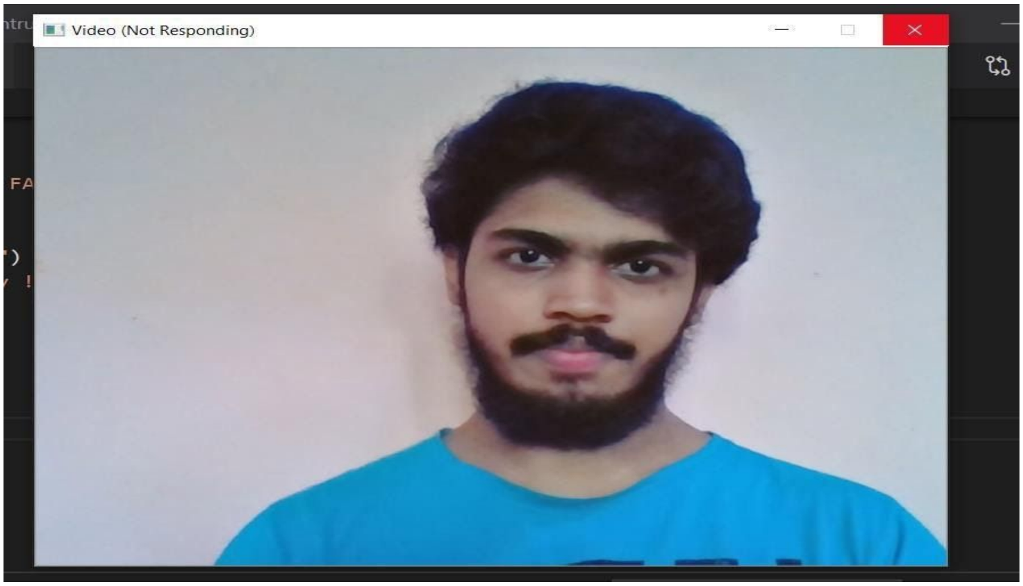


Unknown

Permission Denied !!

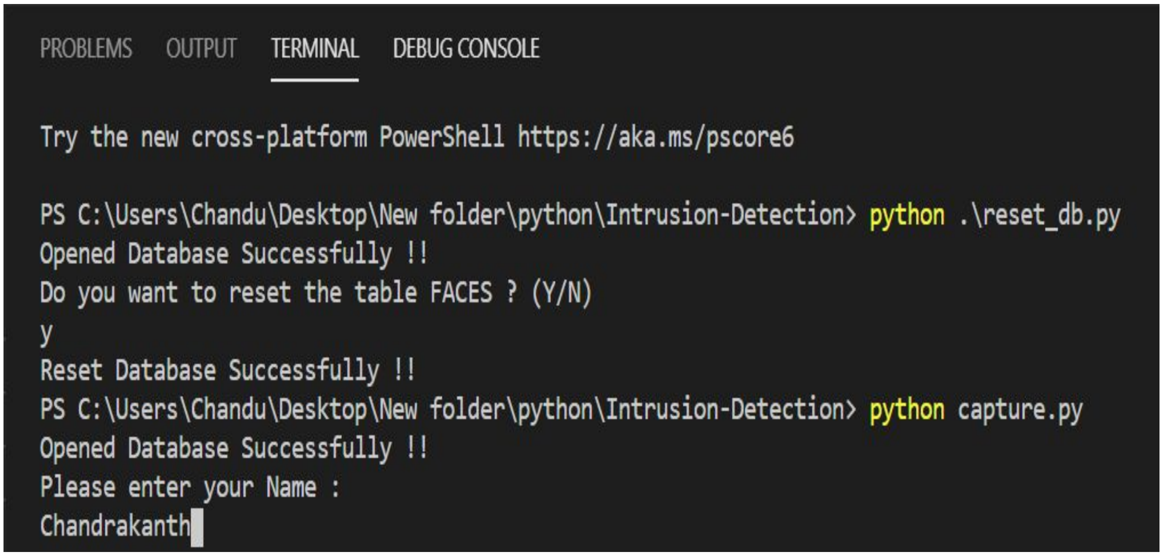
In the above image, it shows Permission Denied !! message, which states that the face does not exist in the database, the image window will display Unknown.

Inorder to come across this, the face does exists in the database. Now adding face of a member to database.

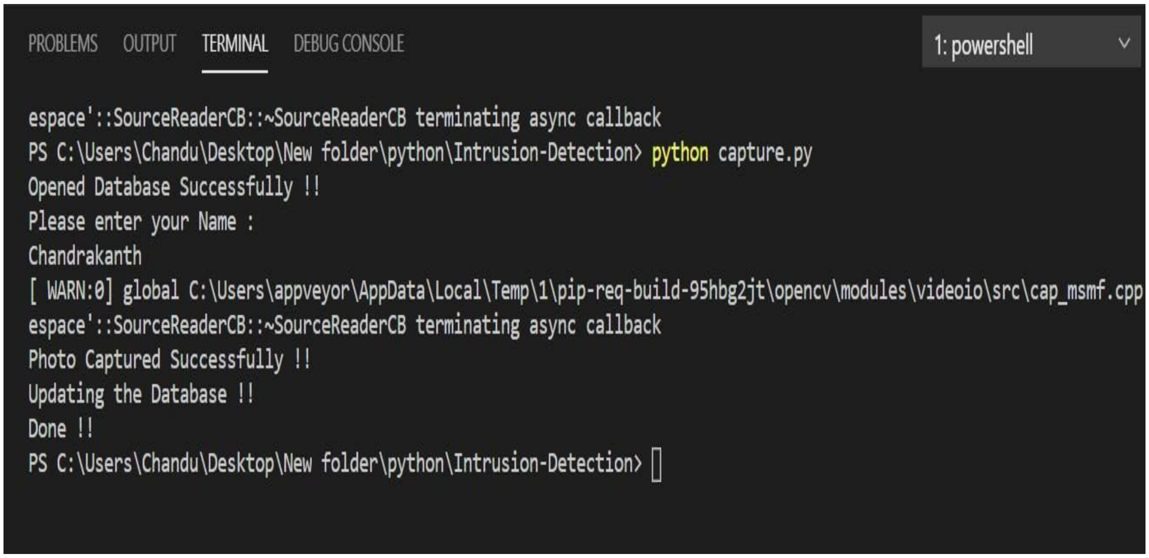


Press C to capture the face.

It will ask to enter the name.



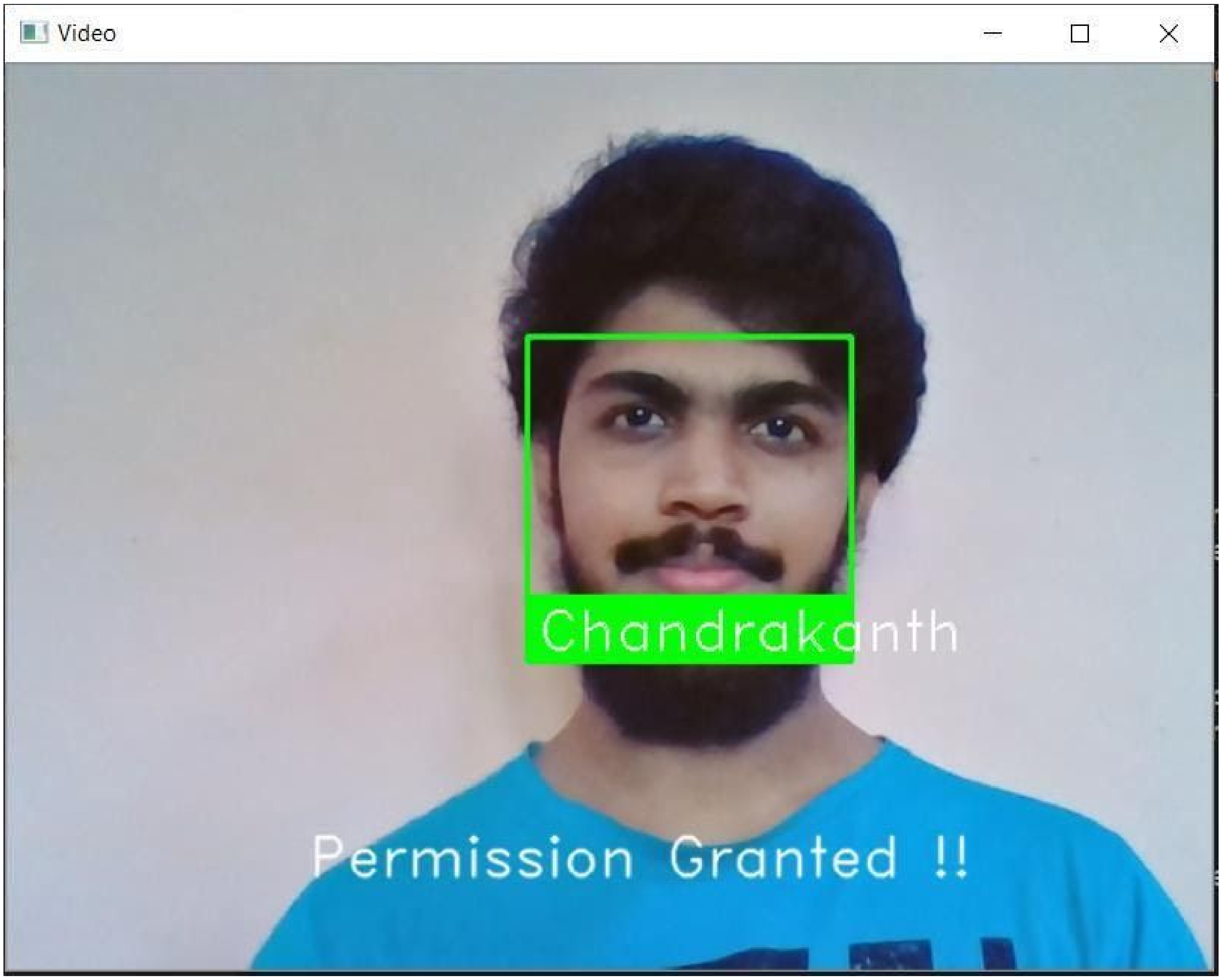
```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Chandu\Desktop\New folder\python\Intrusion-Detection> python .\reset_db.py
Opened Database Successfully !!
Do you want to reset the table FACES ? (Y/N)
y
Reset Database Successfully !!
PS C:\Users\Chandu\Desktop\New folder\python\Intrusion-Detection> python capture.py
Opened Database Successfully !!
Please enter your Name :
Chandrakanth
```



```
espace'::SourceReaderCB::~SourceReaderCB terminating async callback
PS C:\Users\Chandu\Desktop\New folder\python\Intrusion-Detection> python capture.py
Opened Database Successfully !!
Please enter your Name :
Chandrakanth
[ WARN:0] global C:\Users\appveyor\AppData\Local\Temp\1\pip-req-build-95hbg2jt\opencv\modules\videoio\src\cap_msmf.cpp
espace'::SourceReaderCB::~SourceReaderCB terminating async callback
Photo Captured Successfully !!
Updating the Database !!
Done !!
PS C:\Users\Chandu\Desktop\New folder\python\Intrusion-Detection>
```

Now the photo is captured successfully and added to the database.

Now following the same method to run the project.



Since the face exits in the database, the image window displayed the person's name and the system prompted Permission Granted !! message.

.

METHODOLOGY;

Why facial recognition?

The purpose of using facial recognition as an Intrusion Detection System using deep learning is for the detection and prevention of crime. Biometric recognition forms a strong bond between a person and his identity as biometric traits cannot be easily shared, lost, or duplicated. Hence, biometric recognition is fundamentally superior and more resistant to social engineering attacks than the two conservative methods of recognition, namely, passwords and tokens.

How is it done?

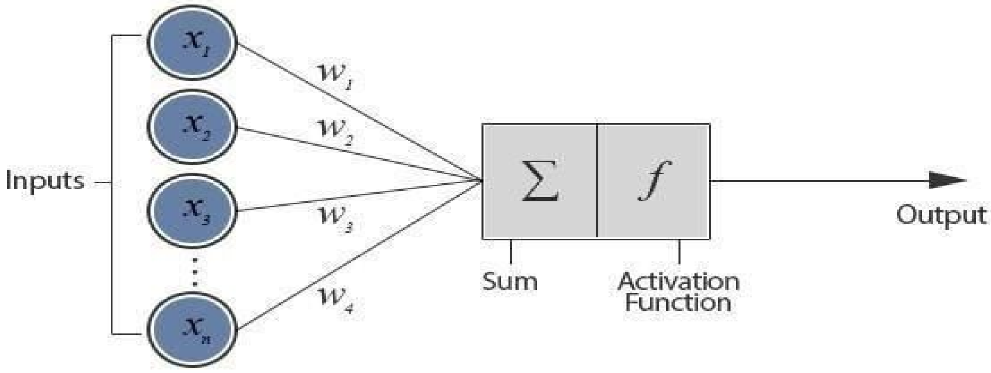
- Convolution Neural Network (DeepLearning).

Convolution Neural Network

Neural Networks + Convolutions = CNN

Neural Networks are composed of Artificial Neurons which simulate biological neurons in a limited way.

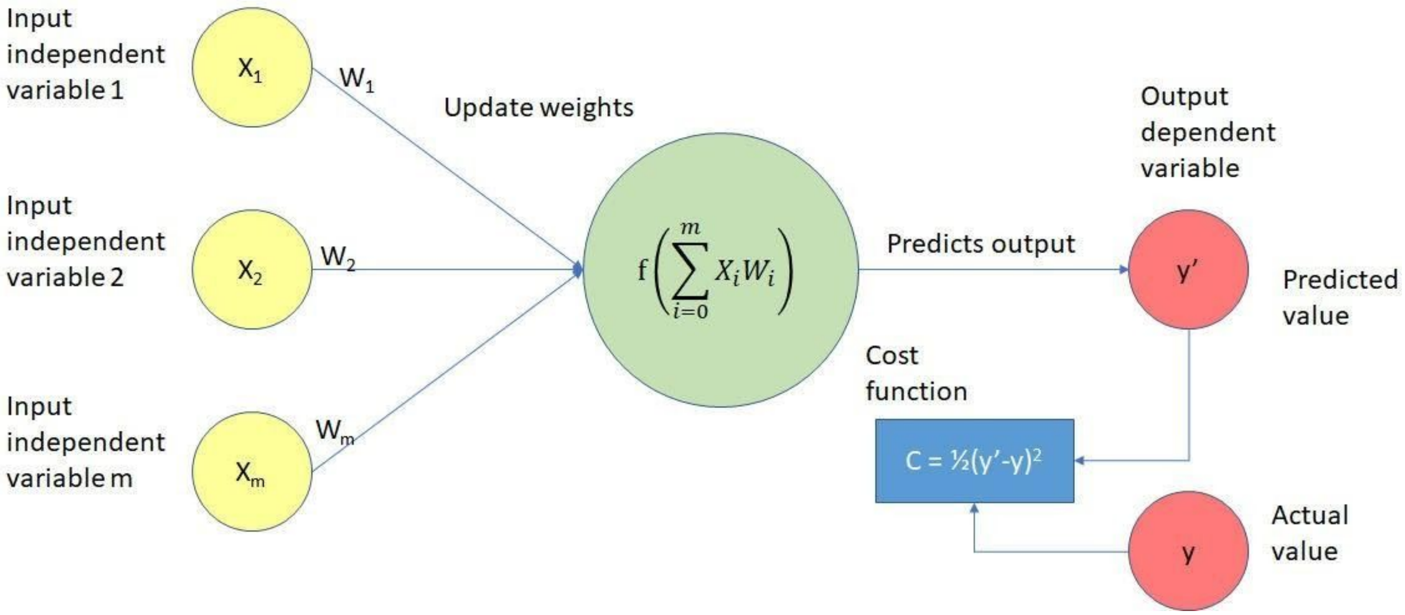
The Artificial Neuron



InputLayer HiddenLayer OutputLayer

The artificial neuron consists of an input layer, a hidden layer, and an output layer. The input layer consists of nodes, each node represents an independent variable. The inputs are passed to the hidden layer which consists of Neurons and the input and hidden layer are connected by a set of weights. The hidden layeris where all the computations take place which consists of activation functions. Few activation functions are threshold function, sigmoid function, hyperbolic tangent function, rectifier function. All the layers consist of neurons. Each neuron in the hidden layer thinks differently, each neuron doesn't consider all the input parameters, it selects the input parameters which are important for it. The neuron picks up those input parameters which it feels important and then the activation function is activated and it will fire up only when certain criteria are met, and that contributes to the value in the output layer.

How do Neural Networks learn?


Input independent variable 1
X_1
W_1
Update weights
Output dependent variable
Input independent variable 2
X_2
W_2
$f\left(\sum_{i=0}^{m} X_i W_i\right)$
Predicts output
y'
Predicted value
Cost function
Input independent variable m
X_m
W_m
C = ½(y'-y)²
Actual value
y

The Neural Networks are trained with predetermined data. Initially, the predicted value is different from the actual value, we use a function called cost function which determines the error in the computation and a process called Back Propagation takes place wherein the weights are adjusted and a new output is obtained. This process takes place recursively until the error in the cost function reaches the optimal minimum.

How are the weights adjusted?

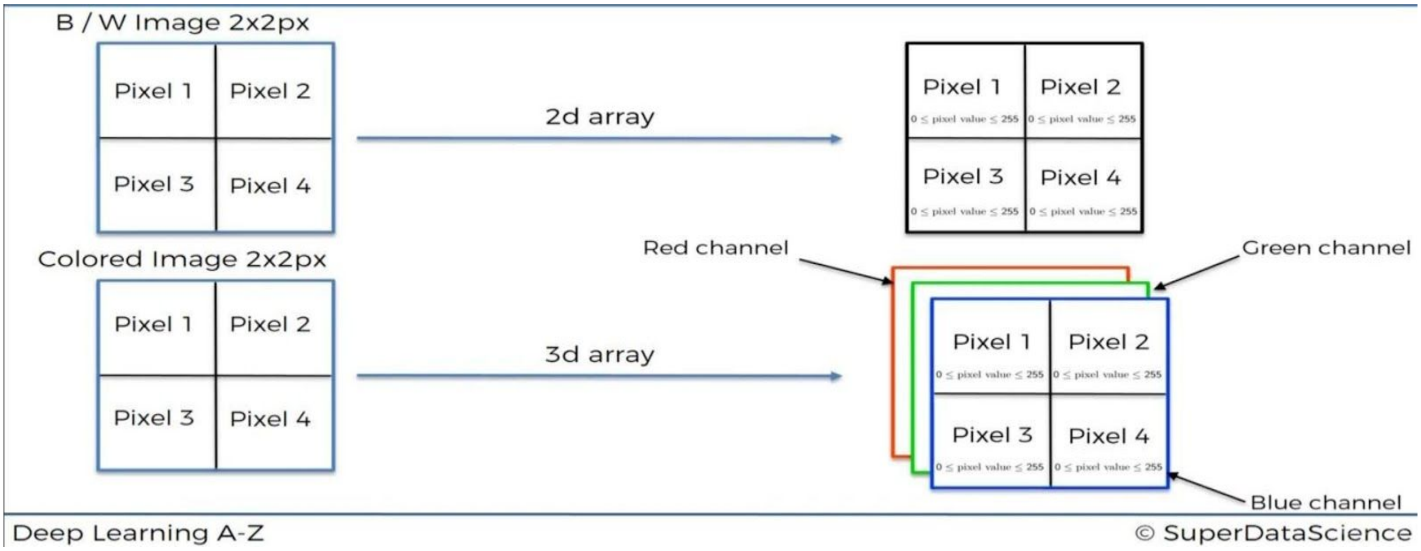
- Bruteforce
- GradientDescent

The brute force approach is applicable when there is a minimal number of weights, and as the number of weights increases it is nearly impossible to randomly select the weights, so the brute force approach is not the way of finding the optimal weights.

Gradient Descent, the most used algorithm to train neural networks is gradient descent. The gradient is a numeric calculation allowing us to know how to adjust the parameters of a network in such a way that its output deviation is minimized.

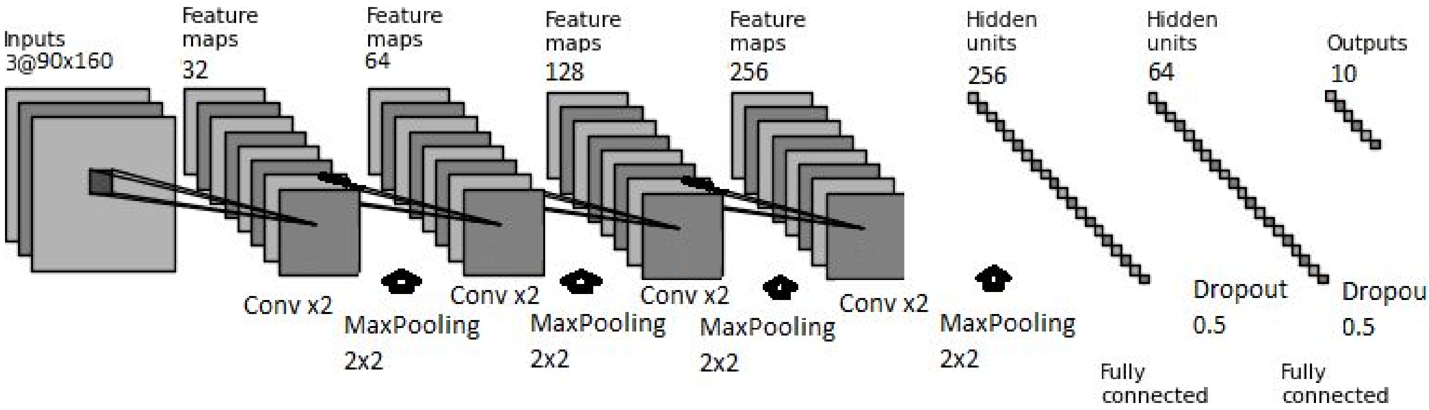
Convolution Neural Networks

Inputimage → → CNN Output Label (image class)

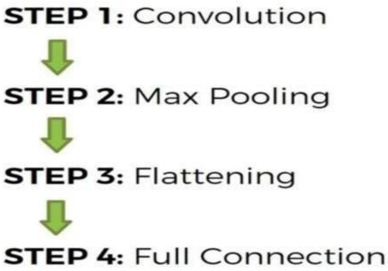


Deep Learning A-Z © SuperDataScience

The input image can be a black and white picture or a colored picture. Neural networks leverage the fact that the black and white image is a two-dimensional array and a colored image is a two-dimensional array. The way we see on the left is just the visual representation. The computer sees it as an array and each pixel consists of values from 0 to 255, the values indicate the intensity of the color. 0-completely black pixel and 255-will be completely white pixels and between them, we have the grayscale range. In the color image, we have three layers blue, green, and red. Each pixel in a color image consists of three values (RGB).



Steps involved:



STEP 1: Convolution

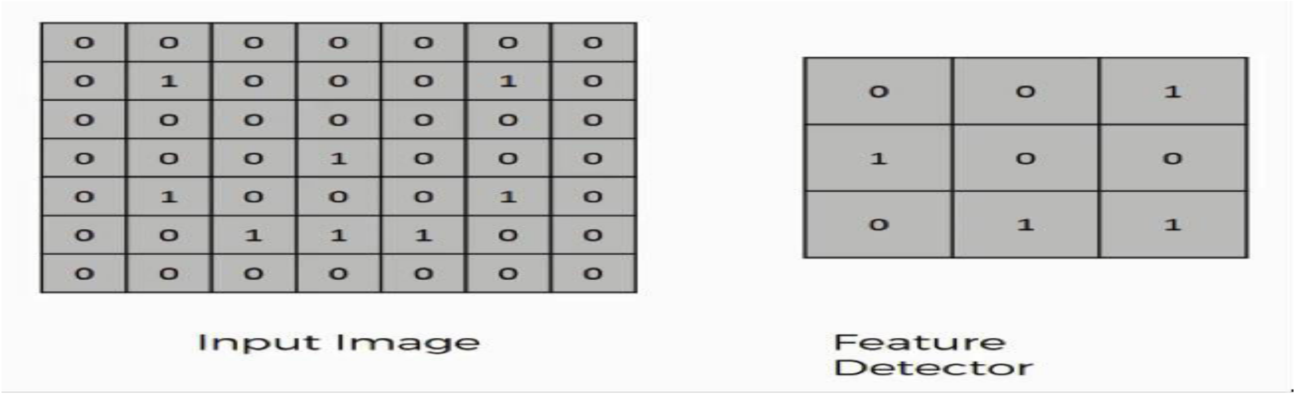
STEP 2: Max Pooling

STEP 3: Flattening

STEP 4: Full Connection

<u>Convolution</u>

The convolution operation:



0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

0	0	1
1	0	0
0	1	1

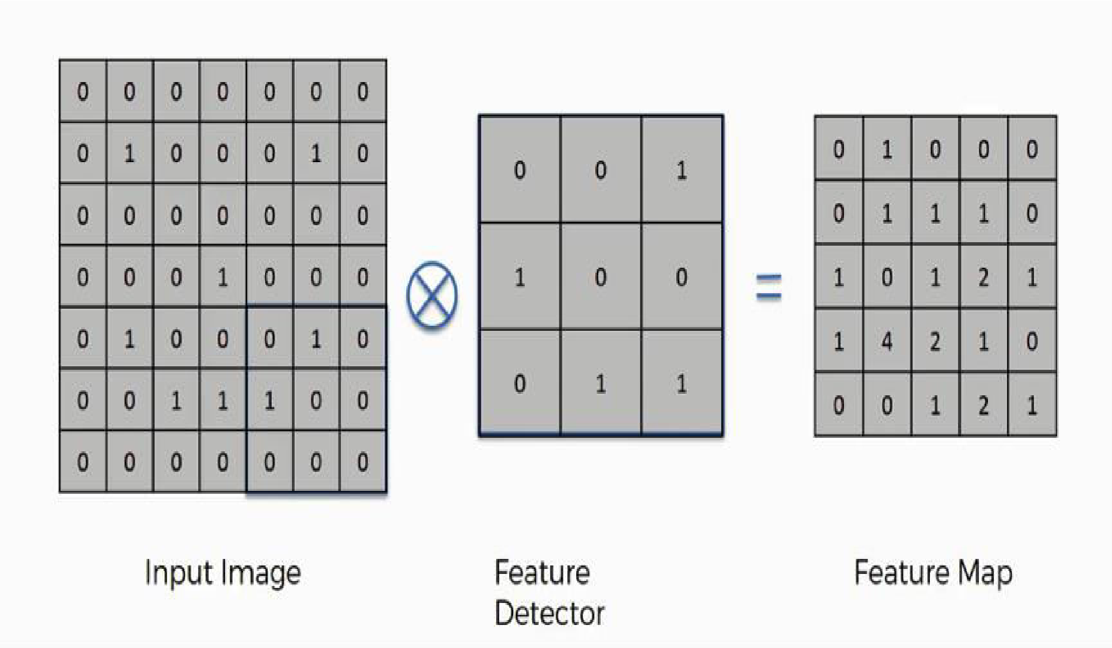
Feature Detector

Sometimes a 5×5 or a 7×7 matrix is used as a feature detector, but the more conventional one is a 3×3 matrix. The feature detector is often referred to as a “kernel” or a “filter”.

- The feature detector is placed over the input image beginning from the top-left corner within the borders we see demarcated above, and then the number of cells in which the feature detector matches the input image are counted.
- The number of matching cells is then inserted in the top-left cell of the featuremap.
- The feature detector moves one cell to the right and does the same thing. This movement is called a stride. Since we are moving the feature detector one cell at a time, that would be called a stride of one pixel.
- What we will find in this example is that the feature detector's middle-left cell with the number 1 inside it matches the cell that it is standing over inside the input image. That's the only matching cell, and so we write “1” in the next cell in the feature map, and so on and so forth.

- After we have gone through the whole first row, we can then move it over to the next row and go through the same process.

It's important not to confuse the feature map with the other two elements. The cells of the feature map can contain any digit, not only 1's and 0's. After going over every pixel in the input image in the example above, we would end up with these results:



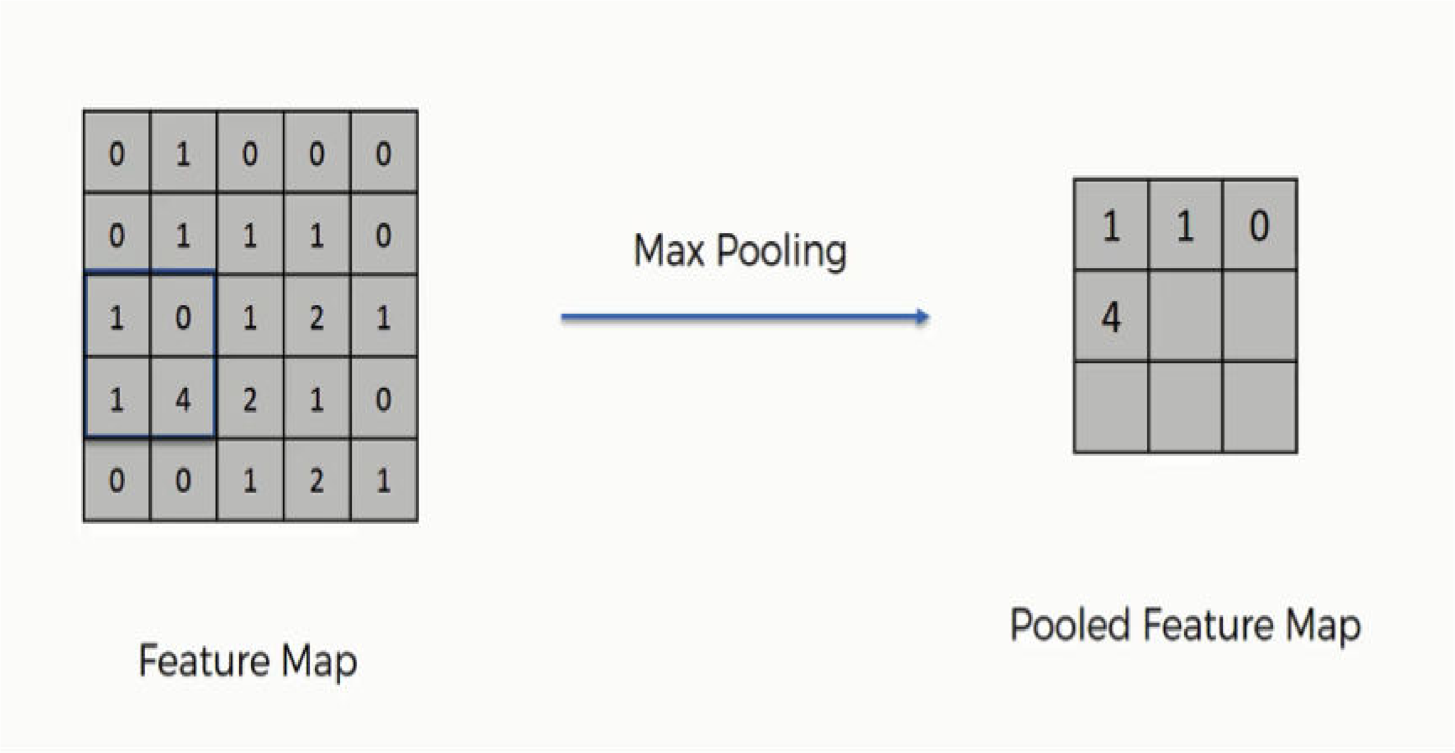
0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

0	0	1
1	0	0
0	1	1

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

The image on the right is called a feature map. We have reduced the size of the image and the reduction of the size depends on the stride of the feature detector. The size of the image is reduced because it will be easier to process it and it will be faster. The feature map that we end up with has fewer cells and therefore less information than the original input image. The very purpose of the feature detector is to detect certain features of certain parts of the image that are integral. The example we gave above is a very simplified one, though. In reality, convolutional neural networks develop multiple feature detectors and use them to develop several feature maps which are referred to as convolutional layers.

Max pooling A pooling layer is another building block of a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network. The pooling layer operates on each feature map independently. The most common approach used in pooling is *max pooling.*



0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

1	1	0
4		

We place a 2×2 box at the top-left corner, and move along the row. For every 4 cells our box stands on, we will find the maximum numerical value and insert it into the pooled feature map.

This process is what provides the convolutional neural network with the "spatial variance" capability. In addition to that, pooling serves to minimize the size of the images as well as the number of parameters which, in turn, prevents an issue of "overfitting" from coming up.

CONCLUSION:

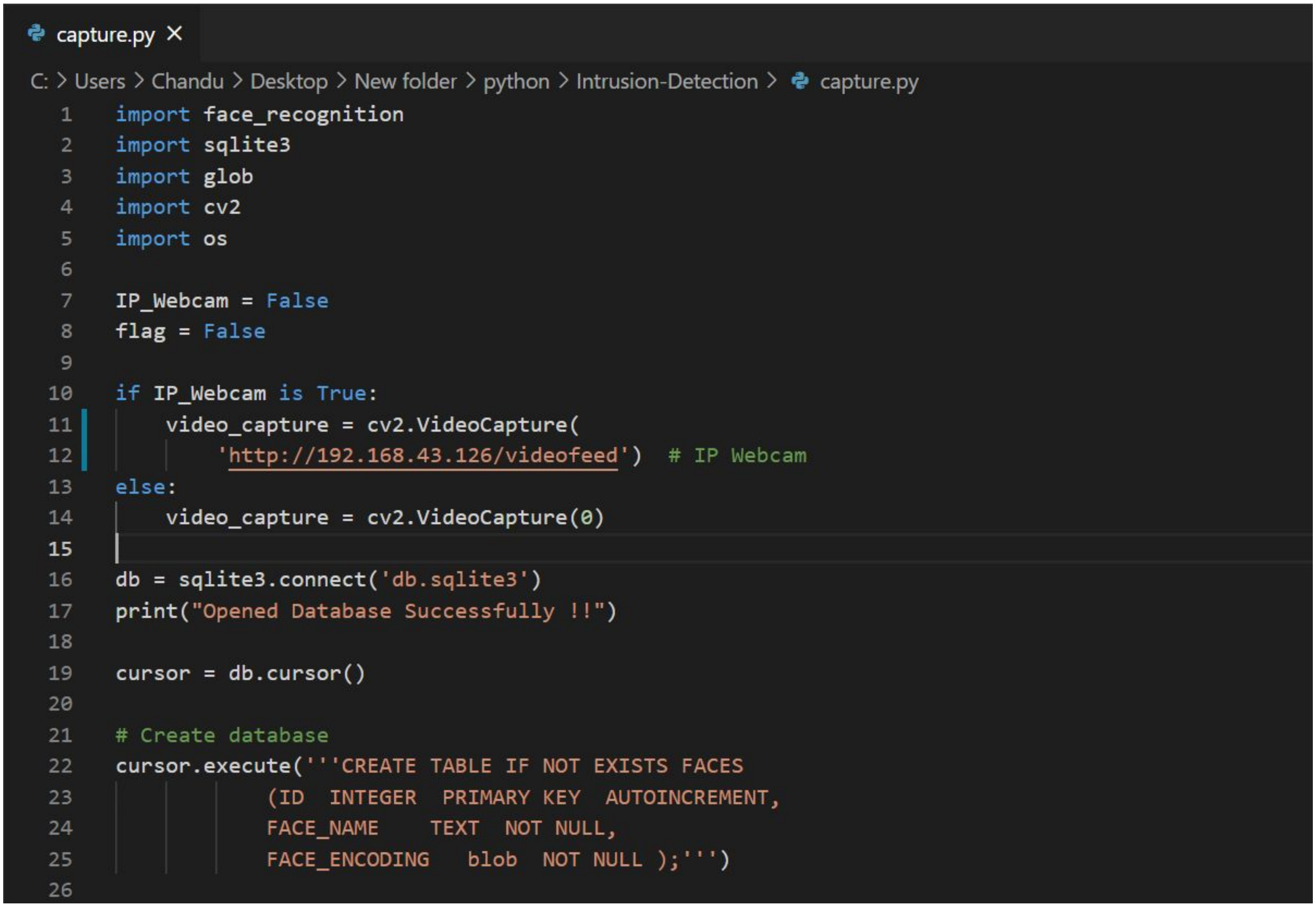
The project is built using dlib's state-of-the-art face recognition library(built with convolutional neural network) and implemented in python 3.9. The application is able to differentiate between an intruder and an authorized user very conveniently. The model has an accuracy of 99.38% on the Labeled Faces in the Wild Benchmark.

FUTURE WORK

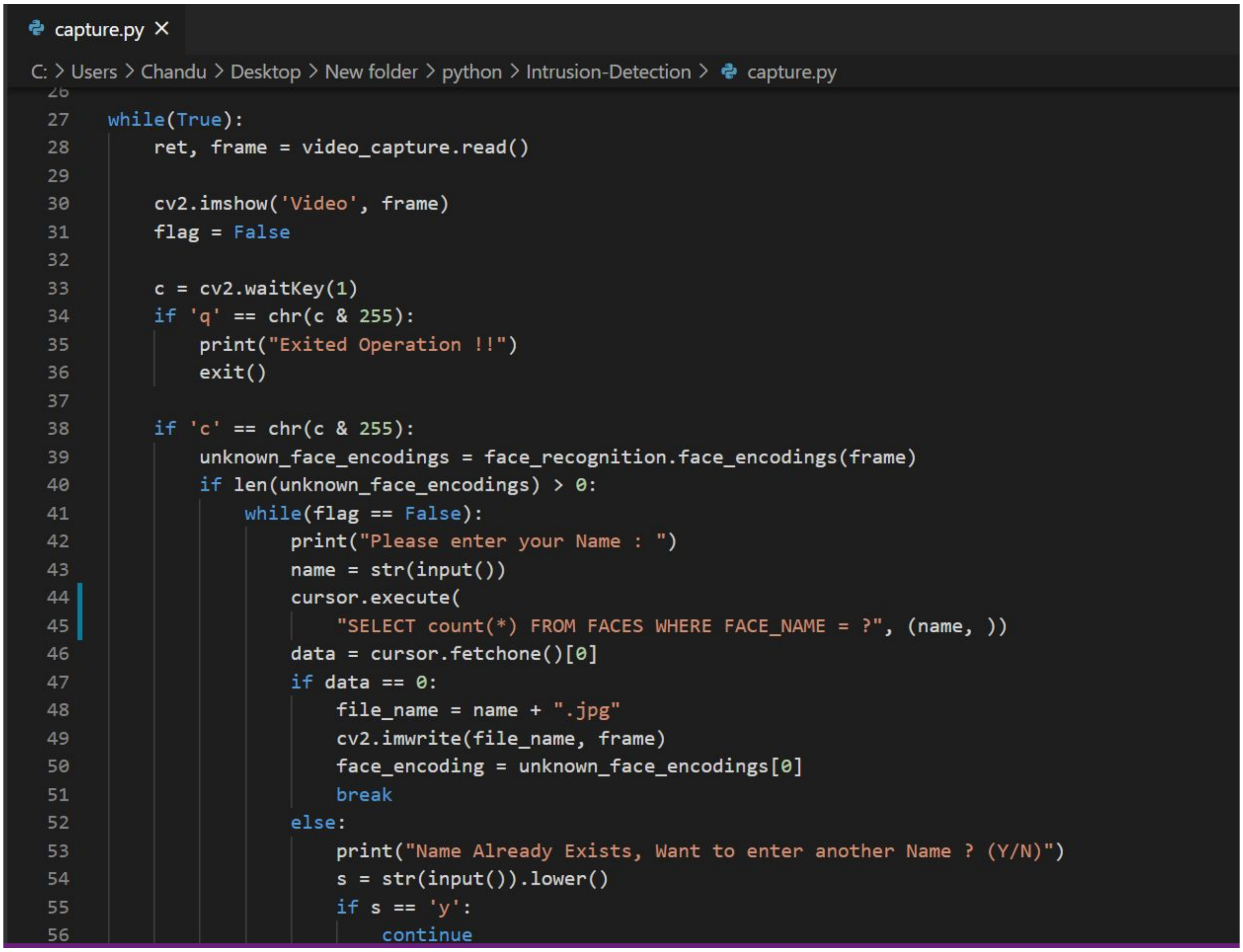
- Generate from one face a multitude of other faces with many or different alterations like long hair etc to expand possibly match.
- Apply some more elements regarding recognition.
- Expand to include other recognition technologies like speechetc.
- Reducing the execution time by simplifying the codestatement.

CODE:

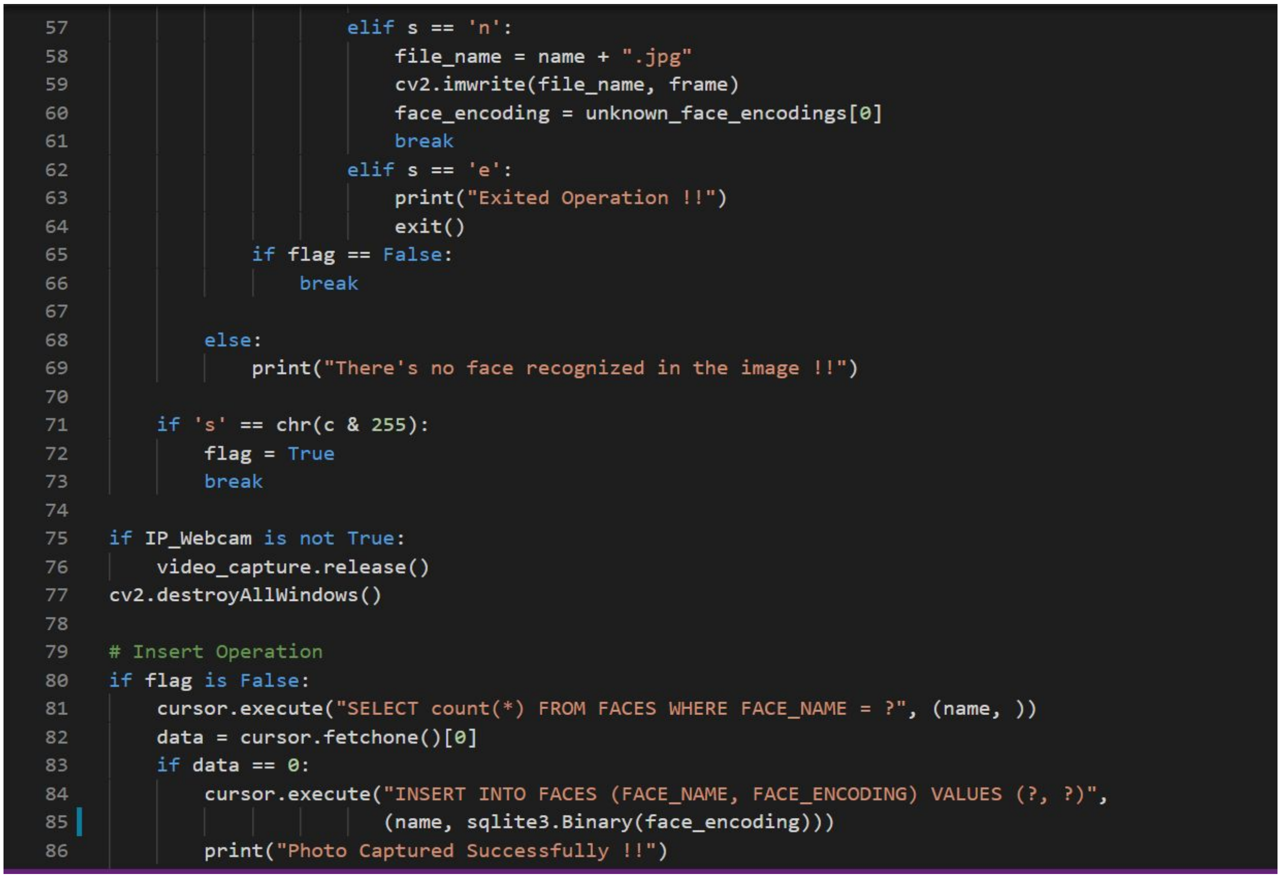
file - capture.py:


capture.py ×
C: > Users > Chandu > Desktop > New folder > python > Intrusion-Detection > capture.py

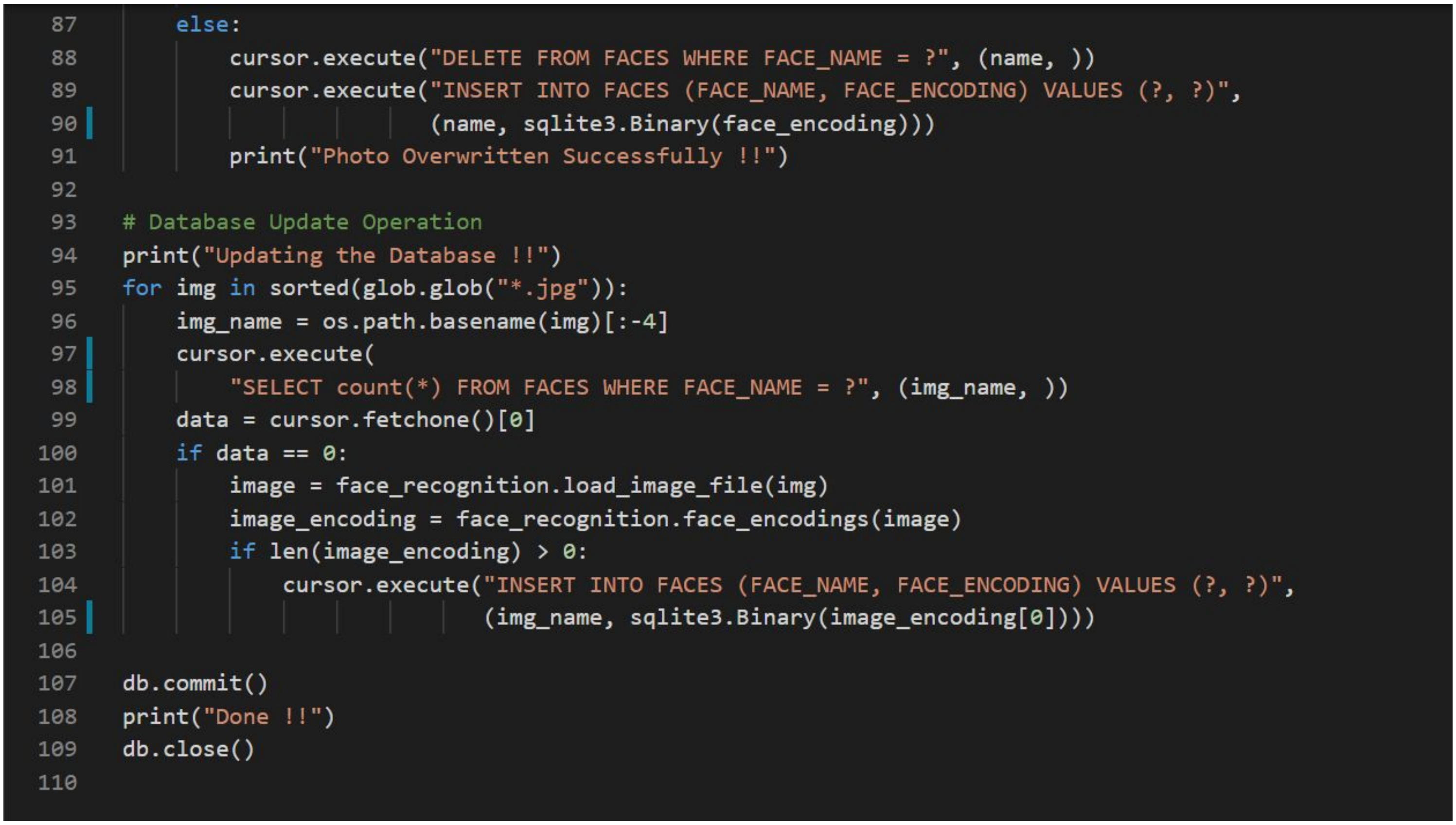
```python
import face_recognition
import sqlite3
import glob
import cv2
import os

IP_Webcam = False
flag = False

if IP_Webcam is True:
    video_capture = cv2.VideoCapture(
        'http://192.168.43.126/videofeed')  # IP Webcam
else:
    video_capture = cv2.VideoCapture(0)

db = sqlite3.connect('db.sqlite3')
print("Opened Database Successfully !!")

cursor = db.cursor()

# Create database
cursor.execute('''CREATE TABLE IF NOT EXISTS FACES
            (ID  INTEGER  PRIMARY KEY  AUTOINCREMENT,
            FACE_NAME    TEXT  NOT NULL,
            FACE_ENCODING   blob  NOT NULL );''')

```


capture.py ×
C: > Users > Chandu > Desktop > New folder > python > Intrusion-Detection > capture.py

```python
while(True):
    ret, frame = video_capture.read()

    cv2.imshow('Video', frame)
    flag = False

    c = cv2.waitKey(1)
    if 'q' == chr(c & 255):
        print("Exited Operation !!")
        exit()

    if 'c' == chr(c & 255):
        unknown_face_encodings = face_recognition.face_encodings(frame)
        if len(unknown_face_encodings) > 0:
            while(flag == False):
                print("Please enter your Name : ")
                name = str(input())
                cursor.execute(
                    "SELECT count(*) FROM FACES WHERE FACE_NAME = ?", (name, ))
                data = cursor.fetchone()[0]
                if data == 0:
                    file_name = name + ".jpg"
                    cv2.imwrite(file_name, frame)
                    face_encoding = unknown_face_encodings[0]
                    break
                else:
                    print("Name Already Exists, Want to enter another Name ? (Y/N)")
                    s = str(input()).lower()
                    if s == 'y':
                        continue
```

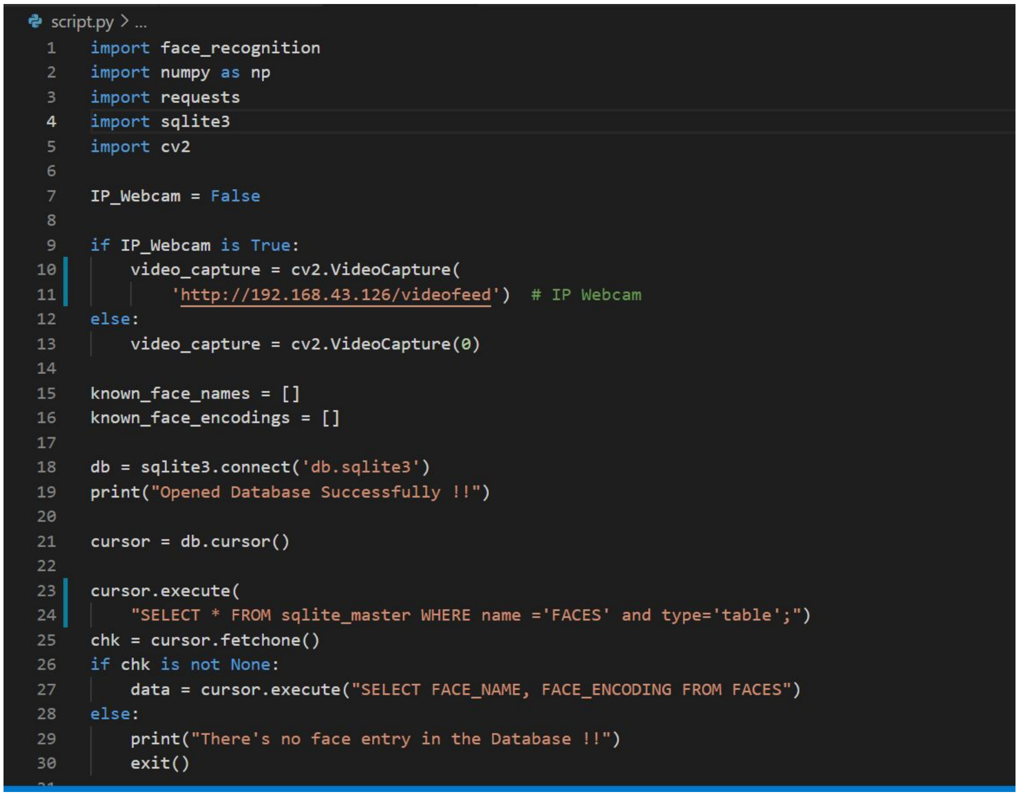


```python
                    elif s == 'n':
                        file_name = name + ".jpg"
                        cv2.imwrite(file_name, frame)
                        face_encoding = unknown_face_encodings[0]
                        break
                    elif s == 'e':
                        print("Exited Operation !!")
                        exit()
            if flag == False:
                break

        else:
            print("There's no face recognized in the image !!")

    if 's' == chr(c & 255):
        flag = True
        break

if IP_Webcam is not True:
    video_capture.release()
cv2.destroyAllWindows()

# Insert Operation
if flag is False:
    cursor.execute("SELECT count(*) FROM FACES WHERE FACE_NAME = ?", (name, ))
    data = cursor.fetchone()[0]
    if data == 0:
        cursor.execute("INSERT INTO FACES (FACE_NAME, FACE_ENCODING) VALUES (?, ?)",
                       (name, sqlite3.Binary(face_encoding)))
        print("Photo Captured Successfully !!")
```

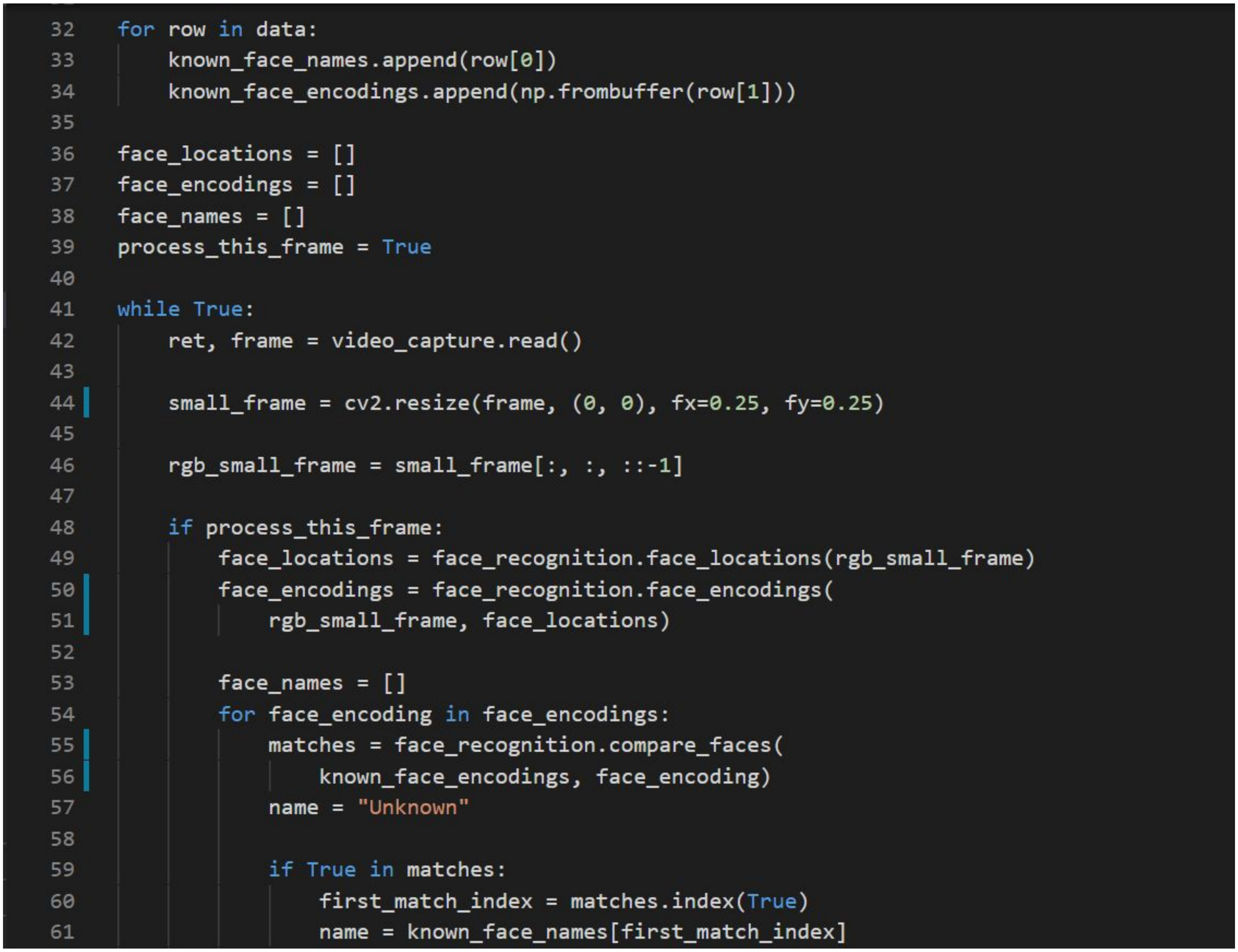


```
    else:
        cursor.execute("DELETE FROM FACES WHERE FACE_NAME = ?", (name, ))
        cursor.execute("INSERT INTO FACES (FACE_NAME, FACE_ENCODING) VALUES (?, ?)",
                       (name, sqlite3.Binary(face_encoding)))
        print("Photo Overwritten Successfully !!")

# Database Update Operation
print("Updating the Database !!")
for img in sorted(glob.glob("*.jpg")):
    img_name = os.path.basename(img)[:-4]
    cursor.execute(
        "SELECT count(*) FROM FACES WHERE FACE_NAME = ?", (img_name, ))
    data = cursor.fetchone()[0]
    if data == 0:
        image = face_recognition.load_image_file(img)
        image_encoding = face_recognition.face_encodings(image)
        if len(image_encoding) > 0:
            cursor.execute("INSERT INTO FACES (FACE_NAME, FACE_ENCODING) VALUES (?, ?)",
                           (img_name, sqlite3.Binary(image_encoding[0])))

db.commit()
print("Done !!")
db.close()
```

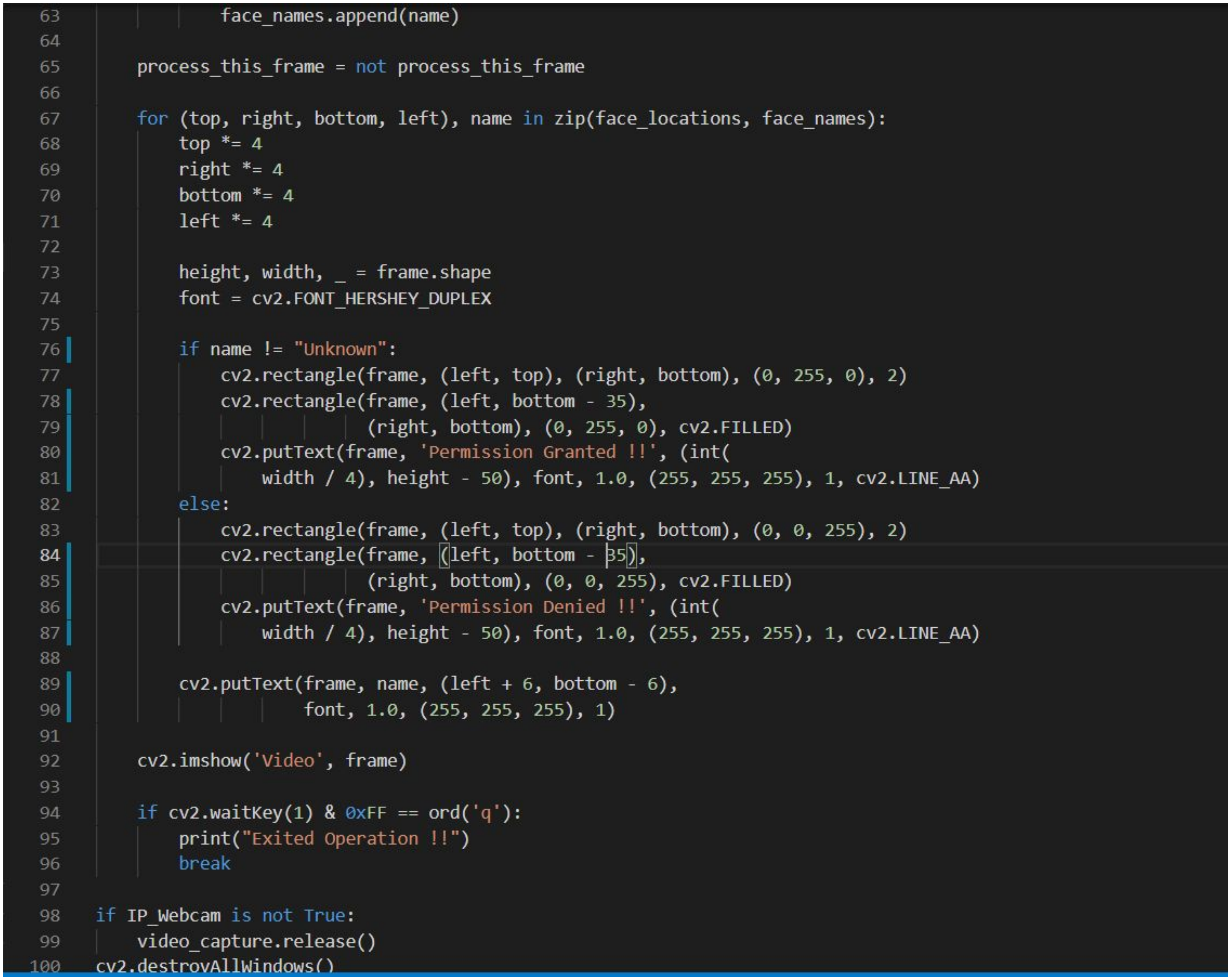
file - script.py



```
import face_recognition
import numpy as np
import requests
import sqlite3
import cv2

IP_Webcam = False

if IP_Webcam is True:
    video_capture = cv2.VideoCapture(
        'http://192.168.43.126/videofeed')  # IP Webcam
else:
    video_capture = cv2.VideoCapture(0)

known_face_names = []
known_face_encodings = []

db = sqlite3.connect('db.sqlite3')
print("Opened Database Successfully !!")

cursor = db.cursor()

cursor.execute(
    "SELECT * FROM sqlite_master WHERE name ='FACES' and type='table';")
chk = cursor.fetchone()
if chk is not None:
    data = cursor.execute("SELECT FACE_NAME, FACE_ENCODING FROM FACES")
else:
    print("There's no face entry in the Database !!")
    exit()
```

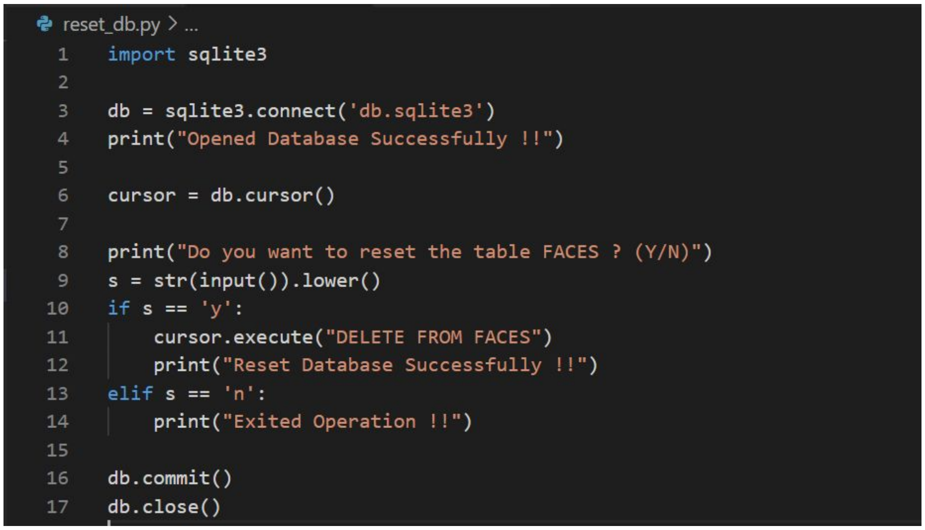


```python
for row in data:
    known_face_names.append(row[0])
    known_face_encodings.append(np.frombuffer(row[1]))

face_locations = []
face_encodings = []
face_names = []
process_this_frame = True

while True:
    ret, frame = video_capture.read()

    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

    rgb_small_frame = small_frame[:, :, ::-1]

    if process_this_frame:
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(
            rgb_small_frame, face_locations)

        face_names = []
        for face_encoding in face_encodings:
            matches = face_recognition.compare_faces(
                known_face_encodings, face_encoding)
            name = "Unknown"

            if True in matches:
                first_match_index = matches.index(True)
                name = known_face_names[first_match_index]
```



```
            face_names.append(name)

    process_this_frame = not process_this_frame

    for (top, right, bottom, left), name in zip(face_locations, face_names):
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4

        height, width, _ = frame.shape
        font = cv2.FONT_HERSHEY_DUPLEX

        if name != "Unknown":
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
            cv2.rectangle(frame, (left, bottom - 35),
                          (right, bottom), (0, 255, 0), cv2.FILLED)
            cv2.putText(frame, 'Permission Granted !!', (int(
                width / 4), height - 50), font, 1.0, (255, 255, 255), 1, cv2.LINE_AA)
        else:
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
            cv2.rectangle(frame, (left, bottom - 35),
                          (right, bottom), (0, 0, 255), cv2.FILLED)
            cv2.putText(frame, 'Permission Denied !!', (int(
                width / 4), height - 50), font, 1.0, (255, 255, 255), 1, cv2.LINE_AA)

        cv2.putText(frame, name, (left + 6, bottom - 6),
                    font, 1.0, (255, 255, 255), 1)

    cv2.imshow('Video', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        print("Exited Operation !!")
        break

if IP_Webcam is not True:
    video_capture.release()
cv2.destroyAllWindows()
```

file - reset_db.py



```
import sqlite3

db = sqlite3.connect('db.sqlite3')
print("Opened Database Successfully !!")

cursor = db.cursor()

print("Do you want to reset the table FACES ? (Y/N)")
s = str(input()).lower()
if s == 'y':
    cursor.execute("DELETE FROM FACES")
    print("Reset Database Successfully !!")
elif s == 'n':
    print("Exited Operation !!")

db.commit()
db.close()
```

REFERENCE AND PUBLICATIONS:

1) M. Coşkun, A. Uçar, Ö. Yildirim and Y. Demir, "Face recognition based on convolutional neural network," 2017 *International Conference on Modern Electrical and Energy Systems (MEES), Kremenchuk,* 2017, pp. 376-379, doi:10.1109/MEES.2017.8248937.
2) Y. Xiao, C. Xing, T. Zhang and Z. Zhao, "An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks," in *IEEE* Access, vol. 7, pp. 42210-42219, 2019, doi: 10.1109/ACCESS.2019.2904620.
3) R. Vinayakumar, K. P. Soman and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," 2017 *International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi*, 2017, pp. 1222-1228, doi: 10.1109/ICACCI.2017.8126009.
4) S. Lawrence, C. L. Giles, Ah Chung Tsoi and A. D. Back, "Face recognition: a convolutional neural-network approach," in *IEEE Transactions on Neural Networks,* vol. 8, no. 1, pp. 98-113, Jan. 1997, doi: 10.1109/72.554195.
5) S. Potluri and C. Diedrich, "Accelerated deep neural networks for enhanced Intrusion Detection System," *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA),* Berlin, 2016, pp. 1-8, doi:10.1109/ETFA.2016.7733515.
6) Face recognition library used for implementation by -- https://github.com/ageitgey/face_recognition