AUTOMATED DRIVER SAFETY SYSTEM

A REPORT

Submitted by

R. VIDHYA LAHARI 16MIS1106

In partial fulfilment for the award

Of

M. Tech. Software Engineering
(5 year Integrated Programme)

School of Computer Science and Engineering

VIT

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

JUNE 2020

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

DECLARATION

I hereby declare that the project entitled **“AUTOMATED DRIVER SAFETY SYSTEM – DROWSINESS DETECTION OF THE CAR DRIVER”** submitted by me to the School of Computing Science and Engineering, Vellore Institute of Technology, Chennai Campus, Chennai 600127 in partial fulfilment of the requirements for the award of the degree of **Master of Technology -Software Engineering (Integrated)** is a record of bonafide work carried out by me**.** I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Signature

R. VIDHYA LAHARI
(16MIS1106)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

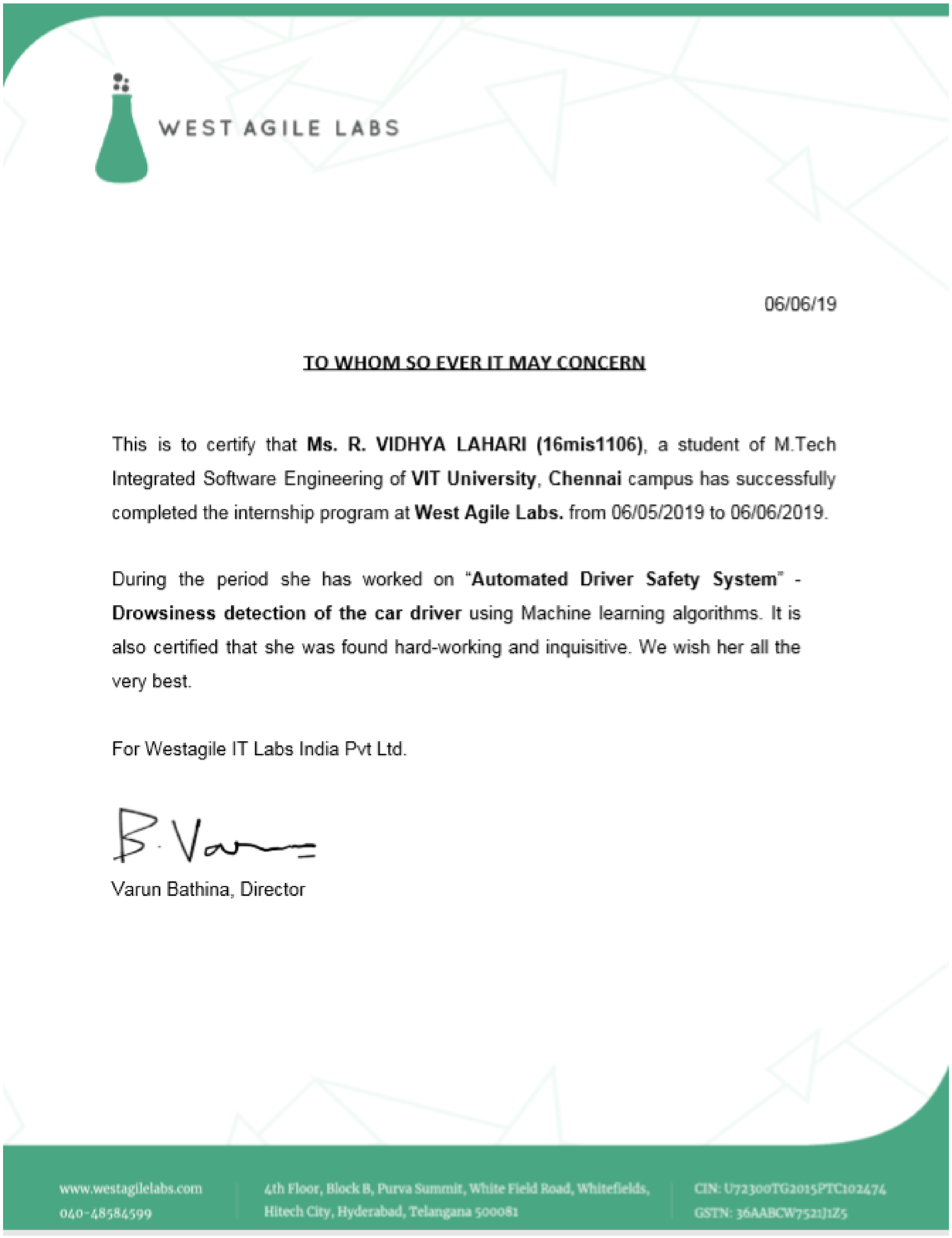
School of Computer Science and Engineering

CERTIFICATE

The project report entitled “” is prepared and submitted by “**R.VIDHYA LAHARI(16MIS1106)”**. It has been found satisfactory in terms of scope, quality and presentation as partial fulfilment of the requirements for the award of the degree of **Master of Technology – Software Engineering (Integrated)** in Vellore Institute of Technology, Chennai, India.

Examined by:

Examiner I **Examiner II**



WEST AGILE LABS

06/06/19

TO WHOM SO EVER IT MAY CONCERN

This is to certify that **Ms. R. VIDHYA LAHARI (16mis1106)**, a student of M.Tech Integrated Software Engineering of **VIT University**, **Chennai** campus has successfully completed the internship program at **West Agile Labs.** from 06/05/2019 to 06/06/2019.

During the period she has worked on "**Automated Driver Safety System**" - **Drowsiness detection of the car driver** using Machine learning algorithms. It is also certified that she was found hard-working and inquisitive. We wish her all the very best.

For Westagile IT Labs India Pvt Ltd.

[illegible]

Varun Bathina, Director

www.westagilelabs.com
040-48584599

4th Floor, Block B, Purva Summit, White Field Road, Whitefields,
Hitech City, Hyderabad, Telangana 500081

CIN: U72300TG2015PTC102474
GSTN: 36AABCW7521J1Z5

ACKNOWLEDGEMENT

Throughout the writing of this dissertation I have received a great deal of support and assistance. I would first like to thank all the people who guided me through the work and whose expertise was invaluable in the Completion of the project.

I would like to acknowledge my colleagues from my internship at **WEST AGILE IT LABS India private limited** for their wonderful collaboration. You supported me greatly and were always willing to help me. I would particularly like to thank all the members for your excellent cooperation and for all of the opportunities I was given to complete the project.

I would also like to thank my tutors, faculties of the university**, (**Head of the Department for 5 years MTech Integrated Software Engineering Vit University Chennai Campus)
Dr. Asnath Victy Phamila Y , Dr. Jagadeesh Kannan R (Dean of SCOPE of Vit University Chennai Campus**), Dr. Geetha S (**Associate Dean of SCOPE of Vit University Chennai Campus**)** for their valuable guidance .You provided me with the tools that I needed to choose the right direction and successfully complete my dissertation.

In addition, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. Finally, there are my friends, who were of great support in deliberating over our problems and findings, as well as providing happy distraction to rest my mind outside of my research.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Expansion
OpenCV	Open Computer Vision
BSD	Berkeley Software Distribution
MATLAB	Matrix Laboratory
MMX	Multi Media Extensions
CUDA	Compute Unified Device Architecture
OpenCL	Open Computing Language
SVM	Support Vector Machine
LBP	Local Binary Pattern
NIR	Near Infra Red

ABSTRACT

The project on which I worked was an approach towards automobile safety and security. When a driver takes his/her seat, the system automatically detects the face of the person and displays the details. This system can be used to prevent the theft. Current developments aim to provide driver assistance in the form of conditional and partial automation. The presence of Computer vision technologies inside the vehicles is expected to grow as the automation levels increase. However, embedding a vision-based driver assistance system supposes a big challenge due to the special features of vision algorithms, the existing constrains and the strict requirements that need to be fulfilled. The aim of this project is to leverage Vision based Artificial Intelligence for assisting the driver to the next level.

Present scenario

- Depends on non-smart sensor data which might fail to produce better results.
- Accuracy when compared to visual data will be less than 50%.
- High quality sensor to be used for all the modules.

Limitations/pitfalls of the present scenario

Depends on non-smart sensor data, which might fail to produce better results, in the proposed system the involvement of Artificial Intelligence, will fetch robust yet best results.
Accuracy when compared to visual data will be less to 50%, where in the proposed system visual Data helps in predicting at a higher accuracy of 9-% with less slip through rate less than 10%.

Cost is high, high quality sensor have to be installed for each and every functionality, wherein the proposed method uses only a camera to do all the functionality.

Problem addressed

Car accident is the major cause of death in which around 1.3 million people die every year. Majority of these accidents are caused because of distraction or the drowsiness of driver. Construction of high-speed highway roads had diminished the margin of error for the driver. The countless number of people drives for long distance every day and night on the highway. Lack of sleep or distractions like the phone call, talking with the passenger, etc may lead to an accident. To prevent such accidents, we propose a system which alerts the driver if the driver gets distracted or feels drowsy. Facial landmarks detection is used with help of image processing of images of the face captured using the camera, for detection of distraction or drowsiness. This whole system is deployed on portable hardware which can be easily installed in the car for use.

Provided Solution:

- In proposed system the involvement of artificial intelligence, will fetch robust yet best results.
- The data helps in predicting at a higher accuracy of 90% with less slip through rate less than 10%.
- In this method usage of single camera yields all the functionalities of modules.
- The proposed system uses open source technologies as the cost decreases as no buying of software is involved.
- The overall accuracy of system is higher than 90% for all the modules.

1.INTRODUCTION

OpenCV

In this project, I have used libraries of OpenCV to detect and recognize the faces.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many start-ups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

To overcome the problem of accidents, we came up with the solution implemented in the form of image processing. To perform image processing, OpenCV open source libraries are used. Python is used as a language to implement the idea.

A web camera is used to continuously track the facial landmark and movement of eyes of the driver. This project mainly targets the landmarks of eyes of the driver. For detection of drowsiness, landmarks of eyes are tracked continuously. Images are captured using the camera at fix frame rate, maximum of 500 frames. These images are passed to image processing module which performs face landmark detection to detect distraction and drowsiness of driver. If the driver is found to be distracted then a voice (audio) alert and is provided and a message is displayed on the screen.

2. METHODOLOGY

SVM (Support Vector Machine)

Here we are used SVM (support vector machine) to classify the components in the input video. While cropping the region of interest components in the video is not accurate. Sometimes it will show regions wrong. To sense the eyes first we have to create boundary boxes for that and a classification algorithm. The algorithm of SVM will not support.

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).

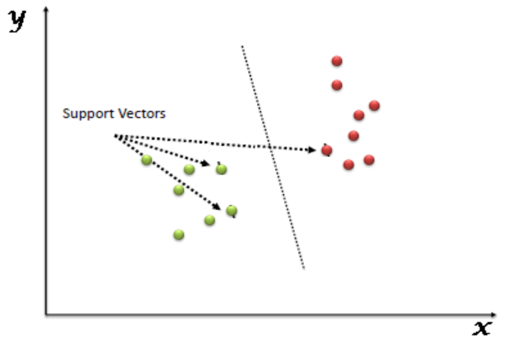


Fig1: support vectors

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is "How can we identify the right hyper-plane?". Don't worry, it's not as hard as you think!

Let’s understand:

- **Identify the right hyper-plane (Scenario-1):**

Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

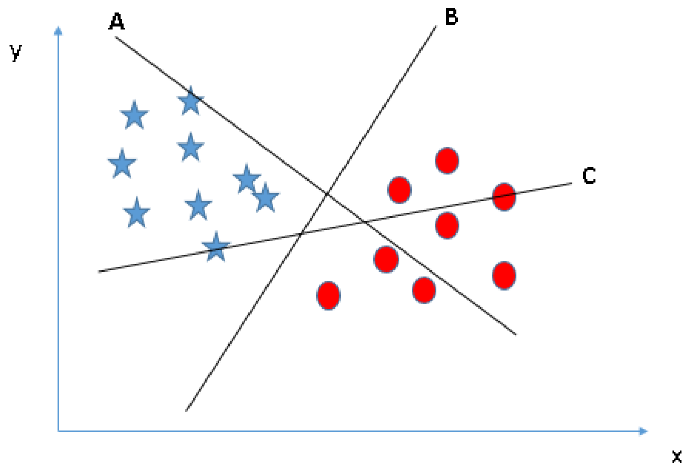


Fig2: identifying right hyperplane

You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

- **Identify the right hyper-plane (Scenario-2):**

  Here, we have three hyper-planes (A, B and C) and all are segregating the classes well.

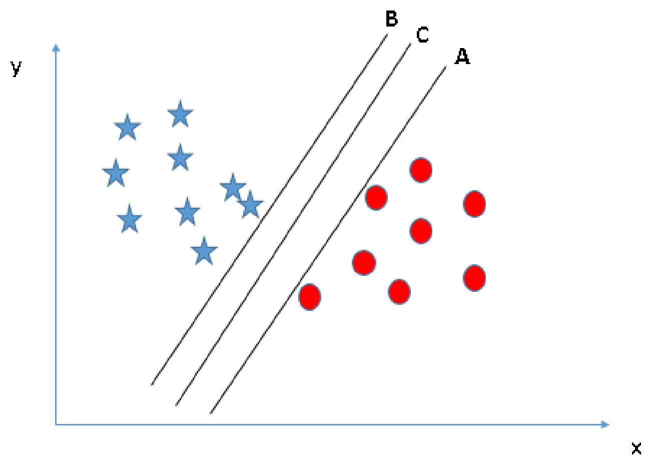


Fig3: Identifying right hyperplane(2)

Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.

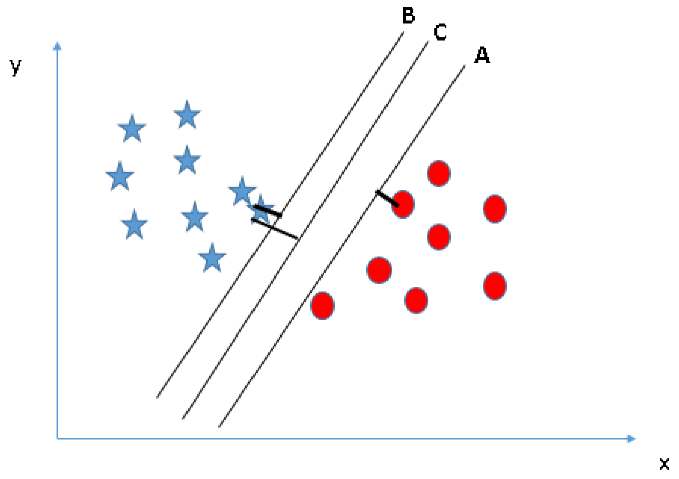


Fig4: margin

Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

- **Identify the right hyper-plane (Scenario-3):**

Hint: Use the rules as discussed in previous section to identify the right hyper-plane

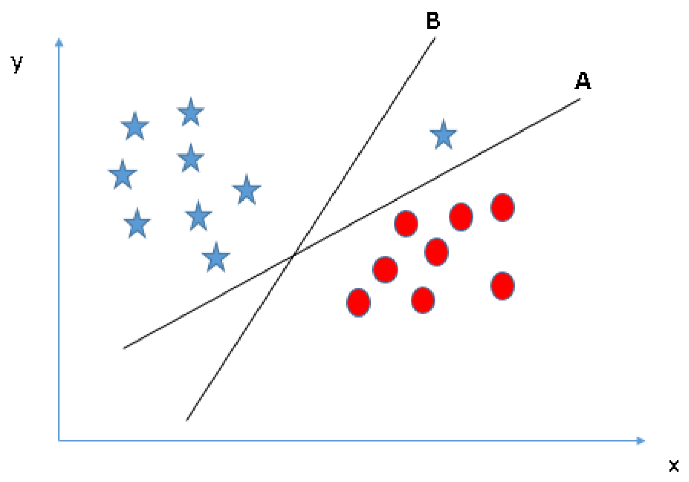


Fig5: Hyperplane substitutes

Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.** But here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

- **Can we classify two classes (Scenario-4)?**

I am unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.

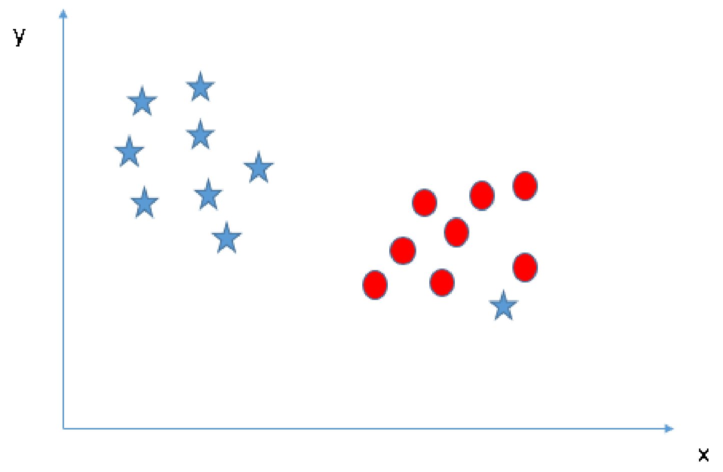


Fig6: segregation of two classes

As I have already mentioned, one star at another end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.

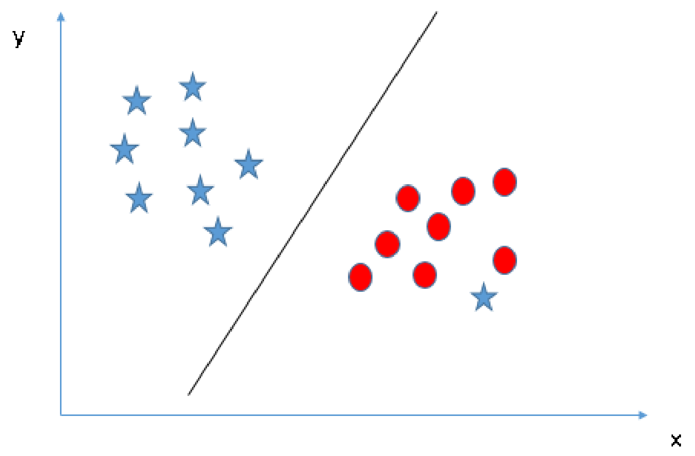


Fig7: ignoring outliers

- **Find the hyper-plane to segregate to classes (Scenario-5):**

  we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.

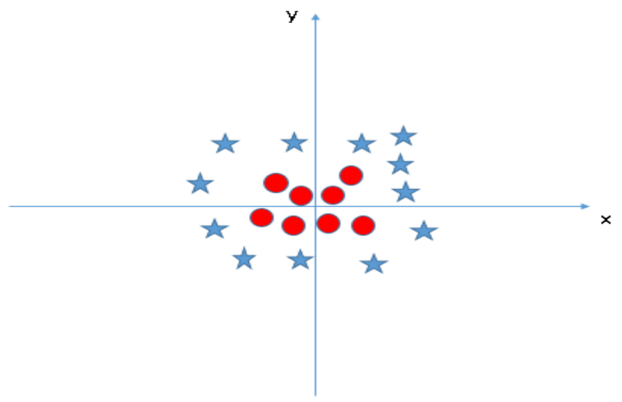


Fig8: segregates classes

SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:

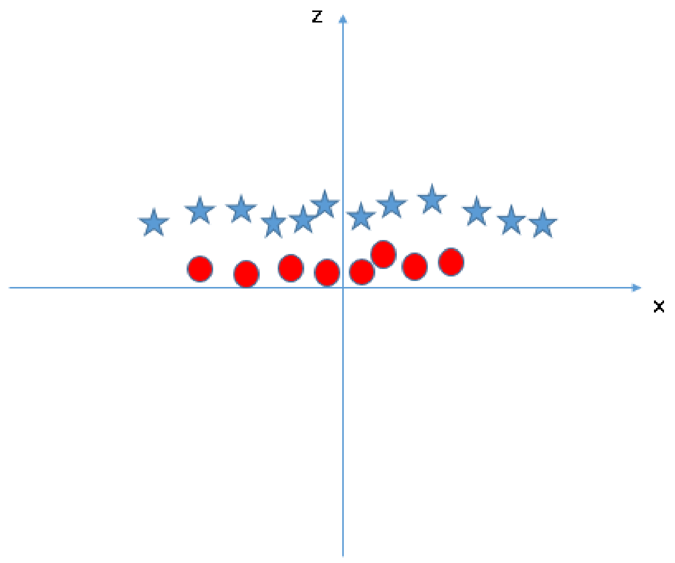


Fig9: hyperplane that segregates classes

In above plot, points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the **kernel trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:

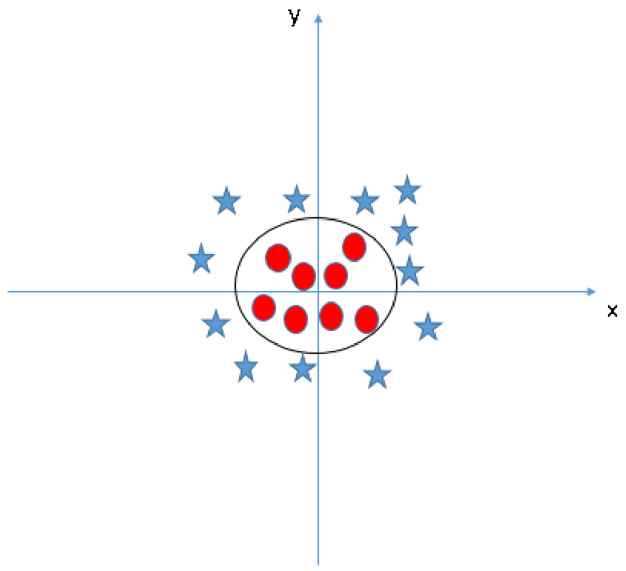


Fig10: possible division of classes

Haar cascade

Haar cascade face detection classifier is the algorithm used to detect the faces.

How haar cascade works

The face detection algorithm proposed by Viola and Jones is used as the basis of our design. The face detection algorithm looks for specific Haar features of a human face. When one of these features is found, the algorithm allows the face candidate to pass to the next stage of detection. A face candidate is a rectangular section of the original image called a sub-window. Generally, this sub-window has a fixed size (typically 24×24 pixels). This sub window is often scaled in order to obtain a variety of different size faces. The algorithm scans the entire image with this window and denotes each respective section a face candidate.

- The algorithm uses an integral image in order to process Haar features of a face candidate in constant time. It uses a cascade of stages which is used to eliminate non-face candidates quickly. Each stage consists of many different Haar features. Each feature is classified by a Haar feature classifier. The Haar feature classifiers generate an output which can then be provided to the stage comparator. The stage comparator sums the outputs of the Haar feature classifiers and compares this value with a stage threshold to determine if the stage should be passed. If all stages are passed the face candidate is concluded to be a face.
- Object Detection using Haar feature-based cascade classifiers is an effective object detection method. 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 below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

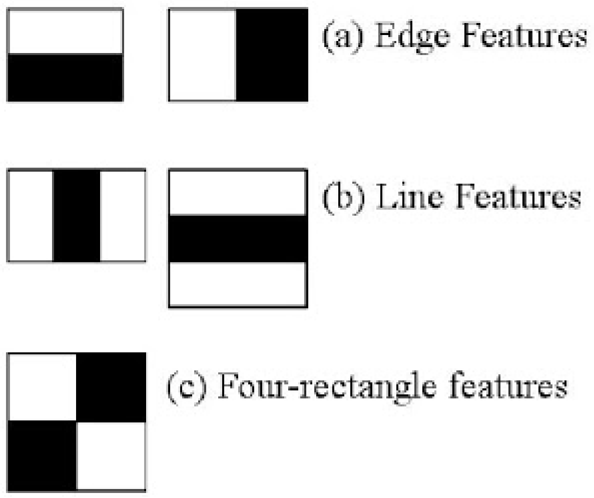


Fig11: Haar features

Haar Features

Now all possible sizes and locations of each kernel is used to calculate plenty of features. (Even a 24x24 window results over 160000 features). For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of sum of pixels, how large may be the number of pixels, to an operation involving just four pixels.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. 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 applying on cheeks or any other place is irrelevant.

Face Detection

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. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies 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 again same process is done. New error rates are calculated. Also new weights. The process is continued until required accuracy or error rate is achieved or required number of features are found).

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. In an image, most of the image region 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. Don't process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region.

For this they introduced the concept of Cascade of Classifiers. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very less number of features). If a window fails the first stage, discard it. We don't consider 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.

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

LOCAL BINARY PATTERN [LBP]:

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications. It can be seen as a unifying approach to the traditionally divergent statistical and structural models of texture analysis. Perhaps the most important property of the LBP operator in real-world applications is its robustness to monotonic gray-scale changes caused, for example, by illumination variations. Another important property is its computational simplicity, which makes it possible to analyze images in challenging real-time settings.

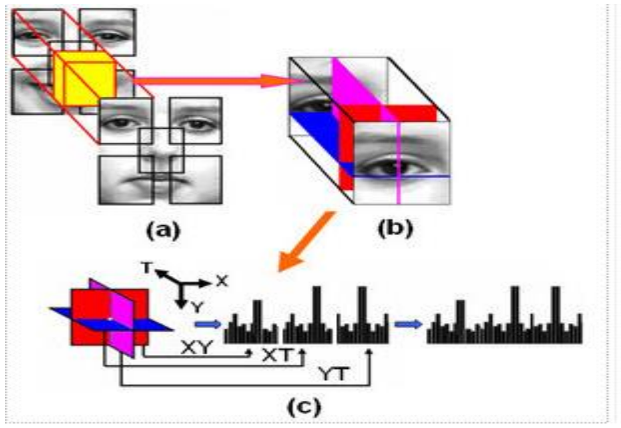


Fig11: facial expression with Local binary pattern

The LBP feature vector, in its simplest form, is created in the following manner:
Divide the examined window into cells (e.g. 16x16 pixels for each cell).
For each pixel in a cell, compare the pixel to each of its 8 neighbours (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counter-clockwise.
Where the centre pixel's value is greater than the neighbour’s value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).
Compute the histogram, over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are greater than the centre). This histogram can be seen as a 256-dimensional feature vector.

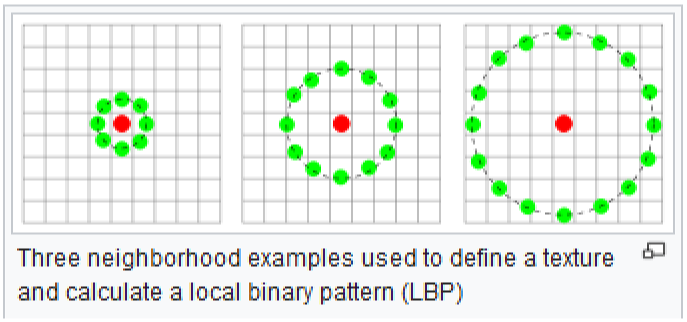

Three neighborhood examples used to define a texture and calculate a local binary pattern (LBP)

Fig12: example of neighbors in LBP

Optionally normalize the histogram.

Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window. The feature vector can now be processed using the Support vector machine, extreme learning machines, or some other machine-learning algorithm to classify images. Such classifiers can be used for face recognition or texture analysis.

In the LBP approach for texture classification, the occurrences of the LBP codes in an image are collected into a histogram. The classification is then performed by computing simple histogram similarities. However, considering a similar approach for facial image representation results in a loss of spatial information and therefore one should codify the texture information while retaining also their locations. One way to achieve this goal is to use the LBP texture descriptors to build several local descriptions of the face and combine them into a global description. Such local descriptions have been gaining interest lately which is understandable given the limitations of the holistic representations. These local feature-based methods are more robust against variations in pose or illumination than holistic methods.

The basic methodology for LBP based face description proposed by Ahonen et al. (2006) is as follows: The facial image is divided into local regions and LBP texture descriptors are extracted from each region independently. The descriptors are then concatenated to form a global description of the face.

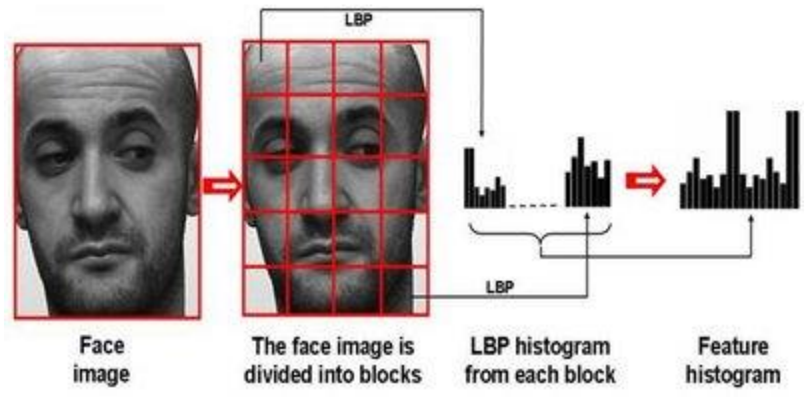

LBP
LBP
Face image
The face image is divided into blocks
LBP histogram from each block
Feature histogram

Fig13: Face description with local binary patterns.

This histogram effectively has a description of the face on three different levels of locality: the LBP labels for the histogram contain information about the patterns on a pixel-level, the labels are summed over a small region to produce information on a regional level and the regional histograms are concatenated to build a global description of the face.

It should be noted that when using the histogram-based methods the regions do not need to be rectangular. Neither do they need to be of the same size or shape, nor do they not necessarily have to cover the whole image. It is also possible to have partially overlapping regions.

The two-dimensional face description method has been extended into spatiotemporal domain (Zhao and Pietikäinen 2007). Figure depicts facial expression description using LBP-TOP. Excellent facial expression recognition performance has been obtained with this approach.

Since the publication of the LBP based face description, the methodology has already attained an established position in face analysis research and applications. A notable example is illumination-invariant face recognition system proposed by Li et al. (2007), combining NIR imaging with LBP features and Adaboost learning. Zhang et al. (2005) proposed the extraction of LBP features from images obtained by filtering a facial image with 40 Gabor filters of different scales and orientations, obtaining outstanding results. Hadid and Pietikäinen (2009) used spatiotemporal LBPs for face and gender recognition from video sequences, while Zhao et al. (2009) adopted the LBP-TOP approach to visual speech recognition achieving leading-edge performance without error-prone segmentation of moving lips.

A useful extension to the original operator is the so-called uniform pattern , which can be used to reduce the length of the feature vector and implement a simple rotation invariant descriptor. This idea is motivated by the fact that some binary patterns occur more commonly in texture images than others. A local binary pattern is called uniform if the binary pattern contains at most two 0-1 or 1-0 transitions. For example, 00010000(2 transitions) is a uniform pattern, 01010100(6 transitions) is not. In the computation of the LBP histogram, the histogram has a separate bin for every uniform pattern, and all non-uniform patterns are assigned to a single bin. Using uniform patterns, the length of the feature vector for a single cell reduces from 256 to 59. The 58 uniform binary patterns correspond to the integers 0, 1, 2, 3, 4, 6, 7, 8, 12, 14, 15, 16, 24, 28, 30, 31, 32, 48, 56, 60, 62, 63, 64, 96, 112, 120, 124, 126, 127, 128, 129, 131, 135, 143, 159, 191, 192, 193, 195, 199, 207, 223, 224, 225, 227, 231, 239, 240, 241, 243, 247, 248, 249, 251, 252, 253, 254 and 255.

The facial landmarks:

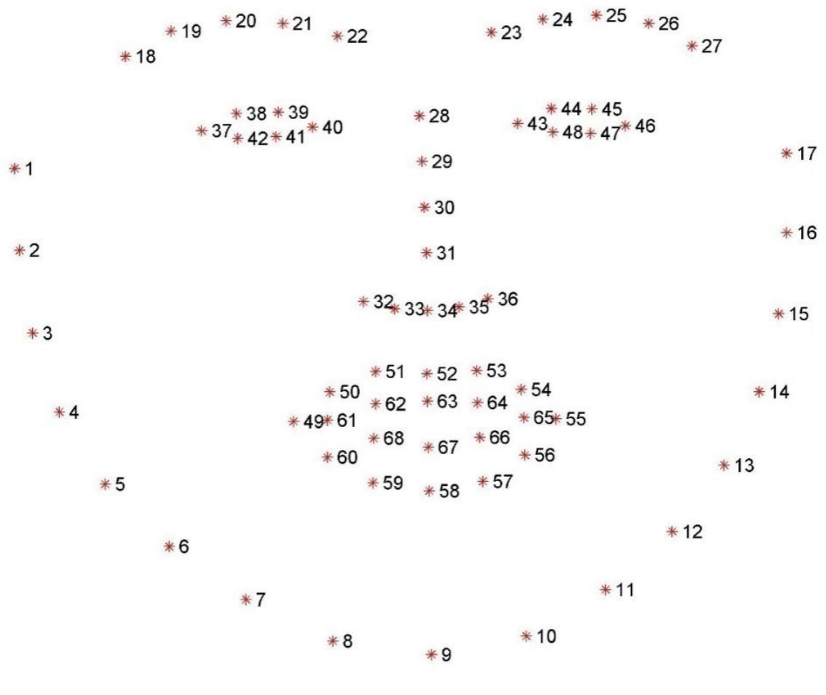


Fig14: Facial landmarks

4. CONCLUSION:

It was a complete useful experience working at WEST AGILE LABS. This experience brought out my strength and also the areas I needed to make up. It added more confidence to my professional approach. It built a stronger positive attitude and taught me how to work in Team as a player.

The primary objective of an internship is to gather a real-life working experience and put their theoretical knowledge in practice.

In review this internship has been an excellent and rewarding experience. I have been able to meet and network with many people that I am sure this will be able to help me with opportunities in the future.

The project was challenging and I thoroughly learnt all the phases of its development. First, the project was explained thoroughly and then the concepts related to the project was also explained. Teammates were very friendly and the work was distributed among us. Our everyday task made us to learn something and try to implement it into the project.

REFERENCES:

https://www.westagilelabs.com/

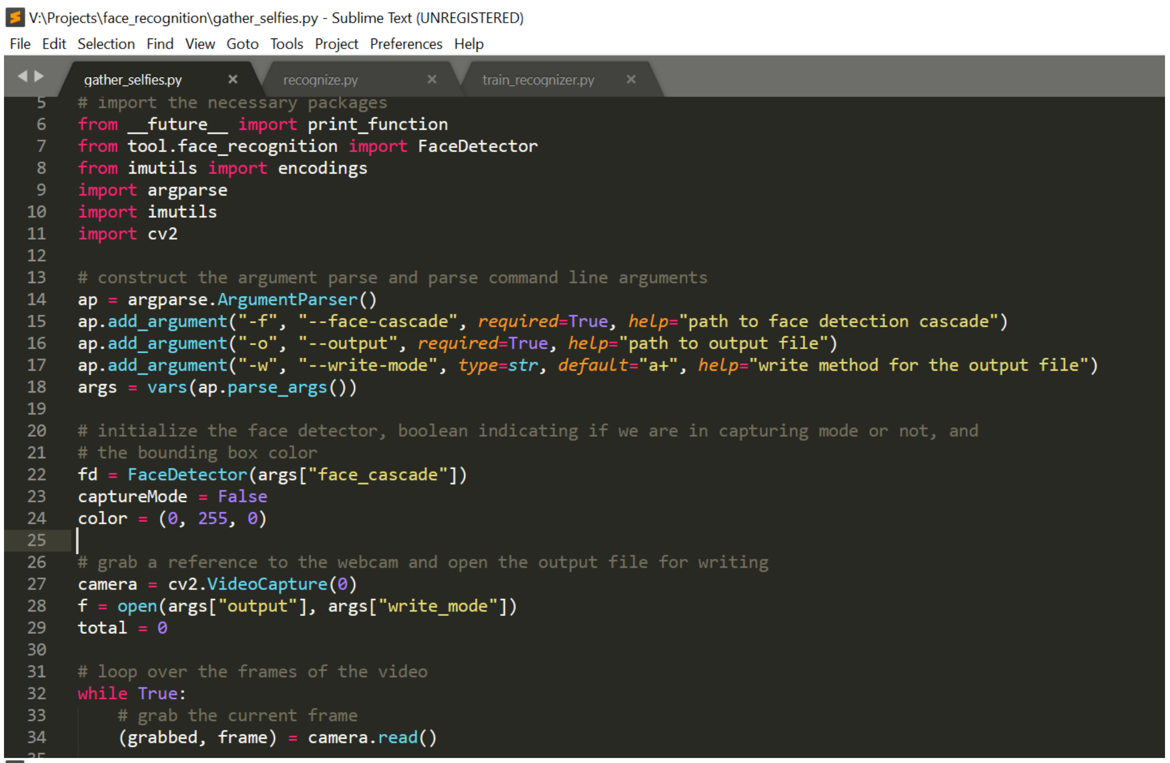
https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/

https://www.geeksforgeeks.org/project-idea-driver-distraction-and-drowsiness-detection-system-dcube/

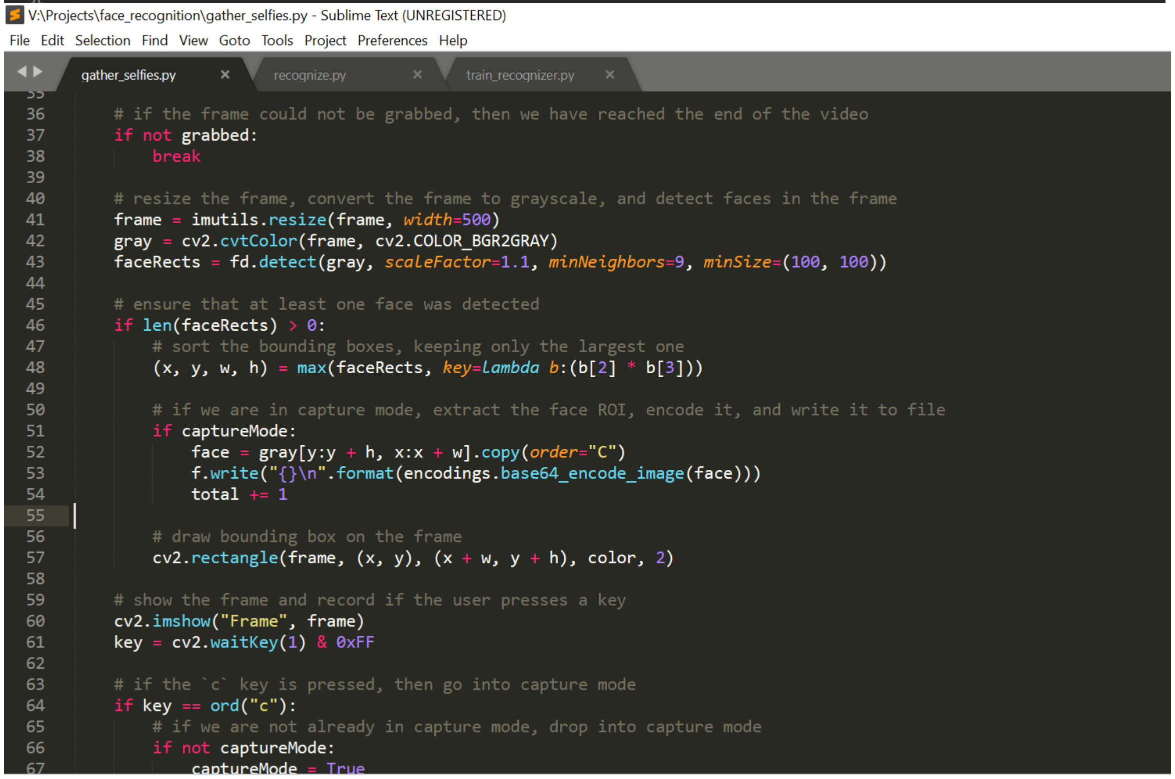
https://www.analyticsvidhya.com/blog/2018/08/a-simple-introduction-to-facial-recognition-with-python-codes/

APPENDIX-1:

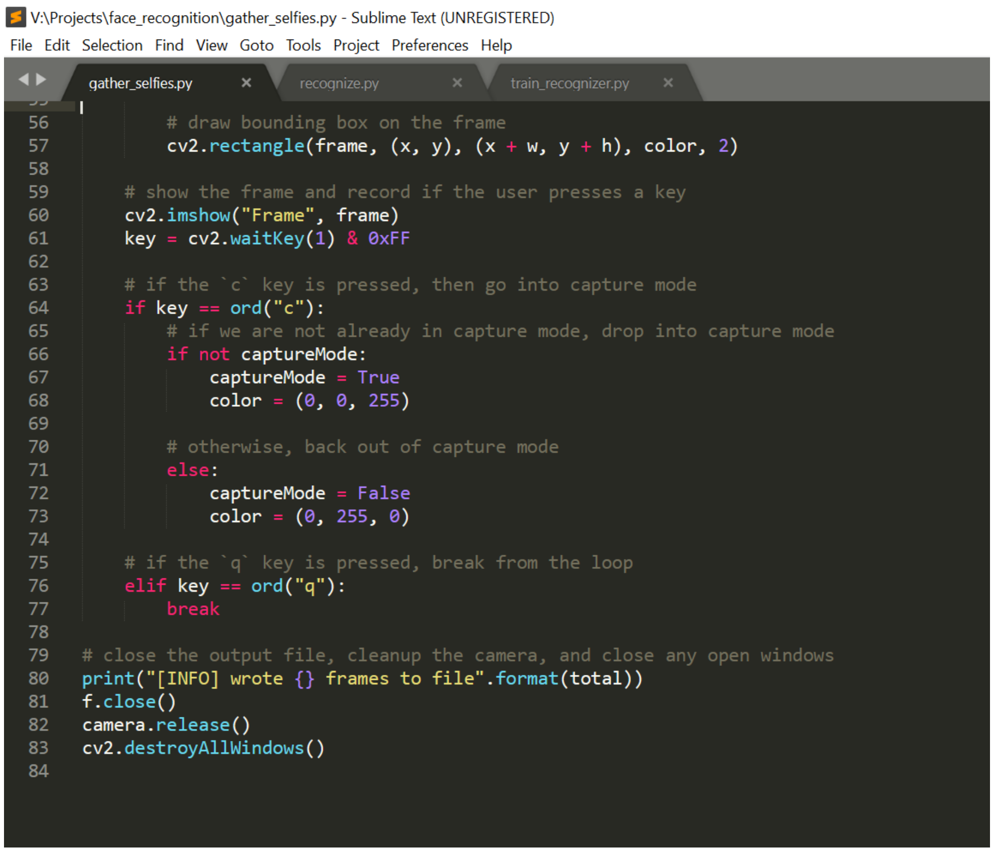
1. GATHERING SELFIES:


V:\Projects\face_recognition\gather_selfies.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
gather_selfies.py
recognize.py
train_recognizer.py

```python
# import the necessary packages
from __future__ import print_function
from tool.face_recognition import FaceDetector
from imutils import encodings
import argparse
import imutils
import cv2

# construct the argument parse and parse command line arguments
ap = argparse.ArgumentParser()
ap.add_argument("-f", "--face-cascade", required=True, help="path to face detection cascade")
ap.add_argument("-o", "--output", required=True, help="path to output file")
ap.add_argument("-w", "--write-mode", type=str, default="a+", help="write method for the output file")
args = vars(ap.parse_args())

# initialize the face detector, boolean indicating if we are in capturing mode or not, and
# the bounding box color
fd = FaceDetector(args["face_cascade"])
captureMode = False
color = (0, 255, 0)

# grab a reference to the webcam and open the output file for writing
camera = cv2.VideoCapture(0)
f = open(args["output"], args["write_mode"])
total = 0

# loop over the frames of the video
while True:
    # grab the current frame
    (grabbed, frame) = camera.read()
```


V:\Projects\face_recognition\gather_selfies.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
gather_selfies.py
recognize.py
train_recognizer.py

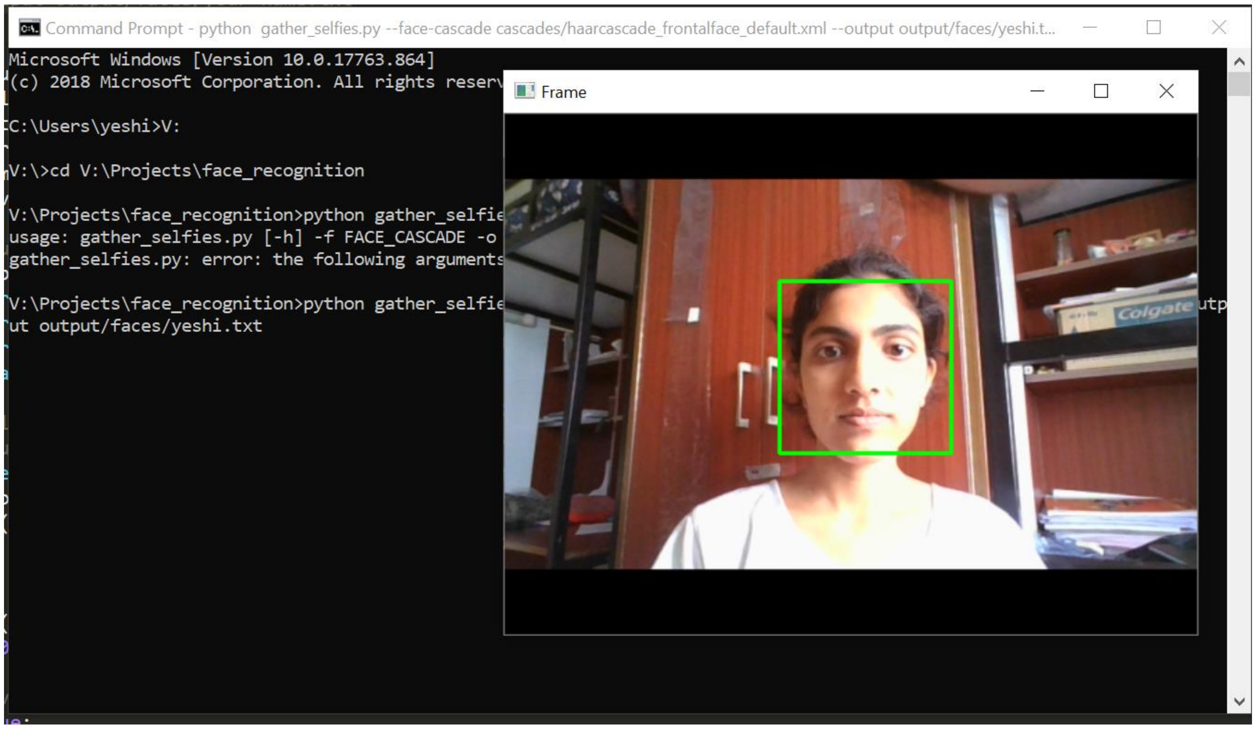
```python
    # if the frame could not be grabbed, then we have reached the end of the video
    if not grabbed:
        break

    # resize the frame, convert the frame to grayscale, and detect faces in the frame
    frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faceRects = fd.detect(gray, scaleFactor=1.1, minNeighbors=9, minSize=(100, 100))

    # ensure that at least one face was detected
    if len(faceRects) > 0:
        # sort the bounding boxes, keeping only the largest one
        (x, y, w, h) = max(faceRects, key=lambda b:(b[2] * b[3]))

        # if we are in capture mode, extract the face ROI, encode it, and write it to file
        if captureMode:
            face = gray[y:y + h, x:x + w].copy(order="C")
            f.write("{}\n".format(encodings.base64_encode_image(face)))
            total += 1

        # draw bounding box on the frame
        cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)

    # show the frame and record if the user presses a key
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `c` key is pressed, then go into capture mode
    if key == ord("c"):
        # if we are not already in capture mode, drop into capture mode
        if not captureMode:
            captureMode = True
```



V:\Projects\face_recognition\gather_selfies.py - Sublime Text (UNREGISTERED)

```python
            # draw bounding box on the frame
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)

        # show the frame and record if the user presses a key
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `c` key is pressed, then go into capture mode
        if key == ord("c"):
            # if we are not already in capture mode, drop into capture mode
            if not captureMode:
                captureMode = True
                color = (0, 0, 255)

            # otherwise, back out of capture mode
            else:
                captureMode = False
                color = (0, 255, 0)

        # if the `q` key is pressed, break from the loop
        elif key == ord("q"):
            break

    # close the output file, cleanup the camera, and close any open windows
    print("[INFO] wrote {} frames to file".format(total))
    f.close()
    camera.release()
    cv2.destroyAllWindows()
```

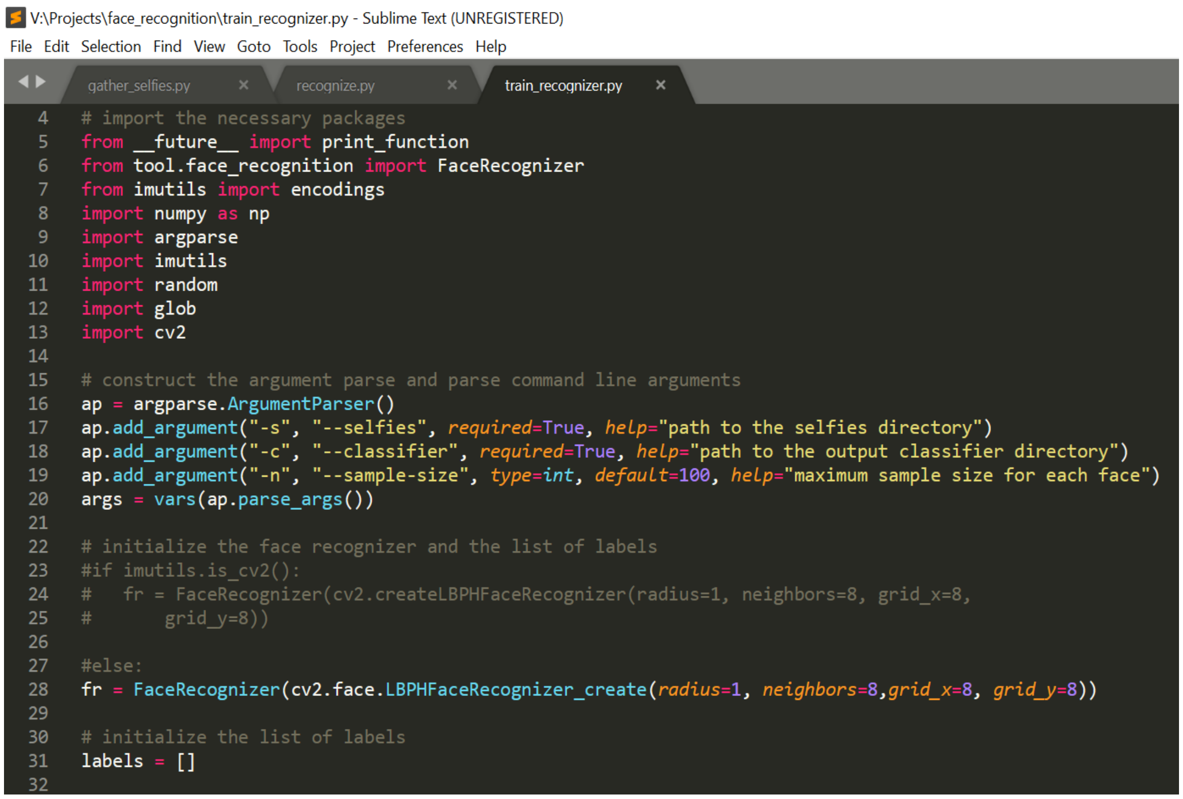
OUTPUT:



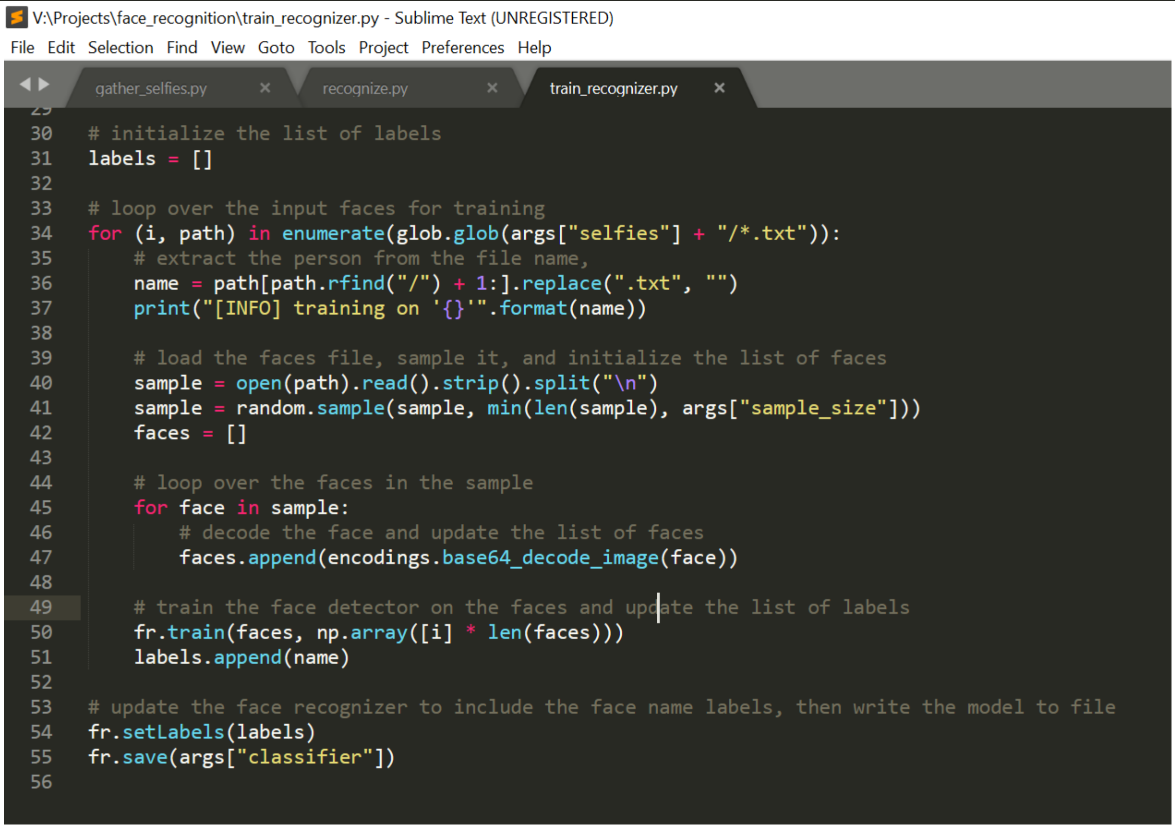
Command Prompt - python gather_selfies.py --face-cascade cascades/haarcascade_frontalface_default.xml --output output/faces/yeshi.t...

```
Microsoft Windows [Version 10.0.17763.864]
(c) 2018 Microsoft Corporation. All rights reserv

C:\Users\yeshi>V:

V:\>cd V:\Projects\face_recognition

V:\Projects\face_recognition>python gather_selfie
usage: gather_selfies.py [-h] -f FACE_CASCADE -o
gather_selfies.py: error: the following arguments

V:\Projects\face_recognition>python gather_selfie                                              utp
ut output/faces/yeshi.txt
```

2. TRAINING THE SAMPLES:

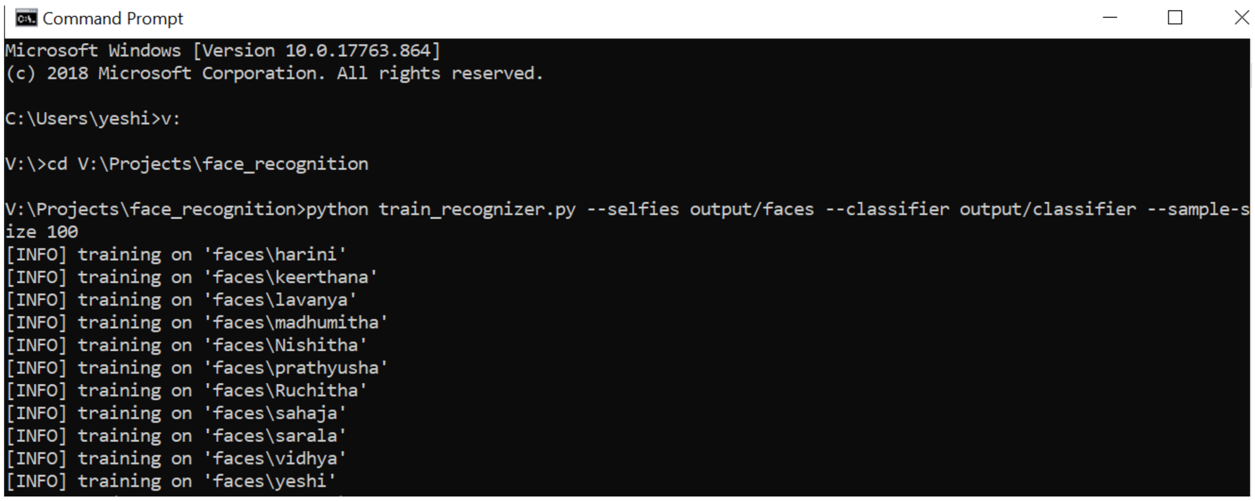

V:\Projects\face_recognition\train_recognizer.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
gather_selfies.py
recognize.py
train_recognizer.py

```python
# import the necessary packages
from __future__ import print_function
from tool.face_recognition import FaceRecognizer
from imutils import encodings
import numpy as np
import argparse
import imutils
import random
import glob
import cv2

# construct the argument parse and parse command line arguments
ap = argparse.ArgumentParser()
ap.add_argument("-s", "--selfies", required=True, help="path to the selfies directory")
ap.add_argument("-c", "--classifier", required=True, help="path to the output classifier directory")
ap.add_argument("-n", "--sample-size", type=int, default=100, help="maximum sample size for each face")
args = vars(ap.parse_args())

# initialize the face recognizer and the list of labels
#if imutils.is_cv2():
#   fr = FaceRecognizer(cv2.createLBPHFaceRecognizer(radius=1, neighbors=8, grid_x=8,
#       grid_y=8))

#else:
fr = FaceRecognizer(cv2.face.LBPHFaceRecognizer_create(radius=1, neighbors=8,grid_x=8, grid_y=8))

# initialize the list of labels
labels = []

```


V:\Projects\face_recognition\train_recognizer.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
gather_selfies.py
recognize.py
train_recognizer.py

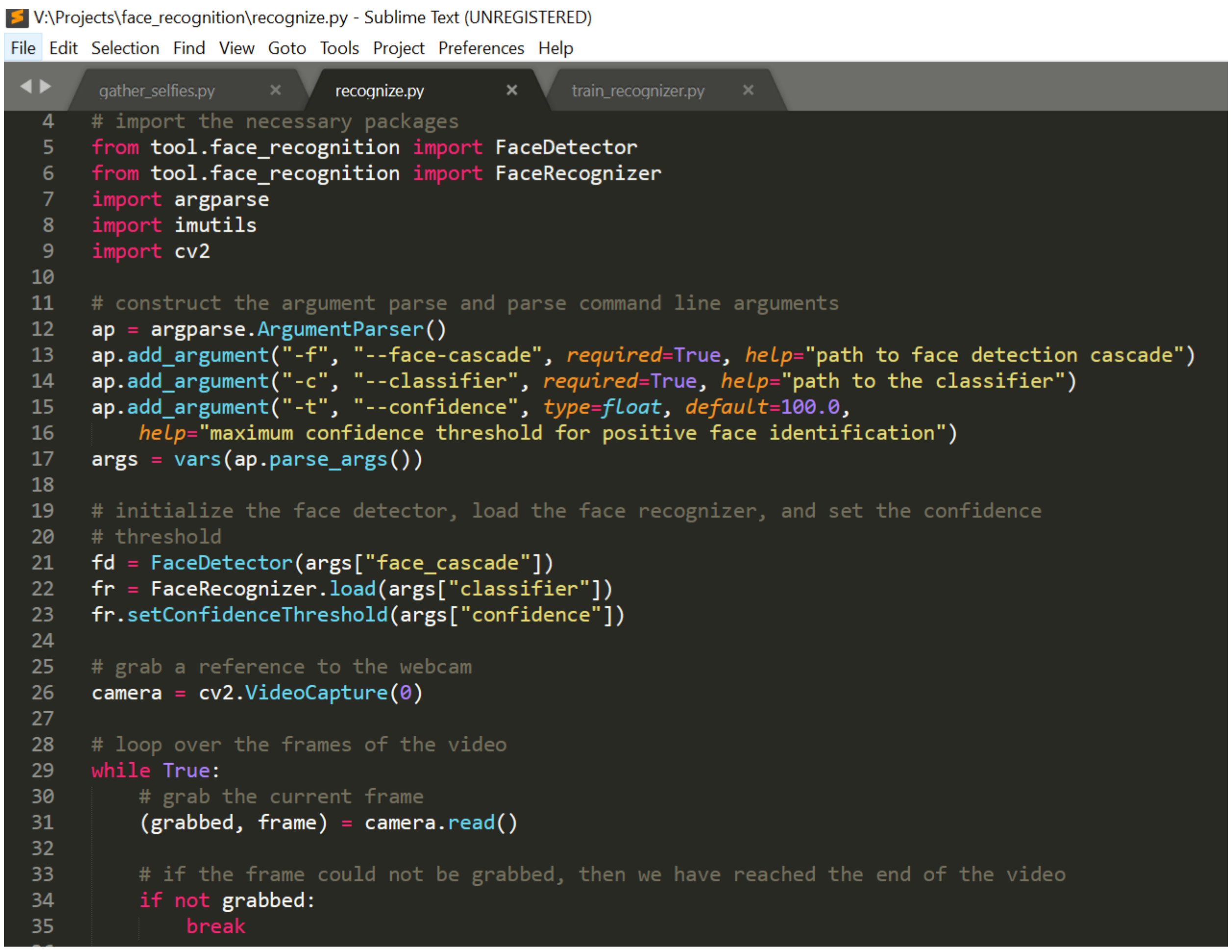
```python
# initialize the list of labels
labels = []

# loop over the input faces for training
for (i, path) in enumerate(glob.glob(args["selfies"] + "/*.txt")):
    # extract the person from the file name,
    name = path[path.rfind("/") + 1:].replace(".txt", "")
    print("[INFO] training on '{}'".format(name))

    # load the faces file, sample it, and initialize the list of faces
    sample = open(path).read().strip().split("\n")
    sample = random.sample(sample, min(len(sample), args["sample_size"]))
    faces = []

    # loop over the faces in the sample
    for face in sample:
        # decode the face and update the list of faces
        faces.append(encodings.base64_decode_image(face))

    # train the face detector on the faces and update the list of labels
    fr.train(faces, np.array([i] * len(faces)))
    labels.append(name)

# update the face recognizer to include the face name labels, then write the model to file
fr.setLabels(labels)
fr.save(args["classifier"])
```

OUTPUT:

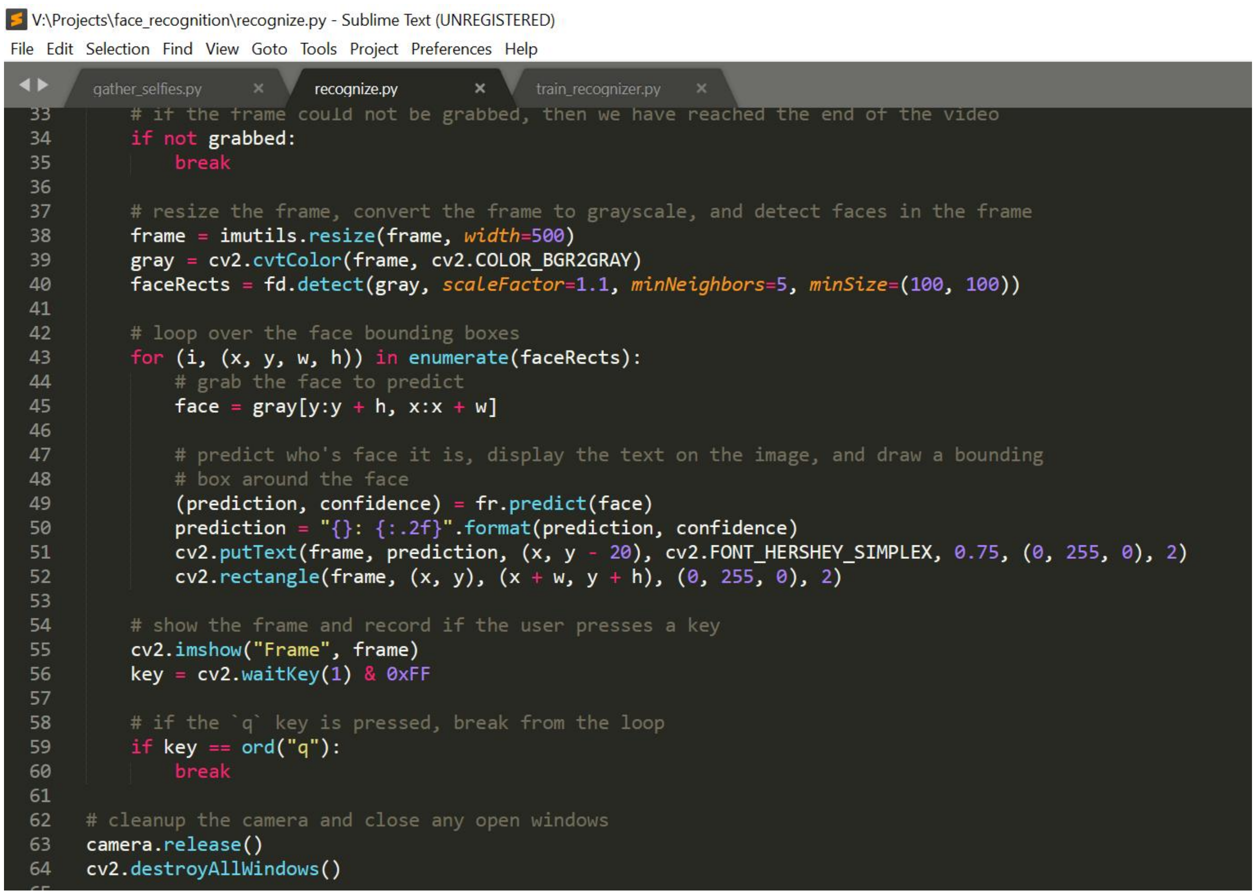


```
Microsoft Windows [Version 10.0.17763.864]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\yeshi>v:

V:\>cd V:\Projects\face_recognition

V:\Projects\face_recognition>python train_recognizer.py --selfies output/faces --classifier output/classifier --sample-s
ize 100
[INFO] training on 'faces\harini'
[INFO] training on 'faces\keerthana'
[INFO] training on 'faces\lavanya'
[INFO] training on 'faces\madhumitha'
[INFO] training on 'faces\Nishitha'
[INFO] training on 'faces\prathyusha'
[INFO] training on 'faces\Ruchitha'
[INFO] training on 'faces\sahaja'
[INFO] training on 'faces\sarala'
[INFO] training on 'faces\vidhya'
[INFO] training on 'faces\yeshi'
```

3. FACE RECOGNITION:

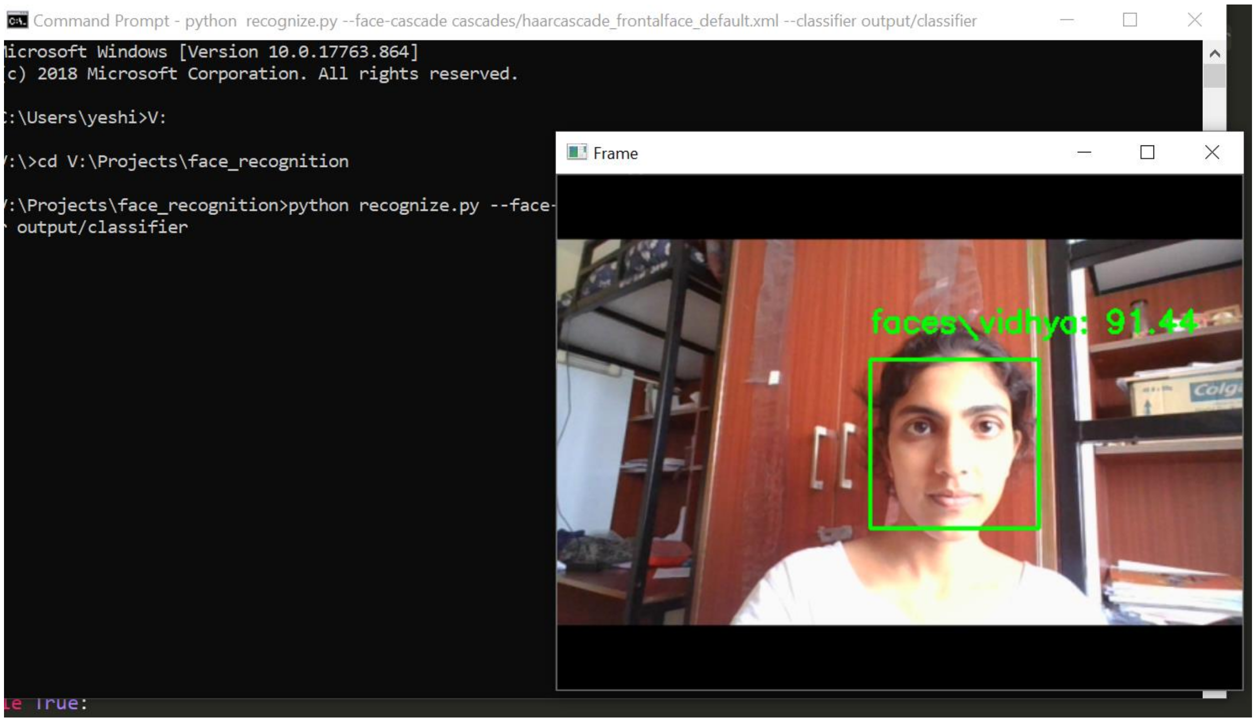

V:\Projects\face_recognition\recognize.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
gather_selfies.py
recognize.py
train_recognizer.py

```python
# import the necessary packages
from tool.face_recognition import FaceDetector
from tool.face_recognition import FaceRecognizer
import argparse
import imutils
import cv2

# construct the argument parse and parse command line arguments
ap = argparse.ArgumentParser()
ap.add_argument("-f", "--face-cascade", required=True, help="path to face detection cascade")
ap.add_argument("-c", "--classifier", required=True, help="path to the classifier")
ap.add_argument("-t", "--confidence", type=float, default=100.0,
    help="maximum confidence threshold for positive face identification")
args = vars(ap.parse_args())

# initialize the face detector, load the face recognizer, and set the confidence
# threshold
fd = FaceDetector(args["face_cascade"])
fr = FaceRecognizer.load(args["classifier"])
fr.setConfidenceThreshold(args["confidence"])

# grab a reference to the webcam
camera = cv2.VideoCapture(0)

# loop over the frames of the video
while True:
    # grab the current frame
    (grabbed, frame) = camera.read()

    # if the frame could not be grabbed, then we have reached the end of the video
    if not grabbed:
        break
```


V:\Projects\face_recognition\recognize.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
gather_selfies.py
recognize.py
train_recognizer.py

```
    # if the frame could not be grabbed, then we have reached the end of the video
    if not grabbed:
        break

    # resize the frame, convert the frame to grayscale, and detect faces in the frame
    frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faceRects = fd.detect(gray, scaleFactor=1.1, minNeighbors=5, minSize=(100, 100))

    # loop over the face bounding boxes
    for (i, (x, y, w, h)) in enumerate(faceRects):
        # grab the face to predict
        face = gray[y:y + h, x:x + w]

        # predict who's face it is, display the text on the image, and draw a bounding
        # box around the face
        (prediction, confidence) = fr.predict(face)
        prediction = "{}: {:.2f}".format(prediction, confidence)
        cv2.putText(frame, prediction, (x, y - 20), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # show the frame and record if the user presses a key
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key is pressed, break from the loop
    if key == ord("q"):
        break

# cleanup the camera and close any open windows
camera.release()
cv2.destroyAllWindows()
```

OUTPUT:


Command Prompt - python recognize.py --face-cascade cascades/haarcascade_frontalface_default.xml --classifier output/classifier
Frame
faces\vidhya: 91.44

```
icrosoft Windows [Version 10.0.17763.864]
c) 2018 Microsoft Corporation. All rights reserved.

:\Users\yeshi>V:

:\>cd V:\Projects\face_recognition

:\Projects\face_recognition>python recognize.py --face-
 output/classifier

le True:
```