The Virtual Soldier: Detecting, Recognizing, Tracing, Informing Criminals as well as crimes in real world.

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

BY

ADITYA TRIPATHI

TAPAN POOJARY

ABHISHEK YADAV

UNDER THE GUIDANCE OF

PROF. JAYA JESWANI

(Prof. Department of Information Technology)

INFORMATION TECHNOLOGY DEPARTMENT

XAVIER INSTITUTE OF ENGINEERING

UNIVERSITY OF MUMBAI

2020– 2021

XAVIER INSTITUTE OF ENGINEERING

MAHIM CAUSEWAY, MAHIM, MUMBAI - 400016.

CERTIFICATE

This to certify that

ADITYA TRIPATHI	(XIEIT171808)
TAPAN POOJARY	(XIEIT171843)
ABHISHEK YADAV	(XIEIT171861)

Have satisfactorily carried out the PROJECT work titled "**The Virtual Soldier: Detecting, Recognizing, Tracing, Informing Criminals as well as crimes in real world.**" in complete fulfillment of the degree of Bachelor of Engineering as laid down by the University of Mumbai during the academic year 2017-2018.

Prof. Jaya Jeswani
Supervisor/Guide

Prof. Vijay Katkar
Head of Department

Dr. Y.D Venkatesh
Principal

PROJECT REPORT APPROVAL FOR B.E.

This project report entitled The Virtual Soldier: Detecting, Recognizing, Tracing, Informing Criminals as well as crimes in real world.

by

ADITYA TRIPATHI (XIEIT171808)

TAPAN POOJARY (XIEIT171843)

ABHISHEK YADAV (XIEIT171861)

is approved for the degree of BACHELOR OF ENGINEERING.

Examiners

1. ______________________

2. ______________________

Supervisors

1. ______________________

2. ______________________

Date:

Place: MAHIM, MUMBAI

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' Ideas or words have been included, I have adequately cited and referenced the original sources.

I also declare that I have adhered to all the principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which thus have not been properly cited or from whom proper permission have not been taken when needed.

Aditya Tripathi (XIEIT171808)	------------------------------
Tapan Poojary (XIEIT171843)	-----------------------------
Abhishek Yadav (XIEIT171861)	-----------------------------

Date:

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

Finding criminals or hunting for people, in a CCTV video footage, after a crime scene or major attack takes place, is a time-consuming task. Pattern Analytical System and Data Mining techniques are powerful tools that helps in removing covered up information from a huge dataset to upgrade exactness and productivity of estimating.

Traditional prediction frameworks are hard to manage the enormous information and exactness of crime detection. As informed to us by cyber cell members, they make multiple members of the department sit with laptops and computers literally to search through the CCTV footage to find and trace the guilty, as they don't have the automated system for doing this task with them. This process is both time and labour intensive.

In this research paper we have tried to survey the existing technologies as well as we propose a new system for criminal Detection & Recognition using Cloud Computing and Machine Learning, which if used by our Crime Agencies would definitely help them to find criminals from CCTV footage. The proposed system can not only help find criminals but if used properly on different sites such as railway stations etc, can also help find missing children and people from the CCTV footage available from the respective site. Existing solutions use traditional face recognition algorithms which can be troublesome in changing Indian environments especially factors like light, weather and especially orientation.

Acknowledgement

We would like to thank Fr. John Rose (Director of XIE) for providing us with such an environment so as to achieve goals of our project and supporting us constantly.

We express our sincere gratitude to our Honorable Principal Mr. Y.D.Venkatesh for encouragement and facilities provided to us.

We would like to place on record our deep sense of gratitude to Dr. Vijay Katkar, Head of Dept Of Information Technology, Xavier Institute of Engineering, Mahim, Mumbai, for her generous guidance help and useful suggestions.

With deep sense of gratitude we acknowledge the guidance of our project guide Prof. Jaya Jeswani . The time-to-time assistance and encouragement by her has played an important role in the development of our project.

We would also like to thank our entire Information Technology staff who have willingly cooperated with us in resolving our queries and providing us all the required facilities on time.

Aditya Tripathi	----------------------------
Tapan Poojary	----------------------------
Abhishek Yadav	----------------------------

CHAPTER 1

INTRODUCTION

The Virtual Soldier: Detecting, Recognizing, Tracing, Informing Criminals as well as crimes in real world.

1.1 Problem Definition:

Circuit Television Cameras (CCTV's) are widely used to control occurrence of crimes in the surroundings. Although CCTV's are deployed at various public and private areas to monitor the surroundings there is no improvement in the control of crimes. This is because CCTV requires human supervision which may lead to human prone errors like missing of some important crime events by human while monitoring so many screens recorded by CCTV's at same time and Retrieval of images with object-of-interest from a vast pool of social media images has been a research interest in cybercrime research community for detecting criminal behaviours in social media. Due to inherent diversity and the low duplicate property of images on social media, it brings forth many challenges in image retrieval, especially in identifying distinct features for a given object-of-interest.

The quick and accurate identification of criminal activity is paramount to securing any residence. With the rapid growth of smart cities, the integration of crime detection systems seeks to improve this security. In the past a strong reliance has been put on standard video surveillance in order to achieve this goal. This often creates a backlog of video data that must be monitored by a supervising official. For large urban areas, this creates an increasingly large workload for supervising officials which leads to an increase in error rate

Automatic recognition of violence between individuals or crowds in videos in real time is very difficult but at the same time, if it is implemented in a right direction can provide a high accurate result.

Our Problem revolves around 4 different types of crimes i.e.

1) Crimes happening by the known criminal in real time.
2) Crimes happening by unknown criminal in real time.
3) Crimes like mob lynching which normally happens in crowd
4) Crimes like stealing of vehicles.

1.2 Scope of Project

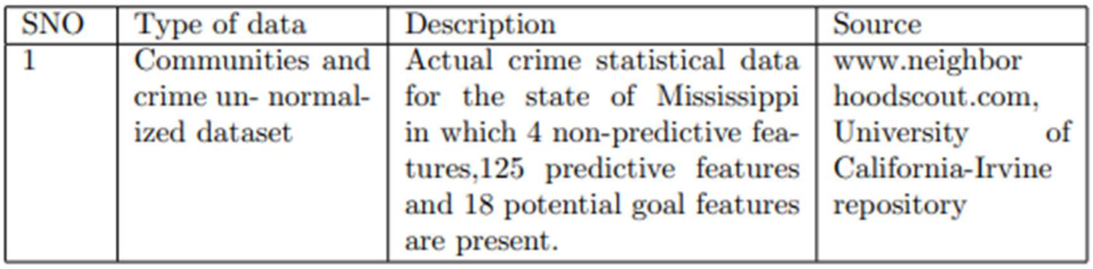
Crimes now days are increasing day by day and with different level of intensity and versatility. The result is great loss to society in terms of monitory loss, social loss and further it enhances the level of threat against the smooth livelihood in the society. To overcome this problem the computing era can help to reduce the crime or even may be helpful in predicting the crime so that sufficient measures can be taken to minimize the loss to property and life. The crime rate prediction strategies can be applied on historical data available in the police records by examining the data at various angles like reason of crime, frequency of similar kind of crimes at specific location with other parameters to prepare model the crime prediction. It is the major challenge to understand the versatile data available with us then model it to predict the future incidence with acceptable accuracy and further to reduce the crime rate.

As a future scope extension of crime detection and analysis will be to generate the crime hot-spots that will help in deployment of police at most likely places of crime for any given window of time, to allow most effective utilization of police resources. The developed model will reduce crimes and will help the crime detection field in many ways that is from arresting the criminals to reducing the crimes by carrying out various necessary measures.

1.3 Existing System

Many researchers have worked on the same techniques to predict the future crime based on certain input parameters. The approaches used by them to evaluate the prediction models include: artificial neural network, support vector and time series analysis etc. We with help of crime data modeling the crime pattern can be detected or worked upon. Crime data analysis can even speed up the resolution to the court cases with ease and timely. The previously unknown information can be better extracted from the historical data. Some researcher used clustering algorithms for crime analysis. Clustering approach can work well in this case as the crime is totally random in nature and the available data cannot effectively predict the future incidences? In this case the clustering process can find similarities between crimes in increasing size pattern and in remaining database. Such types of crime prediction approach work on pattern of crimes in terms of similarity of attack or crime such as time of incidence, the methodology of crime is same etc.

Table 1 Existing Crime Investigation System



SNO	Type of data	Description	Source
1	Communities and crime un- normal-ized dataset	Actual crime statistical data for the state of Mississippi in which 4 non-predictive fea-tures,125 predictive features and 18 potential goal features are present.	www.neighbor hoodscout.com, University of California-Irvine repository

SNO	Type of data	Description	Source
2	Cybercrimes by motives	Data regarding various cyber-crime like revenge, greed, extortion, harassment and others.	Https://data.gov. in /catalog/cases-registered-under-cyber-crimes-motives
3	Communities and crime dataset	Multivariate characteristics of data with real attribute characteristic associated with regression, which consists of 1994 instances and 128 attributes.	Http://archive.i cs.uci.edu/ml /datasets
4	Persons arrested for crime against women	The various crimes covered for a given time period, like rape kidnapping and abduction, insult to the modesty of women, immoral traffic act, dowry deaths, commission of sati prevention act and assault on women	Https://data.gov. in/catalog/perso ns-arrested-under-crime-against-women
5	Victims of murder	A total number of cases reported under murder and number of victims of murder by gender on the basis of age group.	Https://data.gov. in/catalog/age-group-wise-victims-murder
6	Criminal victimization data	Data consists of burglary, robbery, theft, fraud, assault, murder, thuggery and rape cases.	In-home, face-to-face personal interview using a stratified multistage random selection procedure
7	Real-word crimes in two cities of the US	Dataset information is based on the national Incident-based reporting system for five years. It consists of 19 attributes with 333068 instances, also gives the exact occurrence time and location of a district where a crime occurred	http://data.denv er gov.org/dataset /city-and-county-of-denver crime. Http://us-city.census.okfn. org/dataset/crime -stats.
8	Crime data of Tamilnadu	Six cities (Chennai, Coimbatore, Salem, Madurai, Thirunelvelli, Thiruchirapalli) crimes are listed for the year 2000-2014 which consist of 1760 instances and 9 attributes.	National crime records bureau (NCRB)

1.4 Proposed System:

Crime can be better predicted if we use clustering approach rather than classification due random nature of crimes. Clustering approach may work better.

Following steps may apply for the crime analysis:

1. Collect the data:

In this phase the data from various sources is collected from various government sources, social media platform about the incidence like facebook, blogs etc. It may expect that the data received may be in unstructured form with different types and size. So oops approach may be better to extract the information. Explicitly NoSql can be used as it run without schema detail.

2. Classification:

The standard algorithm useful for classification which may be applied is Naïve bayes Classifier. This classifier will the probability of falling into different classes of crimes and the crime predicted to belong to specific class the probability is highest. Naïve bayes classifier is more efficient as compared to regression in terms of convergence rate.

3. Identify pattern:

Next phase in the methodology is to find the sequence of crimes which are similar in nature and belongs to same class. Such pattern may be identified as suggested in literature through the apriori algorithm. Apriori algorithm work on principle of association rules. Association rules highlight the trend the crime pattern database.

Pattern identification outcome may be location based pattern detection. Pattern information may help the government or police department to effectively speedily work on resolving the criminal cases.

4. Predict the Crime:

For the prediction outcome the decision trees are used. Here in decision tree each internal note has test for occurrence on an incidence and outcome is normally yes or no and based on the outcome again the next level of internal node has another question test and so on to reach to final decision for prediction.

CHAPTER 2

REVIEW OF LITERATURE

The first paper in this survey is also a survey of existing technologies as well as a new system for criminal Detection & Recognition using Cloud Computing and Machine Learning, which if used by our Crime Agencies would definitely help them to find criminals from CCTV footage. The proposed system can not only help find criminals but if used properly on different sites such as railway stations etc, can also help find missing children and people from the CCTV footage available from the respective site. Existing solutions use traditional face recognition algorithms which can be troublesome in changing Indian environments especially factors like light, weather and especially orientation. Some CCTV are in a bad place and can get tilted resulting in a wild increase in inaccuracy. This research paper proposes to use Microsoft Azure Cognitive services and Cloud system for implementation of the proposed system. The next phase this research is trying to compare this proposed methodology with traditional techniques like HAAR cascade to judge performance of the proposed System, as it is important to have a high accuracy, for a project of this sensitivity [1].

The second paper proposed a real-time violence detector based on deep-learning methods. The proposed model consists of CNN as a spatial feature extractor and LSTM as temporal relation learning method with a focus on the three-factor (overall generality - accuracy - fast response time). The suggested model achieved 98% accuracy with speed of 131 frames/sec [2].

In Third Paper we saw a machine learning based predicting policing algorithm exploiting the modus operandi features of a recent crime and the existing prior criminal records. The proposed model predicts and probabilistically shortlist the potential suspects who may be involved in a recent crime. Thereby, meaningful leads are produced to aid the investigation process. In this paper, we take the Punjab province of Pakistan as a case study and apply the proposed model on the real data collected from various police stations. The results show that the identification of potential suspects proved vital for the cases which involves criminals who had previous criminal record.[3]

Fourth we saw a deep learning model based on 3D convolutional neural networks, without using hand-crafted features or RNN architectures exclusively for encoding temporal information. The improved internal designs adopt compact but effective bottleneck units for learning motion patterns and leverage the Dense Net architecture to promote feature reusing and channel interaction, which is proved to be more capable of capturing spatiotemporal features and requires relatively fewer parameters. The performance of the proposed model is validated on three standard datasets in terms of recognition accuracy compared to other advanced approaches. Meanwhile, supplementary experiments are carried out to evaluate its effectiveness and efficiency [4].

It presents different techniques for detection of such activities from the video proposed in the recent years. This research study reviews various state-of-the-art techniques of violence detection. In this paper, the methods of detection are divided into three categories that is based on classification techniques used: violence detection using traditional machine learning, using Support Vector Machine (SVM) and using Deep Learning [5]

In this work, they employ statistical analysis methods and machine learning models for predicting different types of crimes in New York City, based on 2018 crime datasets.

They have combined weather, and its temporal attributes like cloud cover, lighting and time of day to identify relevance to crime data. They note that weather-related attributes play a negligible role in crime forecasting. They have evaluated the various performance metrics of crime prediction, with and without the consideration of weather datasets, on different types of crime committed. their proposed methodology will enable law enforcement to make effective decisions on appropriate resource allocation, including backup officers related to crime type and location [6].

In this work, an end-to-end deep neural network model for the purpose of recognizing violence in videos is proposed. The proposed model uses a pre-trained VGG-16 on ImageNet as spatial feature extractor followed by Long Short-Term Memory (LSTM) as temporal feature extractor and sequence of fully connected layers for classification purpose. The achieved accuracy is near state-of-the-art. Also, they contribute by introducing a new benchmark called Real- Life Violence Situations which contains 2000 short videos divided into 1000 violence videos and 1000 non-violence videos. The new benchmark is used for fine-tuning the proposed models achieving a best accuracy of 88.2% [7].

This paper proposed deep representation-based model using concept of transfer learning for violent scenes detection to identify aggressive human behaviours. The result reports that proposed approach is outperforming state-of-the art accuracies by learning most discriminating features achieving 99.28% and 99.97% accuracies on Hockey and Movies datasets respectively, by learning finest features for the task of violent action recognition in videos [9].

In this paper we came up with Crime Intension Detection System that detects crime in real time videos, images and alerts the human supervisor to take the necessary actions. To alert the supervisors or nearby police station about the occurrence of crime.

We added SMS sending module to our system which sends SMS to concern person whenever crimes are detected. The proposed system is implemented using Pre-trained deep learning model VGGNet19 which detects gun and knife in the hand of person pointing to some other person. We also compared the working of two different pre-trained models like Google Net InceptionV3 in training. The results obtained with VGG19 are more accurate in terms of training accuracy. This motivated us to use VGG19 with little fine tuning to detect crime intention in videos and images to overcome the issues with existing approaches with more accuracy. And we made use of Fast RCNN and RCNN these algorithms are well known as Faster RCNN this helps us to draw the bounding box over objects in images like person, gun, knife and some untrained images are marked with N/A. Algorithms help for detection and classifications of objects over images [10].

In next research papers, study is based on the analysis of faces, emotions, Ages and genders to identify the suspects. Face recognition, emotion, age and gender identifications are implemented using deep learning-based CNN approaches. Suits identification is based on LeNet architecture. In the implementation phase for the classification purpose, Keras deep learning library is used, which is implemented on top of TensorFlow. IMDb is the dataset used for the whole training purpose. Training is performed using in AWS cloud which is more powerful and capable way of training instead of using local machines. Real- time Video and images are taken for the experiment. Results of the training and predictions are discussed below in brief [11].

In this paper the purpose is to classify each facial image as one of the seven facial expressions considered in this study. According to the characteristics of facial expression recognition, a new convolution neural network structure is designed which uses convolution kernel to extract implicit features and max pooling to reduce the dimensions of the extracted implicit features. In comparison to AlexNet network, we can improve the recognition accuracy about 4% higher on the CK+ facial expression database by the aid of Batch Normalization (BN) layer to our network. A facial expression recognition system is constructed in this paper for the convenience of application, and the experimental results show that the system could reach the real-time needs [12].

In this paper, heterogeneous data-based danger detection (HDDD) mechanism is proposed for reducing crime rates. The HDDD mechanism is a mechanism for detecting dangerous situations (e.g., homicide and violence) based on heterogeneous data that are data gathered from multiple devices such as closed-circuit television (CCTV) cameras, smartphones, and wearable devices. The HDDD enables immediate detection of dangerous situations and then reduces crime rates by responding to the dangerous situations [14].

In this paper, they present CNN (Convolutional Neural Network) in the use of detect knife, blood and gun from an image. Detecting these threatening objects from image can give us a prediction whether a crime occurred or not and from where the image is taken. They emphasized on the accuracy of detection so that it hardly gives us wrong alert to ensure efficient use of the system. This model uses Rectified Linear Unit (ReLU), Convolutional Layer, fully connected layer and dropout function of CNN to reach a result for the detection. We use TensorFlow, an open source platform to implement CNN to achieve our expected output. The proposed model achieves 90.2% accuracy for the tested dataset [15].

This table is a brief description of all the papers we surveyed and also took into considerations while proposing this project and to get the best results.

Table 2: Brief description of all the papers

Reference	**Publisher**	**Method Used**	**Strength**	**Weakness**	**Tool Used**	**Dataset**
1)	IEEE 2019	Haar Cascade and Microsoft azure	Very Detailed	Not an exact solution	NA	NA
2)	IEEE 2019	CNN and LSTM	Accurate, robust and fast	Complex	TensorFlow and CNN (vgg19)	combination of hockey fight, movie fight, crowd violence
3)	IEEE 2019	K-means and GMM	Real Data from Police St.	Inaccurate	MATLAB and clustering algo.	NA
4)	IEEE 2019	3D CNN	High Accuracy	Different	CNN	combination of hockey fight, movie fight, crowd violence
5)	IEEE Access 2019	Deep Learning and SVM	Informative	Not an exact solution	SVM, ML Algo	NA
6)	IEEE 2019	Machine Learning	New Way of Solving	Large dataset required to train	Classification Algo.	2018 New York Crime Dataset
7)	IEEE 2019	VGG-16 on ImageNet and LSTM	Accurate	Big Dataset	NA	combination of hockey fight, movie fight, crowd violence
8)	IEEE 2018	DCNN and HDL	Easy to understand	Requires time	NA	NA
9)	IEEE 2018	DCNN and Google Net	High Accuracy	Complex	Google Net and MATLAB	Hockey and Movie dataset
10)	IEEE 2018	VGGNet-19 and RCNN	Descriptive and Accurate	Requires large dataset	Keras	NA

11)	IEEE 2018	CNN and LeNet	Accurate and New Approac h	Complex and Required time	Keras, TensorFlow, AWS	IMDb Dataset
12)	IEEE 2017	CNN, C++ and computer vision	High Accuracy	Only Detection	OpenCV	CK+ and JAFFE
13)	SSRN-ELSE V IER 2019	OpenCV	Fast	Inaccurat e	OpenCV	NA
14)	IEEE 2018	HDDD	Fast and detailed	Not an exact solution	NA	NA
15)	IEEE 2018	CNN, ReLU	High Accuracy	Large dataset required	TensorFlow and CNN (vgg16)	NA

CHAPTER 3

SYSTEM DESIGN AND ANALYSIS

3.1 DESIGN

1. The input for the model will consist of the image and video feeds.
2. Once the dataset is ready with all the known criminal's data, we can pass it to our system which is a live CCTV Surveillance. So due to this if any particular moment camera recognizes a criminal and it gets matched with our dataset…It will give a message and alarm to the guard who will be at that moment monitoring the surveillance and also one message will go directly to the command centre of police with the camera no. and location of the particular criminal.
3. If a crime is done by a normal person then without having any information about it. Police cannot catch him, in that with our system since it is capturing face of all the people passing through it…will try to scan that particular image to all the official websites, Facebook, Instagram etc.
4. Crimes like Mob-Lynching, fights between groups are often seen in India and all the people standing their they take videos of crime instead of calling the police. So, in this scenario our system with help of machine learning and artificial intelligence will try to recognize the crime scene as a normal scene or any anomaly or unwanted happening in the scene. If it detects something uneven the images, videos, and location of that crime will be directly sent to the nearest police station.
5. Suppose there is a situation wherein someone parks his/her Car or Bike in the parking lot of

their society and in night, a thief robbed the bike and leaving the societies gate, so our system will work as an alarm system wherein in our database we will have all the entries of cars and bikes number plate in that society with their owners name and phone number. so, our system using the cameras located on the gate will detect and recognize the number plate and will send a SMS by saying that your bike/car just leaved the gate.so by this instead of owner getting understand about the robbery of bike in morning he/she will get and instant message and can catch the criminal faster.

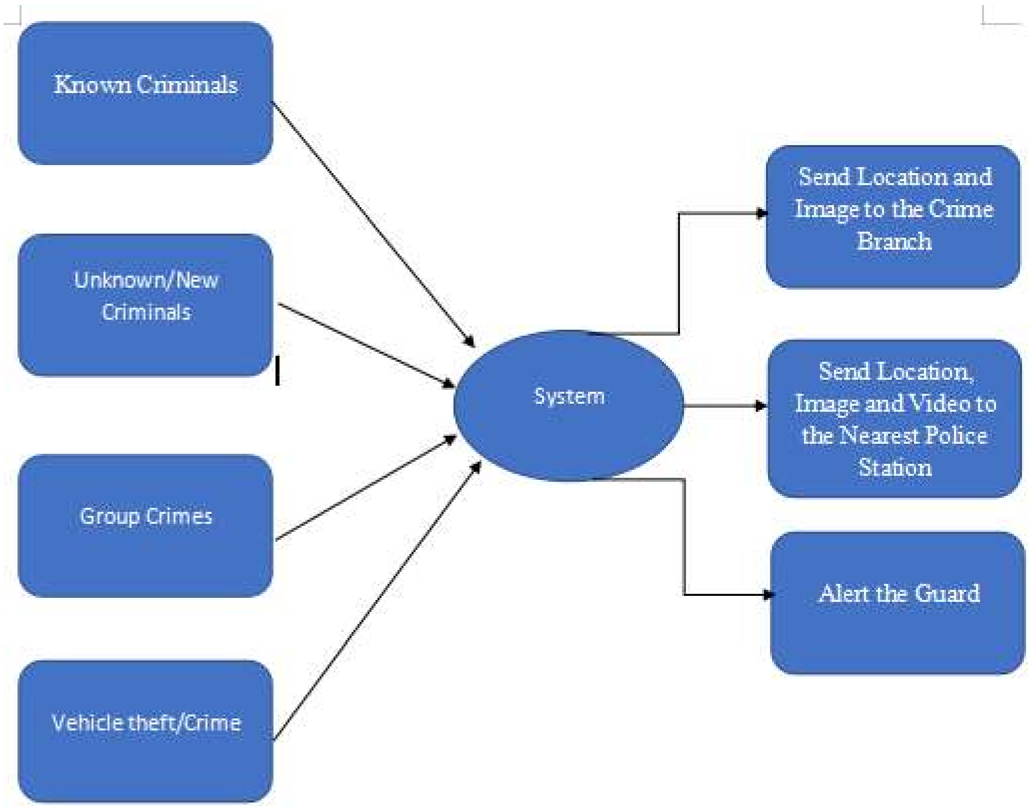

Known Criminals
Unknown/New Criminals
Group Crimes
Vehicle theft/Crime
System
Send Location and Image to the Crime Branch
Send Location, Image and Video to the Nearest Police Station
Alert the Guard

Figure 1: Overview of System

3.2 ALGORITHM

Step 1 : Start

Step 2 : Accept the input image or video

Step 3 : Perform an analysis of the captured images by comparing it with trained data.

Step 4 : Objects are classified then localized using object detection algorithm

Step 5 : Combination of classification and localization results in the identification of an object.

Step 6 : Based on identification predictions are made and probability is found.

Step 7 : The prediction having the largest probability will be the output.

Step 8 : The output tells the crime is detected or not.

Step 9 : Alert the respective police authority for the location of the crime.

Step 10 : Look if the crime persists or not.

3.3 DESIGN ANALYSIS

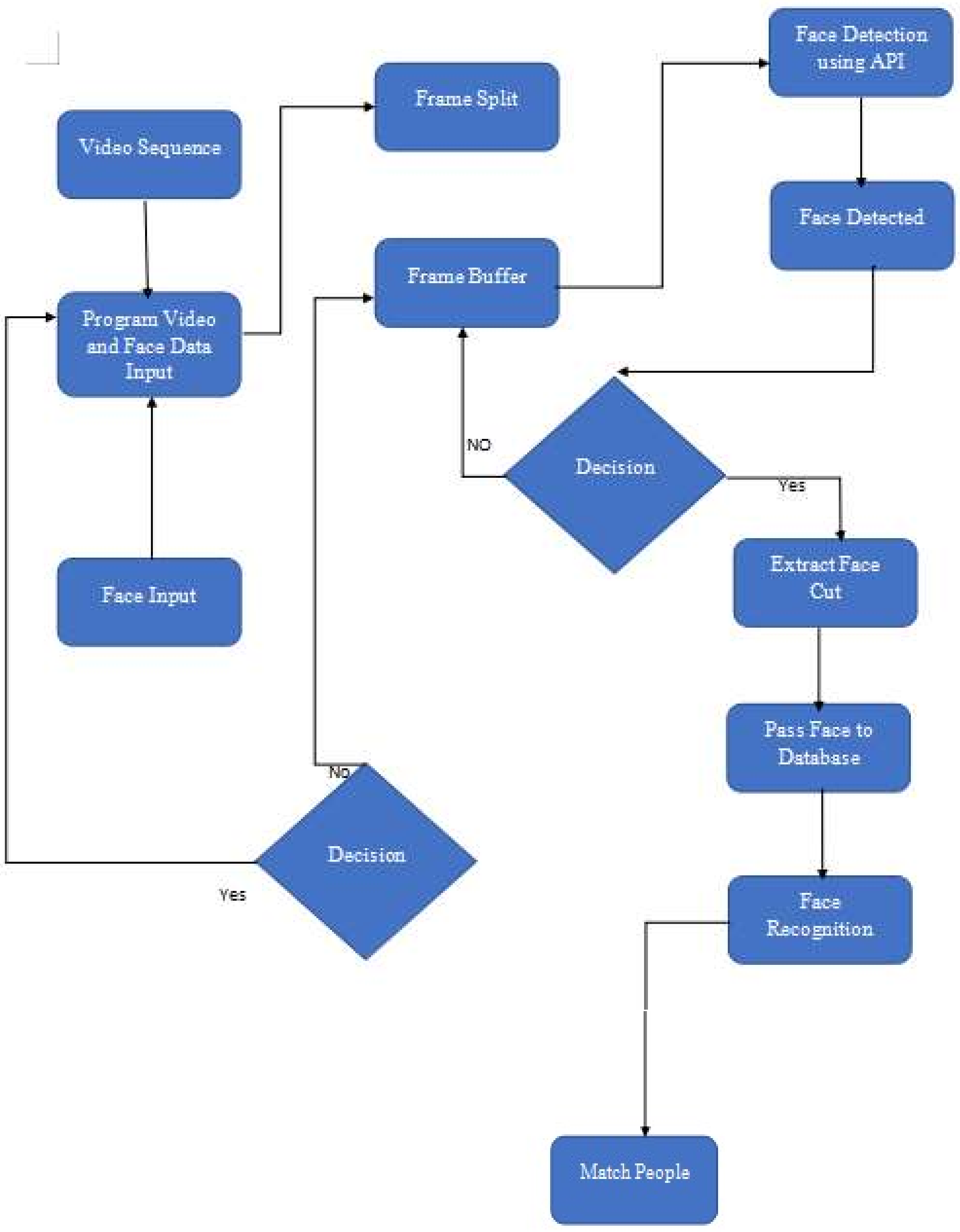


Figure 2: Flowchart of Module 1

3.4 HARDWARE AND SOFTWARE REQUIREMENTS

Recommended System Requirements:

- Processors: Intel Core i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores,2 threads per core), 8 GB of DRAM
- Disk space:3 to 5 GB
- Operating systems: Windows 10, Mac OS, and Linux
- Webcam

Minimum System Requirements:

- Processors: Intel Core i3 processor
- Disk space: 2 GB
- Operating systems: Windows 10, MacOS, and Linux

Software:

- Anaconda with installed libraries
- Apache, CSV
- Web Browser such as Google Chrome
- Opencv
- Machine Learning and Neural Networks Libraries

CHAPTER 4

IMPLEMENTATION, TESTING AND RESULT

We have divided our project in 4 modules and in the end it will be a whole system in which all these features will be implanted and can be directly used by any of the CCTV Camera.

Module 1: Known Criminals like e.g. Dawood using HAAR Cascade.

Since we have a lot of images of Dawood and lot of information to…we can train our data with its face, eyes etc. Once the dataset is ready with all the known criminal's data, we can pass it to our system which is a live CCTV Surveillance. So due to this if any particular moment camera recognizes a criminal and it gets matched with our dataset…It will give a message and alarm to the guard who will be at that moment monitoring the surveillance and also one message will go directly to the command centre of police with the camera no. and location of the particular criminal.

Due to this…if police want to catch the criminal it can go directly catch or if police want to track the criminal or navigates its whole plan…then it can also put an eye on it and try to recognize its plan…or the reason of coming to that location.

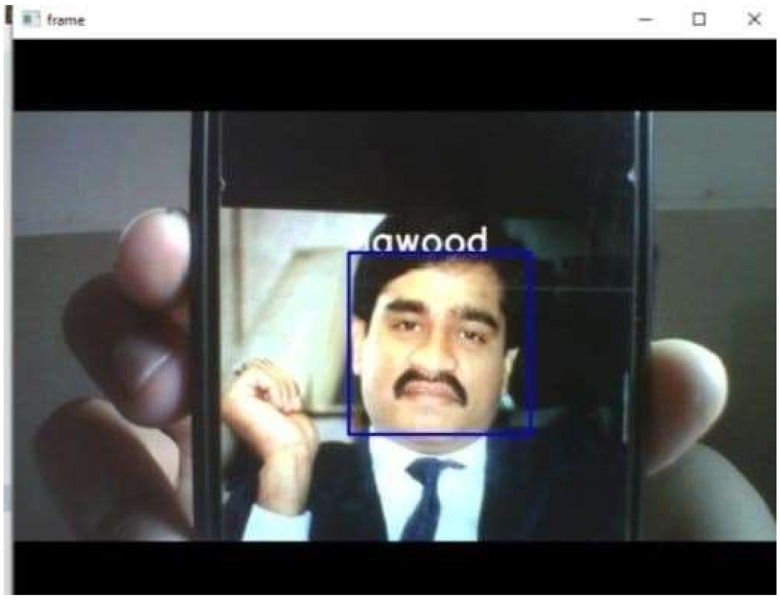


Figure 3: Image of criminal detected

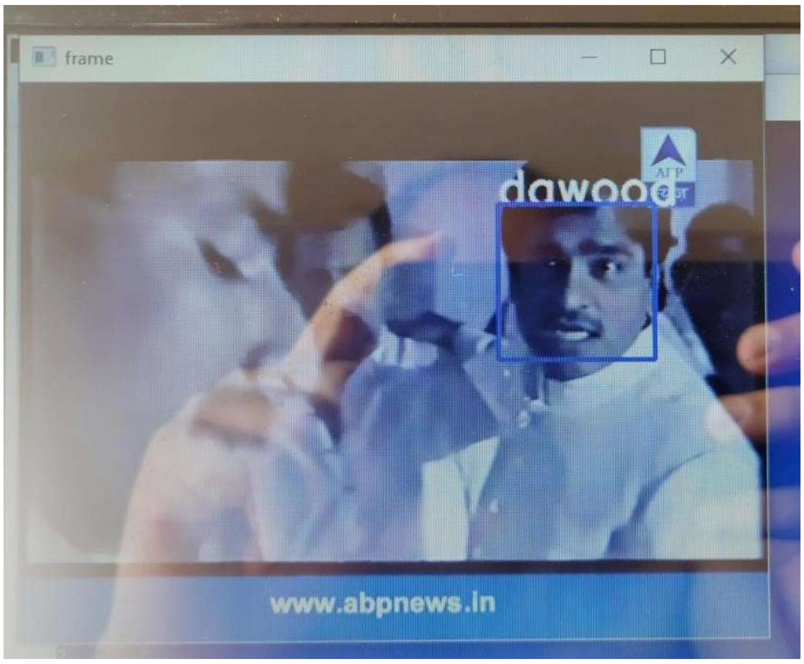
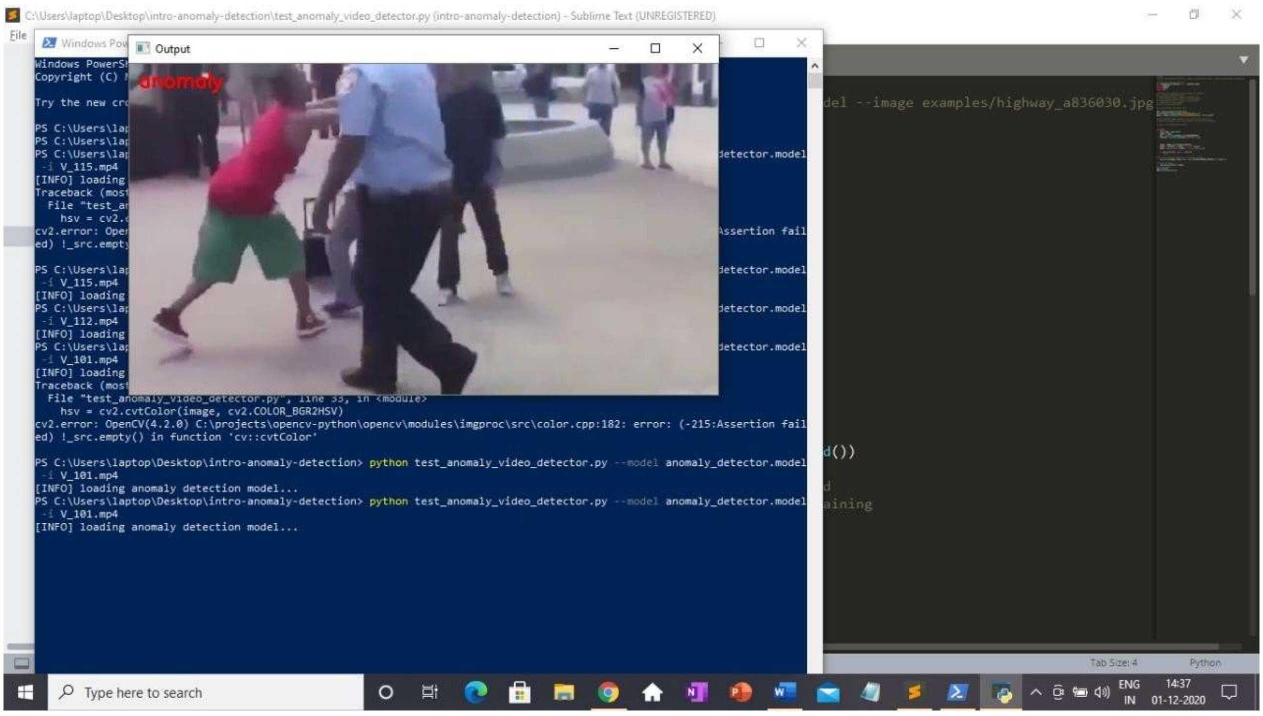


Figure 4: Video consisting frame of Criminal detected

Module 2: Live Crime detecting and passing it to the control room (Mob-Lynching):

Crimes like Mob-Lynching, fights between groups are often seen in India and all the people standing their they take videos of crime instead of calling the police. So, in this scenario our system with help of machine learning and artificial intelligence will try to recognize the crime scene as a normal scene or any anomaly or unwanted happening in the scene. If it detects something uneven the images, videos, and location of that crime will be directly sent to the nearest police station.

All this will be happening at a real time. In today's scenario…first the crime is happened and after then police investigates but with our system in between crime police can reach to the location at a fast rate.


C:\Users\laptop\Desktop\intro-anomaly-detection\test_anomaly_video_detector.py (intro-anomaly-detection) - Sublime Text (UNREGISTERED)
Output
anomaly
Type here to search

```
  File "test_anomaly_video_detector.py", line 33, in <module>
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
cv2.error: OpenCV(4.2.0) C:\projects\opencv-python\opencv\modules\imgproc\src\color.cpp:182: error: (-215:Assertion fail
ed) !_src.empty() in function 'cv::cvtColor'

PS C:\Users\laptop\Desktop\intro-anomaly-detection> python test_anomaly_video_detector.py --model anomaly_detector.model
 -i V_101.mp4
[INFO] loading anomaly detection model...
PS C:\Users\laptop\Desktop\intro-anomaly-detection> python test_anomaly_video_detector.py --model anomaly_detector.model
 -i V_101.mp4
[INFO] loading anomaly detection model...
```

Figure 5: Live crime detection

Module 3: Vehicle Theft:

Suppose there is a situation wherein someone parks his/her Car or Bike in the parking lot of their society and in night, a thief robbed the bike and leaving the societies gate, so our system will work as an alarm system wherein in our database we will have all the entries of cars and bikes number plate in that society with their owners name and phone number. so, our system using the cameras located on the gate will detect and recognize the number plate and will send a SMS by saying that your bike/car just leaved the gate.so by this instead of owner getting understand about the robbery of bike in morning he/she will get and instant message and can catch the criminal faster.

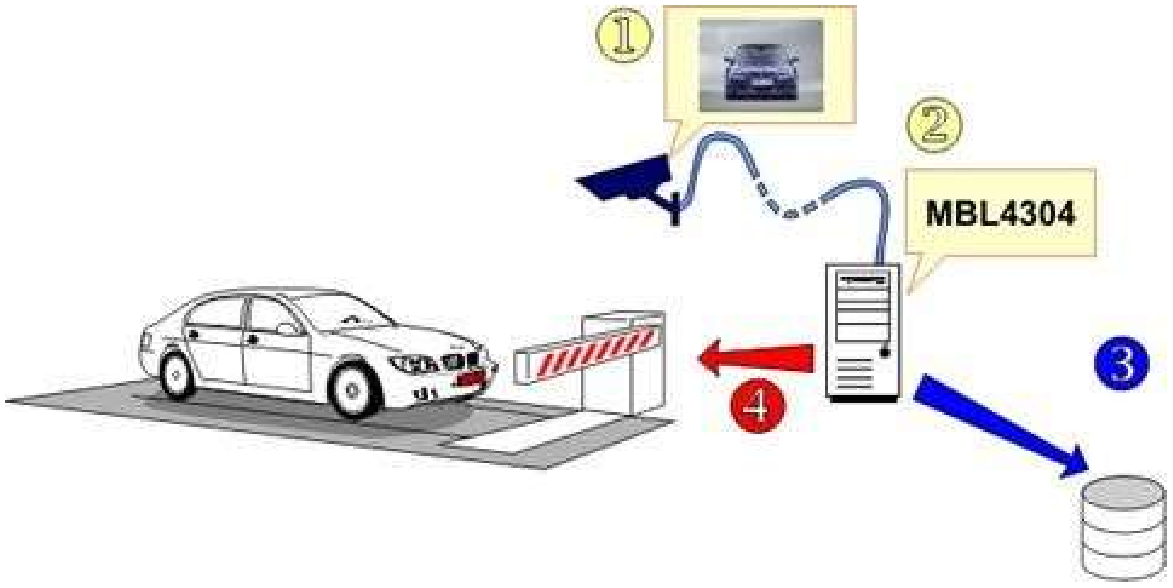


Figure 6: LPN detecting method

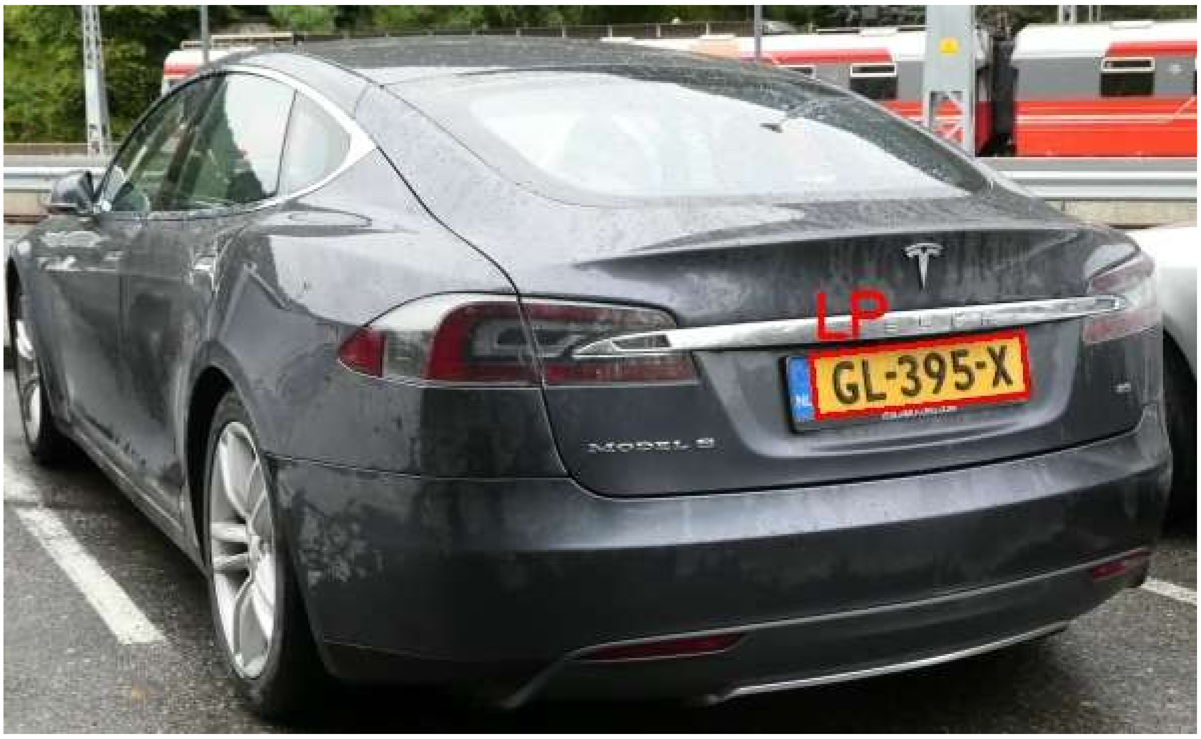
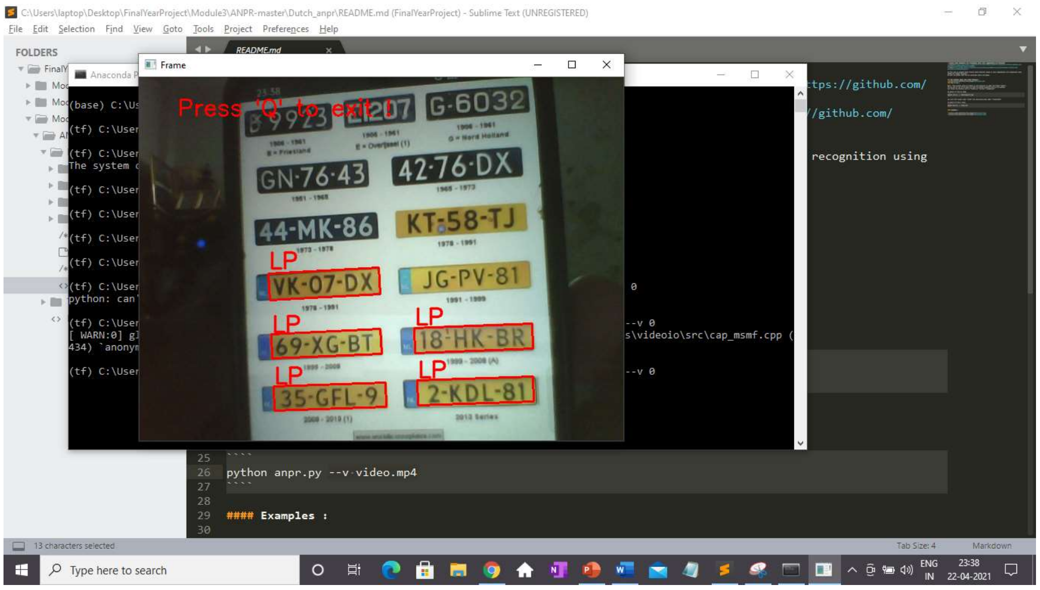


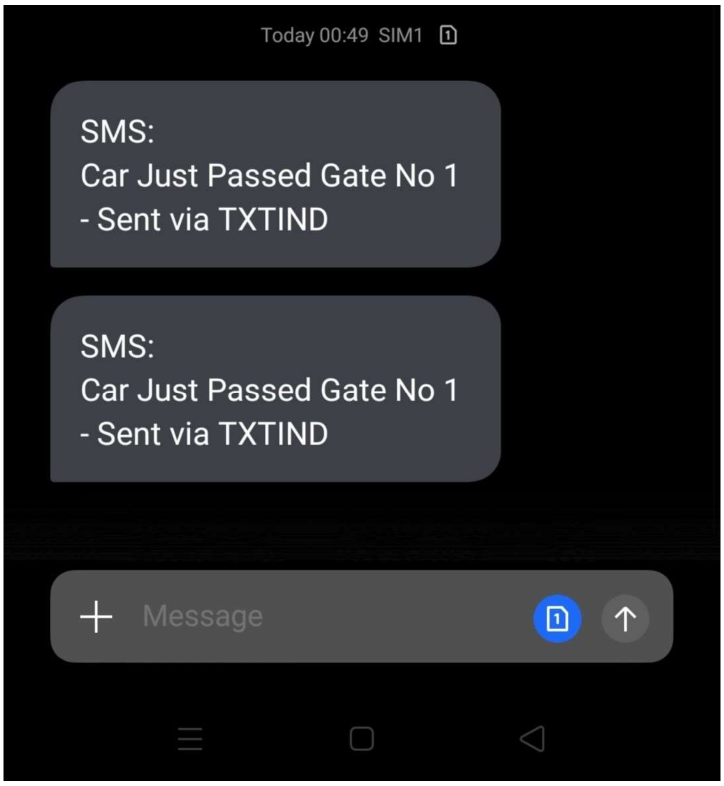
Figure 7: License Plate Number detection in Single Car



```
(base) C:\Us
(tf) C:\User
(tf) C:\User
The system c
(tf) C:\User
(tf) C:\User
(tf) C:\User
(tf) C:\User
(tf) C:\User
python: can'
(tf) C:\User
[ WARN:0] gl
434) `anonym
(tf) C:\User
```

```
```
python anpr.py --v video.mp4
```
#### Examples :
```

Figure 8: License Plate Number detection in Multiple Car


Today 00:49 SIM1

SMS:
Car Just Passed Gate No 1
- Sent via TXTIND

SMS:
Car Just Passed Gate No 1
- Sent via TXTIND

Figure 9: The correct number plate detected and Message Sent

Code:

Module 1: Known Criminals like e.g. Dawood using HAAR Cascade.

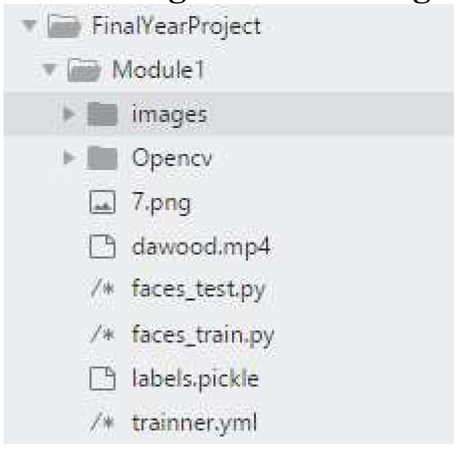


Figure 10: Overview of Module1

Faces_test.py:

```
import numpy as np
import cv2
import pickle

face_cascade =
cv2.CascadeClassifier('C:\\Users\\laptop\\Desktop\\FinalYearProject\\Module1\\Opencv\\data\\h
aarcascade_frontalface_alt2.xml')

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("trainner.yml")




labels = {"person_name": 1}
with open("labels.pickle", 'rb') as f:
    og_labels = pickle.load(f)
    labels = {v:k for k,v in og_labels.items()}

cap = cv2.VideoCapture('dawood.mp4')

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

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
    for (x,y,w,h) in faces:
            # print(x,y,w,h)
            roi_gray = gray[y:y+h, x:x+w]
            roi_color = frame[y:y+h, x:x+w]

            # recognize ? deep learned models predict using keras, tensorflow , pytorch ,
scikitlearn
            id_, conf = recognizer.predict(roi_gray)
            if conf>=45:
                print(id_)
                print(labels[id_])
                font = cv2.FONT_HERSHEY_SIMPLEX
                name = labels[id_]
                color = (255, 255, 255)
                stroke = 2
                cv2.putText(frame, name, (x,y), font, 1, color, stroke, cv2.LINE_AA)

            img_item = "7.png"
            cv2.imwrite(img_item, roi_color)
            # cv2.imwrite(img_item, roi_color)

            color = (255, 0, 0)  #BGR
            stroke = 2
            end_cord_x = x + w
            end_cord_y = y + h
            cv2.rectangle(frame, (x, y), (end_cord_x, end_cord_y), color, stroke)


    # cv2.imshow('gray', gray)
    cv2.imshow('frame', frame)

    if cv2.waitKey(20) & 0xFF == ord('q'):
            break

cap.release()
cv2.destroyAllWindows()
```

Faces_train.py:

```
import cv2
import os
import numpy as np
from PIL import Image
import pickle


BASE_DIR = os.path.dirname(os.path.abspath(__file__))

image_dir = os.path.join(BASE_DIR, "images")

face_cascade =
cv2.CascadeClassifier('C:\\Users\\laptop\\Desktop\\Opencv\\Opencv\\data\\haarcascade_frontalf
ace_alt2.xml')

recognizer = cv2.face.LBPHFaceRecognizer_create()

current_id = 0
label_ids = {}

y_labels = []
x_train = []
```

```
for root, dirs, files in os.walk(image_dir):
    for file in files:
            if file.endswith("png") or file.endswith("jpg"):
                    path = os.path.join(root, file)
                    label = os.path.basename(os.path.dirname(path)).replace(" ", "-").lower()
                    # print(label, path)
                    if not label in label_ids:
                            label_ids[label] = current_id
                            current_id += 1

                    id_ = label_ids[label]
                    # print(label_ids)
                    # y_labels.append(label) #some number
                    # x_train.append(path)   #verify this image, turn into a numpy array,
grayimage
                    pil_image = Image.open(path).convert("L") #grayscale
                    size = (550, 550)
                    final_image = pil_image.resize(size, Image.ANTIALIAS)
                    image_array = np.array(final_image, "uint8")
                    # print(image_array)
                    faces = face_cascade.detectMultiScale(image_array, scaleFactor=1.5,
minNeighbors=5)

                    for (x,y,w,h) in faces:
                            roi = image_array[y:y+h, x:x+h]
                            x_train.append(roi)
                            y_labels.append(id_)


# print(y_labels)
# print(x_train)

with open("labels.pickle", 'wb') as f:
    pickle.dump(label_ids, f)


recognizer.train(x_train, np.array(y_labels))
recognizer.save("trainner.yml")
```

Dataset:

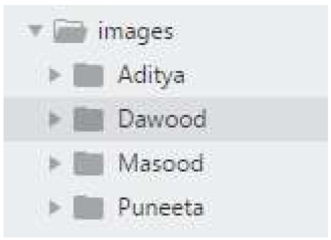


Figure 11: Dataset Image

Module 2: Live Crime detecting and passing it to the control room (Mob-Lynching):

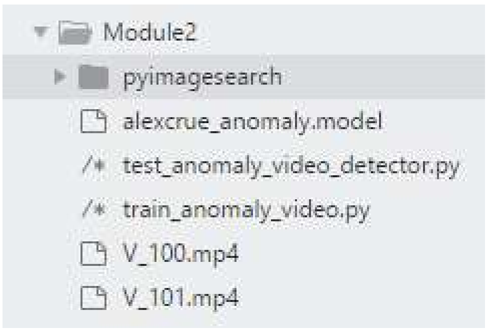


Figure 12: Overview of Module2

Train_anomaly_video.py:

```
# USAGE
# python test_anomaly_detector.py --model anomaly_detector.model --image
examples/highway_a836030.jpg

# import the necessary packages
from pyimagesearch.features import quantify_image
import argparse
import pickle
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-m", "--model", required=True,
    help="path to trained anomaly detection model")
ap.add_argument("-i", "--image", required=True,
    help="path to input image")
args = vars(ap.parse_args())

# load the anomaly detection model
print("[INFO] loading anomaly detection model...")
model = pickle.loads(open(args["model"], "rb").read())

# load the input image, convert it to the HSV color space, and
# quantify the image in the *same manner* as we did during training
image = cv2.imread(args["image"])
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
features = quantify_image(hsv, bins=(3, 3, 3))

# use the anomaly detector model and extracted features to determine
# if the example image is an anomaly or not
preds = model.predict([features])[0]
label = "anomaly" if preds == -1 else "normal"
color = (0, 0, 255) if preds == -1 else (0, 255, 0)

# draw the predicted label text on the original image
cv2.putText(image, label, (10,  25), cv2.FONT_HERSHEY_SIMPLEX,
    0.7, color, 2)
```

```
# display the image
cv2.imshow("Output", image)
cv2.waitKey(0)
```

Test_anomaly_video_detector.py:

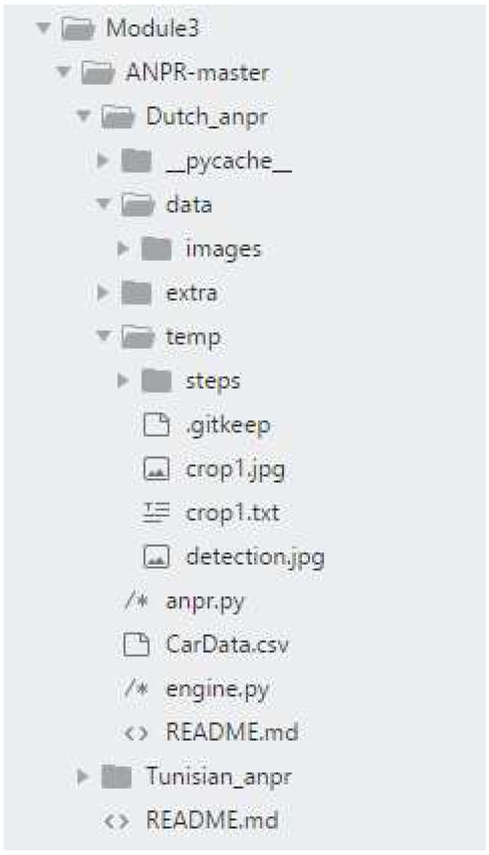
```
# USAGE
# python test_anomaly_detector.py --model anomaly_detector.model --image
examples/highway_a836030.jpg

# import the necessary packages
from pyimagesearch.features import quantify_image
import argparse
import pickle
import cv2

# construct the argument parser and parse the arguments
# ap = argparse.ArgumentParser()
# ap.add_argument("-m", "--model", required=True,
#   help="path to trained anomaly detection model")
# ap.add_argument("-i", "--image", required=True,
#   help="path to input image")
# args = vars(ap.parse_args())

# load the anomaly detection model

cam = cv2.VideoCapture('V_101.mp4')
print("[INFO] loading anomaly detection model...")
model = pickle.loads(open("alexcrue_anomaly.model", "rb").read())

# load the input image, convert it to the HSV color space, and
# quantify the image in the *same manner* as we did during training

# image = cv2.imread(args["image"])


while(1):
    ret, frame = cam.read()
    image = frame
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    features = quantify_image(hsv, bins=(3, 3, 3))



    preds = model.predict([features])[0]
    label = "anomaly" if preds == -1 else "normal"
    color = (0, 0, 255) if preds == -1 else (0, 255, 0)

    if cv2.waitKey(1) & 0xFF == ord('q'):
            break

# draw the predicted label text on the original image
    cv2.putText(image, label, (10,  25), cv2.FONT_HERSHEY_SIMPLEX,0.7, color, 2)

# display the image
```

```
    cv2.imshow("Output", image)
# cv2.waitKey(0)
cam.release()
cv2.destroyAllWindows()
```

features.py:

```
# import the necessary packages
from imutils import paths
import numpy as np
import cv2

def quantify_image(image, bins=(4, 6, 3)):
    # compute a 3D color histogram over the image and normalize it
    hist = cv2.calcHist([image], [0, 1, 2], None, bins,
            [0, 180, 0, 256, 0, 256])
    hist = cv2.normalize(hist, hist).flatten()

    # return the histogram
    return hist

def load_dataset(datasetPath, bins):
    # grab the paths to all images in our dataset directory, then
    # initialize our lists of images
    imagePaths = list(paths.list_images(datasetPath))
    data = []

    # loop over the image paths
    for imagePath in imagePaths:
            # load the image and convert it to the HSV color space
            image = cv2.imread(imagePath)
            image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

            # quantify the image and update the data list
            features = quantify_image(image, bins)
            data.append(features)

    # return our data list as a NumPy array
    return np.array(data)
```

Module 3: Vehicle Theft:



```
Module3
  ANPR-master
    Dutch_anpr
      __pycache__
      data
        images
      extra
      temp
        steps
        .gitkeep
        crop1.jpg
        crop1.txt
        detection.jpg
      anpr.py
      CarData.csv
      engine.py
      README.md
    Tunisian_anpr
    README.md
```

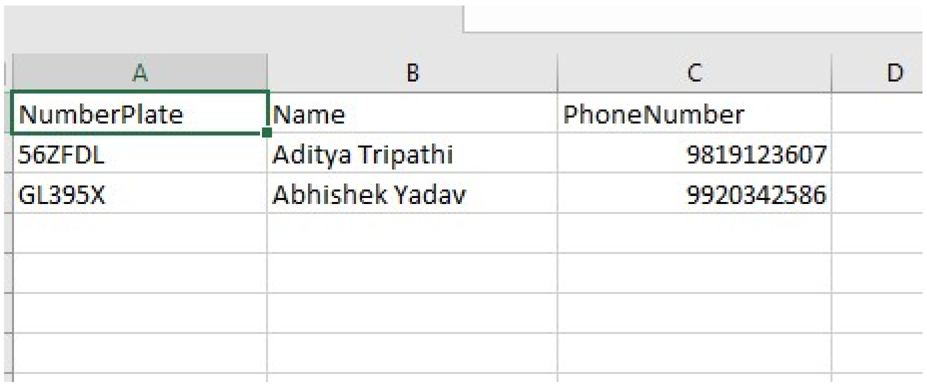
Figure 13: Overview of Module3

Anpr.py:

```
from engine import detect, process, recognise, detect_belg, post_process
import cv2
import pandas as pd
import numpy as np
import argparse
import os
import glob
import sys
import time
import requests
from xlrd import open_workbook
import csv
from csv import DictReader
url = "https://www.fast2sms.com/dev/bulk"

parser = argparse.ArgumentParser()
parser.add_argument('--i', '-image', help="Input image path", type= str)
parser.add_argument('--v', '-video', help="Input video path", type= str)


args = parser.parse_args()
abs_path = os.path.dirname(sys.executable)


if args.i:
    start = time.time()
```

```
try:
    os.mkdir('temp')
except:
    files = glob.glob('tmp')
    for f in files:
        os.remove(f)

input_image = cv2.imread(args.i)
detection, crops = detect(input_image)

i = 1
for crop in crops:

    crop = process(crop)

    cv2.imwrite('temp/crop' + str(i) + '.jpg', crop)
    recognise('temp/crop' + str(i) + '.jpg', 'temp/crop'+str(i))
    post_process('temp/crop' + str(i) + '.txt')



    i += 1
    details = r'C:\\Users\\laptop\\Desktop\\FinalYearProject\\Module3\\ANPR-
master\\Dutch_anpr\\CarData.csv'
    sheet1 = pd.read_csv(details)
    print(sheet1)

    with open("temp\crop1.txt", "r") as f:
        adi = f.readline().replace('\n',"")
        # word = adi.read.replace('\n',"")
        print(adi)


    # with open("CarData.csv", "rb") as f:
    #     csvreader = csv.reader(f, delimiter=",")
    #     for row in csvreader:
    #         if "56ZFDL" in row[0].values:
    #             print("56ZFDL spotted")

    with open('CarData.csv', 'r') as read_obj:
        csv_dict_reader = DictReader(read_obj)
        for row in csv_dict_reader:
            # print(adi)
            # print(row['NumberPlate'], row['Name'])
            if adi == row['NumberPlate'] :
                Flag = 1
                print(row['PhoneNumber'])
                break
                # print("\nThis value exists in Dataframe")

            else:
                # print("\nThis value does not exists in Dataframe")
                Flag = 0
        if Flag == 1:
            print("exists")
            payload = "sender_id=FSTSMS&message=Car Just Passed Gate No
1&language=english&route=p&numbers="+ row['PhoneNumber'] + ""
            headers = {
                        'authorization':
"FkGn92thyupYK3aDmTRoicl6MCH7VJEjgbqfAXUPz4ZB0OvrL1eoM062IUZDsicThzG3Ab
JKjO8a7Cd1",
                        'Content-Type': "application/x-www-form-urlencoded",
                        'Cache-Control': "no-cache",
                        }
            response = requests.request("POST", url, data=payload, headers=headers)
            print(response.text)

        else:
            print("Not Existts")

        # creating a Dataframe object
        # df = pd.DataFrame(details, columns = ['NumberPlate', 'Name', 'PhoneNumber'],index =
['0', '1'])
        # # print(df.NumberPlate)
        # # print("Dataframe: \n\n", df)

        # # chcek 'Ankit' exist in dataframe or not
        # if '56ZFDL' in df.values :
        #     print("\nThis value exists in Dataframe")

        # else :
        #     print("\nThis value does not exists in Dataframe")

    cv2.imwrite('temp/detection.jpg', detection)
    finish = time.time()
    print('Time processing >>>>>>  '+ str(finish-start))
elif args.v:
    cap = cv2.VideoCapture(0)

    # Check if camera opened successfully
    if cap.isOpened() == False:
        print("Error opening video stream or file")

    while (cap.isOpened()):
        # Capture frame-by-frame
        ret, frame = cap.read()
        if ret == True:

            frame, crop = detect(frame)
            # Display the resulting frame

            cv2.putText(frame, 'Press \'Q\' to exit !',(50, 50),cv2.FONT_HERSHEY_SIMPLEX,1,(0,
0, 255), 2)
            cv2.imshow('Frame', frame)

            # Press Q on keyboard to  exit
            if cv2.waitKey(25) & 0xFF == ord('q'):
                break

        # Break the loop
        else:
            break

    # When everything done, release the video capture object
    cap.release()

    # Closes all the frames
    cv2.destroyAllWindows()

else:
     print("--i : input image file path\n--v : input video file path")



# def runCode():
#     payload = "sender_id=FSTSMS&message=Car Number
mesg&language=english&route=p&numbers=9819123607"
#     headers = {
#               'authorization':
"FkGn92thyupYK3aDmTRoicl6MCH7VJEjgbqfAXUPz4ZB0OvrL1eoM062IUZDsicThzG3Ab
JKjO8a7Cd1",
#               'Content-Type': "application/x-www-form-urlencoded",
#               'Cache-Control': "no-cache",
#               }
#     response = requests.request("POST", url, data=payload, headers=headers)
#     print(response.text)
```

Data for Messaging (Verified Users):



A	B	C	D
NumberPlate	Name	PhoneNumber	
56ZFDL	Aditya Tripathi	9819123607	
GL395X	Abhishek Yadav	9920342586	

Figure 14: Verified Users

Engine.py:

```
import cv2
import numpy as np
import os
import pathlib
```

```
def order_points(pts):
    # initialzie a list of coordinates that will be ordered
    # such that the first entry in the list is the top-left,
    # the second entry is the top-right, the third is the
    # bottom-right, and the fourth is the bottom-left
    rect = np.zeros((4, 2), dtype="float32")

    # the top-left point will have the smallest sum, whereas
    # the bottom-right point will have the largest sum
    s = pts.sum(axis=1)
    rect[0] = pts[np.argmin(s)]
    rect[2] = pts[np.argmax(s)]

    # now, compute the difference between the points, the
    # top-right point will have the smallest difference,
    # whereas the bottom-left will have the largest difference
    diff = np.diff(pts, axis=1)
    rect[1] = pts[np.argmin(diff)]
    rect[3] = pts[np.argmax(diff)]

    # return the ordered coordinates
    return rect


def four_point_transform(image, pts):
    # obtain a consistent order of the points and unpack them
    # individually
    rect = order_points(pts)
    (tl, tr, br, bl) = rect

    # compute the width of the new image, which will be the
    # maximum distance between bottom-right and bottom-left
    # x-coordiates or the top-right and top-left x-coordinates
    widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
    widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
    maxWidth = max(int(widthA), int(widthB))

    # compute the height of the new image, which will be the
    # maximum distance between the top-right and bottom-right
    # y-coordinates or the top-left and bottom-left y-coordinates
    heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
    heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))

    # now that we have the dimensions of the new image, construct
    # the set of destination points to obtain a "birds eye view",
    # (i.e. top-down view) of the image, again specifying points
    # in the top-left, top-right, bottom-right, and bottom-left
    # order
    dst = np.array([
        [0, 0],
        [maxWidth - 1, 0],
        [maxWidth - 1, maxHeight - 1],
        [0, maxHeight - 1]], dtype="float32")

    # compute the perspective transform matrix and then apply it
    M = cv2.getPerspectiveTransform(rect, dst)
    warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))

    # return the warped image
    return warped


def automatic_brightness_and_contrast(image, clip_hist_percent=10):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Calculate grayscale histogram
    hist = cv2.calcHist([gray],[0],None,[256],[0,256])
    hist_size = len(hist)

    # Calculate cumulative distribution from the histogram
    accumulator = []
    accumulator.append(float(hist[0]))
    for index in range(1, hist_size):
        accumulator.append(accumulator[index -1] + float(hist[index]))

    # Locate points to clip
    maximum = accumulator[-1]
    clip_hist_percent *= (maximum/100.0)
    clip_hist_percent /= 2.0

    # Locate left cut
    minimum_gray = 0
    while accumulator[minimum_gray] < clip_hist_percent:
        minimum_gray += 1

    # Locate right cut
    maximum_gray = hist_size -1
    while accumulator[maximum_gray] >= (maximum - clip_hist_percent):
        maximum_gray -= 1

    # Calculate alpha and beta values
    alpha = 255 / (maximum_gray - minimum_gray)
    beta = -minimum_gray * alpha

    auto_result = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
    return auto_result, alpha, beta


def detect(img_rgb):

    img = img_rgb.copy()
    input_height = img_rgb.shape[0]
    input_width = img_rgb.shape[1]
    hsv_frame = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2HSV)
    # yellow color
    low_yellow = np.array([20, 100, 100])
    high_yellow = np.array([30, 255, 255])
    yellow_mask = cv2.inRange(hsv_frame, low_yellow, high_yellow)
    yellow = cv2.bitwise_and(yellow_mask, yellow_mask, mask=yellow_mask)

    cv2.imwrite("temp/steps/1_yellow_color_detection.png", yellow)
    # Close morph
    k = np.ones((5, 5), np.uint8)
    closing = cv2.morphologyEx(yellow, cv2.MORPH_CLOSE, k)

    cv2.imwrite("temp/steps/2_closing_morphology.png", closing)
    # Detect yellow area
    contours, hierarchy = cv2.findContours(closing, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    # List of final crops
    crops = []

    # Loop over contours and find license plates
    for cnt in contours:
        x, y, w, h = cv2.boundingRect(cnt)

        # Conditions on crops dimensions and area
        if h*6 > w > 2 * h and h > 0.1 * w and w * h > input_height * input_width * 0.0001:

            # Make a crop from the RGB image, the crop is slided a bit at left to detect bleu area
            crop_img = img_rgb[y:y + h, x-round(w/10):x]
            crop_img = crop_img.astype('uint8')

            # Compute bleu color density at the left of the crop
            # Bleu color condition
            try:
                hsv_frame = cv2.cvtColor(crop_img, cv2.COLOR_BGR2HSV)
                low_bleu = np.array([100,150,0])
                high_bleu = np.array([140,255,255])
                bleu_mask = cv2.inRange(hsv_frame, low_bleu, high_bleu)
                bleu_summation = bleu_mask.sum()

            except:
                bleu_summation = 0

            # Condition on bleu color density at the left of the crop
            if bleu_summation > 550:

                # Compute yellow color density in the crop
                # Make a crop from the RGB image
                imgray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)
                crop_img_yellow = img_rgb[y:y + h, x:x+w]
                crop_img_yellow = crop_img_yellow.astype('uint8')

                # Detect yellow color
                hsv_frame = cv2.cvtColor(crop_img_yellow, cv2.COLOR_BGR2HSV)
                low_yellow = np.array([20, 100, 100])
                high_yellow = np.array([30, 255, 255])
                yellow_mask = cv2.inRange(hsv_frame, low_yellow, high_yellow)

                # Compute yellow density
                yellow_summation = yellow_mask.sum()

                # Condition on yellow color density in the crop
                if yellow_summation > 255*crop_img.shape[0]*crop_img.shape[0]*0.4:

                    # Make a crop from the gray image
                    crop_gray = imgray[y:y + h, x:x + w]
                    crop_gray = crop_gray.astype('uint8')

                    # Detect chars inside yellow crop with specefic dimension and area
                    th = cv2.adaptiveThreshold(crop_gray, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
                    contours2, hierarchy = cv2.findContours(th, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

                    # Init number of chars
                    chars = 0
                    for c in contours2:
                        area2 = cv2.contourArea(c)
                        x2, y2, w2, h2 = cv2.boundingRect(c)
                        if w2 * h2 > h * w * 0.01 and h2 > w2 and area2 < h * w * 0.9:
                            chars += 1

                    # Condition on the number of chars
                    if 20 > chars > 4:
                        rect = cv2.minAreaRect(cnt)
                        box = cv2.boxPoints(rect)
                        box = np.int0(box)
                        pts = np.array(box)
                        warped = four_point_transform(img, pts)
                        crops.append(warped)

                        # Using cv2.putText() method
                        img_rgb = cv2.putText(img_rgb, 'LP', (x, y),
cv2.FONT_HERSHEY_SIMPLEX,1, (0, 0, 255), 2, cv2.LINE_AA)

                        cv2.drawContours(img_rgb, [box], 0, (0, 0, 255), 2)

    return img_rgb, crops

def detect_belg(src):
    img, alpha, beta = automatic_brightness_and_contrast(src)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, th = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)

    contours, hierarchy = cv2.findContours(th, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```

```
    crops = []
    for cnt in contours:
        x, y, w, h = cv2.boundingRect(cnt)

        if h * 6 > w > 2 * h and h > 0.1 * w and w * h > img.shape[0] * img.shape[1] * 0.0001:
            crop = th[y:y + h, x:x + w]

            # Compute sum of white pixels
            white_summation = crop.sum()
            if white_summation > w * h * 0.4 * 255:
                # Compute sum of red pixel
                crop = img[y:y + h, x:x + w]
                crop_img = crop.astype('uint8')
                hsv = cv2.cvtColor(crop_img, cv2.COLOR_BGR2HSV)
                lower_red = np.array([160, 100, 100])
                upper_red = np.array([179, 255, 255])
                red_mask = cv2.inRange(hsv, lower_red, upper_red)
                red_summation = red_mask.sum()

                if red_summation > 510:
                    crop_img = img[y:y + h, x - round(w / 10):x + w]
                    crop_img = crop_img.astype('uint8')
                    hsv = cv2.cvtColor(crop_img, cv2.COLOR_BGR2HSV)
                    low_bleu = np.array([100, 150, 0])
                    high_bleu = np.array([140, 255, 255])
                    bleu_mask = cv2.inRange(hsv, low_bleu, high_bleu)
                    bleu_summation = bleu_mask.sum()

                    if bleu_summation > 255:

                        crop = gray[y:y + h, x:x + w]
                        crop_img = crop.astype('uint8')
                        th2 = cv2.adaptiveThreshold(crop_img, 255,
  cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,
                                      11, 2)

                        contours2, hierarchy = cv2.findContours(th2, cv2.RETR_TREE,
  cv2.CHAIN_APPROX_SIMPLE)
                        j = 0
                        for c in contours2:
                            area2 = cv2.contourArea(c)
                            x2, y2, w2, h2 = cv2.boundingRect(c)
                            if w2 * h2 > h * w * 0.01 and h2 > w2 and area2 < h * w * 0.9:
                                j += 1

                        if 12 > j > 4:
                            rect = cv2.minAreaRect(cnt)
                            box = cv2.boxPoints(rect)
                            box = np.int0(box)
                            pts = np.array(box)
                            warped = four_point_transform(src, pts)
                            crops.append(warped)
                cv2.drawContours(src, [box], 0, (0, 255, 0), 2)
    return src, crops

def process(src):

    # Brigthness and contrast adjustment
    cv2.imwrite("temp/steps/3_detected_plate.png", src)
    adjusted, a, b = automatic_brightness_and_contrast(src)
    cv2.imwrite("temp/steps/4_Brigthness_contrast_adjustment.png", adjusted)
    # BGR to gray
    gray = cv2.cvtColor(adjusted, cv2.COLOR_BGR2GRAY)
    cv2.imwrite("temp/steps/5_gray.png", gray)
    # Binary thresh
    #ret, th = cv2.threshold(gray, 140, 255, cv2.THRESH_BINARY)
    ret, th = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    cv2.imwrite("temp/steps/6_threshold.png", th)
    return th

def post_process(input_file_path):
    f = open(input_file_path, "r")
    text = f.readline()
    puncts = [',', '.', '"', ':', ')', '(', '-', '!', '?', '|', ';', "'", '$', '&', '/', '[', ']', '>', '%', '=',
            '#', '*', '+', '\\', '•', '~', '@', '£',
            '·', '_', '{', '}', '©', '^', '®', '`', '<', '→', '°', '€', '™', '›', '♥', '←', '×', '§', '″', '′',
            'Â', '█', '½', '…',
            '“', '★', '”', '–', '●', '►', '−', '¢', '²', '¬', '░', '¶', '↑', '±', '¿', '▾', '═', '¦', '║', '―',
            '¥', '▓', '—', '‹', '─',
            '▒', '：', '¼', '⊕', '▼', '▪', '†', '■', '’', '▀', '¨', '▄', '♫', '☆', '¯', '♦', '¤', '▲', '¸', '¾',
            'Ã', '⋅', '‘', '∞',
            '∙', '）', '↓', '、', '│', '（', '»', '，', '♪', '╩', '╚', '³', '・', '╦', '╣', '╔', '╗', '▬', '♡', 'Ø',
            '¹', '≤', '‡', '√', '«', ' ']

    for punct in puncts:
        if punct in text:
            text = text.replace(punct, '')

    f.close()
    f = open(input_file_path, "w")
    f.write(text)
    f.close()
    return text

def recognise(src_path, out_path):

    # Tesseract command for recognition
    cmd = str(pathlib.Path().absolute()) + '/extra/tesseract.exe '+ src_path + ' ' + out_path + ' -l eng
--psm 6 --dpi 300 --oem 1'
    os.system(cmd)

    return 0
```

CHAPTER 5

CONCLUSION:

In this research, we encountered several face recognition algorithms along the way. Some were really accurate while some were really slow. Our research found that Striking a balance between the Accuracy & Speed is really difficult. Some of the algorithms we came across through means of this research are HAAR , Eigen Faces, Cam shift , CNNs, Viola-Jones Algorithm, Gaussian , Eucledian distance, AdaBoost etc.

The most famous detection algorithm we came across through this research was HAAR. We also came across and learned about the Microsoft Azure, which bundle of the cloud computing services that are created by Microsoft which can be used for building, testing, as well as deploying, and managing the applications and services using a global network of the data centres managed by Microsoft , which has the various Face API services which is basically cloud based service which provides algorithms for analysing the human faces found in images and videos. It is fast and accurate and thus our survey and the research helped us choose & use of Microsoft Azure to take care of our face recognition need.

The above proposed approach will be implemented in the next phase of this project which will help us superior verify the performance and high accuracy of this approach with very low latency. Accuracy is of paramount importance for a project of this sensitivity. Also important is the ability to run on any machine regardless of the how powerful the hardware is. These both lead us to choose the Azure face API. The proposed methodology when implemented successfully could definitely be of great help to our Criminal agencies in detecting and finding of criminals, also search for missing people.

REFERENCES:

1. S.Shirsat, A.Naik, D.Tamse, J.Yadav, P. Shetgaonkar, S.Aswale, "Proposed System for Criminal Detection and Recognition on CCTV Data Using Cloud and Machine Learning", 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), **ISBN:** 978-1-5386-9353-7,
2. DOI: 10.1109/ViTECoN.2019.8899441.
3. AMR.Abdali, RFA.Tuma, "Robust Real-Time Violence Detection in Video Using CNN And LSTM", 2019 2nd Scientific Conference of Computer Sciences (SCCS), University of Technology - Iraq, ISBN: 978-1-7281-0761-5,
4. DOI: 10.1109/SCCS.2019.8852616
5. S. Ali, SA. Alvi, Atiq Ur Rehman, "The Usual Suspects: Machine Learning Based Predictive Policing for Criminal Identification", 2019 13th International Conference on Open Source Systems and Technologies (ICOSST), ISBN: 978-1- 7281-4613-3, DOI: 10.1109/ICOSST48232.2019.9043925.
6. J.Li, X.Jiang, T.Sun, K.Xu, "Efficient Violence Detection Using 3D Convolutional Neural Networks", 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), ISBN: 978-1-7281-0990-9, DOI: 10.1109/AVSS.2019.8909.
7. M.Ramzan, A. Abid, H. Khan, S. Awan, A. Ismail, M. Ahmed, M. Ilyas, A. Mahmood, "A Review on state-of-the-art Violence Detection Techniques", DOI 10.1109/ACCESS.2019.2932114, IEEE Access.
8. L.Elluri, V.Mandalapu, N.Roy, "Developing Machine Learning based Predictive Models for Smart Policing", 2019 IEEE International Conference on Smart Computing (SMARTCOMP), ISBN: 978-1-7281-1689-1, DOI: 10.1109/SMARTCOMP.2019.00053.
9. M.Soliman, M.Kamal, M.Nashed, Y.Mostafa, B.Chawky, D.Khattab, "Violence Recognition from Videos using Deep
10. Learning Techniques", 2019 IEEE Ninth International Conference on Intelligent Computing and Information Systems (ICICIS),
11. ISBN: 978-1-7281-3995-1, DOI: 10.1109/ICICIS46948.2019.9014714.
12. S.Chackravarthy, S.Schmitt, Li.Yang," Intelligent Crime Anomaly Detection in Smart Cities using Deep Learning", 2018 IEEE 4th International Conference on Collaboration and Internet Computing, ISBN: 978-1-5386-9502-9, DOI: 10.1109/CIC.2018.00060.
13. A.Mumtaz, Ab.Sargano, Z.Habib, "Violence Detection in Surveillance Videos with Deep Network using Transfer Learning", 2018 2nd European Conference on Electrical Engineering and Computer Science (EECS), **ISBN:** 978-1-7281-1929-8, DOI: 10.1109/EECS.2018.00109.
14. U. Navalgund, Priyadarshini.K, "Crime Intention Detection System Using Deep Learning", 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), ISBN: 978-1-5386-0576-9, DOI: 10.1109/ICCSDET.2018.8821168.
15. K.Rasanayagam, K.S.D.D.C, W.A.D.D, S.N.T, Dr.P.Samarashinge,
16. S.E.R.Siriwardana, "CIS: An Automated Criminal Identification System", 2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS), ISBN: 978-1-5386-9418-3, DOI: 10.1109/ICIAFS.2018.8913367.
17. X. Chen, X. Yang, M. Wang,Ji. Zou, "Convolution Neural Network for Automatic Facial Expression Recognition", 2017 International Conference on Applied System Innovation (ICASI), ISBN: 978-1-5090-4897-7, DOI: 10.1109/ICASI.2017.7988558.
18. H. Park, E. Kwon, ES.Jung, H-W.Lee, Y-T.Lee," Heterogeneous Data based Danger

Detection for Public Safety" ,2018 IEEE International Conference on Consumer Electronics (ICCE), ISBN: 978-1-5386-3025-9, DOI: 10.1109/ICCE.2018.8326323.

19. M.Nakib, Md.Hassan, J.Uddin, RT.Khan, "Crime Scene Prediction by Detecting Threatening Objects Using Convolutional Neural Network", 2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2), ISBN: 978-1-5386-4775-2, DOI: 10.1109/IC4ME2.2018.84655
20. https://leadingindia.ai/downloads/projects/CM/cm_1.pdf
21. https://iopscience.iop.org/article/10.1088/1742-6596/1000/1/012046/pdf