Intelligent Framework for Crime Detection and Prediction using Behavioural Tracking and Motion Analysis

submitted in partial fulfillment of the requirement
for the award of the Degree of

Bachelor of Technology
in
Information Technology

by

Rajat Shenoy
Deepak Yadav
Harshita Lakhotiya

under the guidance of

Prof. Jignesh Sisodia

Department of Information Technology
Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
(Autonomous Institute Affiliated to University of Mumbai)
Munshi Nagar, Andheri-West, Mumbai-400058
University of Mumbai
2021-2022

Approval Certificate

This is to certify that the Project entitled "Intelligent Framework for Crime Detection and Prediction using Behavioural Tracking and Motion Analysis" by Rajat Shenoy, Harshita Lakhotiya and Deepak Yadav is approved for the partial fulfillment of Major Project - I for the Final Year Project towards obtaining the Degree of Bachelor of Technology in Information Technology from University of Mumbai.

Project Guide

(signature)

Name:

Date:

External Examiner

(signature)

Name:

Date:

Head of Department

Principal

Seal of the Institute

Declaration

I wish to state that the work embodied in this work titled "Intelligent Framework for Crime Detection and Prediction using Behavioural Tracking and Motion Analysis" forms my own contribution to the work carried out under the guidance of Prof. Jignesh Sisodia at the Sardar Patel Institute of Technology. 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 principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

Rajat Shenoy	**Deepak Yadav**	**Harshita Lakhotiya**
2018140056	**2018140064**	**2018140034**

Abstract

Closed Circuit Television Systems are now being deployed in most public spaces to make the city more secure. Manual observation of these clips for the prevention of crime would take up a lot of manpower. This paper proposes an Intelligent Framework using the power of Artificial Intelligence to ensure the safety of the surroundings. The system will use different Computer Vision techniques for video analysis. It will monitor CCTV footage for any criminal offenders, violent objects, and suspicious behavior which could lead to crime. SSD MobileNet Model, an architecture for concealed object detection, is trained for labeling weapons in the frame. The images captured are processed using Face Detection algorithms to identify human faces. Facial Recognition API using libraries in python is implemented to recognize the offenders from criminal records. A ResNet-GRU Model was trained for human behavior analysis which detects suspicious actions. An alert is generated when there are signs of crime and concerned authorities are notified. The proposed framework aims to make societies secure by correctly identifying criminals and crime-related objects.

List of Figures

1 Class Diagram . . . 13
2 Data Flow Diagram 0 . . . 14
3 Data Flow Diagram 1 . . . 15
4 Use Case Diagram . . . 16
5 Sequence Diagram . . . 17
6 Block Diagram . . . 18
7 Face Detection . . . 19
8 Code for face Detection . . . 20
9 Code for Embeddings . . . 21
10 Creation of Embeddings . . . 22
11 Face Recognition Code . . . 23
12 Face Recognition Code . . . 24
13 Object Detection Code . . . 25
14 Object Detection Code . . . 26
15 SSD Mobile Net Architecture . . . 27
16 CAVIAR Dataset . . . 29
17 XML Data . . . 30
18 Frames to be processed . . . 31
19 OOP Representation of the Dataset . . . 32
20 Loading and preparing the dataset . . . 33
21 Extract Boxes position from the data . . . 34
22 Extract Data Code . . . 34
23 The Driver Code . . . 35
24 Labeling Suspicious Behaviour in Data . . . 36
25 Time Distributed Layer . . . 37
26 Gated Resnet Unit . . . 38
27 Long Short Term Memory Cell . . . 39
28 Recurrent Neural Network . . . 40
29 Data Creation Code using XML Files . . . 41
30 Load and Clean Data . . . 42
31 Building Architecture of the Model using Keras . . . 43
32 Architecture of our Model . . . 44
33 Training our model . . . 45
34 Multiple Process Running Simultaneously . . . 46
35 Multiprocessing Code in Python . . . 47
36 Mailing Code in Python . . . 47
37 Testing and Running our model against actual test cases . . . 48
38 Displaying suspicious and non suspicious boxes . . . 49
39 Face Recognition model Recognizing the name from the embeddings . . . 50
40 Face Recognition model Recognizing the name from the embeddings . . . 51
41 Face Recognition model Recognizing the name from the embeddings . . . 52
42 SSD Mobilenet Model detecting all the objects . . . 53

43 Suspicious behaviour detected by our model 54
44 Non Suspicious behviour detected by our model 55
45 Model Accuracy 55
46 Multiple models running parallely using multiprocessing 56
47 Mail Sent to the authorities 56

List of Tables

1 Literature Survey 12

Contents

1 Introduction **8**
1.1 Problem Statement . . . 8
1.2 Objectives . . . 8
1.3 Scope . . . 8
1.4 Technologies Used . . . 9
1.5 Assumptions And Constraints . . . 9

2 Literature Survey **10**

3 Analysis **13**
3.1 Class Diagram . . . 13
3.2 Data Flow Diagrams . . . 14
3.3 Use Case Diagram . . . 16
3.4 Sequence Diagram . . . 17

4 Design and Methodology **18**
4.1 Face Detection and Recognition . . . 19
4.2 Concealed Object Detection . . . 25
4.3 Suspicious Behaviour Prediction . . . 28
4.4 Integrated System For Crime Detection . . . 46

5 Results and Discussion **50**
5.1 Face Recognition . . . 50
5.2 Object Recognition . . . 53
5.3 Suspicious Behaviour Prediction . . . 54
5.4 Identifying Crime using Multiprocessing . . . 56

6 Conclusion and Further Work **57**
6.1 Conclusion . . . 57
6.2 Future Work . . . 57

7 Research Paper **58**

8 Plagiarism Report **64**

1 Introduction

1.1 Problem Statement

In order to protect society from crime, a system for real-time crime prevention and detection is required.
Develop a system considering following points:-

1. It should be able to compute frames and extract features from the video captured.
2. Detect suspicious behaviour and objects for crime prevention.
3. Identify crime incidents and criminals.
4. General alerts and notify concerned authorities.

1.2 Objectives

- To develop a framework for crime prediction and detection which works irrespective of the conditions.
- To explore all the possibilities of using deep learning approach to develop an efficient system.
- To identify criminal activities and offenders.
- To generate alerts and notify concerned authorities in real time to ensure safety of common people.

1.3 Scope

- Effectively utilize power of Computer Vision to prevent and detect crime in real-time.
- Implement different behaviour tracking techniques to identify suspicious behaviour and predict crimes.
- Use Intelligent Motion Detection to detect and analyze different crime scenarios.
- Detect concealed object which can be potential crime weapons.
- Implement techniques for facial recognition to identify criminals.

1.4 Technologies Used

Algorithms	Technology Used	Datasets
Face Detection and Facial Recognition Algorithms **Use: Face-Recognition Library**	OpenCV	Inbuilt database of Face Recognition Library
Object Detection Algorithms and Concealed Object Detection Algorithms **Use: SSD MobileNet**	OpenCV, PyTorch	Dasci Weapon Dataset
Suspicious Behaviour Detection Algorithms **Use: Resnet**	Tensorflow	CAVIAR Dataset, UCF Crime Dataset, KTH, HMDB

Datasets:

- UCF Crime Dataset : Consists of surveillance videos which cover 13 different anomalies: Vandalism, Stealing, Shoplifting, Shooting, Robbery, Road Accident, Abuse, Arrest, Arson, Burglary, Explosion, Fighting and Assault
- KTH Dataset: It is an action database containing six different types of human actions like running, clapping hands and waving, boxing ,running, walking.
- CAVIAR Dataset: These include video clips of people window shopping, walking alone meeting, entering inside shops and leaving shops, fighting ,passing and falling down and leaving a package or bag in a public place
- HMDB : These include video clips of General facial expressions and facial expressions with objects, generic body movements, body movement with object or human interaction (dangerous activities as well).

1.5 Assumptions And Constraints

- We have used demo input video for Suspicious Behaviour Detection
- The distance between the person and web cam should be considerably less for face detection and weapon detection

2 Literature Survey

No.	Paper Title	Work Done	Observations/Findings
1	Deep Learning Approach for Suspicious Activity Detection from Surveillance Video	Analysis of Human Behaviour,Preprocessing of input video frames, Train CNN Models	This technique can be used to predict crime before it happens. It will be a precautionary measure for safer surroundings.
2	Behavior tracking model in dynamic situation using the risk ratio EM	Object Assessment, Situation Assessment, Risk ratio based ML Model	This can help us design a model considering external factors like weather, time information, location, etc
3	A Framework for Crime Prediction and Analysis in Real-Time for Smart Home	Face Recognition to identify criminals, Object Detection, Event-driven approach	We can analyse video frames to detect objects like guns, knives, etc. At the same time, detect faces and use them to identify criminals if any.
4	i-SURVEILLANCE CRIME MONITORING AND PREVENTION USING NEURAL NETWORKS	Motion Detection to track moving objects, Motion analysis using HMDB database.	Input video frames can be used to detect violent actions like hitting, kicking, mob lynching. HMDB database can be efficiently used for the same.
5	AI Based Automatic Robbery/Theft Detection using Smart Surveillance in Banks	Multimedia analysis, Image Processing, Motion detection, Faster R-CNN, Sliding Window	Faster R-CNN is a good method for motion analysis for theft detection. One region is RPN which will produce regions and the second region will detect objects in this region.
6	Crime forecasting: A Machine Learning and Computer Vision approach to Crime Prediction and Prevention	Face Recognition to identify criminals, Object Detection, Event-driven approach	We can analyse video frames to detect objects like gun, knife, etc. At the same detect faces and use them to identify criminals if any.
7	Automated Detection of Firearms and Knives in a CCTV Image	Edge Detection, Feature analysis, Pattern Recognition, Fuzzy Classifier	Sliding window mechanism can be used to find the potential crime objects in the frame. Canny edge detection algorithm can be used for filtering and detecting edges which can be used for further analysis

8	Crime forecasting: a machine learning and computer vision approach to crime prediction and prevention	Comparative analysis of different methods, Machine Learning, Computer Vision	Marker-less vision-based human motion analysis has the potential to provide a non-obtrusive solution for the evaluation of body poses. Different supervised and unsupervised methods can be used for training purposes.
9	Real-time gun detection in CCTV: An open problem	Object detection, CNN, Deep Learning, Data Augmentation	Faster R-CNN with FPN is used for small object detection. Model developed has balanced precision and inference speed. The architecture had resnet backbone.
10	An application of a deep learning algorithm for automatic detection of unexpected accidents under bad CCTV monitoring conditions in tunnels	Object tracking algorithm, tunnel accident detection in insufficient light, Faster R-CNN for object detection	Faster R-CNN algorithm was used for accurate object detection. SORT algorithm is used for object tracking. To avoid false detection we can label objects before itself for e.g. - In Tunnel accident detection, tunnel light etc are labelled as false indications.
11	AUTOMATED CRIMINAL IDENTIFICATION BY FACE RECOGNITION USING OPEN COMPUTER VISION CLASSIFIERS	Automated Surveillance, Face detection, Face recognition, Comparison based results	Recognize criminal person in the frame by using face detection and later comparing the face with images in the database. The Haar Cascade algorithm can be used for face detection for getting high accuracy.
12	A Motion Detection-Based Framework for Improving Image Quality of CCTV Security Systems	Improve quality of CCTV footage for further analysis, Camera position translation analysis, Zoom controlling	Video from two cctv cameras is used to estimate position transition analysis and find superimposed region. With the help of object length and zooming time the resolution of the image is improved.
13	Spatio-Temporal Crime HotSpot Detection and Prediction: A Systematic Literature Review	A systematic literature survey was conducted to study relation between space time and crime numbers. Predictions systems were designed	We can include time series data and analysis techniques to further improve crime prediction systems. We can also include spatial data.

14	Initial evidence on the relationship between the coronavirus pandemic and crime in the United States	SARIMA model was used to forecast the crime numbers in 2020 in the future where there was no pandemic.	Comparisons were made between predictions from the model and the actual crime numbers during the onset of the pandemic

Table 1: Literature Survey

3 Analysis

3.1 Class Diagram

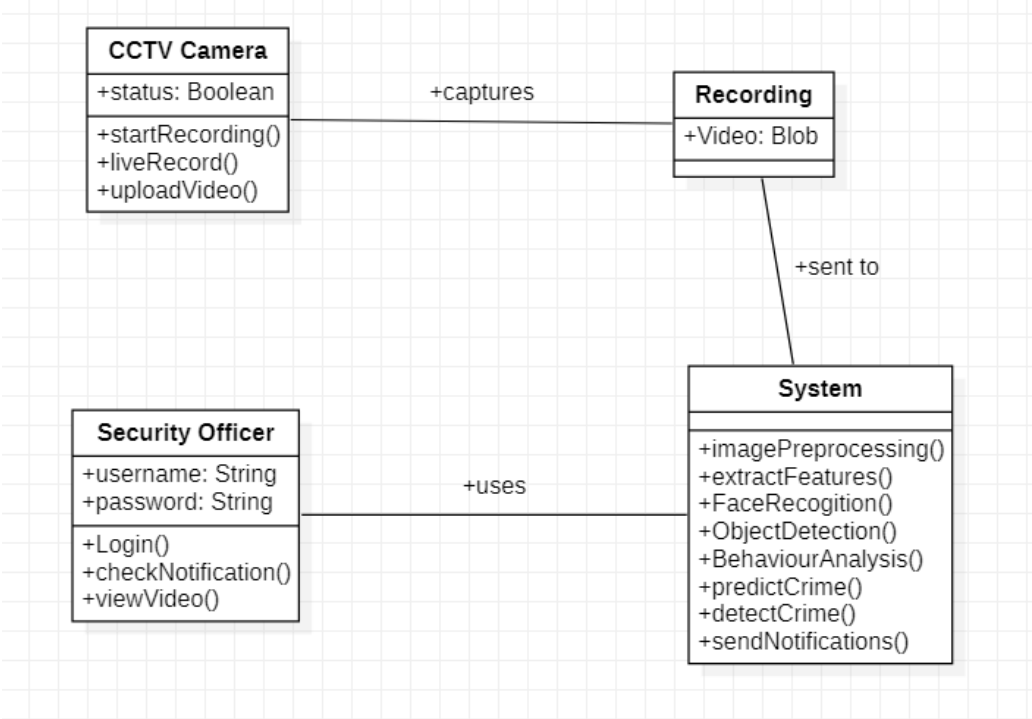


Figure 1: Class Diagram

3.2 Data Flow Diagrams

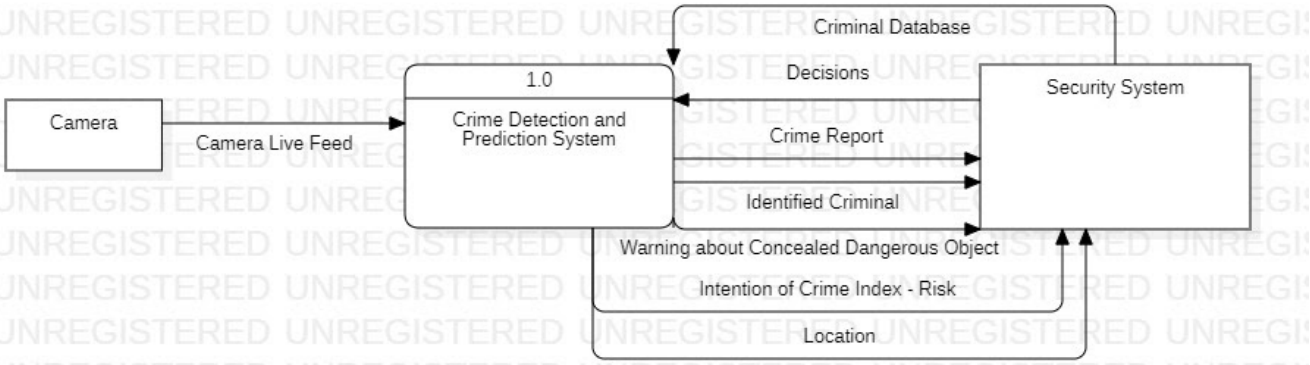


Figure 2: Data Flow Diagram 0

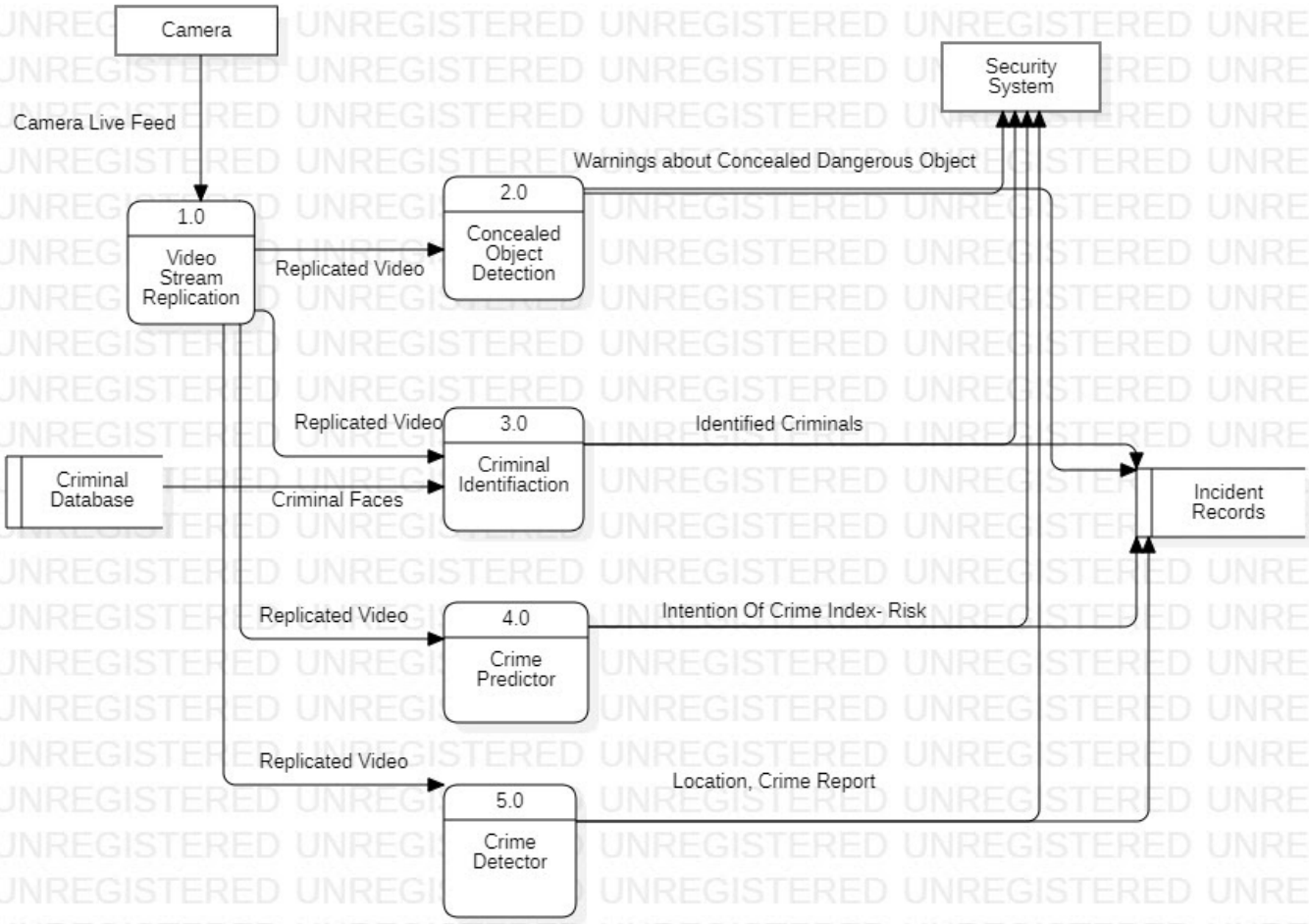


Figure 3: Data Flow Diagram 1

3.3 Use Case Diagram

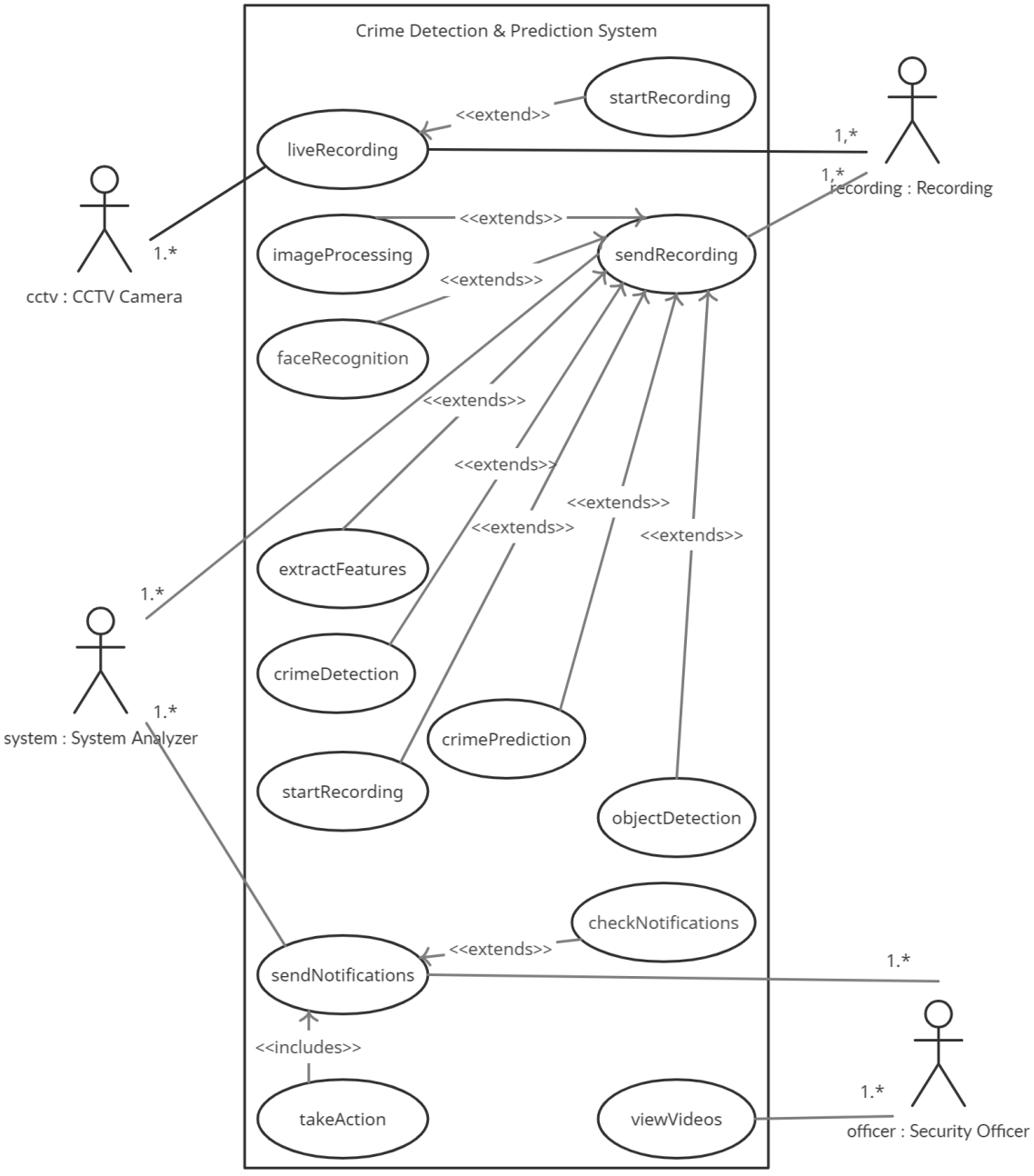


Figure 4: Use Case Diagram

3.4 Sequence Diagram

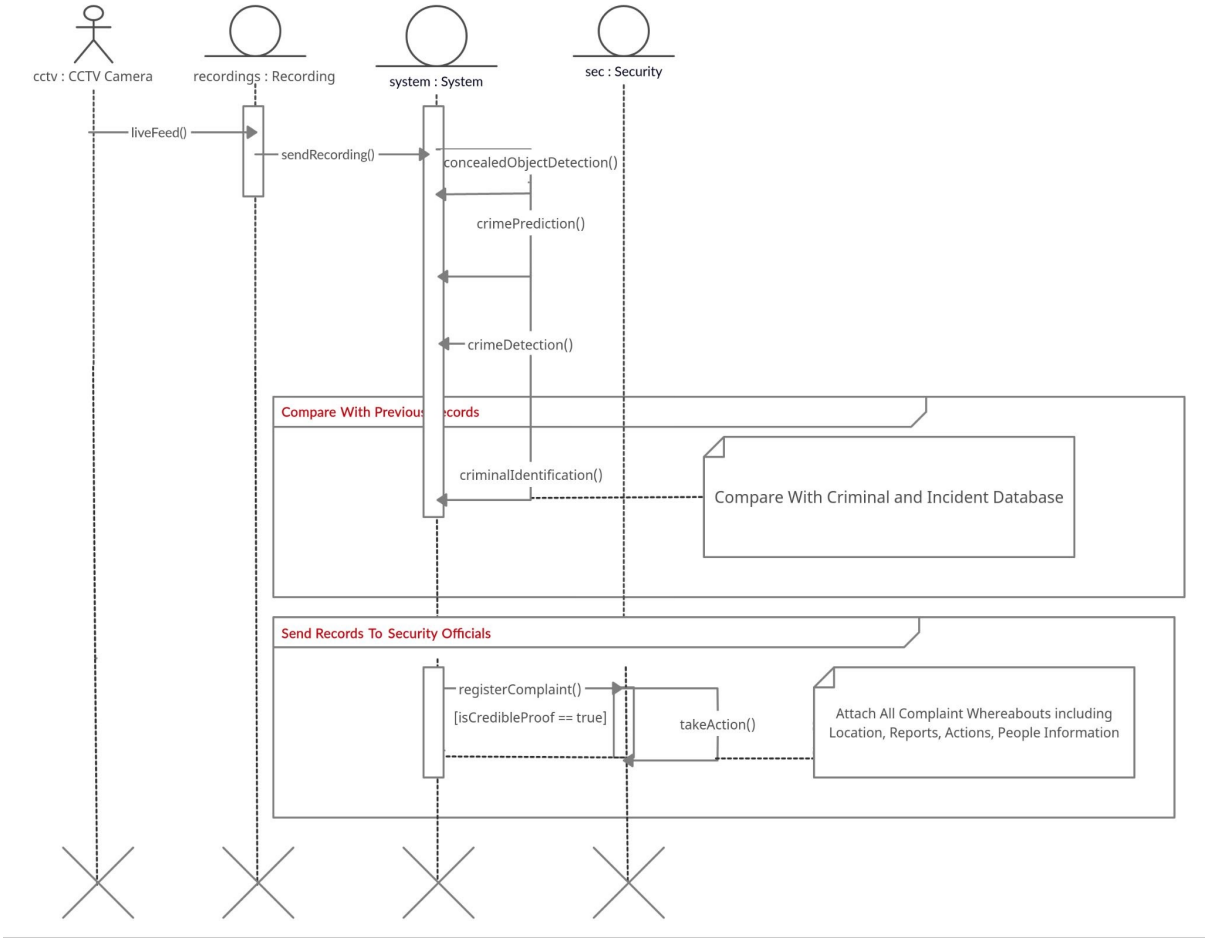


Figure 5: Sequence Diagram

4 Design and Methodology

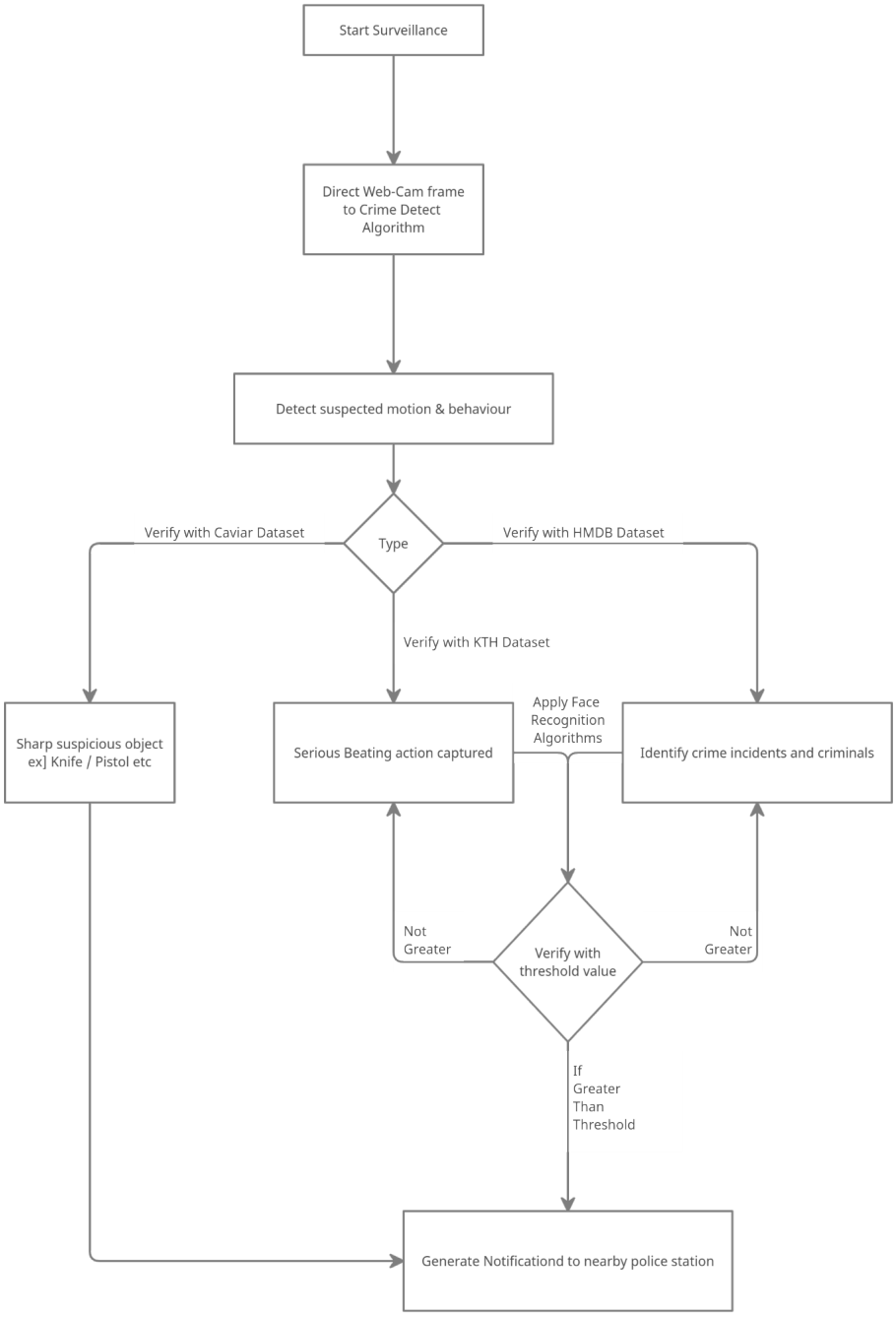


Figure 6: Block Diagram

4.1 Face Detection and Recognition

- Face Detection was implemented using Haar-Cascade Classifier
- All the faces in the snapshot are detected and cropped part is fed for recognition
- Samples were collected for comparison/testing purposes using webcam in openCV
- Embedding for all the known faces is stored in a file for further analysis
- Face recognition library in python was used for Recognising known/unknown faces
- The continuous frames are recorded from live feed using webcam of laptop and detected faces are embedded which are then compared with embeddings of all known faces
- If face is ones amongst the known faces of the system, it will recognise the person else the face will be unknown

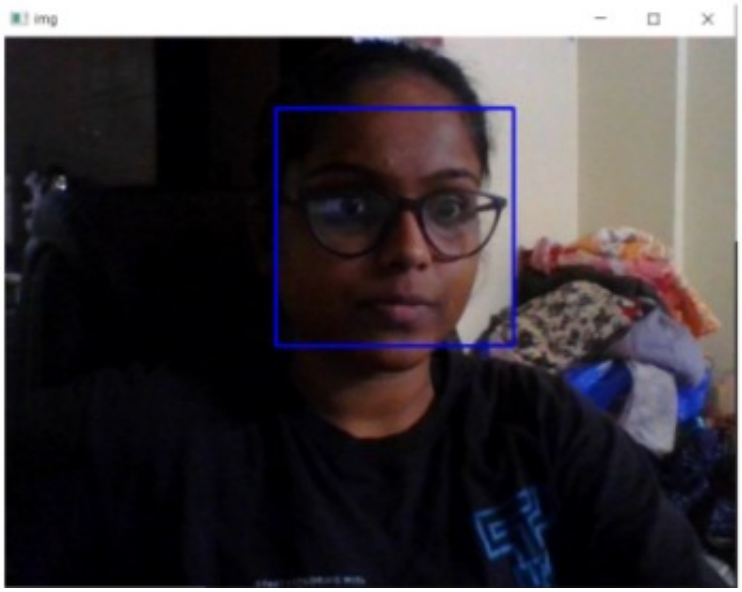
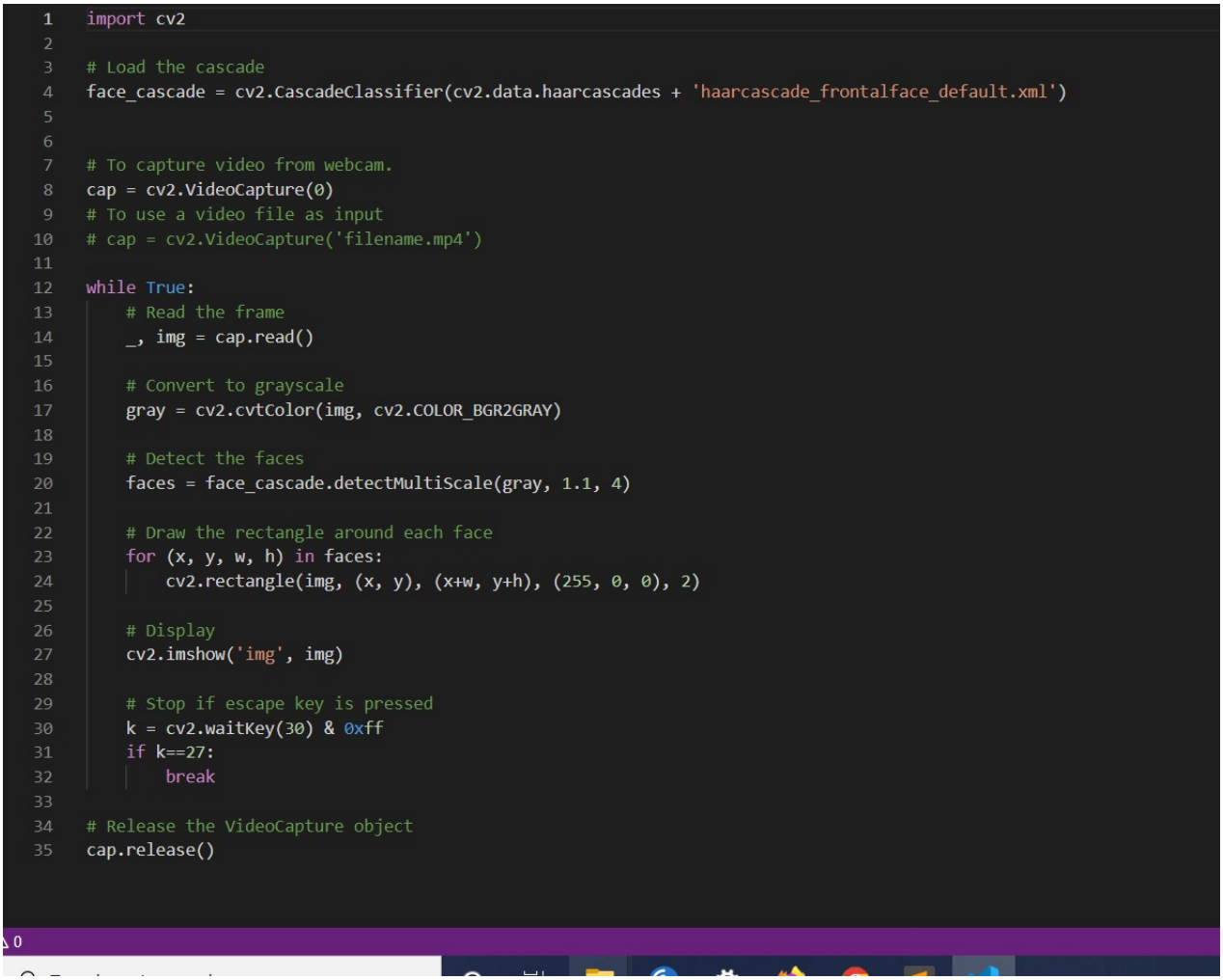
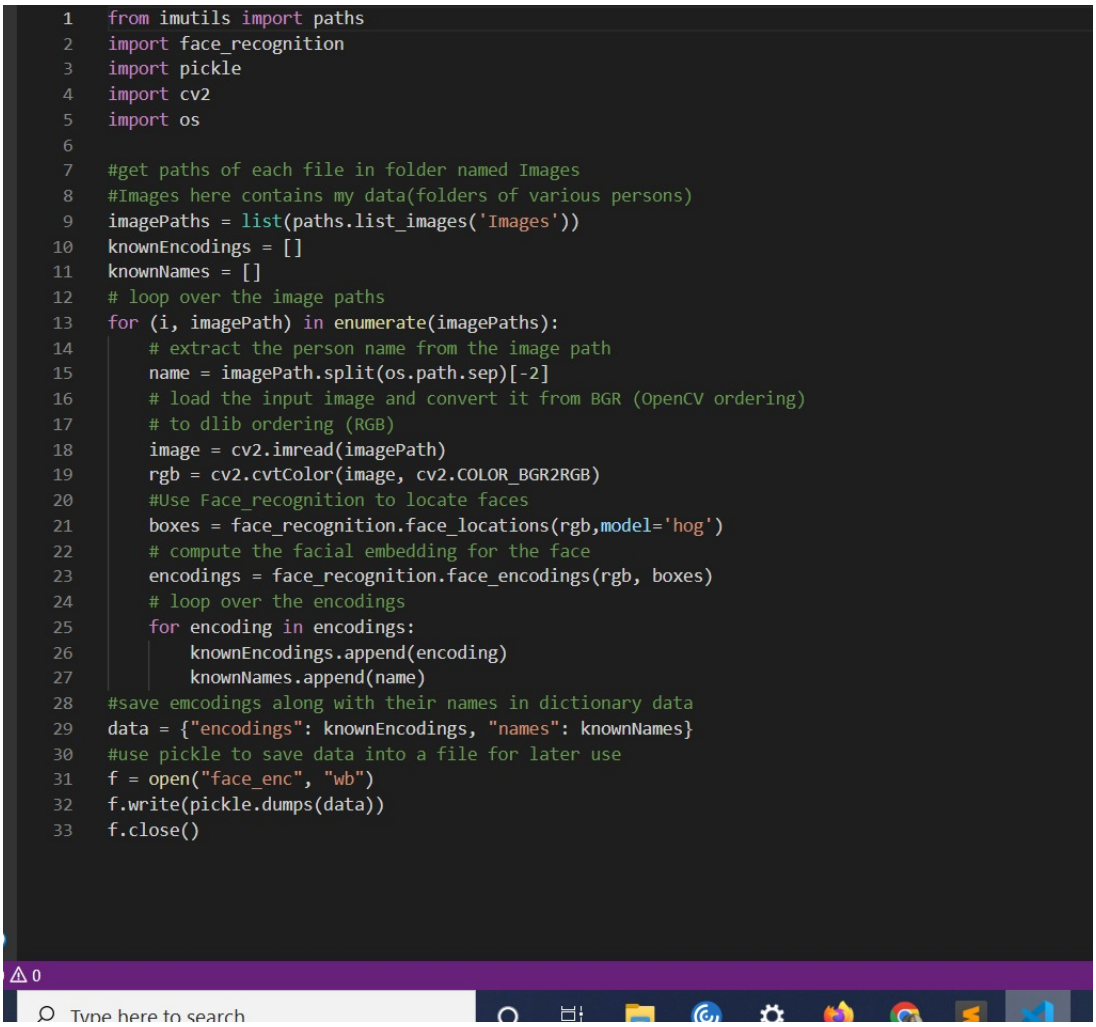


Figure 7: Face Detection



```python
import cv2

# Load the cascade
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')


# To capture video from webcam.
cap = cv2.VideoCapture(0)
# To use a video file as input
# cap = cv2.VideoCapture('filename.mp4')

while True:
    # Read the frame
    _, img = cap.read()

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

    # Detect the faces
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    # Draw the rectangle around each face
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

    # Display
    cv2.imshow('img', img)

    # Stop if escape key is pressed
    k = cv2.waitKey(30) & 0xff
    if k==27:
        break

# Release the VideoCapture object
cap.release()
```

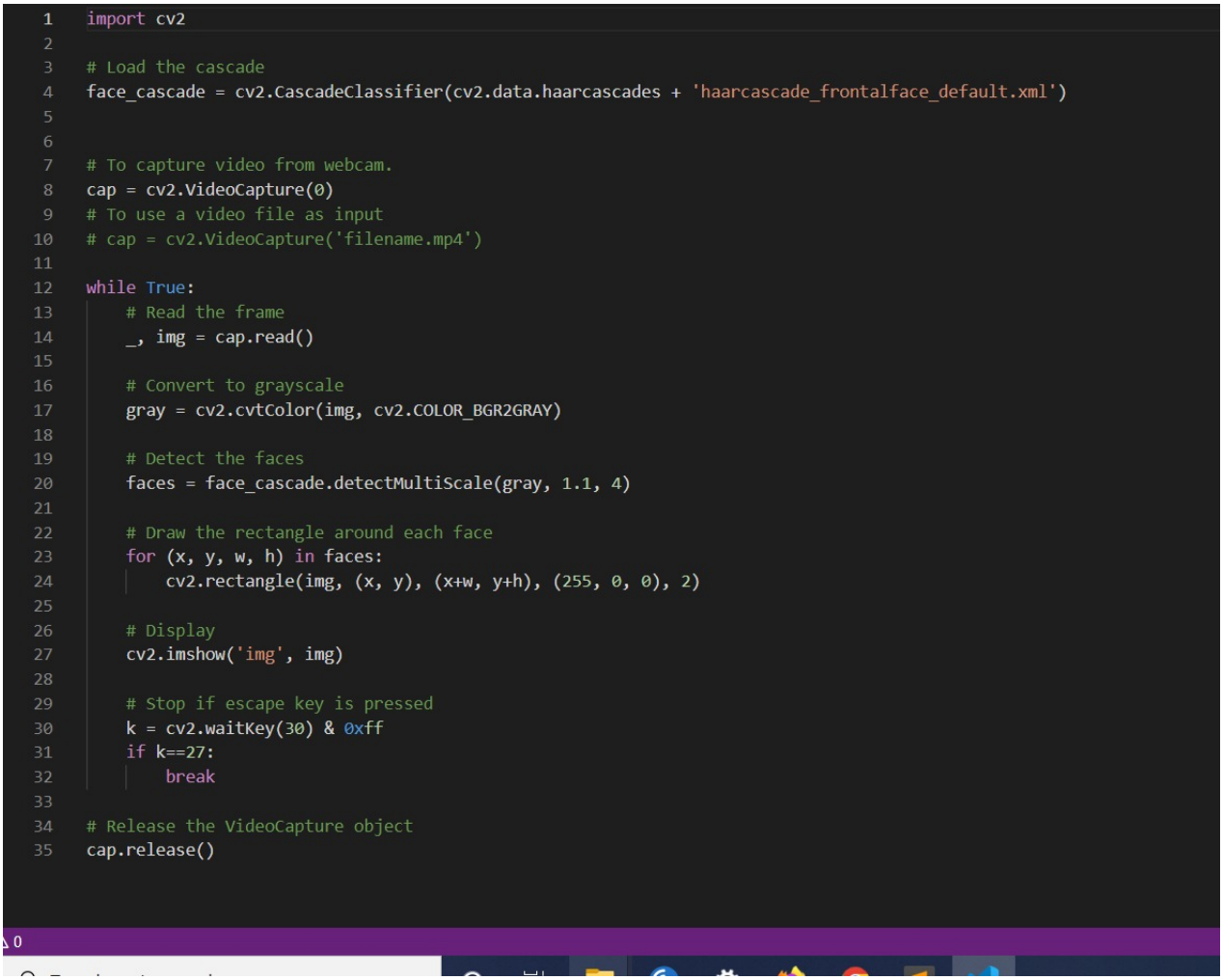
Figure 8: Code for face Detection



```python
from imutils import paths
import face_recognition
import pickle
import cv2
import os

#get paths of each file in folder named Images
#Images here contains my data(folders of various persons)
imagePaths = list(paths.list_images('Images'))
knownEncodings = []
knownNames = []
# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
    # extract the person name from the image path
    name = imagePath.split(os.path.sep)[-2]
    # load the input image and convert it from BGR (OpenCV ordering)
    # to dlib ordering (RGB)
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    #Use Face_recognition to locate faces
    boxes = face_recognition.face_locations(rgb,model='hog')
    # compute the facial embedding for the face
    encodings = face_recognition.face_encodings(rgb, boxes)
    # loop over the encodings
    for encoding in encodings:
        knownEncodings.append(encoding)
        knownNames.append(name)
#save emcodings along with their names in dictionary data
data = {"encodings": knownEncodings, "names": knownNames}
#use pickle to save data into a file for later use
f = open("face_enc", "wb")
f.write(pickle.dumps(data))
f.close()
```

Figure 9: Code for Embeddings

Figure 10: Creation of Embeddings



```python
import cv2

# Load the cascade
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')


# To capture video from webcam.
cap = cv2.VideoCapture(0)
# To use a video file as input
# cap = cv2.VideoCapture('filename.mp4')

while True:
    # Read the frame
    _, img = cap.read()

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

    # Detect the faces
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    # Draw the rectangle around each face
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

    # Display
    cv2.imshow('img', img)

    # Stop if escape key is pressed
    k = cv2.waitKey(30) & 0xff
    if k==27:
        break

# Release the VideoCapture object
cap.release()
```

Figure 11: Face Recognition Code

```
        if True in matches:
            #Find positions at which we get True and store them
            matchedIdxs = [i for (i, b) in enumerate(matches) if b]
            counts = {}
            # loop over the matched indexes and maintain a count for
            # each recognized face face
            for i in matchedIdxs:
                #Check the names at respective indexes we stored in matchedIdxs
                name = data["names"][i]
                #increase count for the name we got
                counts[name] = counts.get(name, 0) + 1
            #set name which has highest count
            name = max(counts, key=counts.get)


        # update the list of names
        names.append(name)
        # loop over the recognized faces
        for ((x, y, w, h), name) in zip(faces, names):
            # rescale the face coordinates
            # draw the predicted face name on the image
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(frame, name, (x, y), cv2.FONT_HERSHEY_SIMPLEX,
             0.75, (0, 255, 0), 2)
    cv2.imshow("Frame", frame)
    k = cv2.waitKey(30) & 0xff
    if k==27:
        break


video_capture.release()
cv2.destroyAllWindows()
```

Figure 12: Face Recognition Code

4.2 Concealed Object Detection

- Concealed Object Detection uses the SSD MobileNet model.
- It is comparatively faster, has small memory, can work with just a CPU and does not require a complicated setup.
- It can work without any external dependencies.
- It takes the model, configuration files and can detect a standard set of 90 types of objects.
- The video was split into each frame and the data of each frame is passed through to the model with the configurations.
- The object detected is judged and listed on the top left in the frame.

```
def object_detector(pqueue):

    object_flag = True
    thres = 0.45  # Threshold to detect object
    nms_threshold = 0.4

    classNames = []
    classFile =
    "E:\\College\\Projects\\FourthYear\\SEM7\\SuspiciousBehaviourRecognition\\ObjectDetector-Deepak\\'coco.names"
    with open(classFile, "rt") as f:
        classNames = f.read().rstrip("\n").split("\n")

    configPath =
    "E:\\College\\Projects\\FourthYear\\SEM7\\SuspiciousBehaviourRecognition\\ObjectDetector-Deepak\\ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt"
    weightsPath =
    "E:\\College\\Projects\\FourthYear\\SEM7\\SuspiciousBehaviourRecognition\\ObjectDetector-Deepak\\frozen_inference_graph.pb"

    net = cv2.dnn_DetectionModel(weightsPath, configPath)
    net.setInputSize(320, 320)
    net.setInputScale(1.0 / 127.5)
    net.setInputMean((127.5, 127.5, 127.5))
    net.setInputSwapRB(True)
    video_capture = cv2.VideoCapture(0, cv2.CAP_DSHOW)
```

Figure 13: Object Detection Code

```
while True:
    ret, img = video_capture.read()
    img1 = copy.copy(img)
    pqueue.put(img1)

    classIds, confs, bbox = net.detect(img, confThreshold=thres)
    bbox = list(bbox)
    confs = list(np.array(confs).reshape(1, -1)[0])
    confs = list(map(float, confs))
    indices = cv2.dnn.NMSBoxes(bbox, confs, thres, nms_threshold)


    for i in indices:
        i = i[0]
        box = bbox[i]
        x, y, w, h = box[0], box[1], box[2], box[3]
        cv2.rectangle(img, (x, y), (x+w, h+y), color=(0, 255, 0), thickness=2)
        cv2.putText(img, classNames[classIds[i][0]-1].upper(), (box[0]+10, box[1]+30),
        cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
        if classNames[classIds[i][0]-1] == "knife":
            if object_flag:
                message = "Knife Detected"
                print("ss")
                # s.sendmail("modgtrek197@gmail.com", "rajatshenoy@gmail.com", message)
                object_flag = False

    cv2.namedWindow('Object Detector')
    cv2.moveWindow('Object Detector', 600,30)
    cv2.imshow("Object Detector", img)
    k = cv2.waitKey(1) & 0xff
    if k==27:
        break

pqueue.put('done')
video_capture.release()
cv2.destroyAllWindows()
```

Figure 14: Object Detection Code

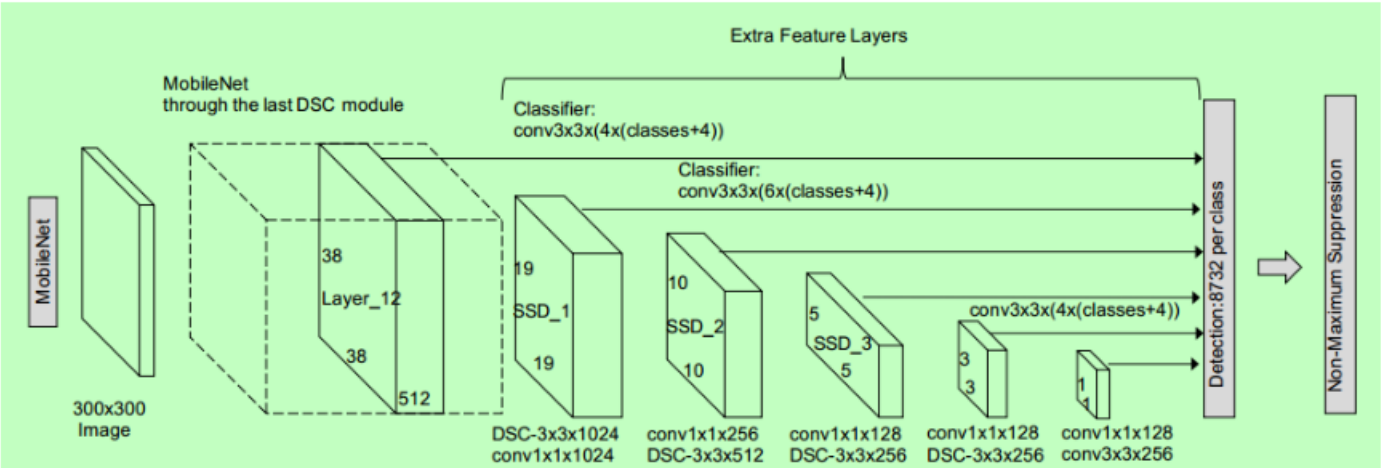


Figure 6. Mobile-Det: SSD-based detection with MobileNet as backbone, modified based on [14]

Figure 15: SSD Mobile Net Architecture

4.3 Suspicious Behaviour Prediction

- The Dataset used for this module was the CAVIAR Dataset.
- CAVIAR Dataset uses XML to store its annotation of the .mpg videos
- The dataset contains a role and context, based on which we can find out whether the action is suspicious or not.
- The XML file had to be parsed and mapped with each frame of the video.
- The video was split into each frame and the data of each frame was found in the XML file.
- We processed the data and categorized the data as suspicious vs not suspicious.
- Then we stored the train and test data in .npy files as loading times are better in .npy files compared to .txt and .csv files
- The Model consists of Resnet Layer followed by a Time Distributed Layer followed by Gated Recurrent Unit followed by two Dropout and Dense Layers.
- We have trained two other models with GRU being replaced by LSTMs and RNNs.
- We load our data from the .npy files.
- Since we wanted our model to make prediction on series of movements and not only on one image we had to reshape our data into 5 dimensional data - (No of batches, No of frames , Width, Height , Channel)
- We trained our model for 50 epochs on TPUs from Google Colab

Resting, slumping or fainting

Person resting in chair	*Rest_InChair.mpg* (12 Mb) Ground truth XML version: *ricgt.xml* JPEGS: *Rest_InChair_jpg.tar.gz* (21 Mb)
Person slump on floor	*Rest_SlumpOnFloor.mpg* (11 Mb) Ground truth XML version: *rsfgt.xml* JPEGS: *Rest_SlumpOnFloor_jpg.tar.gz* (19 Mb)
Person wiggle on floor	*Rest_WiggleOnFloor.mpg* (15 Mb) Ground truth XML version: *rwgt.xml* JPEGS: *Rest_WiggleOnFloor_jpg.tar.gz* (28 Mb)
Person fall down immobile	*Rest_FallOnFloor.mpg* (12 Mb) Ground truth XML version: *rffgt.xml* JPEGS: *Rest_FallOnFloor_jpg.tar.gz* (21 Mb)

Leaving bags behind

Person leaving bag by wall	*LeftBag.mpg* (17 Mb) Ground truth XML version: *lb1gt.xml* JPEGS: *LeftBag_jpg.tar.gz* (31 Mb)
Person leaving bag at chairs	*LeftBag_AtChair.mpg* (13 Mb) Ground truth XML version: *lb2gt.xml* JPEGS: *LeftBag_AtChair_jpg.tar.gz* (24 Mb)
Person leaving bag behind chairs	*LeftBag_BehindChair.mpg* (13 Mb) Ground truth XML version: *lbbcgt.xml* JPEGS: *LeftBag_BehindChair_jpg.tar.gz* (23 Mb)
Person leaving box	*LeftBox.mpg* (10 Mb) Ground truth XML version: *lbgt.xml* JPEGS: *LeftBox_jpg.tar.gz* (18 Mb)
Person leaving bag but then pick it up again	*LeftBag_PickedUp.mpg* (16 Mb) Ground truth XML version: *lbpugt.xml* JPEGS: *LeftBag_PickedUp_jpg.tar.gz* (29 Mb)

Figure 16: CAVIAR Dataset

```
<frame number="0">

    <objectlist>

        <object id="0">
            <orientation>150</orientation>
            <box xc="168" yc="118" w="48" h="32"/>
            <appearance>appear</appearance>
            <hypothesislist>
                <hypothesis id="1" prev="0.0" evaluation="1.0">
                    <movement evaluation="1.0">active</movement>
                    <role evaluation="1.0">walker</role>
                    <context evaluation="1.0">immobile</context>
                    <situation evaluation="1.0">inactive</situation>
                </hypothesis>
            </hypothesislist>
        </object>

    </objectlist>

    <grouplist/>

</frame>

<frame number="1">

    <objectlist>

        <object id="0">
            <orientation>150</orientation>
            <box xc="168" yc="118" w="48" h="32"/>
            <appearance>visible</appearance>
            <hypothesislist>
                <hypothesis id="1" prev="1.0" evaluation="1.0">
                    <movement evaluation="1.0">inactive</movement>
                    <role evaluation="1.0">walker</role>
                    <context evaluation="1.0">immobile</context>
                    <situation evaluation="1.0">inactive</situation>
                </hypothesis>
            </hypothesislist>
        </object>

    </objectlist>

    <grouplist/>

</frame>
```

Figure 17: XML Data

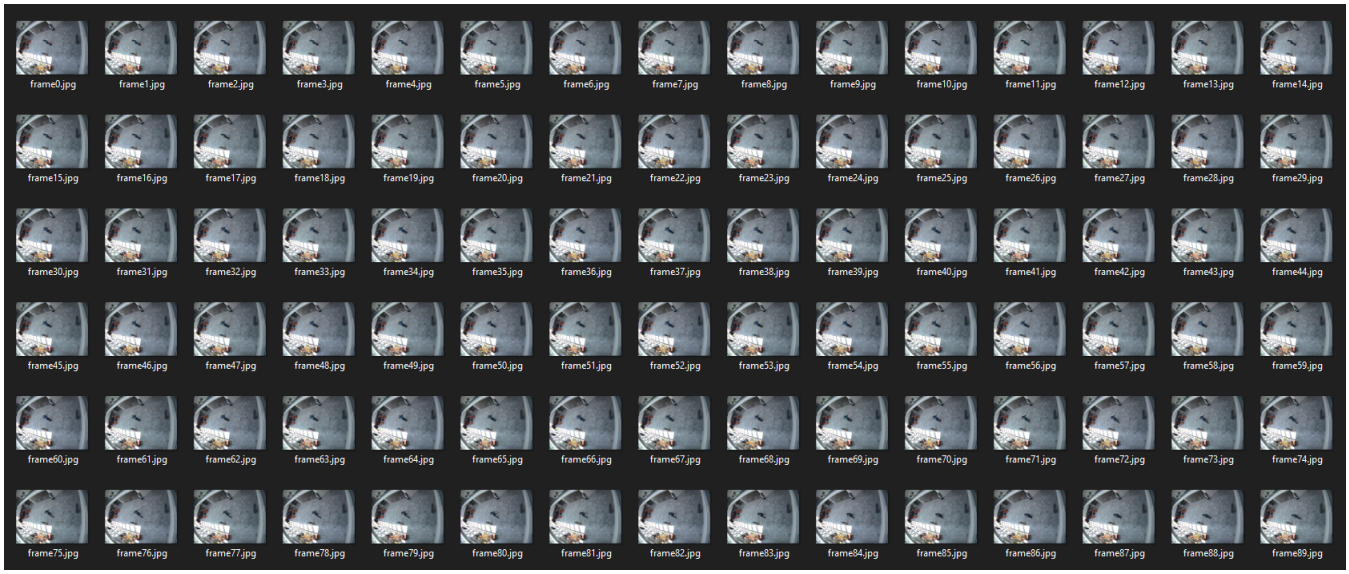


Figure 18: Frames to be processed

```python
class CaviarDataset():

    def __init__(self, class_map=None):
        self._image_ids = []
        self.image_info = []
        self.class_info = [{"source": "", "id": 0, "name": "BG"}]
        self.source_class_ids = {}

    def add_class(self, source, class_id, class_name):
        for info in self.class_info:
            if info['source'] == source and info["id"] == class_id:
                return
        self.class_info.append({
            "source": source,
            "id": class_id,
            "name": class_name,
        })

    def add_image(self, source, image_id, path, **kwargs):
        image_info = {
            "id": image_id,
            "source": source,
            "path": path,
        }
        image_info.update(kwargs)
        self.image_info.append(image_info)
```

Figure 19: OOP Representation of the Dataset

```
def prepare(self, class_map=None):

    def clean_name(name):
        """Returns a shorter version of object names for cleaner display."""
        return ",".join(name.split(",")[:1])

    self.num_classes = len(self.class_info)
    self.class_ids = np.arange(self.num_classes)
    self.class_names = [clean_name(c["name"]) for c in self.class_info]
    self.num_images = len(self.image_info)
    self._image_ids = np.arange(self.num_images)
    self.sources = list(set([i['source'] for i in self.class_info]))
    self.source_class_ids = {}
    for source in self.sources:
        self.source_class_ids[source] = []
        for i, info in enumerate(self.class_info):
            if i == 0 or source == info['source']:
                self.source_class_ids[source].append(i)

def load_dataset(self, data_dir, dataset_dir, obj_class, is_train=True):
    self.add_class(dataset_dir, 1, obj_class)
    annotations_dir = data_dir + fr'\annots'
    for im_id, filename in enumerate(os.listdir(dataset_dir)):
        img_path = os.path.join(dataset_dir, filename)
        print(img_path)
        annots = os.path.join(annotations_dir, os.listdir(annotations_dir)[0])
        self.add_image('dataset', image_id=im_id, path=img_path, annotation=annots)
```

Figure 20: Loading and preparing the dataset

```
def extract_boxes(self, filename, im_id,all_roles,all_context):
    tree = ElementTree.parse(filename)
    root = tree.getroot()
    boxes, all_labels = [], []
    for f, frames in enumerate(root.findall('frame')):
        xc, yc, h, w, sits, roles, movements, contexts = [], [], [], [], [], [], [], []
        if f == im_id:
            for box in frames.findall('.//box'):
                xc.append(box.get('xc'))
                yc.append(box.get('yc'))
                h.append(box.get('h'))
                w.append(box.get('w'))
            for role in frames.findall('.//role'):
                roles.append(role.text)
            for context in frames.findall('.//context'):
                contexts.append(context.text)

            coors = np.ones(shape=(len(xc), 4))
            a, b = coors.shape
            for i in range(a):
                coors[i, :] = [h[i], w[i], xc[i], yc[i]]
                all_labels.append([all_roles[roles[i]], all_context[contexts[i]]])
            boxes.append(coors)
    return boxes, all_labels
```

Figure 21: Extract Boxes position from the data

```
def extract_frames_of_video(video_path, new_frame_path):
    cap = cv2.VideoCapture(video_path)
    count = 0
    if os.path.exists(new_frame_path):
        shutil.rmtree(new_frame_path)
        os.mkdir(new_frame_path)
    else:
        os.mkdir(new_frame_path)
    while cap.isOpened():
        ret, frame = cap.read()
        if ret:
            cv2.imwrite(new_frame_path + r'\frame{:d}.jpg'.format(count), frame)
            count += 1
            cap.set(25, count)
        else:
            cap.release()
            cv2.destroyAllWindows()
            break
```

Figure 22: Extract Data Code

```
def run_from_video(data_dir, frame_path, action_name, caviar_obj,all_roles,all_context):
    caviar_obj.load_dataset(data_dir, frame_path, action_name, is_train=True)
    caviar_obj.prepare()
    num_frame = len(os.listdir(frame_path))
    X = []
    y = np.zeros([int((num_frame - 251)/3)+1, 2], dtype=object)

    for i, ids in enumerate(np.arange(250, num_frame - 1, 3)):
        f = os.path.join(frame_path, 'frame{0}.jpg'.format(ids))
        image = pl.imread(f)
        X.append(image)

        path = caviar_obj.image_info[ids]['annotation']
        box, mov = caviar_obj.extract_boxes(path, ids,all_roles,all_context)

        if len(box) == 0:
            break
        box = np.array(box[0])
        mov = np.array(mov)
        temp = np.zeros([1, len(box)], dtype=object)

        if len(mov) > 0:
            for w in range(len(box)):
                temp[0, w] = mov[w]
            y[i, 0] = [z[0] for z in temp[0]]
            y[i, 1] = [z[1] for z in temp[0]]
        else:
            y[i,:] = [],[]
    return X,y
```

Figure 23: The Driver Code

```
def suspicious_behavior_labels(labels):
    very_suspicious_index = 2
    suspicious_index = 1
    not_suspicious_index = 0
    num_frames = len(labels)

    new_labels = np.array([],dtype=int)
    check_role = np.array([])
    check_context = np.array([])

    for i in range(num_frames):
        if labels[i,0] != [] and labels[i,1] != []:
            check_role = np.append(check_role, np.max(labels[i,0]))
            check_context = np.append(check_context, np.max(labels[i,1]))
        else:
            check_role = np.append(check_role, 0)
            check_context = np.append(check_context, 0)

    for i in range(num_frames):
        if check_role[i] == very_suspicious_index  or check_context[i] == very_suspicious_index:
            new_labels = np.append(new_labels, 2)
        elif check_role[i] == suspicious_index  or check_context[i] == suspicious_index:
            new_labels = np.append(new_labels, 1)
        else:
            new_labels = np.append(new_labels, 0)

    return new_labels.reshape(len(new_labels), 1)
```

Figure 24: Labeling Suspicious Behaviour in Data

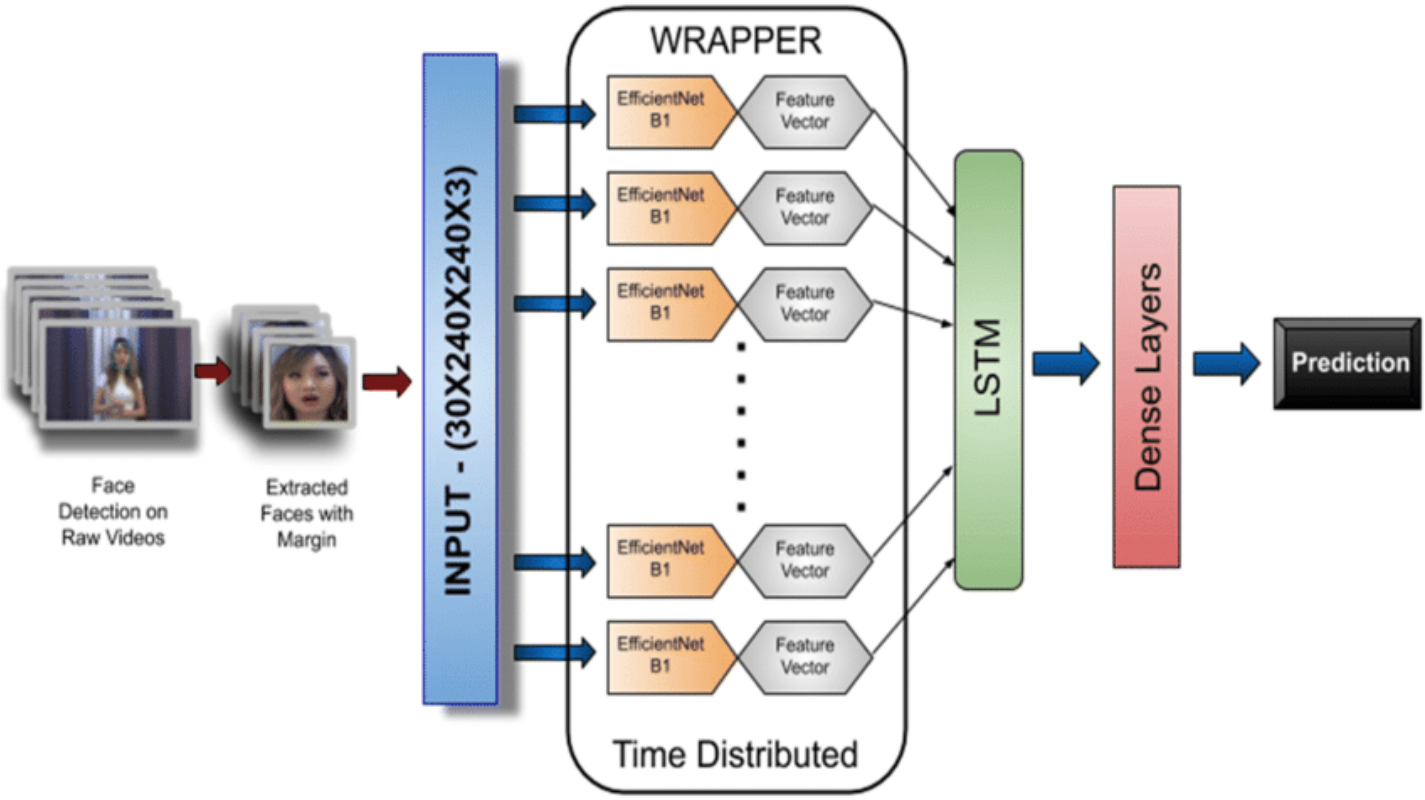


Figure 25: Time Distributed Layer

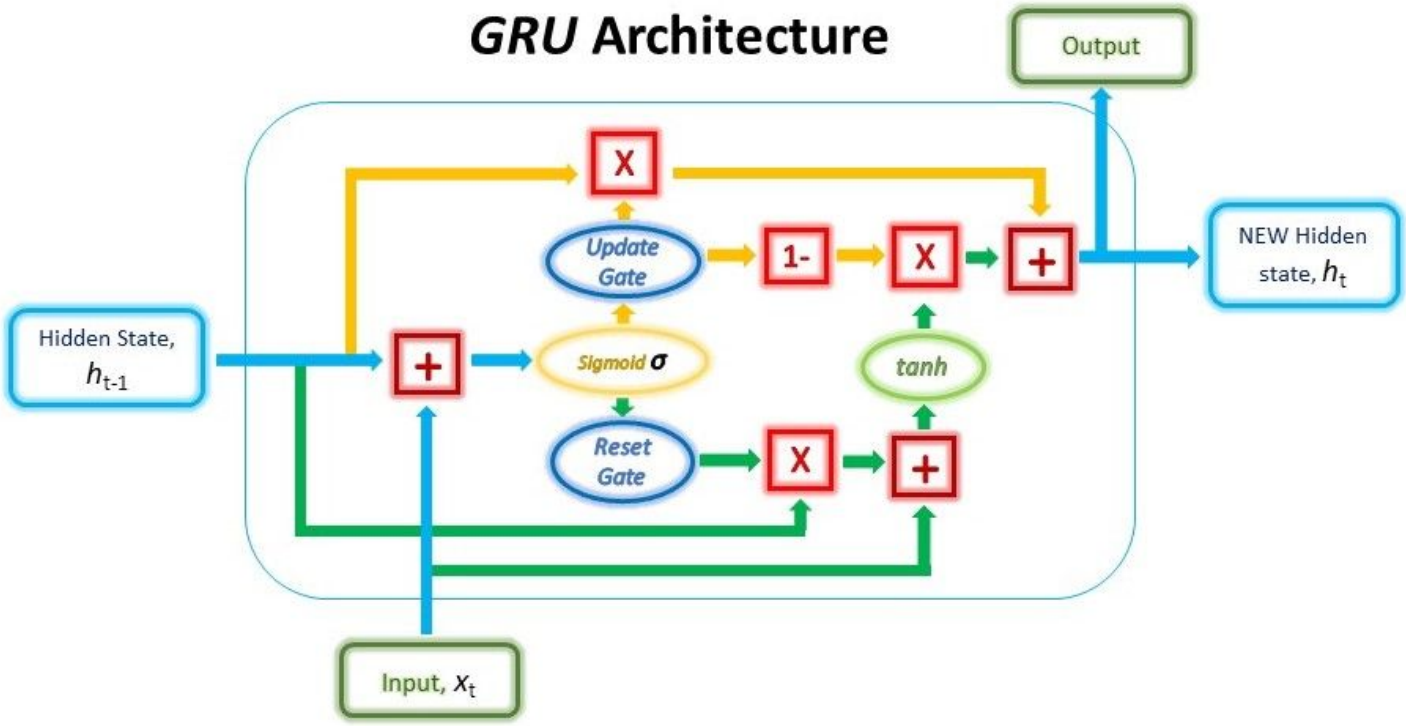


Figure 26: Gated Resnet Unit

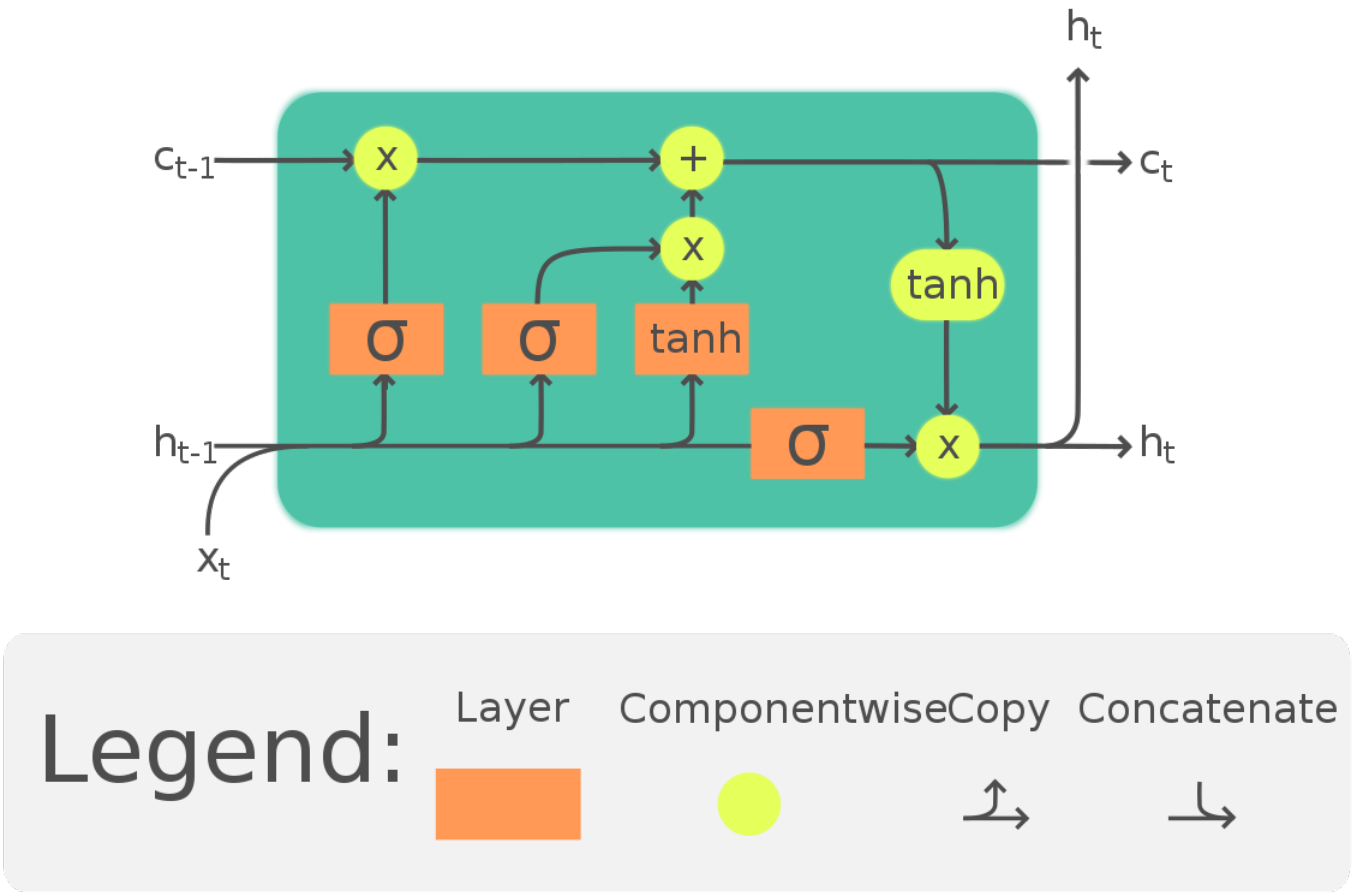


Figure 27: Long Short Term Memory Cell

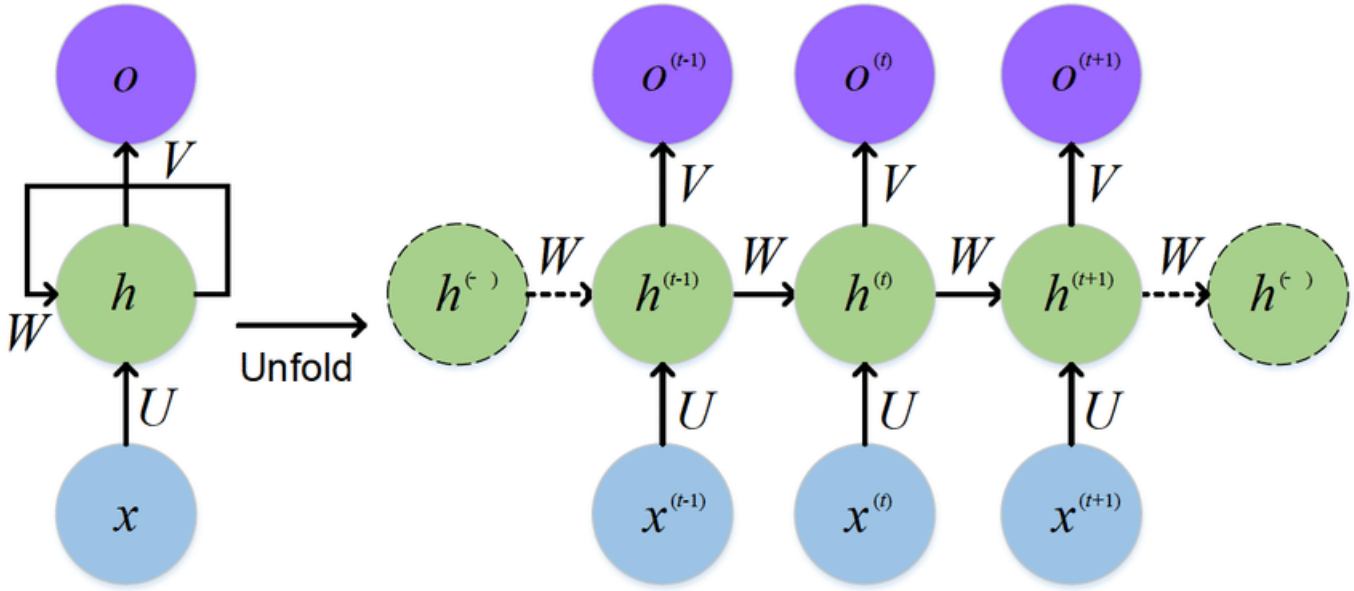


Figure 28: Recurrent Neural Network

```python
if __name__ == '__main__':

    # Change this string to the path of the dataset
    main_dir = r'E:\College\Projects\FourthYear\SEM7\SuspiciousBehaviourRecognition\Caviar-Dataset'

    actions = ['Browse', 'Fight', 'Groups_Meeting', 'LeftBag', 'Rest', 'Walk']
    all_roles = {'fighters': 2, 'fighter': 2, 'leaving object': 2, 'browser': 1, 'browsers': 1,
     'walkers': 0, 'meet': 0, 'meeters': 0, 'walker': 0}
    all_context = {'fighting': 2, 'leaving': 2, 'drop down': 2, 'browsing': 1, 'immobile': 0,
     'walking': 0, 'meeting': 0, 'windowshop': 0, 'shop enter': 0, 'shop exit': 0, 'shop reenter': 0, 'none':0}


    for id_dataset in range(6):
        # id_dataset = int(input('Choose action'))
        action_dir = os.path.join(main_dir, actions[id_dataset])

        train_testX, train_testy = [], []
        for id_vid in range(1,4):
            data_dir = os.path.join(action_dir, actions[id_dataset]) + str(id_vid)

            train_set = CaviarDataset()

            new_frames_file = action_dir + '\\' + actions[id_dataset] + str(id_vid) +'\\new'

            video_file = os.path.join(data_dir + r'\video', os.listdir(data_dir + r'\video')[0])

            extract_frames_of_video(os.path.join(data_dir, video_file) , new_frames_file)

            X_frames,y_frames = run_from_video(
                data_dir, new_frames_file, actions[id_dataset], train_set, all_roles,all_context)

            # print('video number {0} \n X_frames = {1} \n y_frames = {2}'.format(id_vid, len(X_frames), len(y_frames)))

            train_testX.append(np.array(X_frames))
            train_testy.append(np.array(y_frames))


        X = np.concatenate((train_testX[0], train_testX[1], train_testX[2]), axis=0)
        y = np.concatenate((train_testy[0], train_testy[1], train_testy[2]), axis=0)
        y = suspicious_behavior_labels(y)

        with open(main_dir+r'\{0}\X.npy'.format(actions[id_dataset]), 'wb') as f:
            np.save(f,X)
        with open(main_dir+r'\{0}\y.npy'.format(actions[id_dataset]), 'wb') as f:
            np.save(f, y)
```

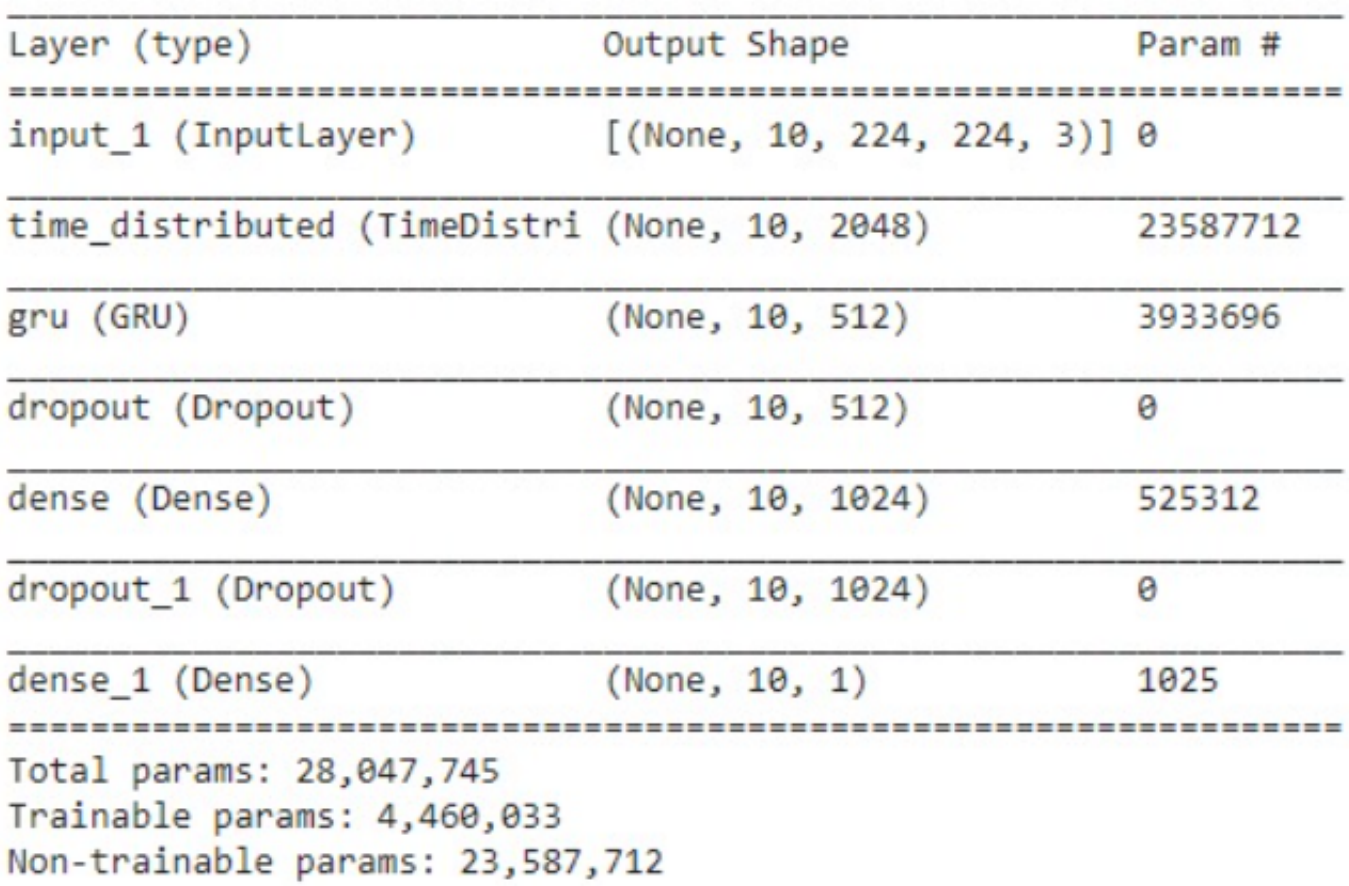
Figure 29: Data Creation Code using XML Files

```
[ ]  1  actions = ['Walk','Browse','Fight','LeftBag','Groups_Meeting','Rest']
     2  X = None
     3  y = None
     4  for action in actions:
     5    with open(action+"/X.npy",'rb') as f:
     6      temp_X = np.load(f,allow_pickle=True)
     7      j = len(temp_X)%10
     8      if X is None:
     9        X = temp_X[:0-j]
    10      else:
    11        X = np.concatenate((X,temp_X[:0-j]),axis=0)
    12    with open(action+"/y.npy",'rb') as f:
    13      temp_y = np.load(f,allow_pickle=True)
    14      j = len(temp_y)%10
    15      if y is None:
    16        y = temp_y[:0-j]
    17      else:
    18        y = np.concatenate((y,temp_y[:0-j]),axis=0)

     1  print(X.shape)
     2  print(y.shape)

(3850, 288, 384, 3)
(3850, 1)

     1  X = X[:,10:285,30:335]
     2  X = np.array([cv2.resize(i, (224,224)) for i in X[:,:]])

[ ]  1  y[y == 1] = 0
     2  y[y == 2] = 1

[ ]  1  X = X.reshape(385, 10 ,224, 224, 3)
     2  y = y.reshape(385, 10, 1)

[ ]  1  X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2)
```

Figure 30: Load and Clean Data

```
video = Input(shape=(10, 224, 224, 3))
cnn_base = ResNet50(input_shape=(224, 224, 3), weights="imagenet", include_top=False)
cnn_out = GlobalAveragePooling2D()(cnn_base.output)
cnn = Model(inputs=cnn_base.input, outputs=cnn_out)
cnn.trainable = False
x = TimeDistributed(cnn)(video)
x = GRU(512, return_sequences=True)(x) # Gated Recurrent Unit
x = Dropout(0.2)(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(1, activation="sigmoid")(x)
model = Model([video], x)
model.summary()

earlystop = EarlyStopping(patience=7)
callbacks = [earlystop]

model.compile(optimizer='adam', loss='binary_crossentropy',metrics=["accuracy"], )
model.fit(X_train, y_train, batch_size=64, epochs=50, shuffle=False, callbacks=callbacks)
```

Figure 31: Building Architecture of the Model using Keras



Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 10, 224, 224, 3)]	0
time_distributed (TimeDistri	(None, 10, 2048)	23587712
gru (GRU)	(None, 10, 512)	3933696
dropout (Dropout)	(None, 10, 512)	0
dense (Dense)	(None, 10, 1024)	525312
dropout_1 (Dropout)	(None, 10, 1024)	0
dense_1 (Dense)	(None, 10, 1)	1025

Total params: 28,047,745
Trainable params: 4,460,033
Non-trainable params: 23,587,712

Figure 32: Architecture of our Model

```
5/5 [==============================] - 14s 3s/step - loss: 0.0613 - accuracy: 0.9779
Epoch 43/50
5/5 [==============================] - ETA: 0s - loss: 0.0613 - accuracy: 0.9776WARNING:te
5/5 [==============================] - 14s 3s/step - loss: 0.0613 - accuracy: 0.9776
Epoch 44/50
5/5 [==============================] - ETA: 0s - loss: 0.0605 - accuracy: 0.9753WARNING:te
5/5 [==============================] - 14s 3s/step - loss: 0.0605 - accuracy: 0.9753
Epoch 45/50
5/5 [==============================] - ETA: 0s - loss: 0.0507 - accuracy: 0.9799WARNING:te
5/5 [==============================] - 14s 3s/step - loss: 0.0507 - accuracy: 0.9799
Epoch 46/50
5/5 [==============================] - ETA: 0s - loss: 0.0452 - accuracy: 0.9870WARNING:te
5/5 [==============================] - 14s 3s/step - loss: 0.0452 - accuracy: 0.9870
Epoch 47/50
5/5 [==============================] - ETA: 0s - loss: 0.0458 - accuracy: 0.9828WARNING:te
5/5 [==============================] - 14s 3s/step - loss: 0.0458 - accuracy: 0.9828
Epoch 48/50
5/5 [==============================] - ETA: 0s - loss: 0.0499 - accuracy: 0.9799WARNING:te
5/5 [==============================] - 14s 3s/step - loss: 0.0499 - accuracy: 0.9799
Epoch 49/50
5/5 [==============================] - ETA: 0s - loss: 0.0447 - accuracy: 0.9857WARNING:te
5/5 [==============================] - 14s 3s/step - loss: 0.0447 - accuracy: 0.9857
Epoch 50/50
5/5 [==============================] - ETA: 0s - loss: 0.0477 - accuracy: 0.9802WARNING:te
5/5 [==============================] - 14s 3s/step - loss: 0.0477 - accuracy: 0.9802
<tensorflow.python.keras.callbacks.History at 0x7f6491952090>
```

Figure 33: Training our model

4.4 Integrated System For Crime Detection

- The three individual models are integrated to live input from webcam and detect crime situations.
- The algorithms are executed parallelly using the concept of multiprocessing to compute the results.
- This significantly helps in improving performance of the systems. Both the computational tasks run in separate processes.
- The code is implemented using multiprocessing library in python
- Multiprocessing queue is used for sharing the webcam through openCV in both the processes.
- The three models run simultaneously in three separate windows
- If any of the frame is found to be suspicious, concerned authorities are notified.

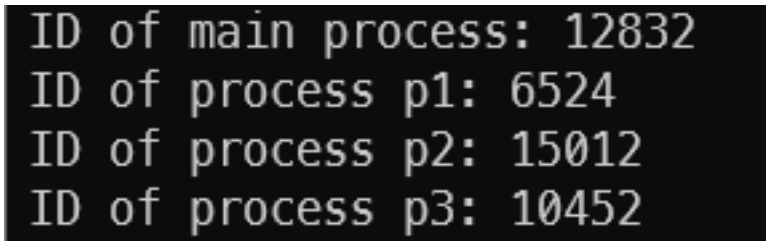


Figure 34: Multiple Process Running Simultaneously

```python
if __name__ == "__main__":

    print("ID of main process: {}".format(os.getpid()))
    pqueue = Queue()
    # p1 = Process(target=test_model,args=("Caviar-Dataset/Test/Leftbox.npy",))
    # p1 = Process(target=test_model,args=("Caviar-Dataset/Test/Browse.npy",))
    p1 = Process(target=test_model,args=("Caviar-Dataset/Test/Fight.npy",))
    p2 = Process(target=face_detector,args=(pqueue,))
    p3 = Process(target=object_detector,args=(pqueue,))

    p1.start()
    p2.start()
    p3.start()

    print("ID of process p1: {}".format(p1.pid))
    print("ID of process p2: {}".format(p2.pid))
    print("ID of process p3: {}".format(p3.pid))

    p1.join()
    p2.join()
    p3.join()
    s.quit()
    print("processes finished execution!")
```

Figure 35: Multiprocessing Code in Python

```python
s = smtplib.SMTP('smtp.gmail.com', 587)
model = load_model('model_resnet_gru_10fps_browse0_3.h5')
s.starttls()
s.login("yourmail@mail.com", "apppassword")
```

Figure 36: Mailing Code in Python

```
def test_model(file_name):

    earlystop = EarlyStopping(patience=7)
    callbacks = [earlystop]
    with open(file_name, 'rb') as f:
        test = np.load(f, allow_pickle=True)
    test = test[:,10:285,30:335]
    test = np.array([cv2.resize(i, (224,224)) for i in test[:,:]])
    j = len(test) % 10
    k = len(test)- j
    test = test[j:]
    test = test.reshape(k//10,10 ,224, 224, 3)
    labels = model.predict(test,batch_size=10,callbacks=callbacks)
    display_all_suspicious_images(test,labels)
```

Figure 37: Testing and Running our model against actual test cases

```python
def display_all_suspicious_images(video, labels):
    sus_flag = True
    img = None
    font = cv2.FONT_HERSHEY_SIMPLEX
    for i in range(video.shape[0]):
        for j in range(video.shape[1]):
            image = copy.copy(video[i,j,:,:,:])
            label = labels[i,j,:]
            im_shape = image.shape
            if label >= 0.5:
                image = cv2.rectangle(image, (1, 1), (im_shape[1]-1, im_shape[0]-1) , [0, 0, 255], 3)
                image = cv2.putText(img=image,text='Suspicious Behavior', org=(24,24), fontFace=font ,
                    fontScale=0.4, color=[0,0,255], lineType=2)
                if sus_flag:
                    message = "Suspicious Behaviour Detected "
                    print("Sssss")
                    # s.sendmail("modgtrek197@gmail.com", "rajatshenoy@gmail.com", message)
                    sus_flag = False
            else:
                image = cv2.putText(img=image,text='Not Suspicious Behavior', org=(24,24), fontFace=font ,
                    fontScale=0.4, color=[255,0,0], lineType=3)

            image = cv2.resize(image, (500, 500))

            cv2.namedWindow('Suspicious Behavior')
            cv2.moveWindow('Suspicious Behavior', 40,30)
            cv2.imshow('Suspicious Behavior',image)
            if id == 1000:
                break
                pl.close('all')

            if cv2.waitKey(0) == ord('q'):
                continue
            else:
                break
```

Figure 38: Displaying suspicious and non suspicious boxes

5 Results and Discussion

5.1 Face Recognition

- The system is tested for live input from a webcam using OpenCV.
- It is observed to detect all the faces in the frame.
- The known person to the system is labeled with the name and the unknown person remains unlabelled.
- The faces detected are highlighted with a red rectangle.
- The model works with high accuracy and very little delay time

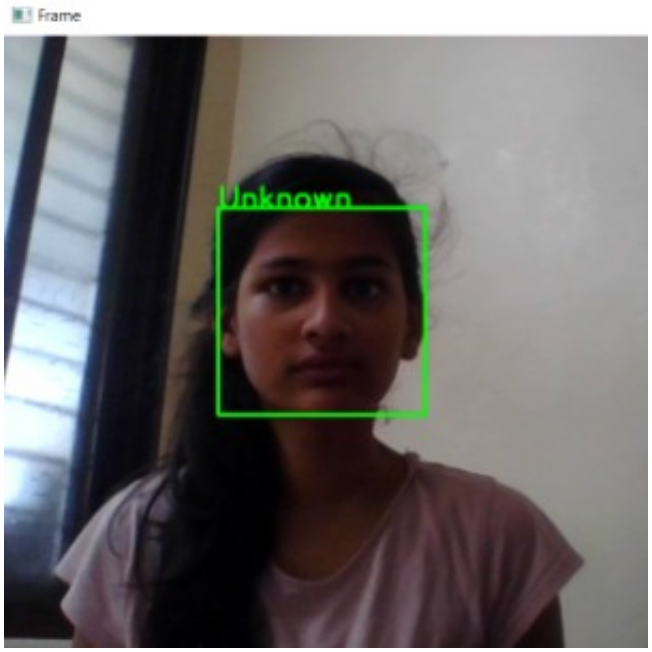


Figure 39: Face Recognition model Recognizing the name from the embeddings

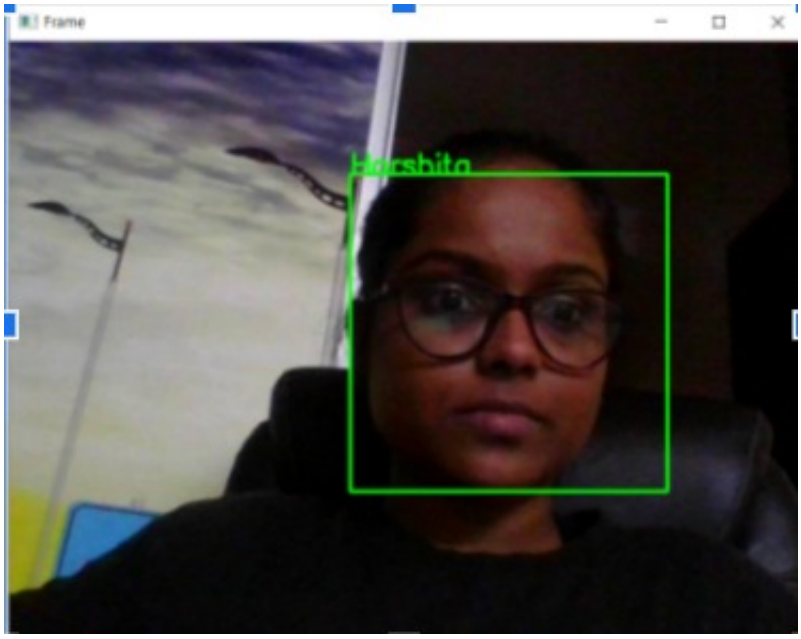


Figure 40: Face Recognition model Recognizing the name from the embeddings

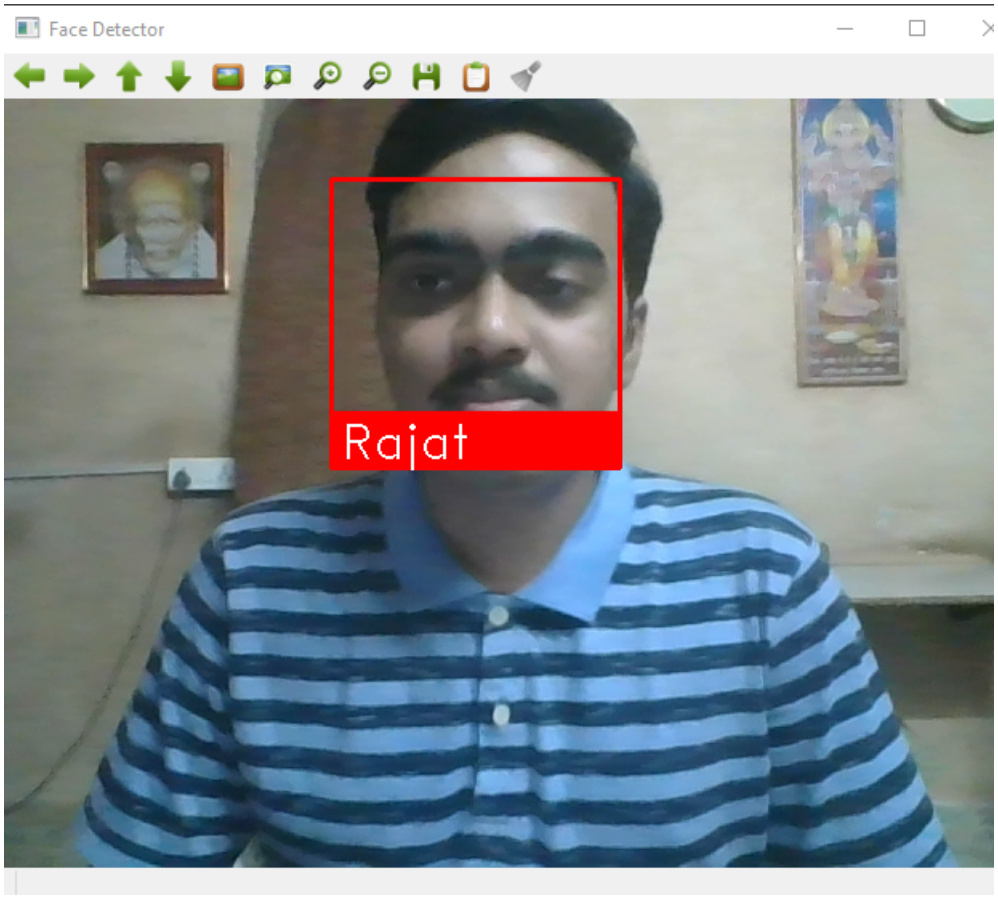


Figure 41: Face Recognition model Recognizing the name from the embeddings

5.2 Object Recognition

- The trained SSD Mobilenet model was found to detect 91 weapons from the DaSCI dataset.
- The SSD MobileNet model is known for its fast execution time and observed to give results with very high inference speed.
- The model was tested using input set of 5 different objects .
- The test set was made up of 20 samples each of some hazardous and daily life objects.
- The overall accuracy obtained was 82
- For live testing, the input frames are captured fro m the video and fed to the system.
- The hazardous objects and common real-life objects are highlighted in a rectangle with their titles.

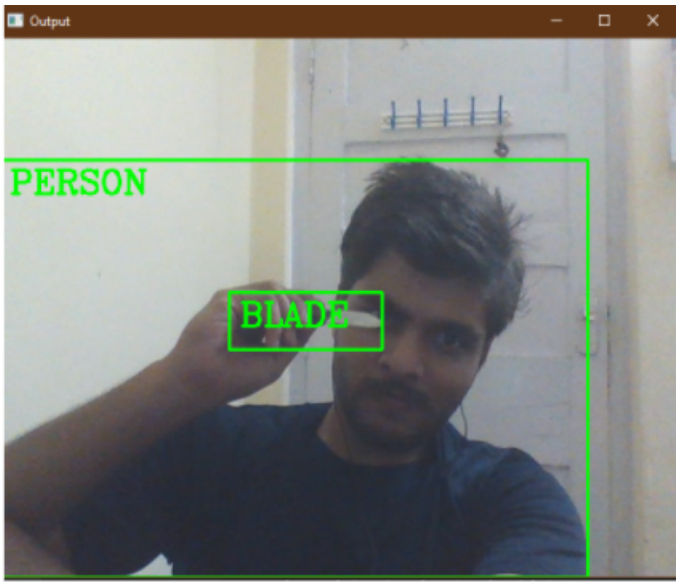


Figure 42: SSD Mobilenet Model detecting all the objects

5.3 Suspicious Behaviour Prediction

- The model detects relations between the frames and provides output only by processing a series of act ions.
- In other words, the model does not label the action as suspicious based on one frame.
- It takes into consideration a series of frames to make its prediction.
- In the test case there is a clip o f people fighting and the frame is labelled as suspicious

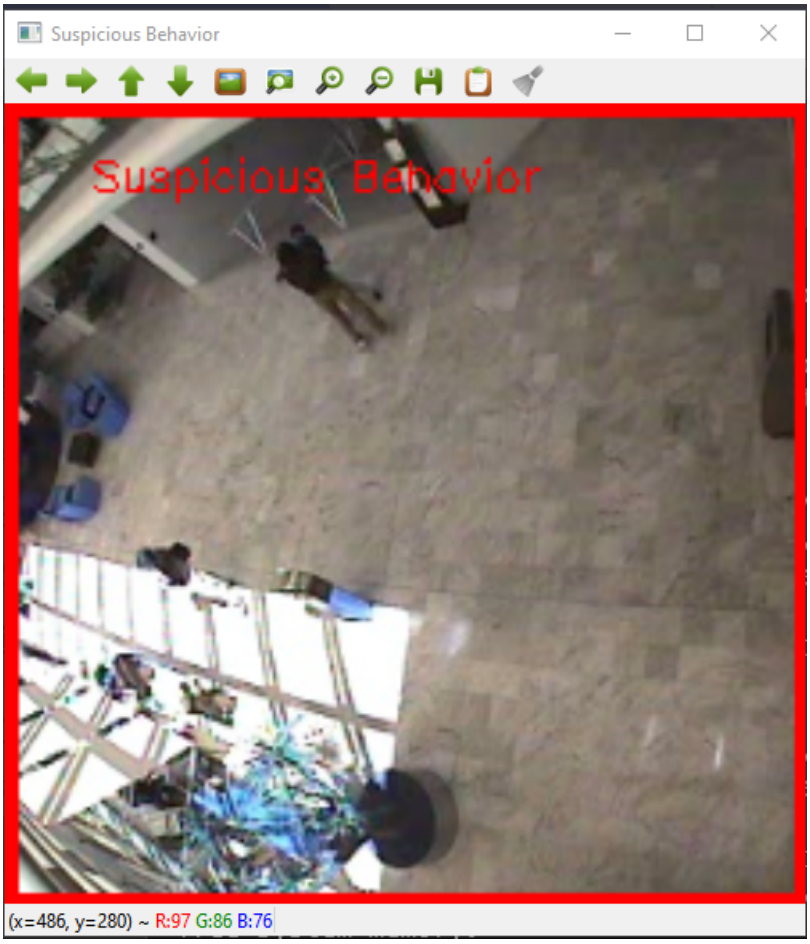


Figure 43: Suspicious behaviour detected by our model

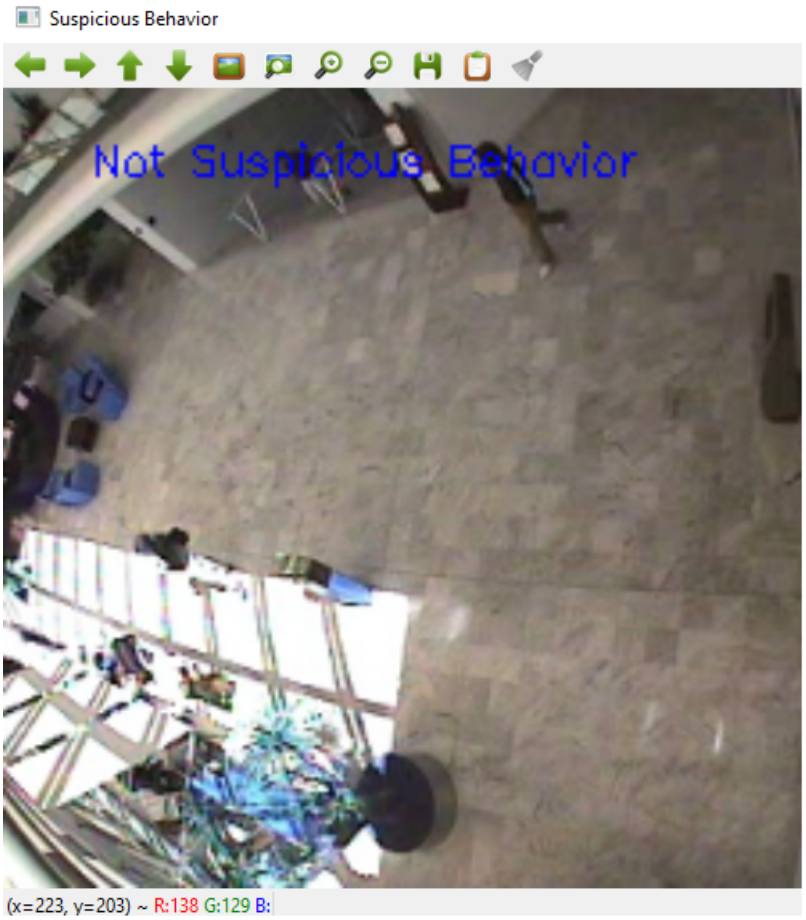


Figure 44: Non Suspicious behviour detected by our model

```
1   model.evaluate(X_test, y_test, batch_size=16)
```

```
5/5 [==============================] - 6s 1s/step - loss: 0.4111 - accuracy: 0.8610
[0.411093533039093, 0.8610389828681946]
```

Figure 45: Model Accuracy

5.4 Identifying Crime using Multiprocessing

- The three models are executed in parallel.
- This increases the speed of output generation when compared to running each model individually.
- The models run simultaneously in separate frames and share a webcam to take input.
- If any frame is found to be suspicious, an email is sent to the concerned authorities.
- These tests were carried out on an Intel core i5- 8300H 8th Gen processor with an 8GB RAM , which clearly indicate that with multiprocessing we were able to process more frames per second with Object Detection SSD model achieving up to 30 fps

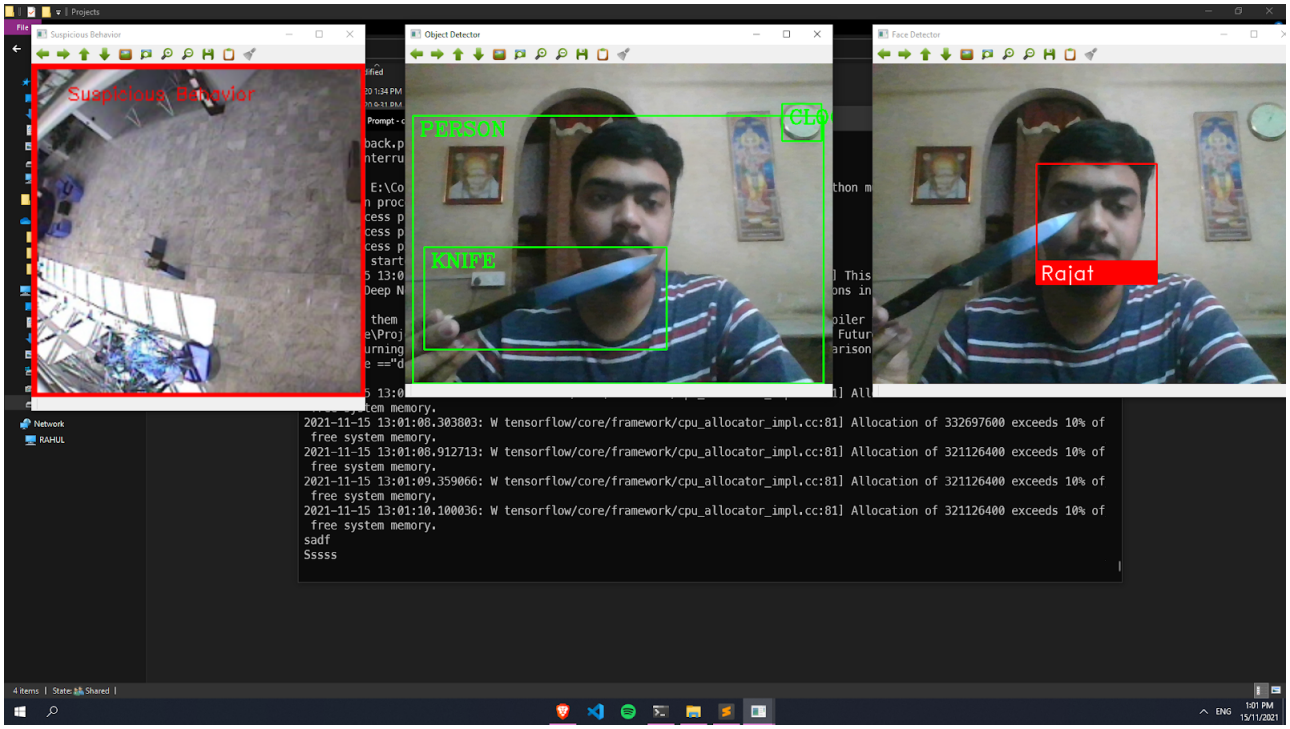


Figure 46: Multiple models running parallely using multiprocessing

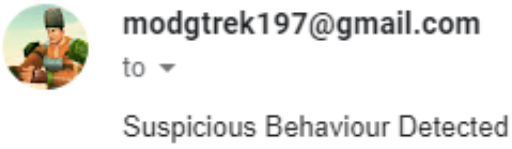


Figure 47: Mail Sent to the authorities

6 Conclusion and Further Work

6.1 Conclusion

In this paper, a crime detection framework was properly designed and implemented. The three components of the system face recognition, object identification, and suspicious behavior detection were tested individually and we obtained highly accurate results with little execution time. Co mparing faces based on similarity learning gave the best results for face recognition with minimu m errors. The model correctly detected faces in the frame. The SSD Mobilenet Model detected weapons with an accuracy of 81and RNN were compared for training the suspicious behavior detection model. GRU was found to give the best accuracy of 95.97and integrated into the final step. The integration of the algorithms using multiprocessing was the integral step in the prediction of criminal activit ies. The multiprocessing helped increase the inference speed of these two models running at the same time and sharing the same resources. The final model successfully managed to detect the suspicious activities in a single input frame. The output was displayed on the screen and an email was sent to the police authorities

6.2 Future Work

The system proposed in this paper has a few limitat ions. Face recognition and object detection work well when the distance between the object of interest and the input camera is considerably less. In real-wo rld scenarios usually, this distance is relatively more and the model developed in this paper wouldn't be an ideal choice fo r crime predict ion. The model designed for suspicious behaviour can be designed to detect different possible crime situations. In the future, we aim to design with a few additions the system will beco me more efficient to predict crime in different situations.

7 Research Paper

An Intelligent Framework For Crime Prediction Using Behavioural Tracking And Motion Analysis

Rajat Shenoy[1]
Department of Information Technology
Sardar Patel Institute of Technology
Mumbai, India
rajat.shenoy@spit.ac.in

Deepak Yadav[1]
Department of Information Technology
Sardar Patel Institute of Technology
Mumbai, India
deepak.yadav@spit.ac.in

Harshita Lakhotiya[1]
Department of Information Technology
Sardar Patel Institute of Technology
Mumbai, India
harshita.lakhotiya@spit.ac.in

Jignesh Sisodia[2]
Department of Information Technology
Sardar Patel Institute of Technology
Mumbai, India
jsisodia@spit.ac.in

***Abstract*— Closed Circuit Television Systems are now being deployed in most public spaces to make the city more secure. Manual observation of these clips for the prevention of crime would take up a lot of manpower. This paper proposes an Intelligent Framework using the power of Artificial Intelligence to ensure the safety of the surroundings. The system will use different Computer Vision techniques for video analysis. It will monitor CCTV footage for any criminal offenders, violent objects, and suspicious behavior which could lead to crime. SSD MobileNet Model, an architecture for concealed object detection, is trained for labeling weapons in the frame. The images captured are processed using Face Detection algorithms to identify human faces. Facial Recognition API using libraries in python is implemented to recognize the offenders from criminal records. A ResNet-GRU Model was trained for human behavior analysis which detects suspicious actions. An alert is generated when there are signs of crime and concerned authorities are notified. The proposed framework aims to make societies secure by correctly identifying criminals and crime-related objects.**

Keywords—*Object Detection, Face Detection, ResNet-GRU, Facial Recognition, Computer Vision, SSDMobileNet*

I. INTRODUCTION

Crime is one of the most common social problems faced worldwide. It has been a part of society for decades. It is a hindrance in the path of a more secure society, better life quality, and socio-economic growth of a nation. In the recent covid times, the nationwide shutdown has left people with uncertainty in employment, financial losses due to the same causing a lot of psychological trauma. It has led to an increase in burglary, shoplifting, and many such criminal activities. Some criminals escaped the legal system and are roaming around freely in society and now are a threat to society. In most developed countries, the government has a dataset of such criminals which can be used to identify them.

Nowadays, CCTV cameras are being installed all around the cities in societies, shops, commercial spaces, etc. All the activities in the locality can be tracked 24X7. This can help us monitor the public spaces for different criminal activities. Inspecting the footage with human intervention would take up a lot of manpower and is a slow process[6]. Automation of this process using Artificial Intelligence would be a better solution. This can help in a faster and more efficient way in detecting crime.

With the proposed methodology in this paper, we aim to develop a system that can help in making society more secure. It will identify criminal faces, detect suspicious actions and objects which can be used for criminal activities. After video analysis, each frame that is found to be a situation of interest will be fed to the system. The framework proposed comprises of following steps mainly:

1. Detecting and Identifying objects in the frame
2. Face Detection
3. Face Recognition
4. Suspicious Human Behavior Analysis

Recent studies in the field of Object Detection consist of models based on region-based CNN, Faster RCNN, and YOLO algorithms[7]. In Faster RCNN, regions are selected after the generation of a convolutional feature map which makes it faster than RCNN. Single Shot Multibox Detector is also one of the popular models. All the objects in the image are detected in one single shot. Face Detection is a prerequisite step for recognizing faces. Face Recognition can be implemented by training neural networks using a dataset of faces. Similarity-based learning which is based on holistic matching of faces is found to be a better approach for face identification [5]. Human motion can be identified and analyzed for suspicious behavior using CNN for extracting features from the video frames and feeding them to Gated Recurrent Units for making predictions [17]. GRU (Gated Recurrent Unit) is a recurrent neural network that seeks to tackle the vanishing gradient problem.

This paper is divided into the following sections: Section II describes related work in this field; Section III describes the methodology used for designing the proposed system; Section IV describes the result and evaluation of our model and in Section V; conclusion is drawn for the paper.

II. RELATED WORK

Swathi BN et al. have designed a system for predicting crime in real-time for Smart Homes. The framework is event-driven which sends data to concerned authorities for quick actions. The method used here is based on intelligent motion detection. Face recognition and object detection begin when some movement is detected in the live feed. The process of face recognition is based on the euclidean distance between features extracted from two images. The YOLOV3 algorithm is used for detecting objects. In this paper, the main focus is on real-time face recognition and weapon detection to predict crime in the surroundings[1].

Zacarro et al. proposed a solution for real-time gun detection which can help in solving crime. This paper focuses on the detection of small objects in the frame. The synthetic and real-time datasets were made for training and testing purposes. The focus was to balance precision and inference speed. FPN feature extractor was considered due to the small sizes of weapons in frames. The model was trained using the Faster RCNN FPN algorithm. The proposed methodology in this paper also aims to maintain the ratio for precision and speed[2].

Apoorva.P et al through their paper have devised an automated system for criminal Identification. The implementation focuses on identifying people in real-time using OpenCV libraries. Haar Cascade Classifier is used for tracking the images and detecting faces. The faces detected are passed to a pre-trained model for face recognition. The training is focused on even identifying faces in different rotations and angles. The system can efficiently identify multiple faces in one frame. The system in this paper was designed to detect and recognize multiple faces all at once[3].

Li Shixin et al. have developed a technique for object detection based on neural network models. The paper works with an SSD object detector and MobileNet model, a pre-trained deep learning model. All the parameters in the SSD method are explained in detail. The algorithm detects multiple objects in a single shot. It detects objects present in the dataset defined by third-party trained models. The classification of an object is based on the presence of small objects in boxes that are united to give the final output. The SSD MobileNet model was considered for weapon detection in the proposed framework[4].

Harsh Jain et al. have devised a method for real-time weapon detection using artificial intelligence. The CNN models based on SSD and Faster RCNN are trained. COCO dataset and a customized dataset are used for training and testing. The results are tabulated and compared for these two models. The Faster RCNN model is found to give slightly better accuracy compared to the SSD Model. The SSD Model is much better in terms of speed. Overall, the SSD Model was found to give better results[6].

Wassima Aitfares, Abdellatif Kobbane and Abdelaziz Kriouile et al. state have developed a low-cost, efficient, and artificial intelligence-based solution to for detecting suspicious behaviour of moving people. The system utilizes analysis of object trajectory, working on the object motion vector. Once a suspicious behaviour suddenly occurs in this trajectory, we segment and track this object during its motion within the camera's field of view [16].

After thorough research on Crowd Density Analysis and Suspicious Activity Detection; Shriya Akella, Privanka Abhang, Vinit Agraharkar and Reena Sonkusare et al. propose an extensive system that makes use of the YOLOv3 algorithm to detect potential suspicious actions in a specific radius. Based on the detection, segmentation and captioning, the algorithm classifies a frame as suspicious or normal. Crowd density has been calculated by detecting the number of people in a frame and suspicion detection has been performed by analysing a frame for suspicious objects like isolated bags, knives and guns [15].

III. PROPOSED SYSTEM

A. System Overview

The system designed in this paper aims to make our society secure by detecting crime in realtime. The development of this framework is based on deep learning models. The live input is captured from a webcam using OpenCV libraries. The frames captured at a speed of around 30 fps are fed to the model. The model will recognise faces and compare them with a criminal database. It will also detect objects and identify if there are any weapons. The frames are analysed to detect suspicious human behaviour. The proposed method is mainly divided in four parts: Face Detection and Recognition, Object Detection, Suspicious Behavior Prediction, Identifying Crime using Multiprocessing.

B. System Architecture

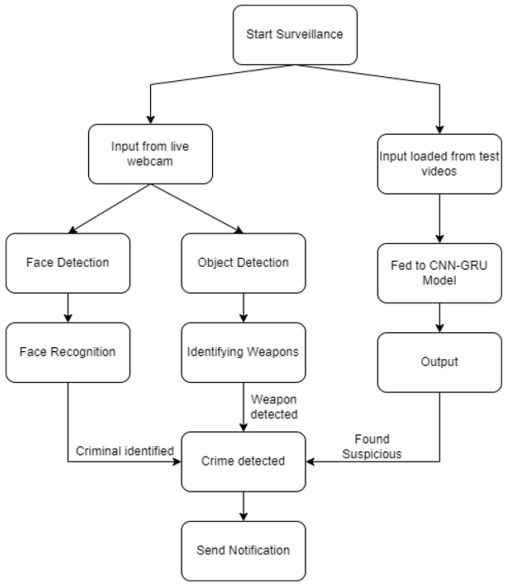


Figure 1. System Architecture

1. Face Detection and Recognition

The first step of identifying criminals is to detect faces in the live feed from a webcam. The popular Haar Cascade Classifier proposed by Viola and Jones is used here[11]. It is just like other object detection algorithms but specially designed for face detection. The model was trained using millions of positive images (which consist of faces) and negative images(which do not consist of faces). The first stage in the process is the calculation of haar features which detect edges and lines in the image. After this, the integral images are created which are just small rectangles. This is done to speed up the process of calculating haar features. Lastly, Adaboost training uses n number of weak classifiers to make one strong classifier that determines the best features. In this designed framework we have used this pre-trained model using OpenCV functions CascadeClassifier(). The xml file used here was haarcascade_frontface_default. Each frame captured is fed to the system to detect human faces in the frame. The faces detected are highlighted with a blue rectangle. The image is further processed for the recognition step if any faces are detected.

To recognize the people from live input, a dataset of known faces needs to be there. A custom dataset was generated using a webcam to capture the faces of around 10 people. For each person, 30 images were captured and stored with their names. Important features are extracted from these images for unique identities. A neural network is trained to generate an output vector from the cropped faces. This output vector highlights the important features of the face. A pre-trained model available in the dlib library in python is used for the implementation. The dataset is passed through this model to generate face embeddings. The embedding data for all the images are stored in an enc file which will be used for comparison in further steps.

After the detection of the face, the rectangular area around the face is cropped and fed to the face recognition system. The new input face is passed through an embedding model to generate an output vector of dimension 128x1. These particular face_embeddings are compared to enc files using face_recognition api in python [12]. The face_recognition.compare_faces() method is based on euclidean distance calculation between two faces. If the value is greater than 0.6 typically the match is found. The output is displayed with the identity of that person.

2. Object Detection

The detection of hazardous weapons is also a very integral part of crime detection. A lot of research is going on in the field of object detection and many models have been developed for recognizing real-life objects in videos and images. In this paper, the focus is on detecting weapons that can be a threat to people around. Dasci Weapon Dataset was primarily used to train the models. This is a very popular dataset to differentiate between weapons and everyday objects. It enabled us to train the system for the detection of around 91 weapons of mass destruction.

During the process, different object detection models were studied. The RCNN algorithm works in two steps, regions are determined and then objects are detected from those regions. Faster RCNN is comparatively faster than RCNN due to direct input of the entire image for feature mapping [9].YOLO models work on the single-shot technique and give much better results. But this model is designed for computers based on GPU and does not work well on CPU. SSD Mobilenet was used for the development of this system. It gives the best result in terms of both accuracy and execution time [10]. As per the description in the figure, VGG 16 is seen to be the base network for the architecture. In addition to this multiple convolutional layers are added towards the end with decreasing sizes. The accuracy of the model is higher due to the detection process which takes in multiple scales.

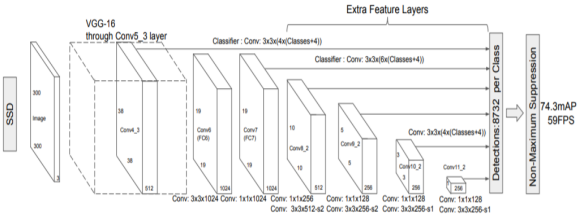


Figure 2. SSDMobileNet Architecture

The data is used to generate corresponding weights and configuration files from the dataset images. Non-maximum suppression techniques are used while processing the frames [14].This technique helps in avoiding multiple suggestions for the same object in the frame. The video is split into several frames and the data of each frame is passed through to the SSDMobileNet model along with the configurations. If the confidence value of the detection is greater than 0.65, then it is displayed onto the screen.

3. Suspicious Behavior Prediction

a. Dataset

The Dataset used for this module was the CAVIAR Dataset. This contains video clips for 6 different human actions. It uses XML to store its annotation of the .mpg videos The dataset contains a role and context, based on which it can be found out whether the action is suspicious or not. The XML file had to be parsed and mapped with each frame of the video. The video was split into each frame and the data of each frame was extracted from the XML file. The processed data was categorized as suspicious vs not suspicious. Then we stored the train and test data in .npy files as loading times are better in .npy files compared to .txt and .csv files.

b. Model Description

Convolutional Neural Networks usually take three-dimensional data i.e., width, height, and channels. This type of framework will only consider the current frame for predictions. This will give highly unstable results as a single frame can't be used to analyze complete human action. A specific number of frames need to be analyzed together to understand the relationship to make a correct prediction. This cannot be possible in normal CNNs and hence we require GRUs. GRUs can hold information about frames for some time which can help us give accurate results [19]. The architecture of the model is as follows - ResNet 50 layer, Time Distributed Unit, Gated Recurrent Unit, two dropout layers, and two dense layers. ResNet is a 50 layered deep

model used for various tasks in Computer Vision [18]. The pre-trained Resnet50 model is loaded using Keras. The input shape is 5-dimensional with the number of batches, number of images, width, height, channels. The different models were trained by replacing GRUs with LSTM and RNN [13]. The models were trained using TPUs from Google Colab for 50 epochs. The model described in fig, was found to give the best performance with high accuracy.

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 10, 224, 224, 3)] 0
_________________________________________________________________
time_distributed (TimeDistri (None, 10, 2048)          23587712
_________________________________________________________________
gru (GRU)                    (None, 10, 512)           3933696
_________________________________________________________________
dropout (Dropout)            (None, 10, 512)           0
_________________________________________________________________
dense (Dense)                (None, 10, 1024)          525312
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 1024)          0
_________________________________________________________________
dense_1 (Dense)              (None, 10, 1)             1025
=================================================================
Total params: 28,047,745
Trainable params: 4,460,033
Non-trainable params: 23,587,712
```

Figure 3. ResNet-GRU Model Summary

The test input data was loaded into the framework using OpenCV. Each frame was converted to .npy files. These npy files were passed to the trained model. It captured the data of 10 frames at a time and predicted the data as suspicious and not suspicious accordingly.

4. Identifying Crime using Multiprocessing

The three components of this framework were built individually. In the final step, these models are integrated for detecting crime in a single step. The live input is taken from a webcam and fed to the face and object recognition system. Some sample video clips are loaded into a framework for suspicious behavior analysis for testing purposes. The three algorithms are executed in parallel using the concept of multiprocessing to compute the results. This significantly helps in improving the performance of the systems. The computational tasks run on separate threads. It was implemented using python. The multiprocessing queue is implemented to share a webcam for input. If any criminal is recognized, a weapon is detected or actions are predicted to be suspicious, the frame is labeled for possible criminal activity. Alert is generated and concerned authorities are notified.

IV. Results and Observations

1. Object Recognition

The trained SSD Mobilenet model was found to detect 91 weapons from the DaSCI dataset. The SSD MobileNet model is known for its fast execution time and observed to give results with very high inference speed. The model was tested using input set of 5 different objects. The test set was made up of 20 samples each of some hazardous and daily life objects. Table 1 shows the no. of correct and incorrect predictions for each object. The overall accuracy obtained was 82%.

Weapon	Correct Predictions	Incorrect Predictions
Knife	17	3
Gun	16	4
Watch	15	5
Scissor	18	2
Pen	16	4

Table 1. Accuracy of weapons

For live testing, the input frames are captured from the video and fed to the system. The hazardous objects and common real-life objects are highlighted in a rectangle with their titles. The blade detected in figure 4, is highlighted with its label

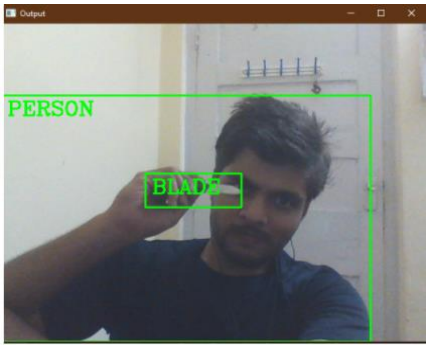


Figure 4. Detection of Blade

2. Face Recognition

The system is tested for live input from a webcam using OpenCV. It is observed to detect all the faces in the frame. The known person to the system is labeled with the name and the unknown person remains unlabelled. The faces detected are highlighted with a red rectangle as seen in figure 5. The model works with high accuracy and very little delay time.

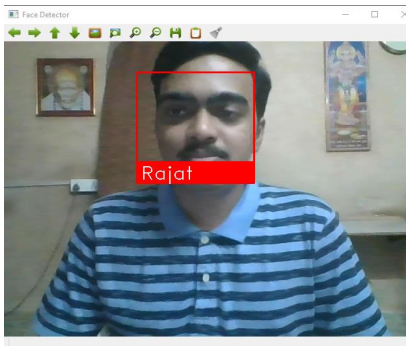


Figure 5. Criminal Detected

3. Suspicious Behavior Prediction

The model is able to detect relations between the frames and provides output only by processing a series of actions.

In other words, the model does not label the action as suspicious based on one frame. It takes into consideration a series of frames to make its prediction. In the test case there is a clip of people fighting and the frame is labelled as suspicious. The results of all the three models on the test dataset after training for 50 epochs is compared in a table 1.

Model	Loss	Accuracy
GRU	9.65%	95.97%
LSTM	17.15%	94.68%
RNN	18.48%	92.99%

Table 2. Comparison of Models

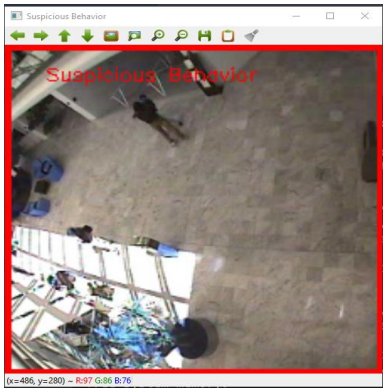


Figure 6. Fighting Detected

4. Identifying Crime using Multiprocessing

The three models are executed in parallel. This increases the speed of output generation when compared to running each model individually. The models run simultaneously in separate frames and share a webcam to take input. If any frame is found to be suspicious, an email is sent to the concerned authorities.

Model	With Multiprocessing	Without Multiprocessing
Crime Prediction	24 fps	15 fps
Object Detection	30 fps	22 fps
Face Recognition	23 fps	18 fps

Table 3. Advantages of multiprocessing

These tests were carried out on an Intel core i5- 8300H 8th Gen processor with an 8GB RAM , which clearly indicate that with multiprocessing we were able to process more frames per second with Object Detection SSD model achieving up to 30 fps.

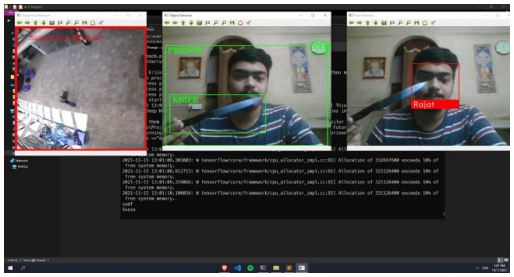


Figure 7. Models executed parallely

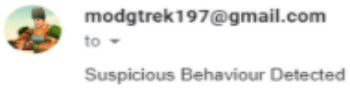


Figure 8. Notification

V. Conclusion

In this paper, a crime detection framework was properly designed and implemented. The three components of the system face recognition, object identification, and suspicious behavior detection were tested individually and we obtained highly accurate results with little execution time. Comparing faces based on similarity learning gave the best results for face recognition with minimum errors. The model correctly detected faces in the frame. The SSD Mobilenet Model detected weapons with an accuracy of 81%. LSTM, GRU, and RNN were compared for training the suspicious behavior detection model. GRU was found to give the best accuracy of 95.97% and integrated into the final step. The integration of the algorithms using multiprocessing was the integral step in the prediction of criminal activities. The multiprocessing helped increase the inference speed of these two models running at the same time and sharing the same resources. The final model successfully managed to detect the suspicious activities in a single input frame. The output was displayed on the screen and an email was sent to the police authorities.

VI. Future Scope

The system proposed in this paper has a few limitations. Face recognition and object detection work well when the distance between the object of interest and the input camera is considerably less. In real-world scenarios usually, this distance is relatively more and the model developed in this paper wouldn't be an ideal choice for crime prediction. The model designed for suspicious behaviour can be designed to detect different possible crime situations. In the future, we aim to design with a few additions the system will become more efficient to predict crime in different situations.

VII. REFERENCES

[1] Swathi BN, Vijaya K. "A Framework for Crime Prediction and Analysis in Real-Time for Smart Home"International Research Journal of Engineering and Technology Volume: 07 Issue: 12 | Dec 2020.

[2] Jose L. Salazar González, Carlos Zaccaro, Juan A. Álvarez-García, Luis M. Soria Morillo, Fernando Sancho Caparrini, Real-time gun detection in CCTV: An open problem, Neural Networks, Volume 132, 2020, Pages 297-308, ISSN 0893-6080, https://doi.org/10.1016/j.neunet.2020.09.013.

[3] Apoorva., P., Impana., H. ., Siri., S. ., Varshitha., M. ., & Ramesh., B. (2019). Automated Criminal Identification by Face Recognition using Open Computer Vision Classifiers. 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC). doi:10.1109/iccmc.2019.8819850

[4] Younis, Ayesha & Shixin, Li & Jn, Shelembi & Hai, Zhang. (2020). Real-Time Object Detection Using Pre-Trained Deep Learning Models MobileNet-SSD CCS Concepts •Computing methodologies→Artificial intelligence→ Computer vision→Computer vision problems→Object detection Keywords. 978-1-4503-7673-0. 44-48.

[5] Nguyen, Hieu & Bai, Li. (2010). Cosine Similarity Metric Learning for Face Verification. ACCV. 6493. 709-720. 10.1007/978-3-642-19309-5_55.

[6] Jain, Harsh & Vikram, Aditya & Mohana, Mohana & Kashyap, Ankit & Jain, Ayush. (2020). Weapon Detection using Artificial Intelligence and Deep Learning for Security Applications. 193-198. 10.1109/ICESC48915.2020.9155832.

[7] S. T. Ratnaparkhi, A. Tandasi and S. Saraswat, "Face Detection and Recognition for Criminal Identification System," 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2021, pp. 773-777, doi: 10.1109/Confluence51648.2021.9377205.

[8] Handalage, Upulie & Kuganandamurthy, Lakshini. (2021). Real-Time Object Detection using YOLO: A review. 10.13140/RG.2.2.24367.66723.

[9] Ren, Shaoqing & He, Kaiming & Girshick, Ross & Sun, Jian. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 39. 10.1109/TPAMI.2016.2577031.

[10] Liu, Wei & Anguelov, Dragomir & Erhan, Dumitru & Szegedy, Christian & Reed, Scott & Fu, Cheng-Yang & Berg, Alexander. (2016). SSD: Single Shot MultiBox Detector. 9905. 21-37. 10.1007/978-3-319-46448-0_2.

[11] Ahad, Md. Atiqur Rahman & Paul, Tanmoy & Shammi, Ummul & Kobashi, Syoji. (2018). A Study on Face Detection Using Viola-Jones Algorithm for Various Backgrounds, Angels and Distances. Applied Soft Computing.

[12] Emami, Shervin & Suciu, Valentin. (2012). Facial Recognition using OpenCV. Journal of Mobile, Embedded and Distributed Systems. 4.

[13] Girshick, Ross & Donahue, Jeff & Darrell, Trevor & Malik, Jitendra. (2013). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 10.1109/CVPR.2014.81.

[14] Rothe, Rasmus & Guillaumin, Matthieu & Van Gool, Luc. (2015). Non-Maximum Suppression for Object Detection by Passing Messages between Windows. LNCS. 9003. 10.1007/978-3-319-16865-4_19.

[15] C. V. Amrutha, C. Jyotsna and J. Amudha, "Deep Learning Approach for Suspicious Activity Detection from Surveillance Video," 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2020, pp. 335-339, doi: 10.1109/ICIMIA48430.2020.9074920.

[16] W. Aitfares, A. Kobbane and A. Kriouile, "Suspicious behavior detection of people by monitoring camera," 2016 5th International Conference on Multimedia Computing and Systems (ICMCS), 2016, pp. 113-117, doi: 10.1109/ICMCS.2016.7905601.

[17] M. Sajjad et al., "A Novel CNN-GRU-Based Hybrid Approach for Short-Term Residential Load Forecasting," in IEEE Access, vol. 8, pp. 143759-143768, 2020, doi: 10.1109/ACCESS.2020.3009537.

[18] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[19] A. Atassi, I. El Azami and A. Sadiq, "The new deep learning architecture based on GRU and word2vec," 2018 International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), 2018, pp. 1-3, doi: 10.1109/ICECOCS.2018.8610611.

8 Plagiarism Report

Project

ORIGINALITY REPORT

5%	3%	1%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Sardar Patel College of Engineering Student Paper	3%
2	ksvuniversity.org.in Internet Source	<1%
3	Submitted to University of Sunderland Student Paper	<1%
4	www.mytechb.com Internet Source	<1%
5	aktu.ac.in Internet Source	<1%
6	oro.open.ac.uk Internet Source	<1%
7	Özertem, Kemal Arda. "A fast automatic target detection method for detecting ships in infrared scenes", Automatic Target Recognition XXVI, 2016. Publication	<1%

References

[1] Amrutha, C.V and Chandran, Jyotsna and Joseph, Amudha. (2020). "Deep Learning Approach for Suspicious Activity Detection from Surveillance Video". 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 335-339. 10.1109/ICIMIA48430.2020.9074920.

[2] Y. Jung and Y. Yoon, "Behavior tracking model in dynamic situation using the risk ratio EM," 2015 International Conference on Information Networking (ICOIN), 2015, pp. 444-448, doi: 10.1109/ICOIN.2015.7057942.

[3] Shah, N., Bhagat, N. and Shah, "Crime forecasting: a machine learning and computer vision approach to crime prediction and prevention".VCIBA Springer Open Art 4, 9 (2021).

[4] Swathi BN, Vijaya K. "A Framework for Crime Prediction and Analysis in Real-Time for Smart Home"International Research Journal of Engineering and Technology Volume: 07 Issue: 12 — Dec 2020.

[5] Malavika Nair, Mathew Gillroy, Neethu Jose, Jasmy Davies "i-SURVEILLANCE CRIME MONITORING AND PREVENTION USING NEURAL NETWORKS"International Research Journal of Engineering and Technology Volume Volume: 05 Issue: 03 — Mar-2018.

[6] Kakadiya, R., Lemos, R., Mangalan, S., Pillai, M., and Nikam, S. (2019). AI Based Automatic Robbery/Theft Detection using Smart Surveillance in Banks. 2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA). doi:10.1109/iceca.2019.8822186

[7] Grega, M., Matiolański, A., Guzik, P., and Leszczuk, M. (2016). Automated Detection of Firearms and Knives in a CCTV Image. Sensors, 16(1), 47. doi:10.3390/s16010047

[8] Salazar González, J. L., Zaccaro, C., Álvarez-García, J. A., Soria Morillo, L. M., and Sancho Caparrini, F. (2020). Real-time gun detection in CCTV: An open problem. Neural Networks, 132, 297–308. doi:10.1016/j.neunet.2020.09.013

[9] Lee, K. B., and Shin, H. S. (2019). An Application of a Deep Learning Algorithm for Automatic Detection of Unexpected Accidents Under Bad CCTV Monitoring Conditions in Tunnels. 2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML). doi:10.1109/deep-ml.2019.00010

[10] Apoorva., P., Impana., H. ., Siri., S. ., Varshitha., M. ., and Ramesh., B. (2019). Automated Criminal Identification by Face Recognition using Open Computer Vision Classifiers. 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC). doi:10.1109/iccmc.2019.8819850

[11] Butt, U. M., Letchmunan, S., Hassan, F. H., Ali, M., Baqir, A., Sherazi, H. H. R. (2020). Spatio-Temporal Crime HotSpot Detection and Prediction: A Systematic Literature Review. IEEE Access, 1–1. doi:10.1109/access.2020.3022808

[12] Jahier Pagliari, D., Chiaro, R., Macii, E., Poncino, M. (2020). CRIME: Input-Dependent Collaborative Inference for Recurrent Neural Networks. IEEE Transactions on Computers, 1–1. doi:10.1109/tc.2020.3021199

[13] Ashby, M. P. J. (2019). Studying crime and place with the crime open database. Research Data Journal for the Humanities and Social Sciences,4(1), 65–80. https ://doi.org/10.1163/24523 666-00401 007.

[14] Lin, Y.-L., Chen, T.-Y., Yu, L.-C. (2017). Using Machine Learning to Assist Crime Prevention. 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI). doi:10.1109/iiai-aai.2017.46

[15] Piza, EL, Welsh, BC, Farrington, DP, Thomas, AL. CCTV Surveillance for crime prevention: A 40-year systematic review with meta-analysis. Criminology Public Policy. 2019; 18: 135– 159. https://doi.org/10.1111/1745-9133.12419

[16] https://lucid.app/documents/dashboard

[17] https://docs.opencv.org/3.4.15/javadoc/org/opencv/videoio/VideoCapture.html

[18] https://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/

[19] https://docs.opencv.org/4.5.2/

[20] https:/https://keras.io/

[21] https://www.tensorflow.org/apidocs/

[22] Piza, EL, Welsh, BC, Farrington, DP, Thomas, AL. CCTV Surveillance for crime prevention: A 40-year systematic review with meta-analysis. Criminology Public Policy. 2019; 18: 135– 159. https://doi.org/10.1111/1745-9133.12419

[23] Emami, Shervin Suciu, Valentin. (2012). Facial Recognition using OpenCV. Journal of Mobile, Embedded and Distributed Systems. 4.

[24] Liu, Wei Anguelov, Dragomir Erhan, Dumitru Szegedy, Christian Reed, Scott Fu, Cheng-Yang Berg, Alexander. (2016). SSD: Single Shot MultiBox Detector. 9905. 21-37. 10.1007/978-3-319-46448-02.

[25] Nguyen, Hieu Bai, Li. (2010). Cosine Similarity Metric Learning for Face Verification. ACCV. 6493. 709-720. 10.1007/978-3-642-19309-555.

[26] Jain, Harsh Vikram, Aditya Mohana, Mohana Kashyap, Ankit Jain, Ayush. (2020). Weapon Detection using Artificial Intelligence and Deep Learning for Security Applications. 193-198. 10.1109/ICESC48915.2020.9155832.

[27] Younis, Ayesha Shixin, Li Jn, Shelembi Hai, Zhang. (2020). Real-Time Object Detection Using Pre-Trained Deep Learning Models MobileNet-SSD CCS Concepts •Computing methodologiesArtificial intelligence Computer visionComputer vision problemsObject detection Keywords. 978-1-4503-7673-0. 44-48.

[28] S. T. Ratnaparkhi, A. Tandasi and S. Saraswat, "Face Detection and Recognition for Criminal Identification System," 2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence), 2021, pp. 773-777, doi: 10.1109/Confluence51648.2021.9377205.

[29] https://face-recognition.readthedocs.io/en/latest/