

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
|  | TEAM : - CODE | | |  |
|  |  | | |  |
|  | |  |  | |
|  | | Team Members  1. Riya Muskan 2. Kumar Snehal 3. Sahil Ali 4. Ayan Sen  Submitting for Rajasthan Police Hackathon 1.0 |  | |

|  |  |
| --- | --- |
| **Chapter** | **Page No.** |
| Problem Address & Proposed Solution and Overview | 1 |
| Algorithm’s Definition | 1-4 |
| Object tracking Algorithm | 5-6 |
| Crime Detection Algorithm | 6-8 |
| Facial Recognizer Algorithm | 9 |
| Vehicle Tracker Algorithm | 10 |
| X v0.0.1 Algorithm | 11-13 |

**Problem Addressed**

Till now Law Enforcements have to search for criminals/suspects in each CCTV cameras and investigate from person to person which definitely gives time to criminals to run away, resulting in delay of the Day of Justice.

**Proposed Solution**

**Overview**

An Ai Vision Camera, Mr. X, which analyzes the environment, predicts crime, generate an alert and tracks the suspect within its vision. Now if the camera is able to get facial data and vehicle number plate of the suspect, then it will store these data in the server and starts a Search Operation in all camera with same software installed.

Now all the cameras that recognize the suspect’s face or same vehicle number plate will give the path history of the suspect, allowing Law enforcement officers to catch the suspect as soon as possible.

**Algorithm’s Definition**

Programming Laguages Used: - Python and C++

Algorithms used: - YOLO, DeepSORT, Kalman Filter, CNN + LSTM

IDE – Visual Studio Code

##### YOLO

YOLO is an object detection algorithm that can detect and locate objects in images and video frames. It's known for its speed and accuracy and is widely used in various applications, including surveillance, autonomous vehicles, and more. The primary idea behind YOLO is to perform object detection in a single pass through the neural network, making it very efficient for real-time processing.

##### DeepSORT

DeepSORT is an extension of the SORT (Simple Online and Realtime Tracking) algorithm, which is designed to track objects in video streams, such as surveillance cameras. The key enhancement in DeepSORT is the use of deep neural networks to improve tracking accuracy and robustness.

##### Kalman Filter

The Kalman Filter is an optimal recursive algorithm for estimating the state of a linear dynamic system. It's particularly useful when you have a system that evolves over time and is observed through noisy sensors. The filter predicts the current state of the system and updates it as new measurements become available

We are using a combination of YOLO and DeepSORT algorithm with Kalman Filter, to build an algorithm to track humans and vehicles in a Video.

##### OCR

Optical Character Recognition (OCR) is a technology that converts printed or handwritten text and characters into machine-encoded text.

##### ConvLSTM or CNN + LSTM

CNN (Convolutional Neural Network):

CNNs are well-known for their ability to extract spatial features from images. They consist of convolutional layers that learn filters to detect patterns like edges, textures, and more complex visual features.

LSTM (Long Short-Term Memory):

LSTMs, on the other hand, are recurrent neural networks (RNNs) that are excellent at handling sequential data. They can capture long-range dependencies and store information over time.

Why using combination of CNN and LSTM:

* Feature Extraction: You can use a CNN to extract features from video frames or image sequences. This is important for identifying objects, including suspects, in each frame.
* Temporal Modeling: Once you have extracted spatial features using the CNN, you can feed these features into an LSTM network. The LSTM can model the temporal dependencies and track the movement of objects or suspects across frames.
* Object Tracking: The LSTM can maintain a memory of past frames and use it to predict the location of objects, allowing you to track suspects as they move through the camera's field of view.
* Sequence Classification: If your goal is to classify certain actions or behaviors of suspects over a sequence of frames, the LSTM can be used for sequence classification tasks.

We are using ConvLSTM Model to create a pre-trained Crime Detection model.

#### Object Tracking Algorithm(YOLO + DeepSORT with Kalman filter)

Libraries used: -

YOLOv8

DeepSort

Tensorflow

Scikit

PyTorch

Opencv-python

Dataset: -

Coco dataset

|\_\_Yolov3.weights

|\_\_Yolov3.cfg

|\_\_Yolov8n.pt

Weight file is a pre-trained model file on coco dataset (330,000 images)

Algorithm

# Objects unique IDs array

objects = []

# Initialize the Kalman Filter for each object

for object in objects:

    object.initialize\_kalman\_filter()

# Process video frames

for frame in video\_frames:

    # Detect objects in the frame

    detected\_objects = detect\_objects(frame)

    # Update tracked objects with new detections

    for object in objects:

# Predict the object's new position

        object.predict()

# Associate the object with detected ojects

        object.update\_association(detected\_objects)

    # Create new tracking objects for unmatched detections

    new\_objects = create\_new\_objects(detected\_objects, objects)

    # Add new objects to the list of tracked objects

    objects.extend(new\_objects)

    # Remove lost objects

    objects = remove\_inactive\_objects(objects)

    # Post process tracked objects

    post\_process\_objects(objects)

    # Draw bounding boxes on the frame for visualization

    draw\_objects\_on\_frame(frame, objects)

    #save the processed frame

    display\_frame(frame)

##### Crime Detection Algorithm (ConvLSTM = CNN + LSTM)

Libraries Used: -

Tensorflow

Scikit-learn

Tensorflow.keras

Opencv-python

Dataset: -

Video dataset of Gun Shooting, Fighting, Running and smoking with 150 video per class.

Model Creation Algorithm: -

# dataset frame collection array

frame\_list = []

# dataset preprocessing like frame extraction, augmentation and  data balancing,

def data\_preprocessing():

    frame = frame\_extraction()

    frame\_list.append(frame)

    create\_dataset()

    label\_dataset()

# dividing dataset into test set and training set (75% and 25% respectivily)

features\_train, features\_test, labels\_train, labels\_test = train\_test\_split(75%,25%)

# creating Model

def create\_conv\_lstm\_model():

    # use a Sequential model for model construction

    model = Sequential()

    # Define the Model Architecture.

    ########################################################################################################################

    model.add(ConvLSTM2D(filters = 4, kernel\_size = (3, 3), activation = 'tanh',data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True, input\_shape = (SEQUENCE\_LENGTH,

                                                                                      IMAGE\_HEIGHT, IMAGE\_WIDTH, 3)))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 8, kernel\_size = (3, 3), activation = 'tanh', data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 14, kernel\_size = (3, 3), activation = 'tanh', data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 16, kernel\_size = (3, 3), activation = 'tanh', data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    model.add(Flatten())

    model.add(Dense(len(CLASSES\_LIST), activation = "softmax"))

  ########################################################################################################################

    # Display the models summary.

    model.summary()

    # Return the constructed convlstm model.

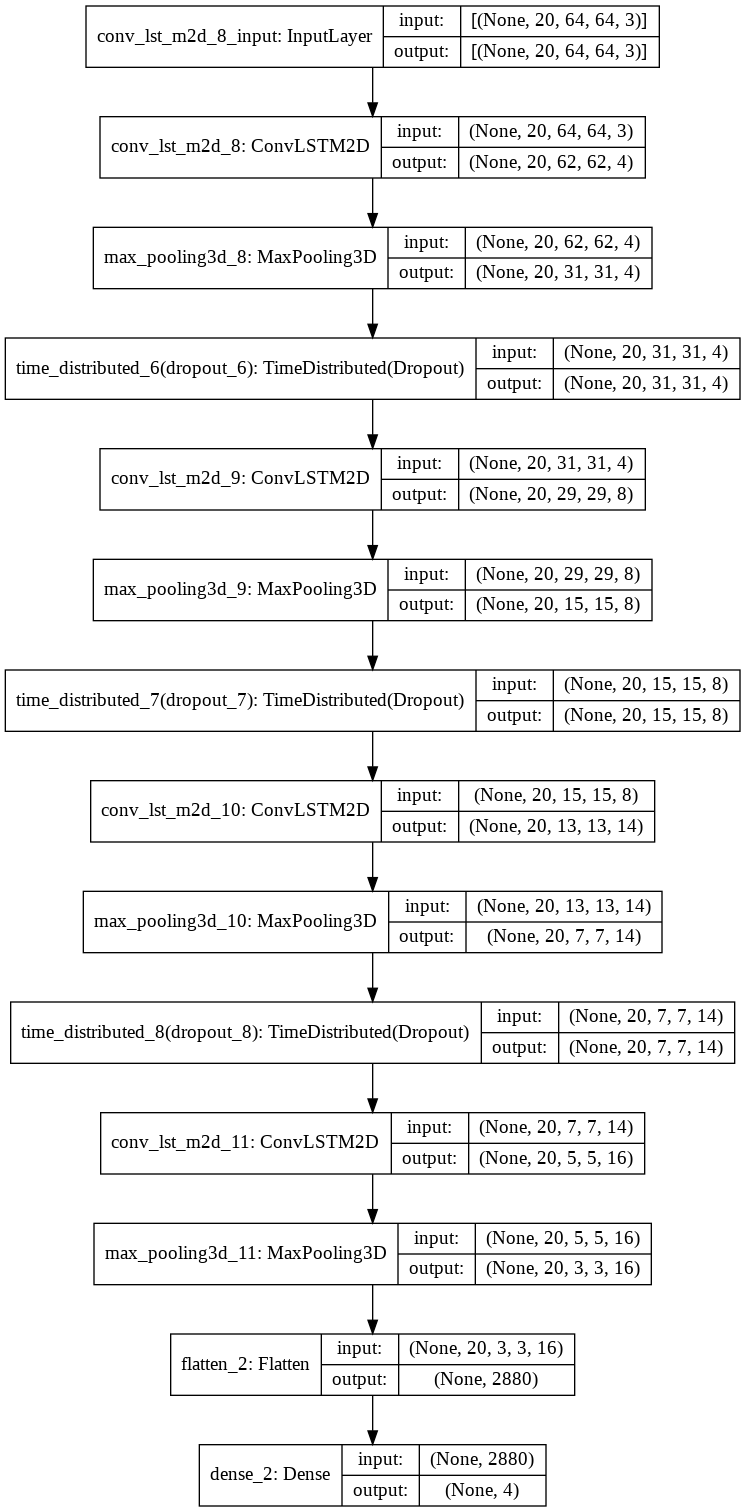
    return model

EarlyStopping(monitor = 'val\_loss', patience = 10, mode = 'min', restore\_best\_weights = True)

convlstm\_model.compile(loss = 'categorical\_crossentropy', optimizer = 'Adam', metrics = ["accuracy"])

convlstm\_model.fit(x = features\_train, y = labels\_train, epochs = 50, batch\_size = 4,shuffle = True, validation\_split = 0.2, callbacks = [early\_stopping\_callback])

##### Model Architecture: -



##### Face Recognizer and data collector Algorithm: -

Libraries Used: -

Opencv-Python

Dataset: -

HaarCascade\_face.xml (pre-trained model file)

Algorithm: -

# Import necessary libraries and load the pre-trained Haar cascade classifier

import cv2

# Load the pre-trained Haar cascade classifier for face detection

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

# Load an image or video frame

frame = cv2.imread('input\_image.jpg')

# Convert the frame to grayscale (Haar cascades work on grayscale images)

gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

# Detect faces in the grayscale frame

faces = face\_cascade.detectMultiScale(gray\_frame, scaleFactor=1.3, minNeighbors=5, minSize=(30, 30))

# stores facial data in local server

Store\_facial\_data()

# Draw rectangles around the detected faces

for (x, y, w, h) in faces:

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Display or save the frame with detected faces

cv2.imshow('Face Detection', frame)

cv2.waitKey(0)

cv2.destroyAllWindows()

##### Vehicle Tracker and data collector Algorithm: -

Libraries: -

Opencv-python

OCR

YOLO

Dataset: -

license\_plate\_detector.pt (image to text reading pre-trained Model)

Algorithm: -

results = {}

path = get\_path()

mot\_tracker = Sort()

# load models

coco\_model = YOLO("yolov8n.pt")

license\_plate\_detector = YOLO("license\_plate\_detector.pt")

# get push data from server to search the number plate

srch\_num = get\_srch()

# load video

cap = cv2.VideoCapture("C:/Users/toufiqhussain/Downloads/test\_2.mp4")

vehicle\_number = []

# read Frame

while True:

frame\_nmr += 1

ret, frame = cap.read()

# detect vehicle

detection = vehicle\_detection()

# track vehicle

tracked\_vehicle\_id += yolo\_track(detection)

# detect number plate on detection

number\_plates = detect\_num\_plate(tracked\_vehicle\_id)

# read number plate

for number\_plate in number\_plates:

num = easyOCR(number\_plates)

vehicle\_number.append(num)

# check if num match, if match then generate alert and add to path

if srch\_num and src\_vhl\_num == num:

alert()

report()

path.append(latitude,longitude)

cap.release()

##### Models Assembling

##### (Object Tracker + Crime Detector + Face recognizer + Vehicle Tracker)

Concept: -

We will use threading concept to run all the algorithms while camera is recording live.

We will save the footage locally at every 10 sec, so now all the algorithms will analyze the video and extract all the important data like all the facial and vehicle’s data that comes in the frame and if crime happens then crime detection model will generate an alert.

Libraries used: -

OpenCV-Python

Threading

Shutil

OS

Algorithm: -

# Crime Detection Model

def crime\_pred(video\_path):

return detection

# Vehicle Tracker Model

def vehicle\_track(video\_path, srch):

store\_data()

if found:

return True

else:

return False

# Facial Recognizer Model

def facial\_recog(video\_path, srch):

store\_data()

if found:

return True

else:

return False

# Run Camera and Start Recording

def main():

# Creating Thread

thread\_1 = threading.Thread(target=crime\_pred)

thread\_2 = threading.Thread(target=vehicle\_track)

thread\_3 = threading.Thread(target=facial\_recog)

# Starting Thread

thread\_1.start()

thread\_2.start()

thread\_3.start()

# starting camera

cap = cv2.VideoCapture(0)

if not cap.isOpened():

print("Error: Could not open camera.")

return

frame\_width = int(cap.get(3))

frame\_height = int(cap.get(4))

fps = int(cap.get(5))

fourcc = cv2.VideoWriter\_fourcc(\*"H264") # Codec for MP4 format

stream\_1 = None

stream\_2 = None

s1\_fileName = None

s2\_fileName = None

recording\_timer = 0

file\_no = 0

# Crime Detection Model

def crime\_pred(video\_path):

return detection

# Vehicle Tracker Model

def vehicle\_track(video\_path, srch):

store\_data()

if found:

return True

else:

return False

# Facial Recognizer Model

def facial\_recog(video\_path, srch):

store\_data()

if found:

return True

else:

return False

# Run Camera and Start Recording

def main():

# Creating Thread

thread\_1 = threading.Thread(target=crime\_pred)

thread\_2 = threading.Thread(target=vehicle\_track)

thread\_3 = threading.Thread(target=facial\_recog)

# Starting Thread

thread\_1.start()

thread\_2.start()

thread\_3.start()

# starting camera

cap = cv2.VideoCapture(0)

if not cap.isOpened():

print("Error: Could not open camera.")

return

frame\_width = int(cap.get(3))

frame\_height = int(cap.get(4))

fps = int(cap.get(5))

fourcc = cv2.VideoWriter\_fourcc(\*"H264") # Codec for MP4 format

stream\_1 = None

stream\_2 = None

s1\_fileName = None

s2\_fileName = None

recording\_timer = 0

file\_no = 0

while True:

ret, frame = cap.read()

# Switching between Threads

if not ret:

print("Error: Could not read frame.")

#join thread if camera closes

thread\_1.join()

thread\_2.join()

thread\_3.join()

break

if recording\_timer == fps \* 0:

stream\_1 = cv2.VideoWriter("path.mp4",

fourcc,

fps,

(frame\_width, frame\_height),

)

if recording\_timer >= fps \* 7 and recording\_timer <= fps \* 17:

stream\_2.write(frame)

if recording\_timer == fps \* 17:

stream\_2.release()

if recording\_timer == fps \* 17:

recording\_timer = 0

else:

recording\_timer += 1

cv2.imshow("Recording", frame)

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

break

cap.release()

cv2.destroyAllWindows()

#join thread if camera closes

thread\_1.join()

thread\_2.join()

thread\_3.join()

if \_\_name\_\_ == "\_\_main\_\_":

main()

Output Example: -



Fig: - Result on CCTV footage of a gun firing in Indore

Conclusion: -

Till now we have created a program to implement it on a camera in real-time, so it can predict if the crime occurs or not, gets the facial data of suspect (if available) and vehicle number plate. It can also search for suspects face data and vehicle number plate if comes in the vision of any camera.

Future of the Project: -

Our program is only use full for tracking if facial data or number plate data is available but what if the suspect hide his face or remove number plate, which happens in most cases.

So, our next target is to build a tracking model that tracks anything without depending on specific data that can be removed to fool the surveillance system and escape.

Most probably we will use a Stereo Vision System and a Reinforcement agent for this purpose.

Links: -

All codes are uploaded on the GitHub. README.md file is provided to instruct how to run the program and implement it.

GitHub Repository link: - <https://github.com/Toufiq0101/ProjectX>

#### Acknowledgments

A special thanks to all team members and our mentor for supporting us to achieve this significant progress.

Credits: -

Sahil Ali – ML in Python Developer

Kumar Snehal – ML in Python Developer

Ayan Sen – Data Analysist

Riya Muskan - ML in Python Developer