Mini Project Report On

SUSPICIOUS ACTIVITY DETECTION

Submitted in partial fulfilment of the requirements for the award of the

Bachelor of Technology

In

Department of Computer Science and Engineering

By

D. Venkata Sai Netaji Nikhil	**21245A0507**
Pilli Maheshwar	**20241A05A6**
Cheema Likith Prem	**20241A0567**
Deepak Majhi	**20241A0573**
Gopu Sai Teja	**20241A0577**

Under the Esteemed guidance of

Dr. P. Chandra Sekhar Reddy Professor

Department of Computer Science and Engineering

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

Mini Project Report On

SUSPICIOUS ACTIVITY DETECTION

Submitted in partial fulfilment of the requirements for the award of the

Bachelor of Technology

In

Department of Computer Science and Engineering

By

D. Venkata Sai Netaji Nikhil	**21245A0507**
Pilli Maheshwar	**20241A05A6**
Cheema Likith Prem	**20241A0567**
Deepak Majhi	**20241A0573**
Gopu Sai Teja	**20241A0577**

Under the Esteemed guidance of

Dr. P. Chandra Sekhar Reddy Professor

Department of Computer Science and Engineering

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

GOKARAJU RANGARAJU

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

CERTIFICATE

This is to certify that the mini project entitled "**Suspicious Activity Detection**" is submitted by **D. Venkata Sai Netaji Nikhil (21245A0507), Pilli Maheshwar (20241A05A6), Cheema Likith Prem (20241A0567), Deepak Majhi (20241A0573), Gopu Sai Teja (20241A0577)** in partial fulfilment of the award of degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during academic year 2022-2023.

INTERNAL GUIDE	HEAD OF THE DEPARTMENT
Dr. G. Srinivas	**Dr. K. MADHAVI**
Associate Professor	**Professor**

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

There are many people who helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First we wish to express our deep gratitude towards our internal guide **Dr. G. Srinivas Assoc Prof,** Department of CSE for her/his support in the completion of our project report. We wish to express our honest and sincere thanks to **Dr. K. Madhavi, HOD, Department of CSE** and to our principal **Dr. J. Praveen** for providing the facilities to complete mini project. We would like to thank all our faculty and friends for their help and constructive criticism during the project completion phase. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

D. Venkata Sai Netaji Nikhil (21245A0507)
Pilli Maheshwar (20241A05A6)
Cheema Likith Prem (20241A0567)
Deepak Majhi (20241A0573)
Gopu Sai Teja (20241A0577)

DECLARATION

We hereby declare that the mini project entitled **“Suspicious Activity detection”** is the work done during the period from **27th Jan 2023 to 25rd May 2023** and is submitted in the partial fulfilment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad).The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

D. Venkata Sai Netaji Nikhil (21245A0507)
Pilli Maheshwar (20241A05A6)
Cheema Likith Prem (20241A0567)
Deepak Majhi (20241A0573)
Gopu Sai Teja (20241A0577)

Table of Contents

NOTE: Paper which is accepted and/or published should attach at last.

ABSTRACT

In response to the increasing instances of fraudulent and offensive activities occurring in real-time, the need for continuous monitoring of CCTV surveillance footage becomes crucial. Human surveillance alone is not feasible due to the sheer volume of footage to be analyzed. Furthermore, it is important to quickly identify and assess frames or sections of the recordings that contain exceptional or suspicious behavior. This project addresses these challenges by utilizing various Deep Learning models, specifically Convolutional Neural Networks (CNN) and Long-term Recurrent Convolutional Networks (LRCN), to detect signs of violence in real-time.

The Deep Learning models are trained on labeled datasets containing examples of violent and non-violent activities. By leveraging the power of CNNs, the models can extract relevant features from video frames, enabling the identification of violent behaviors. LRCNs are employed to capture temporal dependencies and analyze the sequence of frames, providing a comprehensive understanding of the activities.

By implementing these Deep Learning models, the project aims to provide a rapid and automated method for identifying and flagging exceptional or suspicious activities. This technology assists security personnel in efficiently monitoring surveillance footage, enabling them to focus their attention on specific frames or sections of the recordings that require immediate assessment or intervention. Ultimately, the goal is to enhance security measures, facilitate timely response, and improve public safety in real-time surveillance environments.

I. INTRODUCTION

1.1 Existing System

The rise in disruptive and offensive activities has indeed increased the focus on security measures in public places. To address this issue, many locations such as shopping centres, avenues, and banks have started installing CCTVs to ensure the safety of individuals. However, manually monitoring these surveillance cameras for suspicious activities is a challenging task that requires constant attention and human resources.

To overcome this challenge and improve efficiency, there is a growing need to automate the surveillance system with high accuracy. Deep learning algorithms, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), can play a crucial role in automating threat recognition systems. These algorithms can be trained to identify signs of aggression and violence in real-time, effectively filtering out irregularities from normal patterns.

Implementing deep learning algorithms in an automated threat recognition system can significantly reduce the reliance on human monitoring, minimizing both time and labour requirements. It enhances the overall security infrastructure and enables a more proactive approach to identifying and addressing potential threats.

1.1.1 Limitations in Existing System

While the automated threat recognition system using deep learning algorithms offers several advantages, it's important to consider its limitations as well. Here are some potential limitations:

1. Training Data Availability: Deep learning models heavily rely on large and diverse datasets for effective training. Obtaining labelled data that accurately represents various types of threat activities can be challenging and time-consuming. Insufficient or biased training data may impact the system's accuracy and generalizability.

2. Interpretability: Deep learning models, especially complex ones like CNN and RNN, often lack interpretability. It can be challenging to understand the exact reasons behind the model's predictions. This lack of interpretability can hinder trust and acceptance of the system, especially in sensitive security situations.

3. Scalability and Cost: Implementing an automated threat recognition system across a large number of cameras or public spaces can be expensive and resource-intensive. It requires significant computational power and storage capacity to process real-time video feeds from multiple sources simultaneously.

1.2 Proposed System

In Real-time there has been huge amount of fraudulent and offensive activities have been taking place. As majority of places are under CCTV surveillance, and its quite impossible by humans to be on a constant watch over the surveillance footage. Additionally, it’s essential to show which frames and sections of the recording has the exceptional activity, which enables a more rapid assessment of the unusual or suspicious behavior. In this project, we are using different Deep Learning models (CNN and LRCN), for identifying sign of violence in the real-time and performance of this Deep Learning models is also evaluated.

1.2.1 Advantages over Existing System

There are few advantages over existing system:

1. **Deep Learning Models:** The use of deep learning models, such as Convolutional Neural Networks (CNN) and Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN), indicates a sophisticated approach to anomaly detection. These models have demonstrated effectiveness in various computer vision tasks, including image and video analysis.

2. **Real-time Detection:** By utilizing deep learning models, the system can analyse surveillance footage in real-time, enabling immediate detection and response to offensive activities.

3. **Identification of Exceptional Activity:** The system aims to identify specific frames and sections of the recording that contain exceptional or suspicious activity. This targeted approach helps prioritize and focus on relevant footage, facilitating faster assessment and response to unusual behavior.

II. LITERATURE SURVEY

Video surveillance systems have gained significant attention in the field of public security. Researchers and developers have focused on various aspects of these systems, including movement recognition and object tracking, to enhance their effectiveness in monitoring and securing public spaces. I can provide you with an overview of the advancements and techniques in these areas.

To recognize violent or aggressive patterns in real-time surveillance applications, the system may indeed need to perform multiple attempts. Here are a few approaches published in an article called Human Behaviour Recognition Model Based Feature and Classifier Selection [2] by GeGeo, Zhixin Li, Zhan Huan, Ying Chen, Jiuzhen Liang, Bangwen Zhou, Chenhul Dong:

1. Action Recognition: Action recognition techniques can be used to identify aggressive or violent behaviour based on specific patterns of movement. Deep learning models, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), can be trained on labelled video data to classify actions as aggressive or non-aggressive.

2. Behaviour Modelling: The system can learn the normal behavioural patterns of individuals within a given environment. Deviations from these patterns can then be flagged as potentially aggressive or violent actions. This approach often involves training machine learning models on historical data to establish baseline behaviours.

Another article by [3] DaeHyoen Joe and Min-Suk Kim proposed about the Deep Learning Based Sequence Casual Long-Term Recurrent Convolutional Neural Network for Data Fusion Using Video Data. It the case proposed that an RNN, gradient vanishing occurs where the previously hidden state value disappears over time. To redeem a defect, an LSTM with cell state added to hidden-state was devised. An LSTM can be divided into three gates: forget, input, and output, as shown in. The forget gate is determined by the previous cell state and the input gate updates to remember the current state. In addition, the output gate can find output results from both the cell state and input data. Both CNN and LSTM components can be combined into LRCN to learn spatial and temporal complex data representations, and the neural network architecture is powerful for processing sequential data.

III. SOFTWARE REQUIREMENT SPECIFICATION

3.1 1. Purpose and Software requirement

Functional Requirements:

The functional requirements describe the core functionality of the application.

- ➢ Importing the dataset
- ➢ Data cleaning
- ➢ Analyzing the dataset
- ➢ Identifying the patterns
- ➢ Train the Model with the datasets
- ➢ Get the accuracy

Non- Functional Requirements:

Non-functionality system requirements are those that are not specifically related to the functionality that the system provides.

To provide maximum accuracy.

- ➢ Ease of use.
- ➢ Availability
- ➢ Reliability
- ➢ Maintainability

2. Scope of the Project

The scope of this project encompasses the development and implementation of a deep learning-based system for suspicious activity detection in CCTV surveillance systems. The primary focus is on detecting fighting's as suspicious activities while distinguishing walking and running as non-suspicious activities and fighting's as suspicious activities. Collecting a diverse dataset of CCTV surveillance videos that capture instances of fighting's and non-suspicious activities. The dataset should encompass various environments, lighting conditions, and camera angles to ensure the model's robustness.

3. Product function

The main function of this project is to develop a deep learning neural network model that can detect and classify suspicious activities in CCTV surveillance videos. The model's primary task is to identify fighting’s as suspicious activities, while considering walking and running as non-suspicious activities. By analyzing video footage, the model aims to detect the activities and prevent potential conflicts or threats. Ultimately, the main function is to enhance security measures by automating the detection of suspicious activities and improving overall public safety in surveillance environments.

4. User characteristics

The main user characteristic of this project is security personnel. These individuals are responsible for monitoring and ensuring the safety and security of the surveillance area. They rely on the deep learning neural network model for suspicious activity detection to assist them in identifying potential threats and taking appropriate action.

The system's primary objective is to support security personnel in their surveillance efforts, providing them with an additional layer of intelligence and enhancing their ability to maintain public safety. Therefore, the system is designed to be user-friendly and efficient, ensuring that security personnel can easily interpret and act upon the information provided by the system.

5. System Requirement Specification

Hardware Minimum Requirements:

- OS: Windows 10 or above/ Linux/ Max
- CPU: Intel i3 or above /AMD Athlon Processor 1Ghz
- RAM: 4GB RAM or more
- Storage: Needs 32GB minimum

Software Requirements:

- Python 3.5
- Google Colab

3.2 Feasibility Study

The purpose of this project is to leverage deep learning techniques to develop a robust system capable of detecting suspicious activities, particularly fighting's, in CCTV surveillance videos. By categorizing fighting's as suspicious and considering walking and running as non-suspicious activities, the model can contribute to public safety and early intervention in potential conflicts.

The project will involve the following steps:
a) Dataset Acquisition: Collecting a diverse and representative dataset comprising both fighting's and non-suspicious activities captured by CCTV surveillance cameras.
b) Model Development: Implementing a deep learning neural network model using state-of-the-art techniques to train the model on the acquired dataset.
c) Model Evaluation: Assessing the accuracy and performance of the model using evaluation metrics and validation techniques.
d) Integration and Deployment: Integrating the trained model into existing surveillance systems and deploying it in targeted environments.
e) Monitoring and Maintenance: Continuously monitoring the system's performance, addressing any issues, and refining the model to adapt to evolving patterns of suspicious activities.

3.2.1 Technical Feasibility:

- Evaluate the accessibility and reliability of CCTV surveillance videos for training and testing the model.

- Assess the hardware and computing resources required for training and deploying the model efficiently.

- Ensure compatibility with the existing infrastructure and determine any necessary modifications or integration efforts**.**

3.2.2 Economical Feasibility:

- Evaluate the costs associated with hardware, software, data storage, computing resources, personnel, and ongoing maintenance.

-Assess the potential benefits of improved security measures, including reduced incidents, enhanced public safety, and potential cost savings from preventing losses or damages.

3.2.3 Operational Feasibility:

- Evaluate the impact on existing surveillance processes and workflows and identify any required changes or training for personnel.

-Assess the project's scalability to different surveillance setups and environments, including small businesses, residential areas, and local community initiatives.

IV. DESIGN

4.1 Project Description

The project aims to develop a robust and accurate system for detecting suspicious activities in CCTV surveillance videos using deep learning techniques. By leveraging the power of neural network models, the project aims to enhance security measures in various settings such as public spaces, transportation hubs, and commercial establishments.

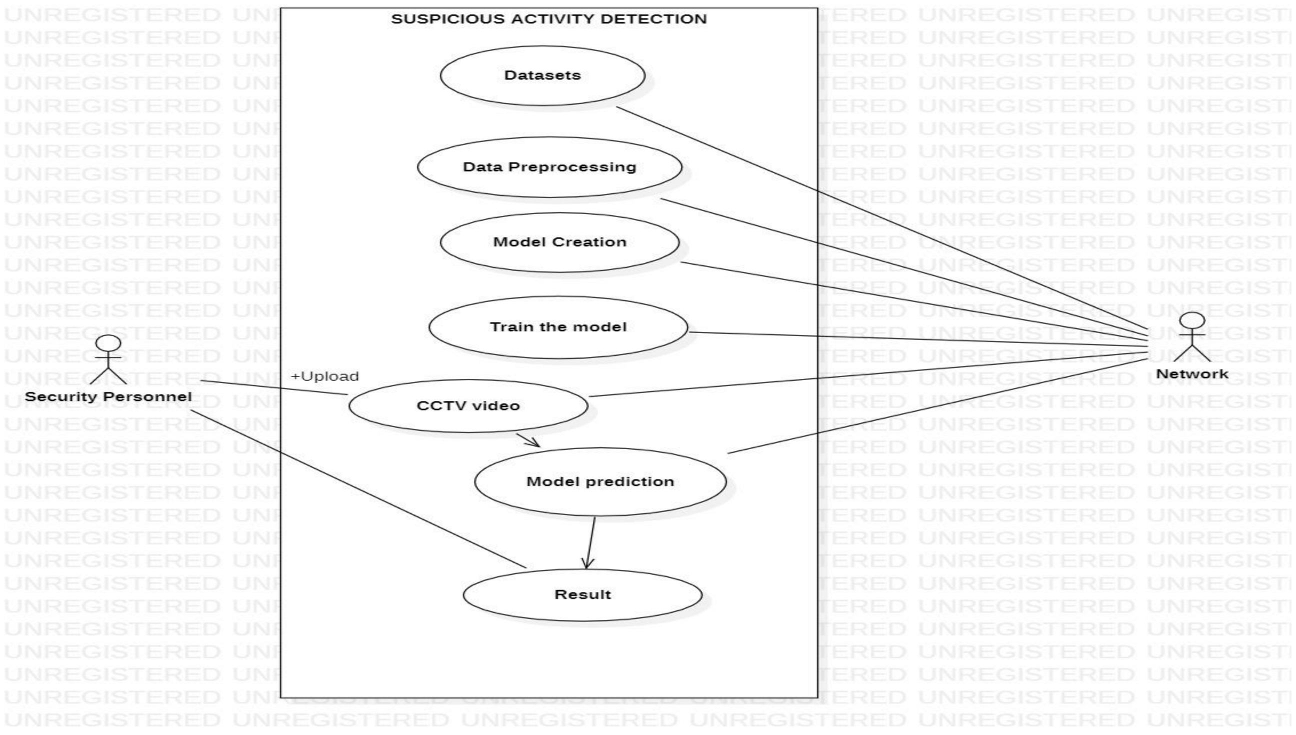
The system will focus on categorizing fighting's as suspicious activities while considering walking and running as non-suspicious activities. The ultimate goal is to enable real-time identification and early intervention in potential conflicts, thus ensuring public safety and preventing harm.

4.1.1 System Use case Diagram

The most common form of system requirements for a software program is a Use Case Diagram In UML. The desired behaviour ("What?") is described in the use cases, not the methods of achieving it ("How?"). Use cases can be displayed both visually and in the form of text (i.e., use case diagram). The potential to design a system from the perspective of the final user is one of the most important concepts int the modelling. This is a useful way to let users know of the system behaviour int their own words by detailing each outwardly apparent software operation. A use case evaluates the necessary cooperation.

- ➢ Use cases are realised by assembling a society of classes and other components that work together to carry out the functionality of a use case.

- ➢ The society of component's static and dynamic structure is modelled.

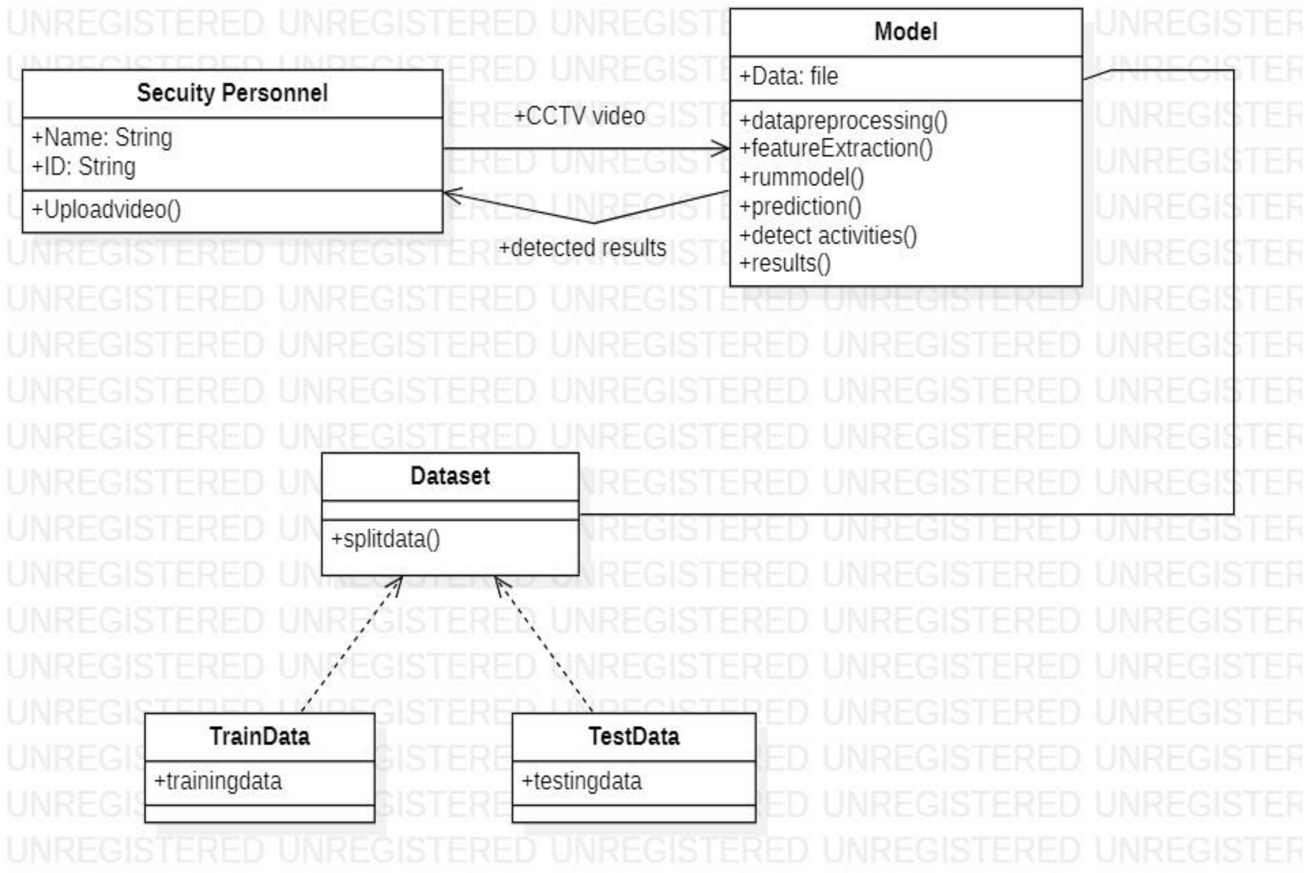


4.1.2 Class Diagram

A class diagram in the Unified Modeling Language (UML) that represents the structure of a system. It shows the system's class, along with their attributes, methods (or operations), and object relationships.

This type of diagrams is particularly useful during the software development process, as it can be directly translated into object-oriented code.

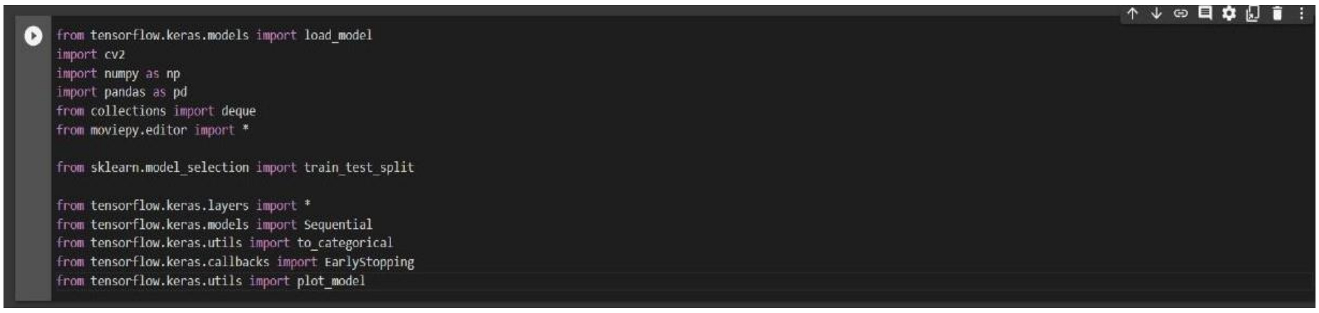
While other UML diagrams, such as activity diagrams and sequence diagrams, focus on the application's flow and sequence, class diagrams emphasize the system's static view. As a result they are frequently used by software developers.



V. IMPLEMENTATION

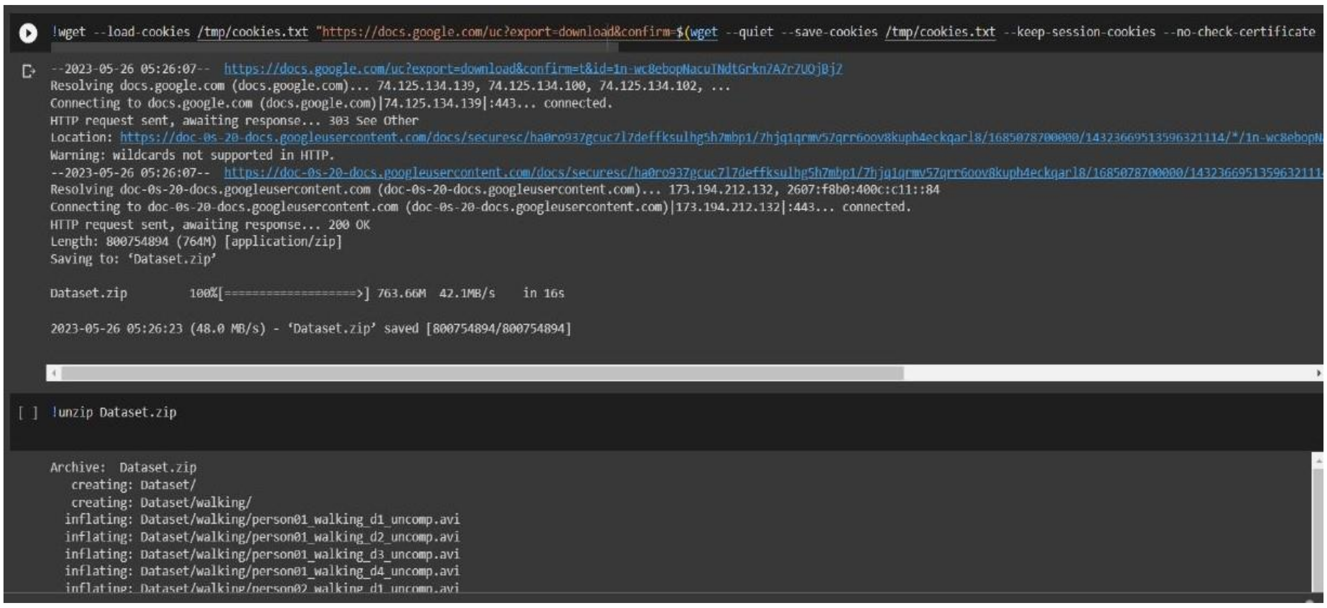
5.1 Source code

Importing libraries



```
from tensorflow.keras.models import load_model
import cv2
import numpy as np
import pandas as pd
from collections import deque
from moviepy.editor import *

from sklearn.model_selection import train_test_split

from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model
```

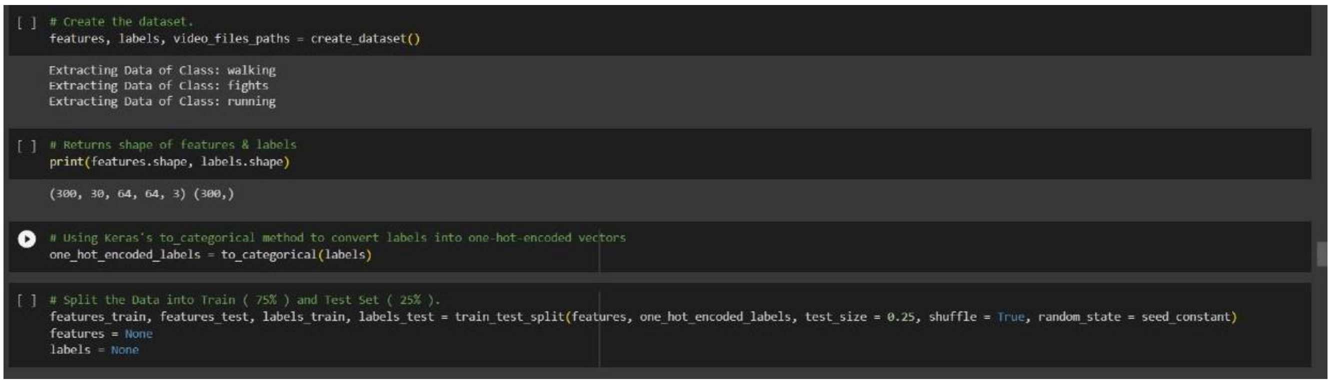
Accessing Datasets



```
!wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies --no-check-certificate

--2023-05-26 05:26:07--  https://docs.google.com/uc?export=download&confirm=t&id=1n-wc8ebopNacuTNdtGrkn7A7r7UOjBj2
Resolving docs.google.com (docs.google.com)... 74.125.134.139, 74.125.134.100, 74.125.134.102, ...
Connecting to docs.google.com (docs.google.com)|74.125.134.139|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-0s-20-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1/7hjq1qrmv57qrr6oov8kuph4eckqarl8/1685078700000/14323669513596321114/*/1n-wc8ebopN
Warning: wildcards not supported in HTTP.
--2023-05-26 05:26:07--  https://doc-0s-20-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1/7hjq1qrmv57qrr6oov8kuph4eckqarl8/1685078700000/1432366951359632111
Resolving doc-0s-20-docs.googleusercontent.com (doc-0s-20-docs.googleusercontent.com)... 173.194.212.132, 2607:f8b0:400c:c11::84
Connecting to doc-0s-20-docs.googleusercontent.com (doc-0s-20-docs.googleusercontent.com)|173.194.212.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 800754894 (764M) [application/zip]
Saving to: ‘Dataset.zip’

Dataset.zip         100%[===================>] 763.66M  42.1MB/s    in 16s

2023-05-26 05:26:23 (48.0 MB/s) - ‘Dataset.zip’ saved [800754894/800754894]
```

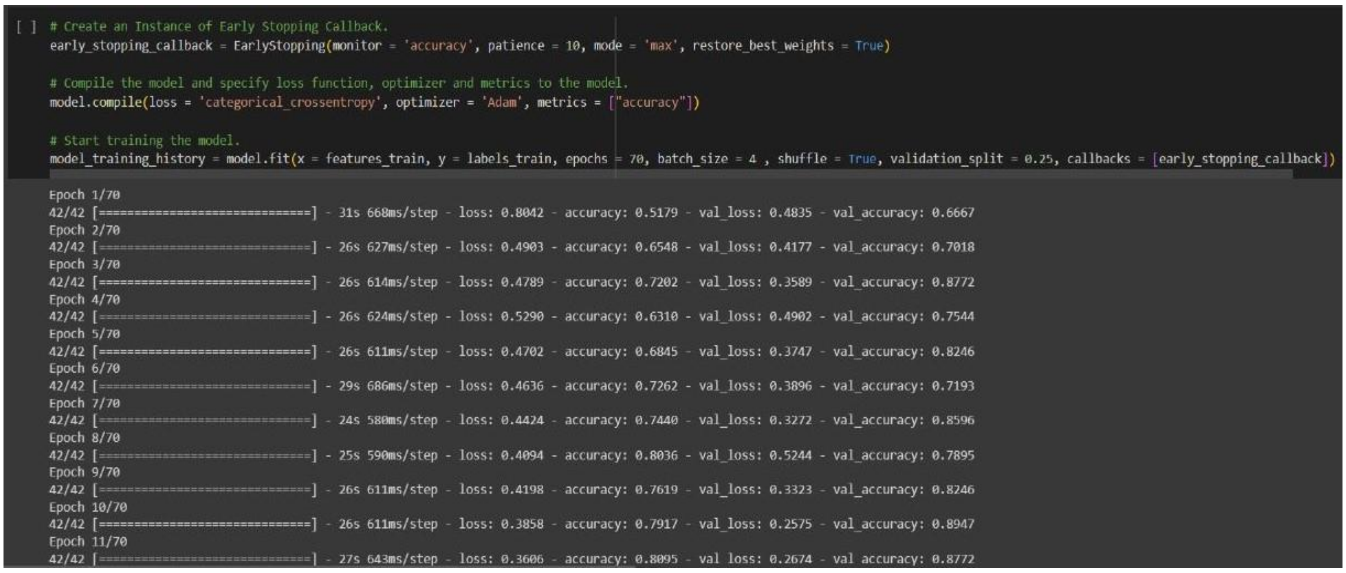
```
[ ] !unzip Dataset.zip

Archive:  Dataset.zip
   creating: Dataset/
   creating: Dataset/walking/
  inflating: Dataset/walking/person01_walking_d1_uncomp.avi
  inflating: Dataset/walking/person01_walking_d2_uncomp.avi
  inflating: Dataset/walking/person01_walking_d3_uncomp.avi
  inflating: Dataset/walking/person01_walking_d4_uncomp.avi
  inflating: Dataset/walking/person02_walking_d1_uncomp.avi
```

Splitting the Datasets



```
[ ] # Create the dataset.
    features, labels, video_files_paths = create_dataset()

    Extracting Data of Class: walking
    Extracting Data of Class: fights
    Extracting Data of Class: running

[ ] # Returns shape of features & labels
    print(features.shape, labels.shape)

    (300, 30, 64, 64, 3) (300,)

    # Using Keras's to_categorical method to convert labels into one-hot-encoded vectors
    one_hot_encoded_labels = to_categorical(labels)

[ ] # Split the Data into Train ( 75% ) and Test Set ( 25% ).
    features_train, features_test, labels_train, labels_test = train_test_split(features, one_hot_encoded_labels, test_size = 0.25, shuffle = True, random_state = seed_constant)
    features = None
    labels = None
```

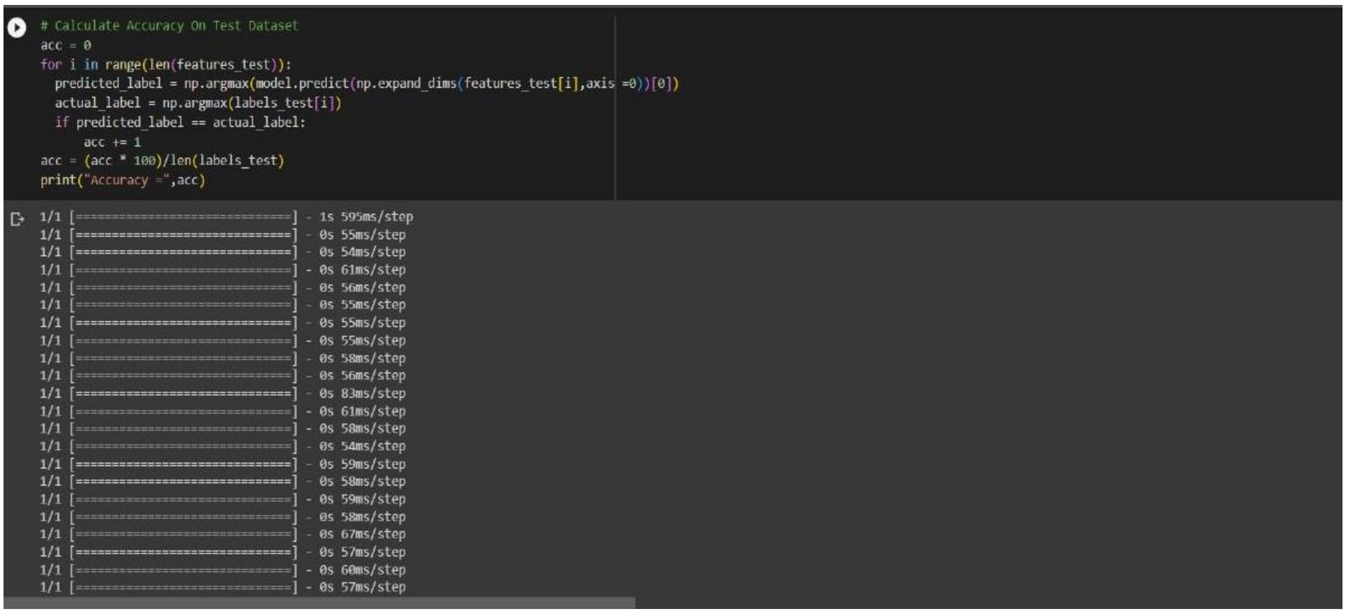
Model Training



```
# Create an Instance of Early Stopping Callback.
early_stopping_callback = EarlyStopping(monitor = 'accuracy', patience = 10, mode = 'max', restore_best_weights = True)

# Compile the model and specify loss function, optimizer and metrics to the model.
model.compile(loss = 'categorical_crossentropy', optimizer = 'Adam', metrics = ["accuracy"])

# Start training the model.
model_training_history = model.fit(x = features_train, y = labels_train, epochs = 70, batch_size = 4 , shuffle = True, validation_split = 0.25, callbacks = [early_stopping_callback])
```

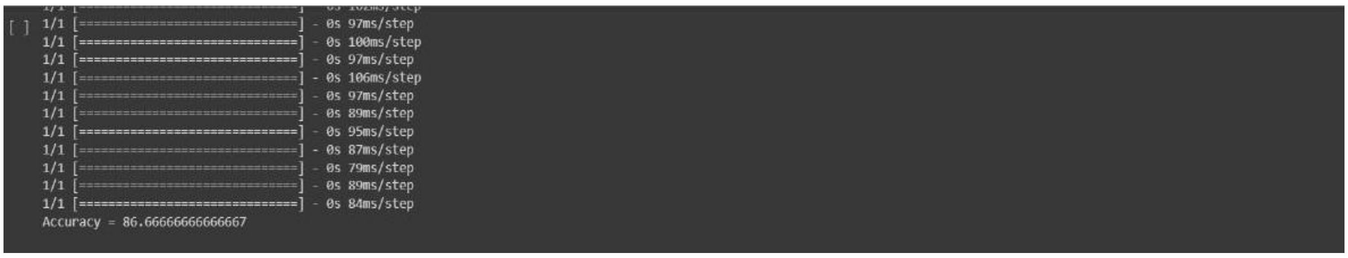
```
Epoch 1/70
42/42 [==============================] - 31s 668ms/step - loss: 0.8042 - accuracy: 0.5179 - val_loss: 0.4835 - val_accuracy: 0.6667
Epoch 2/70
42/42 [==============================] - 26s 627ms/step - loss: 0.4903 - accuracy: 0.6548 - val_loss: 0.4177 - val_accuracy: 0.7018
Epoch 3/70
42/42 [==============================] - 26s 614ms/step - loss: 0.4789 - accuracy: 0.7202 - val_loss: 0.3589 - val_accuracy: 0.8772
Epoch 4/70
42/42 [==============================] - 26s 624ms/step - loss: 0.5290 - accuracy: 0.6310 - val_loss: 0.4902 - val_accuracy: 0.7544
Epoch 5/70
42/42 [==============================] - 26s 611ms/step - loss: 0.4702 - accuracy: 0.6845 - val_loss: 0.3747 - val_accuracy: 0.8246
Epoch 6/70
42/42 [==============================] - 29s 686ms/step - loss: 0.4636 - accuracy: 0.7262 - val_loss: 0.3896 - val_accuracy: 0.7193
Epoch 7/70
42/42 [==============================] - 24s 580ms/step - loss: 0.4424 - accuracy: 0.7440 - val_loss: 0.3272 - val_accuracy: 0.8596
Epoch 8/70
42/42 [==============================] - 25s 590ms/step - loss: 0.4094 - accuracy: 0.8036 - val_loss: 0.5244 - val_accuracy: 0.7895
Epoch 9/70
42/42 [==============================] - 26s 611ms/step - loss: 0.4198 - accuracy: 0.7619 - val_loss: 0.3323 - val_accuracy: 0.8246
Epoch 10/70
42/42 [==============================] - 26s 611ms/step - loss: 0.3858 - accuracy: 0.7917 - val_loss: 0.2575 - val_accuracy: 0.8947
Epoch 11/70
42/42 [==============================] - 27s 643ms/step - loss: 0.3606 - accuracy: 0.8095 - val_loss: 0.2674 - val_accuracy: 0.8772
```

Accuracy on Test Dataset



```
# Calculate Accuracy On Test Dataset
acc = 0
for i in range(len(features_test)):
  predicted_label = np.argmax(model.predict(np.expand_dims(features_test[i],axis =0))[0])
  actual_label = np.argmax(labels_test[i])
  if predicted_label == actual_label:
      acc += 1
acc = (acc * 100)/len(labels_test)
print("Accuracy =",acc)
```

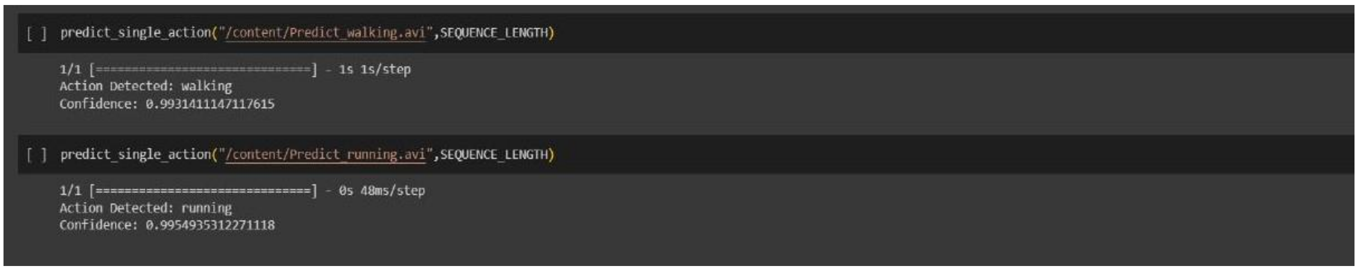
```
1/1 [==============================] - 1s 595ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 83ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 57ms/step
```



```
1/1 [==============================] - 0s 97ms/step
1/1 [==============================] - 0s 100ms/step
1/1 [==============================] - 0s 97ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 97ms/step
1/1 [==============================] - 0s 89ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 87ms/step
1/1 [==============================] - 0s 79ms/step
1/1 [==============================] - 0s 89ms/step
1/1 [==============================] - 0s 84ms/step
Accuracy = 86.66666666666667
```

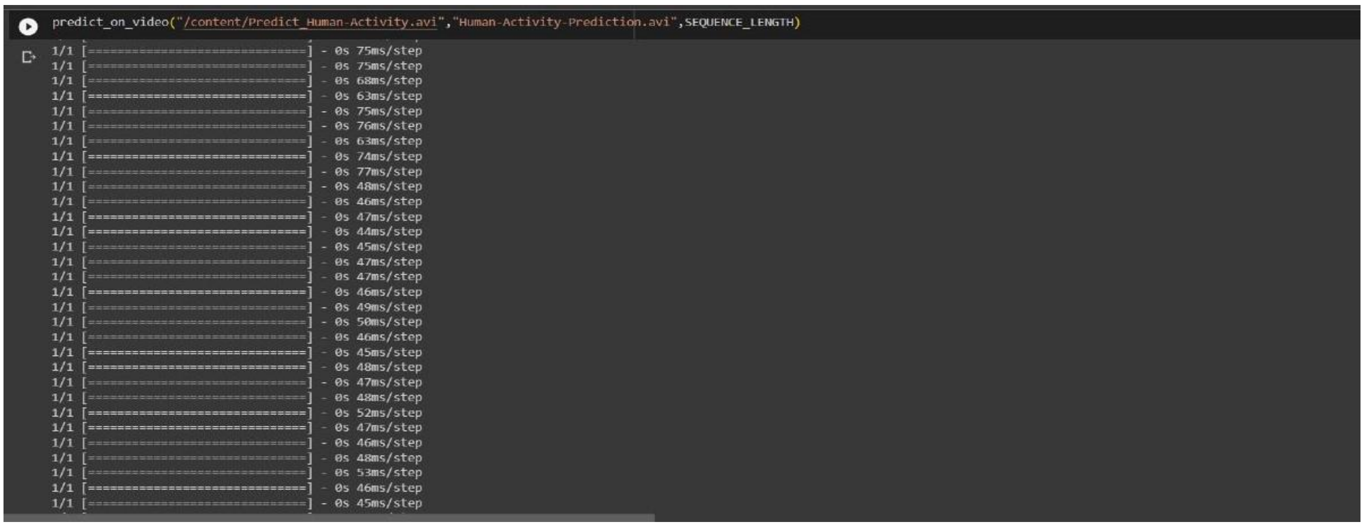
5.2 Results

Detecting Single Action

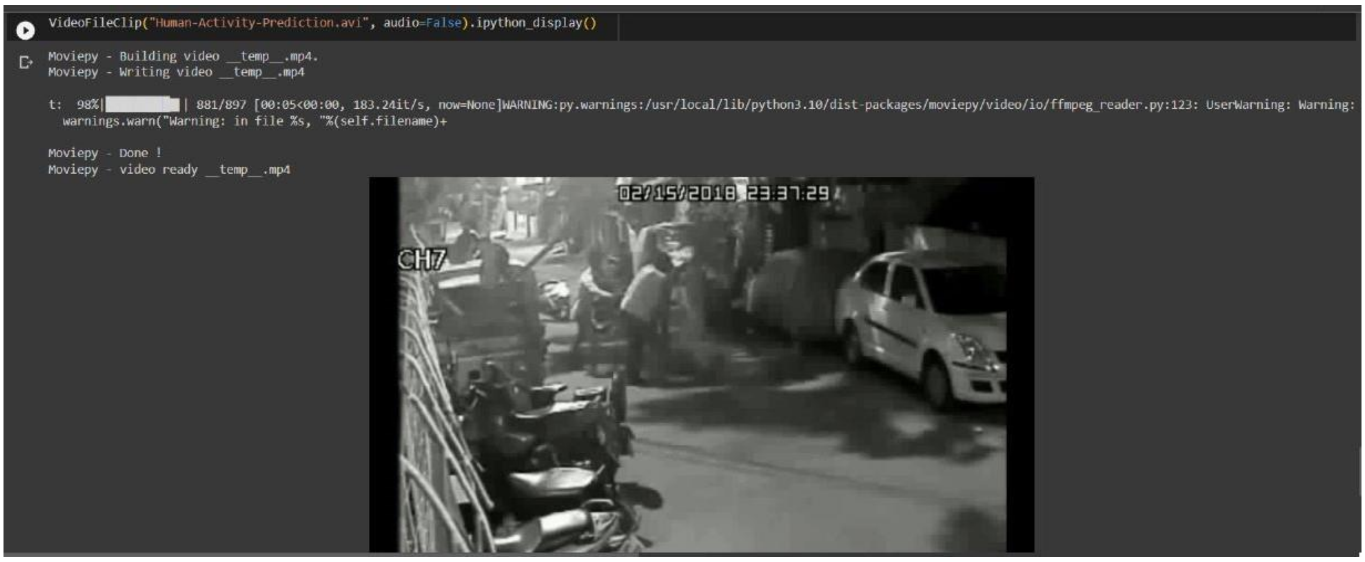


```
[ ] predict_single_action("/content/Predict_walking.avi",SEQUENCE_LENGTH)

    1/1 [==============================] - 1s 1s/step
    Action Detected: walking
    Confidence: 0.9931411147117615

[ ] predict_single_action("/content/Predict_running.avi",SEQUENCE_LENGTH)

    1/1 [==============================] - 0s 48ms/step
    Action Detected: running
    Confidence: 0.9954935312271118
```

Detecting Multiple Action



```
predict_on_video("/content/Predict_Human-Activity.avi","Human-Activity-Prediction.avi",SEQUENCE_LENGTH)
1/1 [==============================] - 0s 75ms/step
1/1 [==============================] - 0s 75ms/step
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 75ms/step
1/1 [==============================] - 0s 76ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 74ms/step
1/1 [==============================] - 0s 77ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 45ms/step
```



```
VideoFileClip("Human-Activity-Prediction.avi", audio=False).ipython_display()

Moviepy - Building video __temp__.mp4.
Moviepy - Writing video __temp__.mp4

t:  98%|█████████ | 881/897 [00:05<00:00, 183.24it/s, now=None]WARNING:py.warnings:/usr/local/lib/python3.10/dist-packages/moviepy/video/io/ffmpeg_reader.py:123: UserWarning: Warning:
  warnings.warn("Warning: in file %s, "%(self.filename)+

Moviepy - Done !
Moviepy - video ready __temp__.mp4
```

Multiple action detected video link: https://drive.google.com/file/d/16unohUf0ARnu4QtU5-lc6he_4NUYtFUm/view?usp=sharing

VI. TESTING

In machine learning, testing refers to the process of evaluating the performance and generalization capabilities of a trained model on unseen or new data. It is an important step to assess how well the model can make predictions or classifications in real-world scenarios beyond the data it was trained on. The testing phase helps measure the model's accuracy, robustness, and suitability for the intended task. The testing set comprises a collection of testing samples that should be separate from the training and validation sets, while sharing the same probability distribution as the training set. The primary objective of machine learning testing is to ensure that the learned logic remains consistent regardless of the number of times the program is invoked.

Model Test Table:

Below shows various tests conducted on the model and whether they have produced the expected result

Test Cases	Test Case Scenario	Expected Result	Actual Result	Status Pass / Fail
1	Test the model’s ability to avoid misclassifying non- suspicious activities as suspicious	The model should correctly classify these non-suspicious activities as non-suspicious	The model correctly classified the non-suspicious activities	Pass
2	Test the model’s performance and real-time capabilities by processing a stream of CCTV surveillance Video footage	The model should detect suspicious activities, ensuring timely intervention and prevention of potential conflicts	The model detected suspicious activities	Pass
3	Test the model’s resilience to environmental changes, such dynamic backgrounds or moving objects	The model should remain effective in detecting and classifying suspicious and non-suspicious activities despite environmental changes and potential distractions	The model detected suspicious and non-suspicious activities despite of environmental changes	Pass
4	Test the model’s performance in low lighting conditions where visibility may be limited	The model should be able to detect and classify suspicious activities accurately, even when in low lighting	The model detected suspicious activities accurately in low lighting	Pass

VII. CONCLUSION & FUTURE SCOPE

Conclusion

The work suggests the implementation of a deep learning neural network model for suspicious activity detection in CCTV surveillance videos has shown promising results. By categorizing fighting's as suspicious activities and considering walking and running as non-suspicious activities, the model can effectively identify potentially dangerous situations and aid in maintaining public safety.

However, it's important that the project's success depends on the quality and variety of the training datasets. To improve accuracy, the neural network model needs to be trained on a diverse range of fighting's and non-suspicious activities, ensuring it can effectively distinguish between them and minimize false alarms. Hence, the overall accuracy of the model is 86.66%.

Future Scope

The result of the detection obtained by the model encompasses several potential advancements and provide the scope for the future work

The functionalities of this project can be scaled up in the future:

- Enhancing the model by refining the training datasets we can improve the accuracy. This could involve increasing the diversity of activities captured in the datasets, considering different environmental conditions.

- Currently, the model has been trained on fighting, running and walking activities. As considering fighting's under suspicious activity, it can be further broadened by taking variety of violence anomalies such as Shooting, Road Accidents, Robbery, Burglar etc.

VIII. REFERENCES

1. Virender Singha, Swati Singha, Dr. Pooja Guptaa. Real-Time Anomaly Recognition Through CCTV Using Neural Networks 1877-0509 © 2020 The Authors. Published by Elsevier B.V

2. Human Behaviour Recognition Model Based Feature and Classifier Selection by GeGeo, Zhixin Li, Zhan Huan, Ying Chen, Jiuzhen Liang, Bangwen Zhou, Chenhul Dong. Published: 23 November 2021*Sensors* 2021, *21*(23), 7791; https://doi.org/10.3390/s21237791

3. DaeHyoen Joe and Min-Suk Kim proposed about the Deep Learning Based Sequence Casual Long-Term Recurrent Convolutional Neural Network for Data Fusion Using Video Data. Published: 24 February 2023
   Electronics 2023, 12*(5)*,1115; https://doi.org/10.3390/electronics12051115

4. Procedia Computer Science 173 (2020) 254–263 www.sciencedirect.com

IX. ACRONYMS

CNN	- Convolutional neural network
LRCN	- Long-Term Recurrent Neural Network
LSTM	- Long Short-Term Memory