UNMASKING THE ABNORMAL EVENTS IN VIDEOS USING DEEP LEARNING

A PROJECT REPORT

SUBMITTED BY

D. GUHAPRIYA (913120104029)

K.KIRUTHIKA (913120104043)

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

VELAMMAL COLLEGE OF ENGINEERING

AND TECHNOLOGY

MADURAI-625 009

ANNA UNIVERSITY: CHENNAI 600 025

APRIL-2024

BONAFIDE CERTIFICATE

Certified that this project report "**UNMASKING THE ABNORMAL EVENTS IN VIDEOS USING DEEP LEARNING**" is the bonafide work of "**GUHAPRIYA D (913120104029), KIRUTHIKA K (913120104043)**" of **VIII Semester B.E Computer Science and Engineering** who carried out the project work under my supervision.

SIGNATURE

Dr.P.ALLI
B.E., Ms., Ph.D
HEAD OF THE DEPARTMENT,
Computer Science and Engineering,
Velammal College of Engineering
and Technology,
Madurai-625009.

SIGNATURE

Mrs. S.PADMADEVI
B.E,M.E (Ph.D)
Assistant Professor,
Computer Science and Engineering,
Velammal College of Engineering
and Technology,
Madurai-625009.

Submitted for the viva voce held on _____ at Velammal College of Engineering and Technology, Madurai.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this project and, in my opinion, this project is adequate in terms of scope and quality for the partial fulfillment for the award of the degree of **BACHELOR of ENGINEERING in COMPUTER SCIENCE AND ENGINEERING.**

Signature :

Name of Supervisor : Mrs. S.Padmadevi

Position: Assistant Professor

Date:

STUDENT DECLARATION

We hereby declare that the project report "**UNMASKING THE ABNORMAL EVENTS IN VIDEOS USING DEEP LEARNING**" is based on our own work carried out during the course of our study under the supervision of **Mrs.S. Padmadevi**, Assistant Professor, Professor.

I assert the statements made and conclusions are drawn are an outcome of my research work. I further certify that

i. The work contained in the report is original and has been done by me under the general supervision of my supervisor.

ii. We have followed the guidelines provided by the university in writing the report.

iii. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and given their details in the references.

Signature: 1) 2)

Student name: 1) Guhapriya D (913120104029)

2) Kiruthika K (913120104043)

PREFACE

This project has been composed with the aim of designing a software on an adaptive traffic signal timer system. A lot of effort has been made to make this project report interesting and a learning experience for us. This report has been explained with the help of diagrams and figures. The running project has been presented through a PowerPoint representation. The subject matter has been compiled in a simple, illustrative and lucid manner.

ACKNOWLEDGEMENT

I wish to express my gratitude to the following person with those help and management that led to successful completion.

First of all, I would like to express my deep gratitude to our beloved and respectable chairman Thiru M.V. Muthuramalingam of Velammal College of Engineering and Technology, Madurai. To provide prospectus infrastructure and environment to execute our tasks with a Smart attitude and conceive vision and mission of technological education.

I express my deep gratitude to Dr. N. Suresh Kumar, Principal of Velammal College of Engineering and Technology for his helpful attitude towards this project. I wish to express my sincere thanks and gratitude to Dr P.Alli, Professor and Head of the Department of Computer Science and Engineering for motivating and encouraging us for every part of my project.

I am proudly grateful to Mrs.S.Padmadevi, Assistant professor as Supervisor for their expert guidance and continuous encouragement throughout to see that this project reaches its target since its commencement to its completion.

Finally, I must express my sincere heartfelt gratitude to all the staff members of the Computer Science and Engineering Department who helped me directly and indirectly during this course of work.

(Guhapriya D) (Kiruthika K)

[Signature of candid]

ABSTRACT

The project is a comprehensive endeavor aimed at developing a robust system for detecting suspicious human activity within video surveillance footage. It leverages advanced deep learning techniques, particularly the Long-term Recurrent Convolutional Network (LRCN) architecture, renowned for its ability to capture both spatial and temporal features from video sequences effectively. The system's efficacy is demonstrated through meticulous preprocessing steps, including frame resizing and pixel normalization, ensuring optimal input data for the model. By training on a diverse dataset comprising videos sourced from multiple databases, encompassing a wide range of human activities, the model can effectively differentiate between normal and anomalous behavior. A significant aspect of the project lies in its implementation of various libraries and modules, including OpenCV for video processing, NumPy for numerical computations, and TensorFlow and Keras for constructing and training deep learning models. These tools streamline the development pipeline, enabling efficient data handling, model construction, and training. Moreover, the project incorporates visualization techniques to provide insights into the dataset's characteristics, facilitating a deeper understanding of the video data and model performance. Ultimately, the project's overarching goal is to contribute to the advancement of surveillance systems by providing a reliable and efficient method for detecting suspicious human activity in video footage. Through the integration of cutting-edge deep learning techniques and meticulous data preprocessing, the system aims to enhance security measures and situational awareness in various surveillance applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

1	CNN	Convolutional Neural Network
2	LSTM	Long Short Term Memory
3	LRCN	Long-term Recurrent Convolutional Network
4	OpenCV	Open-source Computer Vision library

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

In the realm of surveillance and security, advancements in **Artificial Intelligence** have catalyzed a paradigm shift, with automated systems increasingly replacing human monitoring. The impetus for this transition stems from various factors including the repetitive nature of surveillance tasks, human limitations in monitoring large volumes of video data, and the need for rapid response times to anomalous events. In response to these challenges, the proposed project introduces a novel methodology leveraging deep learning techniques for video-based anomaly detection. Our project addresses the pressing need for enhanced security measures and situational awareness by developing a sophisticated anomaly detection system. Central to this system is the **Long-term Recurrent Convolutional Network (LRCN)** architecture, which integrates both temporal and spatial features extracted from video sequences. This integration is achieved through the fusion of **long short-term memory (LSTM)** layers for capturing temporal relationships and **convolutional neural networks (CNN)** for extracting spatial features. By amalgamating these techniques, the system demonstrates superior performance in detecting and categorizing both normal and anomalous human activities within surveillance footage.

The key components of the project include comprehensive data preprocessing techniques to ensure optimal learning from video frames. These techniques encompass resizing frames, normalization of pixel values, and extraction of relevant features. The dataset utilized for training and evaluation comprises diverse video sequences sourced from multiple databases, encompassing a wide spectrum of human activities and anomalous events. The implementation of the proposed methodology is facilitated by the Python programming language, leveraging

powerful libraries such as,

1. TensorFlow
2. Keras
3. OpenCV
4. NumPy

These libraries provide essential tools for data manipulation, model construction, and training, enabling seamless development of the anomaly detection system. In summary, the project represents a significant advancement in the domain of surveillance applications, offering a robust methodology for video-based anomaly detection. By harnessing the capabilities of deep learning and leveraging state-of-the-art techniques, the system demonstrates remarkable proficiency in enhancing security awareness and detecting suspicious activities within surveillance footage.

1.2 PYTHON LANGUAGE

Python serves as the foundation for your deep learning project, providing a versatile and powerful programming language that is well-suited for tasks such as data preprocessing, model development, and evaluation. Python's simplicity and readability make it an ideal choice for implementing complex deep learning algorithms and workflows. Its straightforward syntax facilitates efficient coding and reduces development time, allowing you to focus on implementing and experimenting with machine learning models.

- **Simplicity and Readability:**
  Python's clear and concise syntax makes it easy to write and understand code, enhancing collaboration and maintainability.
- **Reduced Development Time:**
  Python's high-level abstractions and extensive libraries streamline development, enabling rapid prototyping and experimentation.

Python boasts a rich ecosystem of libraries and frameworks specifically designed for machine learning and deep learning tasks. Libraries such as TensorFlow, Keras, and PyTorch provide powerful tools and APIs for building and training neural networks, simplifying the implementation of complex algorithms.

- **Machine Learning Libraries:**
  Python's ecosystem includes popular machine learning libraries such as TensorFlow, Keras, and PyTorch, which offer robust support for deep learning tasks.
- **Extensive Libraries:** Access a wide range of libraries for data manipulation, visualization, and statistical analysis, further enhancing the capabilities of your project.

Python's versatility extends to its compatibility with various platforms and operating systems, ensuring seamless deployment and execution of your deep learning models across different environments.

- **Cross-Platform Compatibility:**
  Python runs on multiple platforms, including Windows, macOS, and Linux, ensuring consistency and compatibility across different systems.
- **Deployment Flexibility:**
  Deploy your trained models in production environments with ease, thanks to Python's versatility and widespread adoption in the industry.

Finally, Python serves as the backbone of your deep learning project, offering a powerful and flexible programming language that enables efficient development, implementation, and deployment of machine learning models. Its simplicity, extensive libraries, cross-platform compatibility, and supportive community make Python an indispensable tool for advancing our project's goals.

1.3 VISUAL STUDIO CODE

Visual Studio Code (VS Code) serves as a versatile and efficient integrated development environment (IDE) for your deep learning project, offering a plethora of features and functionalities tailored to enhance productivity and streamline development workflows. Visual Studio Code offers seamless integration with Git, allowing for efficient version control management directly within the IDE. This enables you to track changes, collaborate with team members, and maintain code consistency effortlessly.

- **Git Integration:**
  VS Code seamlessly integrates with Git, simplifying collaboration and version control management.
- **Efficient Collaboration:**
  Built-in Git commands and source control views facilitate smooth collaboration among team members.
- **Version Tracking:**
  Easily track project milestones and changes, ensuring code integrity and transparency throughout the development process.

With robust support for Python development, including features such as IntelliSense, syntax highlighting, and built-in debugging tools, VS Code provides an optimal environment for coding deep learning algorithms.

- **Python Development:**
  Extensive support for Python development enhances coding productivity and efficiency.
- **Debugging Capabilities:**
  Built-in debugging tools simplify the process of identifying and resolving coding issues, ensuring smoother development cycles.

CHAPTER 2
REVIEW OF LITERATURE

2.1 EXISTING SYSTEM

- The existing systems for video anomaly detection predominantly rely on conventional methods, such as handcrafted features and learning algorithms.
- These approaches often exhibit limitations in accurately capturing intricate temporal and spatial patterns inherent in surveillance videos.
- Handcrafted features may not sufficiently represent the diverse array of anomalies, resulting in diminished detection accuracy and reliability.
- Additionally, these systems typically necessitate extensive manual tuning and domain-specific knowledge for optimal performance, rendering them less scalable and adaptable to various surveillance scenarios.

2.2 PROPOSED SYSTEM

- The proposed system for video anomaly detection integrates deep learning methodologies, specifically the Long-term Recurrent Convolutional Network model, to enhance surveillance capabilities.
- By combining CNN for spatial feature extraction and LSTM for temporal context modeling, the system can effectively analyze complex spatiotemporal patterns inherent in surveillance videos.
- Through meticulous preprocessing and data augmentation techniques, the system ensures the robustness and diversity of the training dataset, enabling the LRCN model to accurately differentiate between normal activities and anomalies.
- During training, optimization techniques such as categorical cross-entropy loss and early stopping mechanisms are employed to enhance model performance and prevent overfitting.

CHAPTER 3
SCOPE AND OBJECTIVES

3.1 SCOPE

- To Enhance Security Measures by identifying and alerting on unusual activities, providing an additional layer of protection against potential threats in surveillance environments.
- To Improve Situational Awareness by quickly identifying and categorizing abnormal events, allowing for rapid responses to potential security breaches.
- To Identify Varied Human Activities that may indicate emergencies, unsafe crossing behaviors, or other critical situations, enabling swift response from emergency services.

3.2 OBJECTIVES

- Identify deviations or irregularities within datasets or systems that may indicate anomalies or unusual behavior.
- Provide early detection and warning of potential issues, allowing for timely intervention or mitigation.
- Enhance security measures by detecting and alerting to potential threats, intrusions, or malicious activities.
- Improve operational efficiency by identifying inefficiencies, errors, or irregularities that may disrupt processes or indicate underlying problems.

CHAPTER 4
SYSTEM REQUIREMENTS AND ANALYSIS

4.1 REQUIREMENT ANALYSIS

The requirement analysis for the video anomaly detection system underscores the necessity for a robust and efficient solution in enhancing surveillance and security measures. By implementing deep learning techniques, particularly the Long-term Recurrent Convolutional Network model, the system aims to automate the detection of anomalies in surveillance footage. This automation not only reduces the reliance on manual intervention but also improves accuracy by effectively capturing complex spatial and temporal patterns within the videos. The primary goal is to minimize human errors and enhance situational awareness by providing a reliable and scalable solution for identifying abnormal behaviors and events in real-time.

4.2 FEASIBILITY STUDY

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of the existing system or proposed venture. In its simplest term, the two criteria to judge feasibility are cost required and value to be attained. As such, a well-designed feasibility study should provide historical background of the project. Generally, feasibility studies precede technical development and project implementation. The assessment of feasibility study is based on the following factors:

a) Technical Feasibility

b) Economic Feasibility

c) Operational Feasibility

4.2.1 TECHNICAL FEASIBILITY

Generally, feasibility studies precede technical development and project implementation. The assessment is based on a system requirement in terms of Input,

Processes, Output, Fields, Programs, and Procedure. This can be quantified in terms of volumes of data, trends, frequency of updating, etc., in order to estimate whether the new system will perform adequately or not. When writing a feasibility report the following should be taken to consideration:

- A brief description of the business
- The part of the business being examined
- The human and economic factors
- The possible solutions to the problems

4.2.2 ECONOMIC FEASIBILITY

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. An entrepreneur must accurately weigh the cost versus benefits before taking an action.

4.2.3 OPERATIONAL FEASIBILITY

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development The operational feasibility of the system can be checked as it solves the problems and reduces the complications occurring in the paper-pencil test.

4.3 SOFTWARE REQUIREMENT SPECIFICATION

A software requirements specification (SRS) is a complete description of the behavior of the software to be developed. It includes a set of use cases that describe all of the interactions that the users will have with the software. In addition to use cases, the SRS contains functional requirements, which define the internal workings of the software: that is, the calculations, technical details, data manipulation and processing, and other specific functionality that shows how the use cases are to be

satisfied. It also contains nonfunctional requirements, which impose constraints on the design or implementation.

The SRS phase consists of two basic activities

1) Problem/Requirement Analysis:
The process is order and more nebulous of the two, deals with understanding the problem, the goal and constraints.

2) Requirement Specification:
Here, the focus is on specifying what has been found, giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity. The Requirement phase terminates with the production of the validated SRS document. Producing the SRS document is the basic goal of this phase.

Role of SRS:
The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium through which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

CHAPTER 5
SYSTEM DESIGN

5.1 PROJECT MODULES

5.1.1 RANDOM

The random module is used to introduce randomness in certain operations, particularly for selecting random elements from lists. By utilizing functions from the random module, such as random.choice(), we randomly select elements from a list of video files within each class. This random selection ensures diversity in the chosen video samples during data visualization, allowing us to visualize different frames from various videos within the dataset.

5.1.2 MATH

The math module is utilized for basic mathematical operations and functions.. In our code, we use functions from the math module for tasks such as calculating skip intervals for frame extraction or performing mathematical calculations within loops or conditional statements.

5.1.3 DATETIME

The datetime module is employed for handling dates and times, primarily for generating timestamps or logging time-related information. While its usage may be limited compared to other libraries, it provides essential functionalities for managing time-related data. For instance, this module generates timestamps indicating when certain operations or events occur, such as when videos are processed or when model training begins and ends.

5.1.4 OS

The OS module in Python provides a way of interacting with the operating system, allowing you to perform operations such as navigating the file system, manipulating

file paths, and executing system commands. In our code, the os module is utilized primarily for file and directory operations. For example, **os.listdir()** to list the contents of directories, os.path.join() to construct file paths, and **os.path.exists()** to check if a file or directory exists. The os module facilitates tasks such as listing video files in directories, constructing file paths dynamically based on database names and class names, and checking for the existence of files or directories. Overall, the os module streamlines file and directory handling in our code, making it more robust and adaptable to different operating systems.

5.1.5 CV2

OpenCV Python is a library of Python bindings designed to solve computer vision problems. cv2.imread() method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

Syntax: cv2.imread(path, flag)

Parameters:

path: A string representing the path of the image to be read.

flag: It specifies the way in which the image should be read. Its default value is cv2.IMREAD_COLOR.

Return Value: This method returns an image that is loaded from the specified file. The image should be in the working directory or a full path of image should be given.

5.1.6 NUMPY

The numpy library plays a crucial role in handling numerical data, particularly in the context of image and video processing. NumPy provides support for multidimensional arrays, which are fundamental data structures for representing images and videos in numerical form. Throughout our code, NumPy is used for various operations, such as resizing frames, normalizing pixel values, and organizing data into arrays. Additionally, NumPy's mathematical functions facilitate

normalization of pixel values by dividing each pixel intensity by 255, thereby scaling them to the range between 0 and 1.

5.2 PROJECT LIBRARIES

5.2.1 OPENCV

- OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting in 2011, OpenCV features GPU acceleration for real-time operations.

- There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in several programming languages have been developed to encourage adoption by a wider audience.

- OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation.

- Wrappers in several programming languages have been developed to encourage adoption by a wider audience. In version 3.4, JavaScript bindings for a selected subset of OpenCV functions was released as OpenCV.js, to be used for web platforms.

- OpenCV's application areas include:
    1. Facial recognition system

2. Gesture recognition
3. Human–computer interaction
4. Object detection
5. Segmentation and recognition
6. Motion tracking
7. Augmented reality

5.3 DATA FLOW DIAGRAMS

- A data flow diagram is a graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed.

- The transformation of data from input to output through processing, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams.

- The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams.. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purposes.

- The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The top-level diagram is often called a "context diagram".

- The DFD provides a clear visualization of how video data moves through different stages of processing, from input to output. It illustrates the flow of data between modules such as preprocessing, model training, and testing.

DFD LEVEL 0

Fig. 5.3.1. DFD LEVEL 0

DFD LEVEL 1

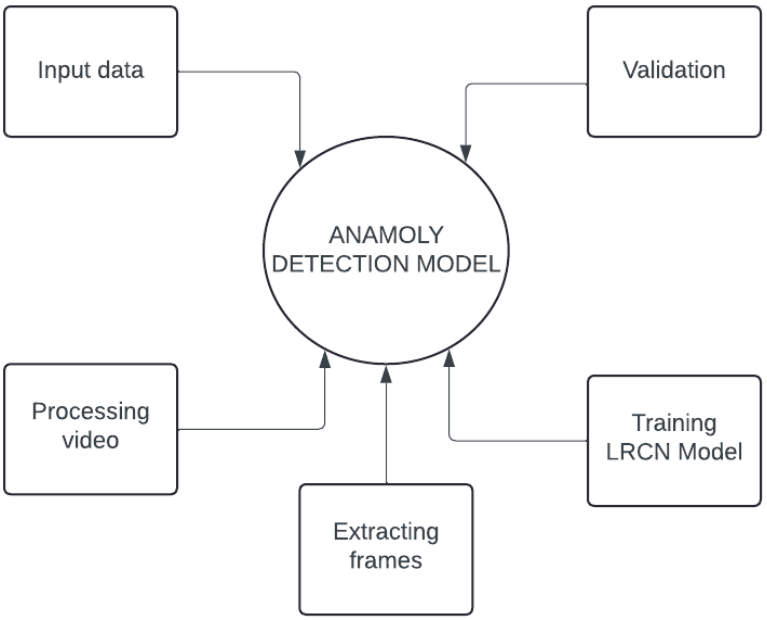

Fig. 5.3.2. DFD LEVEL 1

5.4 UML DIAGRAMS

- UML stands for Unified Modeling Language. It is a third generation method for specifying, visualizing and documenting the artifacts of an object oriented

system under development.

- Object modeling is the process by which the logical objects in the real world (problem space) are represented (mapped) by the actual objects in the program (logical or a mini world).
- This visual representation of the objects, their relationships and their structures is for the ease of understanding. This is a step while developing any product after analysis.

USE CASE DIAGRAM

A use case diagram is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will be accompanied by other types of diagrams.

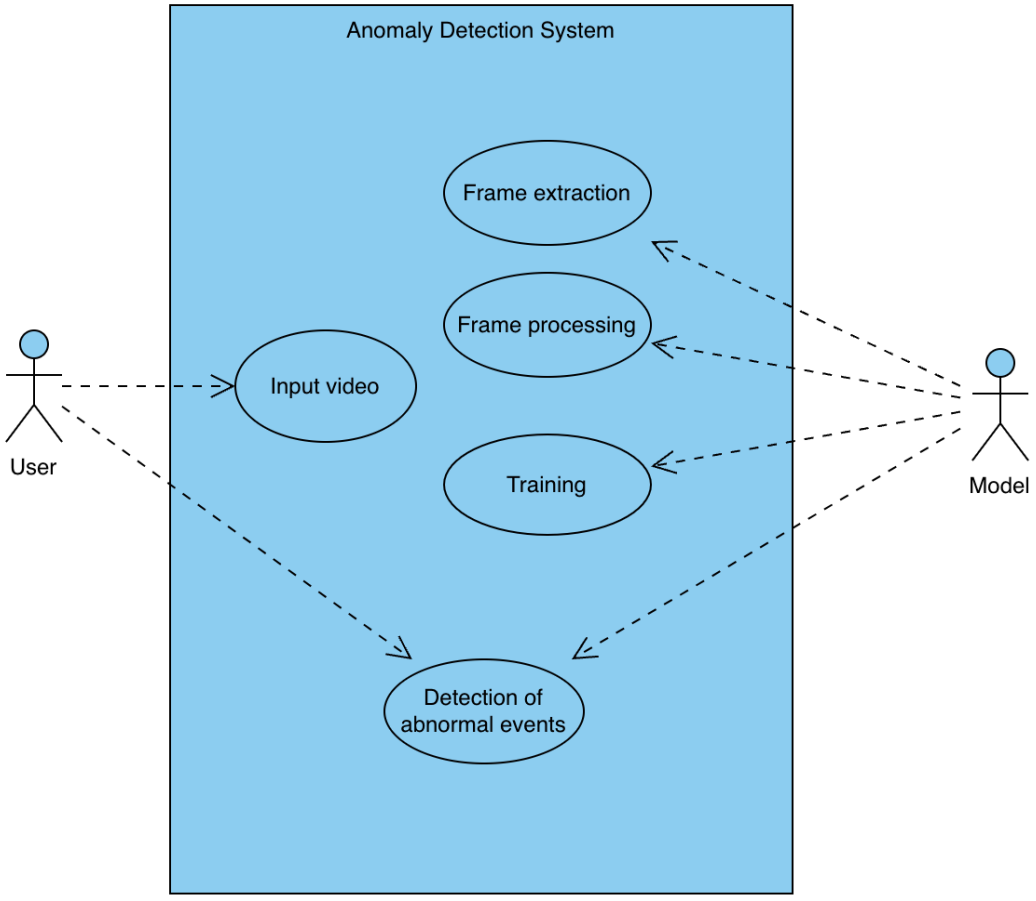

Fig. 5.4.1. Use Case diagram

ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

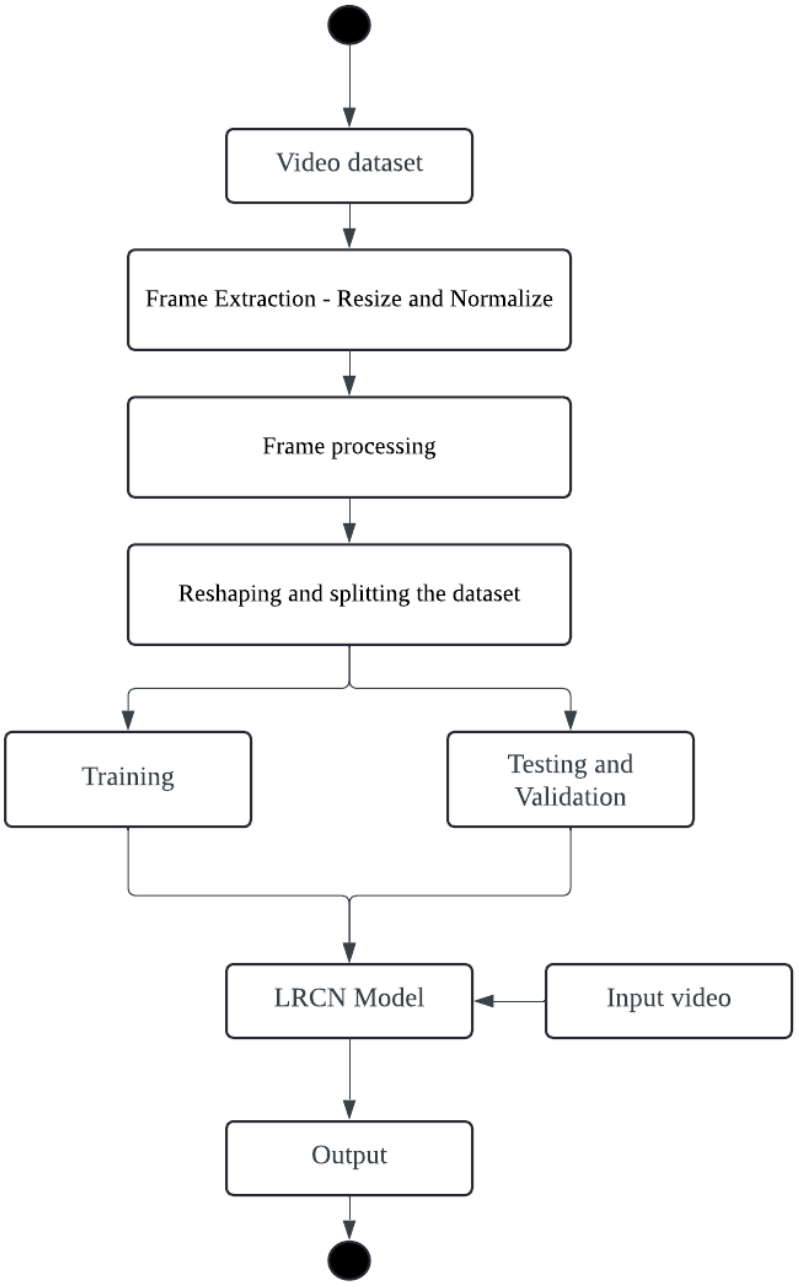

Fig. 5.4.2. Activity diagram

SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

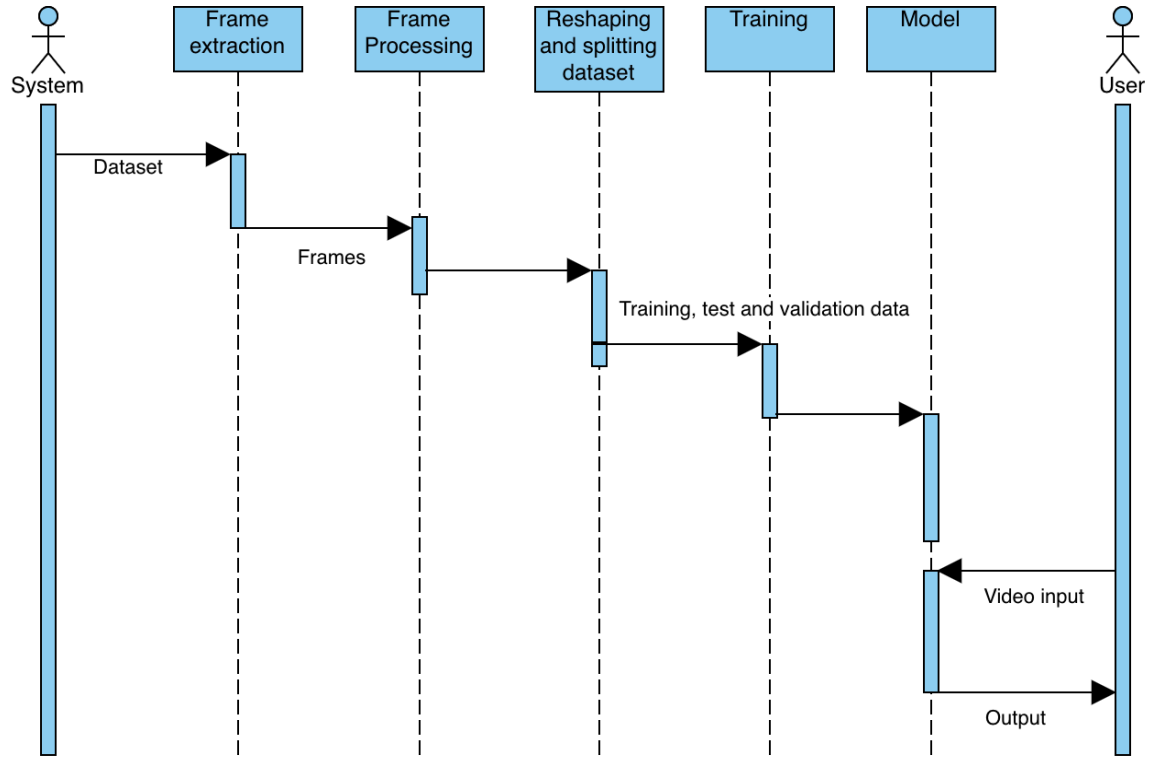

Fig. 5.4.3. Sequence diagram

CHAPTER 6
HARDWARE AND SOFTWARE REQUIREMENTS

The introduction to software requirements specification states the goals and objectives of the software, describing it in the context of the computer-based system, software engineering is refined by establishing a complete information description, a detailed functional description, a representation of system behavior, an indication of performance requirement and design.

6.1 HARDWARE REQUIREMENTS

Desktop computer or laptop with:

- Processor: Intel Pentium 4 / AMD Processor
- Minimum RAM: 2GB RAM
- Hard disk: 25 GB
- Operating System: Windows 7 or higher version of OS (either X86 or X64)

6.2 SOFTWARE REQUIREMENTS

- Visual studio code
- OpenCV
- Anaconda Navigator
- NumPy
- Python

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 CODE

main.py

```
import os
import cv2
import math
import random
import numpy as np
import datetime as dt
import tensorflow as tf
from collections import deque
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow import keras;
from keras.layers import *
from keras.models import Sequential
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
from keras.utils import plot_model
seed_constant = 5
np.random.seed(seed_constant)
random.seed(seed_constant)
tf.random.set_seed(seed_constant)
DB_NAMES = ['Human Activity Recognition - Video Dataset', 'HMDB_dataset',
'Peliculas']
VD = [file for file in os.listdir('Dataset/Human Activity Recognition - Video Dataset') if
not file.startswith('.')]
HMDB = [file for file in os.listdir('Dataset/HMDB_dataset') if not file.startswith('.')]
NF = [file for file in os.listdir('Dataset/Peliculas') if not file.startswith('.')]
allDB = VD+NF+HMDB
print(allDB)
plt.figure(figsize = (20, 20))
all_classes_names = allDB
print(all_classes_names)
for counter, random_index in enumerate(range(len(all_classes_names)), 1):
    selected_class_Name = all_classes_names[random_index]
```

```
    # DB Name get
    for item in VD:
        if selected_class_Name == item:
            db_Name = 'Human Activity Recognition - Video Dataset'
    for item in HMDB:
        if selected_class_Name == item:
            db_Name = 'HMDB_dataset'
    for item in NF:
        if selected_class_Name == item:
            db_Name = 'Peliculas'
    # print(selected_class_Name +" "+db_Name)

    video_files_names_list = [file for file in
os.listdir(f'Dataset/{db_Name}/{selected_class_Name}') if not file.startswith('.')]
    selected_video_file_name = random.choice(video_files_names_list)

    video_reader =
cv2.VideoCapture(f'Dataset/{db_Name}/{selected_class_Name}/{selected_video_file_
name}')
    video_reader.set(1, 25)
    _, bgr_frame = video_reader.read()
    bgr_frame = cv2.resize(bgr_frame ,(224,224))
    video_reader.release()

    rgb_frame = cv2.cvtColor(bgr_frame, cv2.COLOR_BGR2RGB)
    cv2.putText(rgb_frame, selected_class_Name, (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 200, 255), 2)

    plt.subplot(5, 4, counter);plt.imshow(rgb_frame);plt.axis('off')
# plt.show()
# Specify the height and width to which each video frame will be resized in our dataset.
IMAGE_HEIGHT , IMAGE_WIDTH = 64, 64

# Specify the number of frames of a video that will be fed to the model as one
sequence.
SEQUENCE_LENGTH = 30

# Specify the directory containing the UCF50 dataset. DATASET_DIR =
"Dataset/Peliculas"
CLASSES_LIST = all_classes_names
def frames_extraction(video_path):
```

```
    '''
    This function will extract the required frames from a video after resizing and
normalizing them.
    Args:
        video_path: The path of the video in the disk, whose frames are to be extracted.
    Returns:
        frames_list: A list containing the resized and normalized frames of the video.
    '''
    # Declare a list to store video frames.
    frames_list = []

    # Read the Video File using the VideoCapture object.
    video_reader = cv2.VideoCapture(video_path)

    # Get the total number of frames in the video.
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))
    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)
    # Iterate through the Video Frames.
    for frame_counter in range(SEQUENCE_LENGTH):
        # Set the current frame position of the video.
        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter *
skip_frames_window)
        # Reading the frame from the video.
        success, frame = video_reader.read()
        # Check if Video frame is not successfully read then break the loop
        if not success:
            break
        # Resize the Frame to fixed height and width.
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))

        # Normalize the resized frame by dividing it with 255 so that each pixel value then
lies between 0 and 1
        normalized_frame = resized_frame / 255

        # Append the normalized frame into the frames list
        frames_list.append(normalized_frame)

    # Release the VideoCapture object.
    video_reader.release()
    # Return the frames list.
    return frames_list
```

```
def create_dataset():
    '''
    This function will extract the data of the selected classes and create the required dataset.
    Returns:
        features:          A list containing the extracted frames of the videos.
        labels:            A list containing the indexes of the classes associated with the videos.
        video_files_paths: A list containing the paths of the videos in the disk.
    '''
    # Declared Empty Lists to store the features, labels and video file path values.
    features = []
    labels = []
    video_files_paths = []

    # Iterating through all the classes mentioned in the classes list
    for class_index, class_name in enumerate(CLASSES_LIST):

        # Display the name of the class whose data is being extracted.
        print(f'Extracting Data of Class: {class_name}')
        # ------------------------
        # DB Name get
        for item in VD:
            if class_name == item:
                db_Name = 'Human Activity Recognition - Video Dataset'
        for item in HMDB:
            if class_name == item:
                db_Name = 'HMDB_dataset'
        for item in NF:
            if class_name == item:
                db_Name = 'Peliculas'
        # ------------------------

        DATASET_DIR = f'Dataset/{db_Name}'

        # Get the list of video files present in the specific class name directory.
        files_list = [file for file in os.listdir(os.path.join(DATASET_DIR, class_name)) if not file.startswith('.')]
        # os.listdir(os.path.join(DATASET_DIR, class_name))
```

```
        # Iterate through all the files present in the files list.
        for file_name in files_list:

            # Get the complete video path.
            video_file_path = os.path.join(DATASET_DIR, class_name, file_name)
            # Extract the frames of the video file.
            frames = frames_extraction(video_file_path)
            if len(frames) == SEQUENCE_LENGTH:
                # Append the data to their repective lists.
                features.append(frames)
                labels.append(class_index)
                video_files_paths.append(video_file_path)
    # Converting the list to numpy arrays
    features = np.asarray(features)
    labels = np.array(labels)

    # Return the frames, class index, and video file path.
    return features, labels, video_files_paths
# Create the dataset.
features, labels, video_files_paths = create_dataset()
# print(features)
# 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
def create_LRCN_model():
    '''
    This function will construct the required LRCN model.
    Returns:
        model: It is the required constructed LRCN model.
    '''
    # We will use a Sequential model for model construction.
    model = Sequential()

    # Define the Model Architecture.

    model.add(TimeDistributed(Conv2D(32, (3, 3), padding='same',activation = 'relu'),
```

```
input_shape = (SEQUENCE_LENGTH, IMAGE_HEIGHT, IMAGE_WIDTH, 3)))
    model.add(TimeDistributed(MaxPooling2D((4, 4))))

    model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same',activation = 'relu')))
    model.add(TimeDistributed(MaxPooling2D((4, 4))))

    model.add(TimeDistributed(Conv2D(128, (3, 3), padding='same',activation = 'relu')))
    model.add(TimeDistributed(MaxPooling2D((2, 2))))

    model.add(TimeDistributed(Conv2D(256, (2, 2), padding='same',activation = 'relu')))
    model.add(TimeDistributed(MaxPooling2D((2, 2))))
    model.add(TimeDistributed(Flatten()))
    model.add(LSTM(32))
    model.add(Dense(len(CLASSES_LIST), activation = 'softmax'))

    # Display the models summary.
    model.summary()

    # Return the constructed LRCN model.
    return model
model = create_LRCN_model()
# plot_model(model, to_file = 'Suspicious_Human_Activity_LRCN_Model.png',
show_shapes = True, show_layer_names = True)
# 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 = 30,
batch_size = 4 , shuffle = True, validation_split = 0.25, callbacks =
[early_stopping_callback])
model.save("Suspicious_Human_Activity_Detection_LRCN_Model.h5")
def plot_metric(model_training_history, metric_name_1, metric_name_2, plot_name):
    # Get metric values using metric names as identifiers.
    metric_value_1 = model_training_history.history[metric_name_1]
    metric_value_2 = model_training_history.history[metric_name_2]

    # Construct a range object which will be used as x-axis (horizontal plane) of the
```

```
graph.
    epochs = range(len(metric_value_1))

    # Plot the Graph.
    plt.plot(epochs, metric_value_1, 'blue', label = metric_name_1)
    plt.plot(epochs, metric_value_2, 'red', label = metric_name_2)

    # Add title to the plot.
    plt.title(str(plot_name))
    plt.legend()
plot_metric(model_training_history, 'loss', 'val_loss', 'Total Loss vs Total Validation
Loss')
plot_metric(model_training_history, 'accuracy', 'val_accuracy', 'Total Accuracy vs Total
Validation Accuracy')
# 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)
```

7.2 IMPLEMENTATION

7.2.1 PROJECT ARCHITECTURE

The project architecture follows a modular and hierarchical design, encompassing data acquisition and preprocessing, model construction, and anomaly detection stages. At the forefront of the architecture is the **Data Acquisition and Preprocessing module**. Here, diverse video datasets containing both normal and anomalous human activities are collected. These videos undergo preprocessing steps, including resizing frames to a uniform size and normalizing pixel values. This ensures uniformity in the data and prepares it for input into the subsequent model construction phase.

Moving to the **Model Construction phase**, the architecture employs a fusion of convolutional neural networks (CNN) and long short-term memory (LSTM) layers. The CNN layers extract spatial features from individual video frames, capturing static information within each frame. Meanwhile, LSTM layers capture temporal dependencies and sequential patterns across multiple frames, enabling the model to learn temporal dynamics and detect anomalies over time. The fusion of CNN and LSTM layers results in a unified representation of spatial and temporal features, which serves as the foundation for anomaly detection.

Finally, the architecture culminates in the **Anomaly Detection stage**, where the unified feature representation undergoes anomaly detection algorithms to identify abnormal human activities within the surveillance footage. This stage leverages the learned spatial and temporal features to distinguish between normal and anomalous behavior, thereby enhancing security measures and situational awareness in surveillance applications. Overall, this modular and hierarchical architecture facilitates the seamless integration of data preprocessing, model construction, and anomaly detection, ultimately leading to the development of an effective system.

7.2.2 METHODOLOGY

In the video-based anomaly detection methodology, the algorithm is a critical component responsible for accurately identifying abnormal activities within surveillance footage. This algorithm combines the power of convolutional neural networks (CNNs) and long short-term memory (LSTM) networks to effectively capture both spatial and temporal features from the video data. The CNN component of the algorithm operates by analyzing individual frames of the video to extract spatial features. Through a series of convolutional layers, the CNN identifies key visual patterns such as shapes, textures, and edges within each frame. Complementing the CNN, the LSTM component focuses on capturing temporal dependencies and sequential patterns across multiple frames. By processing sequences of spatial features extracted by the CNN, the LSTM learns to model the temporal dynamics inherent in human activities. This enables the algorithm to

recognize not only individual frames but also the context and progression of actions over time.

7.2.3 DESIGN OF PROPOSED METHODOLOGY

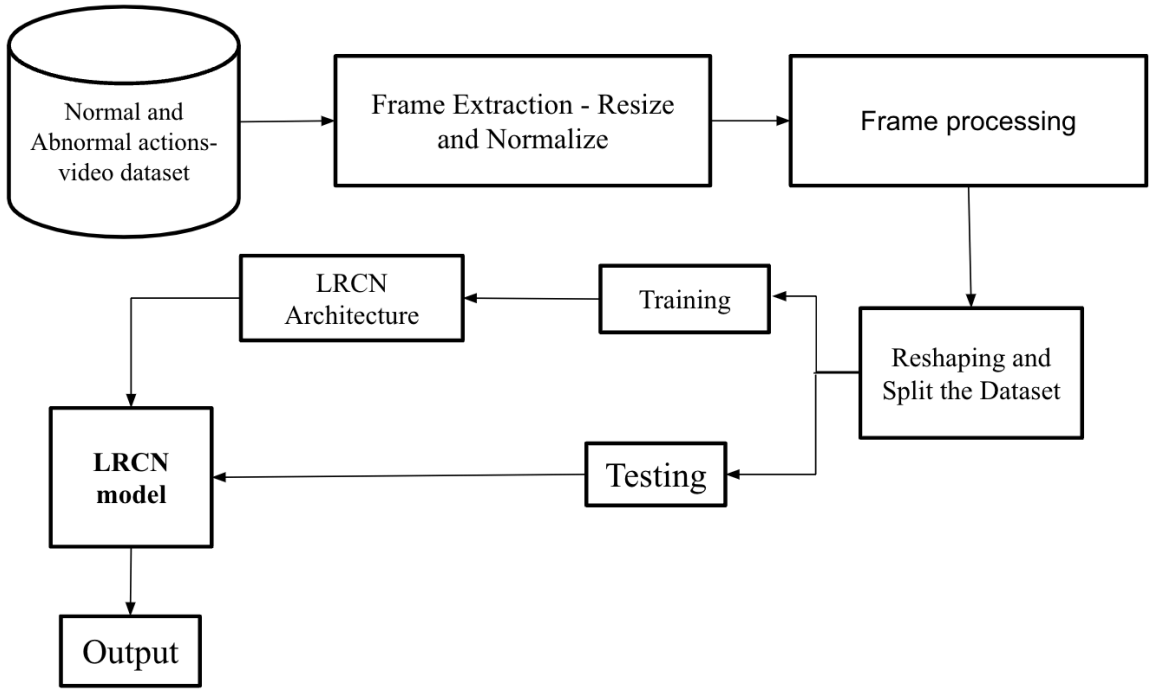

Fig. 7.2.3. Design of proposed methodology

7.2.4 DETECTION ALGORITHM

The detection algorithm is based on the Long-term Recurrent Convolutional Network (LRCN) model, which has been trained to recognize spatial and temporal patterns indicative of both normal and anomalous activities. Once trained, the LRCN model is applied to the surveillance video streams, where it processes each frame in sequential order. The model utilizes its convolutional neural network (CNN) layers to extract spatial features from individual frames, capturing visual patterns and characteristics. Simultaneously, the long short-term memory (LSTM) layers capture temporal dependencies across frames, enabling the model to understand the context and progression of actions over time. By analyzing both spatial and temporal information, the LRCN model can accurately distinguish between normal behaviors

and anomalies within the surveillance footage. When an anomaly is detected, the algorithm can trigger appropriate responses, such as generating alerts or notifications, thereby enhancing security measures and situational awareness in diverse environments. Through this detection algorithm, the system can effectively identify and categorize anomalous events, contributing to improved surveillance and safety protocols.

7.2.5 DATA PROCESSING

In the data processing phase of the proposed methodology, we begin by acquiring three distinct datasets capturing various human activities, including both normal behaviors and anomalies. These datasets undergo meticulous preprocessing to ensure consistency and compatibility. Video resolutions are standardized, and individual frames are extracted from each video while normalizing the durations. To enhance the dataset's diversity and robustness, we apply data augmentation techniques, introducing variations in lighting conditions and perspectives. Following preprocessing, the datasets are amalgamated while preserving clear labels to distinguish between different sources. This careful labeling is crucial for accurate model training and evaluation. Finally, the frames are resized to uniform dimensions, and additional preprocessing steps, such as normalization, are applied to prepare the data for subsequent model training and evaluation.

7.2.6 TRAINING

During the training phase, the Long-term Recurrent Convolutional Network (LRCN) model undergoes rigorous training on the compiled dataset. This dataset is judiciously split into training, validation, and test sets to ensure comprehensive model evaluation. The training set is utilized to train the model's parameters, allowing it to learn spatial and temporal features from the video sequences. The model is optimized using the Adam optimizer and categorical cross-entropy loss function, aiming to minimize prediction errors. To prevent overfitting and ensure generalization, an early stopping mechanism is employed, halting training when the

model's performance on the validation set no longer improves. Through this iterative process, the model learns to accurately identify normal human activities and detect anomalies within surveillance footage, ultimately enhancing security measures and situational awareness

7.2.7 ANALYSIS

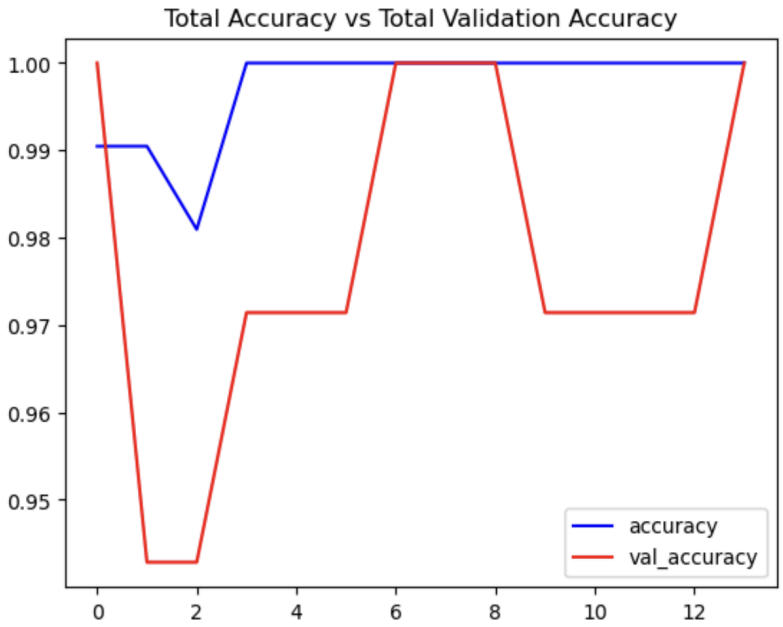

Fig. 7.2.7.1. Total Accuracy vs Validation Accuracy

Fig. 7.2.7.2. Total Loss vs Validation Loss

CHAPTER 8
SOFTWARE TESTING

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected.

8.1 TYPES OF TESTING

Unit testing:

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements. Each module can be tested with following strategies:

Black box testing:

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access
- Performance errors
- Initialization and termination errors
- The logical flow of the data is not checked.

White Box testing:

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

a) Guarantee that all independent paths have been executed.

b) Execute all logical decisions on their true and false sides.

c) Execute all loops at their boundaries and within their operational bounds.

d) Execute internal data structures to ensure their validity.

Integrating Testing:

Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

System Testing:

Involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user that the system meets all requirements of the client's specifications.

Acceptance Testing:

It is a pre-delivery testing in which the entire system is tested at the client's site on real world data to find errors. Validation: The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled.

Compilation Test:

It was a good idea to do our stress testing early on, because it gave us time to fix some of the unexpected deadlocks and stability problems that only occurred when components were exposed to very high transaction volumes.

Execution Test:

This program was successfully loaded and executed. Because of good programming there was no execution error.

8.2 TEST CASES

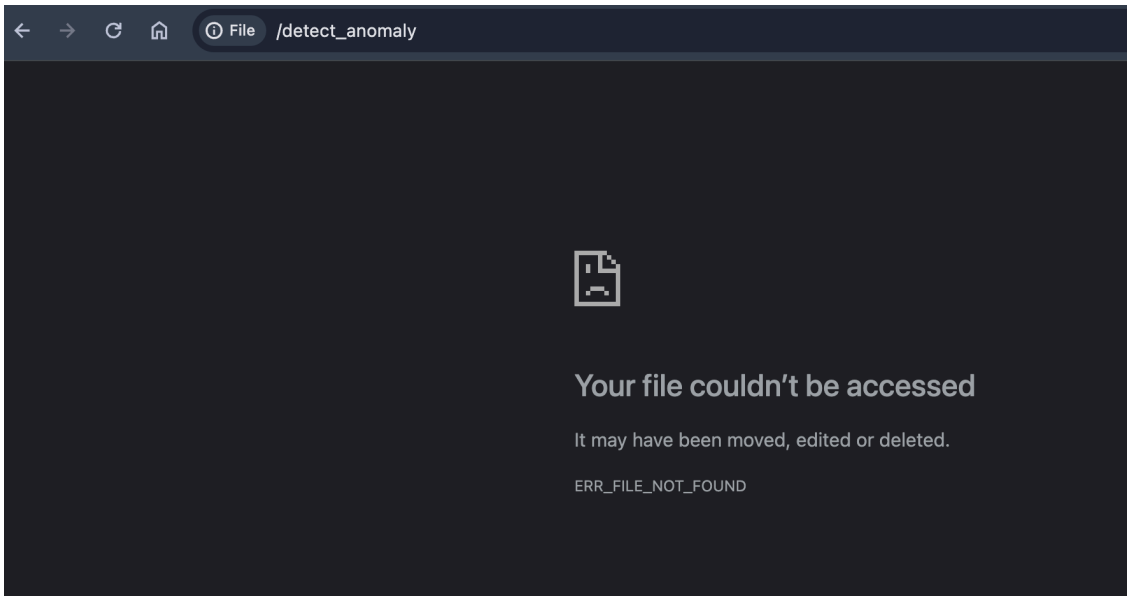

Fig. 8.2.1. Run without starting the local server

```
priya-pt7258@priya-pt7258 Anamoly detection % /usr/local/bin/python3 "/Users/priya-pt7258/Desktop/Anamoly detection/app.py"
Traceback (most recent call last):
  File "/Users/priya-pt7258/Desktop/Anamoly detection/app.py", line 12, in <module>
    model = load_model("Suspicious_Human_Activity_Detection_LRCN_Model.h5")
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/keras/src/saving/saving_api.py", line 183, in load_model
    return legacy_h5_format.load_model_from_hdf5(filepath)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/keras/src/legacy/saving/legacy_h5_format.py", line 116, in l
from_hdf5
    f = h5py.File(filepath, mode="r")
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/h5py/_hl/files.py", line 562, in __init__
    fid = make_fid(name, mode, userblock_size, fapl, fcpl, swmr=swmr)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/h5py/_hl/files.py", line 235, in make_fid
    fid = h5f.open(name, flags, fapl=fapl)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "h5py/_objects.pyx", line 54, in h5py._objects.with_phil.wrapper
  File "h5py/_objects.pyx", line 55, in h5py._objects.with_phil.wrapper
  File "h5py/h5f.pyx", line 102, in h5py.h5f.open
FileNotFoundError: [Errno 2] Unable to open file (unable to open file: name = 'Suspicious_Human_Activity_Detection_LRCN_Model.h5', errno = 2, error
'No such file or directory', flags = 0, o_flags = 0)
```

Fig. 8.2.2. Run without generating the model

CHAPTER 9

SCREENSHOTS

9.1 OUTPUT

Layer (type)	Output Shape	Param #
time_distributed (TimeDistributed)	(None, 30, 64, 64, 32)	896
time_distributed… (TimeDistributed)	(None, 30, 16, 16, 32)	0
time_distributed… (TimeDistributed)	(None, 30, 16, 16, 64)	18,496
time_distributed… (TimeDistributed)	(None, 30, 4, 4, 64)	0
time_distributed… (TimeDistributed)	(None, 30, 4, 4, 128)	73,856
time_distributed… (TimeDistributed)	(None, 30, 2, 2, 128)	0
time_distributed… (TimeDistributed)	(None, 30, 2, 2, 256)	131,3…
time_distributed… (TimeDistributed)	(None, 30, 1, 1, 256)	0
time_distributed… (TimeDistributed)	(None, 30, 256)	0
lstm (LSTM)	(None, 32)	36,992
dense (Dense)	(None, 7)	231

Total params: 261,799 (1022.65 KB)
Trainable params: 261,799 (1022.65 KB)
Non-trainable params: 0 (0.00 B)

Fig. 9.1.1. LRCN Model

```
Epoch 1/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 6s 625ms/step - accuracy: 0.2367 - loss: 2.0043 - val_accuracy: 0.2000 - val_loss: 2.0408
Epoch 2/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 303ms/step - accuracy: 0.2367 - loss: 1.8471 - val_accuracy: 0.4000 - val_loss: 2.1996
Epoch 3/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 313ms/step - accuracy: 0.4117 - loss: 1.8400 - val_accuracy: 0.2000 - val_loss: 2.3132
Epoch 4/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 298ms/step - accuracy: 0.3200 - loss: 1.8188 - val_accuracy: 0.2000 - val_loss: 2.3283
Epoch 5/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 278ms/step - accuracy: 0.3200 - loss: 1.7532 - val_accuracy: 0.2000 - val_loss: 2.2700
Epoch 6/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 272ms/step - accuracy: 0.4567 - loss: 1.6417 - val_accuracy: 0.2000 - val_loss: 2.1407
Epoch 7/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 275ms/step - accuracy: 0.4567 - loss: 1.5708 - val_accuracy: 0.2000 - val_loss: 2.1829
Epoch 8/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 268ms/step - accuracy: 0.5333 - loss: 1.3849 - val_accuracy: 0.2000 - val_loss: 1.9836
Epoch 9/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 269ms/step - accuracy: 0.4400 - loss: 1.2721 - val_accuracy: 0.4000 - val_loss: 1.6710
Epoch 10/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 287ms/step - accuracy: 0.7200 - loss: 1.0379 - val_accuracy: 0.2000 - val_loss: 1.7051
Epoch 11/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 293ms/step - accuracy: 0.6017 - loss: 0.9194 - val_accuracy: 0.4000 - val_loss: 1.7698
Epoch 12/70
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 281ms/step - accuracy: 0.6517 - loss: 1.1037 - val_accuracy: 0.4000 - val_loss: 2.2302
```

Fig. 9.1.2. Training

```
priya-pt7258@priya-pt7258 Anamoly detection % /usr/local/bin/python3 "/Users/priya-pt7258/Desktop/Anamoly detection/app.py"
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train
 * Serving Flask app 'app'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 167-052-740
INFO:werkzeug:127.0.0.1 - - [24/Mar/2024 00:21:39] "GET / HTTP/1.1" 200 -
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 732ms/step
['Meet and Split', 'Clapping', 'noFights', 'fights', 'kick', 'hit', 'catch']
noFights
INFO:werkzeug:127.0.0.1 - - [24/Mar/2024 00:21:52] "POST /detect_anomaly HTTP/1.1" 200 -
```

Fig. 9.1.3. Running the app in local server

Fig. 9.1.4. Video Input - Fight

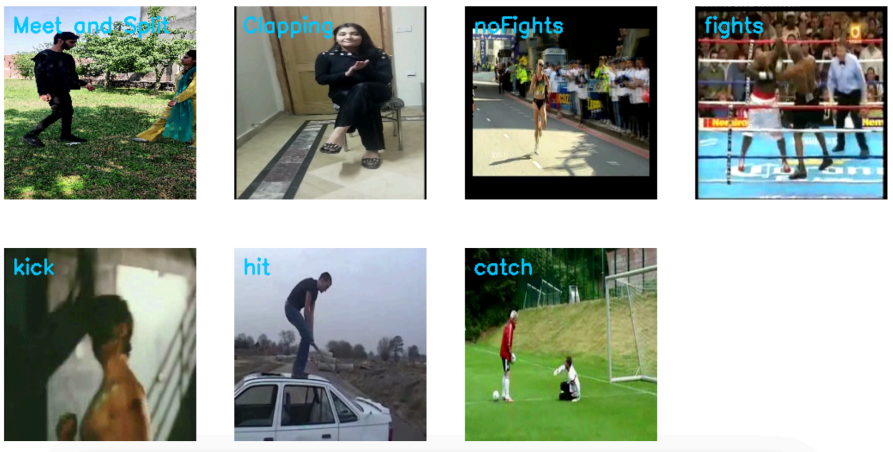

Fig. 9.1.5. Labeled data

CHAPTER 10
CONCLUSION AND FUTURE SCOPE

10.1 CONCLUSION

In summary, our project represents a significant advancement in the domain of anomaly detection. By harnessing the capabilities of deep learning, our LRCN model showcases remarkable skill in identifying and categorizing human actions and unusual events in surveillance footage. The extensive dataset, created by combining various data sources, provides a diverse and robust training environment, which improves the model's capacity to adapt to real-world situations. The achieved accuracy of approximately 98.36% on the test dataset underscores the model's remarkable performance, emphasizing its potential for bolstering security and safety. Moreover, our data preprocessing techniques ensure that the model effectively learns spatial and temporal features, while careful model architecture design facilitates this learning process.

10.2 FUTURE SCOPE

In the future, we envision several exciting avenues for improving our video-based anomaly detection system. To start, we plan to enhance the dataset by including a wider range of anomalies and activities. This will provide the model with a more comprehensive understanding of real- world situations, making it better equipped to handle complex scenarios. Moreover, we aim to optimize the model's architecture and investigate advanced techniques like attention mechanisms and generative adversarial networks. These efforts have the prospect of further improving the accuracy of our system for detecting anomalies. Finally, engaging in partnerships with professionals specializing in video forensics and anomaly detection could lead to opportunities for cross- disciplinary research, guaranteeing improvements in our system's usefulness across various security and surveillance scenarios.

ANNEXURE

REFERENCES

Here are some related works done by various authors using various techniques with a detailed view of the publishing date, their journal title and the techniques used.

[1] Duong, Huu-Thanh, Viet-Tuan Le, and Vinh Truong Hoang, "Deep Learning-Based Anomaly Detection in Video Surveillance: A Survey."Sensors 23, no. 11 (2023): 5024.

[2] Franklin, Ruben J., and Vidyashree Dabbagol, "Anomaly detection in videos for video surveillance applications using neural networks." In 2020 Fourth International Conference on Inventive Systems and Control (ICISC), pp. 632-637. IEEE, 2020.

[3] Abbas, Zainab K., and Ayad A. Al-Ani, "Detection of anomalous events based on deep learning-BiLSTM." Iraqi Journal of Information and Communication Technology 5, no. 3 (2022): 34-42.

[4] Khan, Sardar Waqar, Qasim Hafeez, Muhammad Irfan Khalid, Roobaea Alroobaea, Saddam Hussain, Jawaid Iqbal, Jasem Almotiri, and Syed Sajid Ullah, "Anomaly detection in traffic surveillance videos using deep learning." Sensors 22, no. 17 (2022): 6563.

[5] Ul Amin, S. "EADN: an efficient deep learning model for anomaly detection in videos. Mathematics 10 (9)(2022).

[6] Sultani, Waqas, Chen Chen, Mubarak Shah, "Real-world anomaly detection in surveillance videos." In Proceedings of the IEEE conference on CV and pattern recognition, pp. 6479-6488. 2018.

[7] Mehta, Parth, Atulya Kumar, and Shivani Bhattacharjee, "Fire and gun violence based anomaly detection system using deep neural networks." In 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 199-204. IEEE, 2020.

[8] Zhong, Jia-Xing, Nannan Li, Weijie Kong, Shan Liu, Thomas H. Li, and Ge Li, ”Graph convolutional label noise cleaner: Train a plug- and- play action classifier for anomaly detection." In Proceedings of the IEEE/CVF conference on CV and pattern recognition, pp. 1237-1246. 2019.

[9] Feng, Jia-Chang, Fa-Ting Hong, and Wei-Shi Zheng, "Mist: Multiple instance self-training framework for video anomaly detection." In Proceedings of the IEEE/CVF conference on CV and pattern recognition, pp. 14009-14018. 2021.

[10] Zhang, Jiangong, Laiyun Qing, and Jun Miao, ”Temporal convolutional network with complementary inner bag loss for weakly supervised anomaly detection."