

**A Project Report on**

**Anomaly Xpert: A deep learning approach to anomaly detection**

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the  
academic requirements for the award of the degree.

**Bachelor of Technology**

**In**

**Computer Science and Engineering**

Submitted by

A.Abhinav  
(20H51A0505)

G.Praneeth  
(20H51A0510)

C.Harishwar  
(20H51A05B9)

Under the esteemed guidance of

B.Gayathri  
(Associate Professor)



**Department of Computer Science and Engineering**

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

(UGC Autonomous)

\*Approved by AICTE \*Affiliated to JNTUH \*NAAC Accredited with A<sup>+</sup> Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

**2020 - 2024**

# **CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



### **CERTIFICATE**

**This is to certify that the Major Project report entitled "Anomaly Xpert: A deep learning approach to anomaly detection" being submitted by A.Abhinav (20H51A0505), G.Praneeth (20H51A0510), C.Harishwar (20H51A05B9) in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering**

Is a record of bonafide work carried out under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree.

**B.Gayathri**  
Associate Professor  
Dept. Of CSE

**Dr. S. Siva Skandha**  
Associate Professor and HOD  
Dept. of CSE

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express our heartfelt gratitude to all the people who helped in making this project a grand success.

We are grateful to **B.Gayathri, Associate Professor**, Department of CSE for her valuable technical suggestions and guidance during the execution of this project work.

We would like to thank, **Dr.Siva Skandha**, Head of the Department of CSE, CMR College of Engineering and Technology, who is the major driving forces to complete our project work successfully.

We are very grateful to **Dr. Ghanta Devadasu**, Dean-Academics, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major Dr. V A Narayana**, Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Dept Name for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary& Correspondent, CMR Group of Institutions, and **Shri Ch Abhinav Reddy**, CEO, CMR Group of Institutions for their continuous care and support.

Finally, we extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly or indirectly in completion of this project work.

A.Abhinav	20H51A0505
G.Praneeth	20H51A0510
C.Harishwar	20H51A05B9

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	LIST OF FIGURES	ii
	LIST OF TABLES	iii
	ABSTRACT	iv
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Problem Statement	2
	1.2 Research Objective	3
	1.3 Project Scope and Limitations	4
<b>2</b>	<b>BACKGROUND WORK</b>	<b>7</b>
	2.1. Camera based surveillance system	8
	2.1.1. Introduction	8
	2.1.2. Merits, Demerits and Challenges	11
	2.1.3. Implementation of Camera based surveillance system	12
	2.2. Suspicious Human Activity recognition	13
	2.2.1. Introduction	13
	2.2.2. Merits, Demerits and Challenges	16
	2.2.3. Implementation of Suspicious human activity recognition	17
	2.3. CCTV surveillance system	18
	2.3.1. Introduction	18
	2.3.2. Merits, Demerits and Challenges	21
	2.3.3. Implementation of CCTV surveillance system	22
<b>3</b>	<b>PROPOSED SYSTEM</b>	<b>23</b>
	3.1. Objective of Proposed Model	24
	3.2. Algorithms Used for Proposed Model	28
	3.3. Designing	29
	3.3.1.UML Diagram	30
	3.4. Stepwise Implementation and Code	50
<b>4</b>	<b>RESULTS AND DISCUSSION</b>	<b>51</b>
	4.1. Performance metrics	59

<b>5</b>	<b>CONCLUSION</b>	<b>60</b>
	5.1 Conclusion and Future Enhancement	62
<b>6</b>	<b>REFERENCES</b>	<b>64</b>
<b>7</b>	<b>GITHUB LINK</b>	<b>65</b>
<b>8</b>	<b>PUBLICATION CERTIFICATES</b>	<b>66</b>

### List of Figures

#### FIGURE

NO.	TITLE	PAGE NO.
3.3	Design	29
3.3.1	Class	31
3.3.2	Use Case	33
3.3.3	Sequence	35
3.3.4	Collaboration	37
4.1	Prediction	52
4.2	Run	52
4.3	CMD run	53
4.4	Suspicious activity	53
4.5	Footage selection	54
4.6	Frame analysis	54
4.7	Generation of frames	55
4.8	Anomaly detection	55
4.9	Frames detection	56
4.10	Suspicious frames folder	56
4.11	Suspicious frames	57
4.12	Image classification	57
4.13	Image recognition	58
4.14	Model learning curve	58

## List of Tables

### FIGURE

NO.	TITLE	PAGE NO.
4.1	Performance metrics	59

## **ABSTRACT**

This project introduces a deep learning-based system specifically tailored for automated detection of robbery and suspicious activities in CCTV footage, enhancing security monitoring capabilities. By leveraging convolutional neural networks (CNNs) for object detection and recurrent neural networks (RNNs) for temporal analysis, the system identifies anomalous behaviors indicative of criminal activities. The utilization of advanced deep learning techniques such as feature extraction and transfer learning further bolsters the system's efficiency and accuracy. Through thorough analysis of each frame, the system accurately timestamps instances of robbery or suspicious behavior throughout the video, providing a detailed timeline for review and investigation.

One of the key contributions of this project lies in its ability to generate real-time alerts and timestamps, facilitating swift review and retrieval of relevant footage by security personnel. This capability significantly enhances security monitoring and response capabilities, allowing for proactive measures to be taken in response to detected anomalies. The applications of anomaly detection span across diverse sectors, including cybersecurity, fraud detection, healthcare monitoring, predictive maintenance, and surveillance systems. Furthermore, anomaly detection plays a crucial role in healthcare systems by identifying abnormal patient vital signs, anomalous medical imaging findings, and potential disease outbreaks. In industrial settings, anomaly detection techniques contribute to predictive maintenance by identifying equipment malfunctions or anomalies in sensor data, minimizing downtime and optimizing operational efficiency.



# **CHAPTER 1**

## **INTRODUCTION**

# CHAPTER 1

## INTRODUCTION

### 1.1. Problem Statement

Anomaly detection is a critical task in various domains, including cybersecurity, healthcare monitoring, industrial processes, and financial fraud detection. The problem statement revolves around the need for robust and accurate anomaly detection systems that can effectively identify abnormal patterns or outliers in large-scale and complex datasets. One of the primary challenges in anomaly detection is the inherent imbalance between normal instances and anomalies, where anomalies are often rare and difficult to characterize. Traditional statistical methods may struggle to capture complex anomalies or adapt to evolving patterns, highlighting the need for advanced machine learning and deep learning techniques.

Additionally, anomaly detection systems must contend with noisy and high-dimensional data, varying data distributions, and dynamic environments where normal behavior can change over time. These factors contribute to the difficulty of accurately detecting anomalies while minimizing false positives and false negatives. Furthermore, the interpretability of anomaly detection models is a significant concern, especially in critical domains like healthcare and finance, where transparent and explainable decision-making is paramount. The problem statement also encompasses the scalability and efficiency of anomaly detection systems, particularly in real-time or streaming data scenarios. The ability to process and analyze vast amounts of data rapidly, while maintaining high detection accuracy, is crucial for timely anomaly detection and response.

## 1.2. Research Objective

**Enhancing Detection Accuracy:** Develop advanced machine learning and deep learning models that can accurately detect anomalies while minimizing false positives and false negatives. Investigate ensemble methods, hybrid approaches, and anomaly scoring techniques to improve overall detection performance.

**Handling Imbalanced Data:** Address the challenge of imbalanced datasets where anomalies are rare compared to normal instances. Explore techniques such as anomaly oversampling, anomaly-specific loss functions, and anomaly detection in streaming data to improve model performance on imbalanced data distributions.

**Adaptation to Dynamic Environments:** Develop anomaly detection algorithms capable of adapting to changing data distributions, concept drift, and evolving anomalies in dynamic environments. Investigate online learning approaches, transfer learning techniques, and adaptive model updating strategies for continuous learning and model refinement.

**Ensuring Interpretability:** Incorporate explainable AI techniques into anomaly detection models to enhance interpretability and transparency. Develop methods for visualizing and interpreting anomaly detection results, identifying influential features, and providing context-specific explanations for detected anomalies.

**Scalability and Real-Time Capabilities:** Design scalable anomaly detection architectures that can handle large volumes of data efficiently and operate in real-time or near real-time. Explore parallel processing, distributed computing, and optimized algorithms to improve scalability and responsiveness in dynamic data streams.

### 1.3. Project Scope and Limitations:

#### **Project Scope:**

- **Objectives:** Clearly define the objectives of the anomaly detection system, such as detecting unusual events or patterns in data, identifying potential security threats, minimizing false positives, and improving overall system efficiency.
- **Data Sources:** Determine the types of data sources to be used for anomaly detection, such as sensor data, log files, network traffic, financial transactions, or any other relevant data sources.
- **Data Preprocessing:** Define data preprocessing steps, including data cleaning, transformation, feature selection, and normalization, to prepare the data for anomaly detection algorithms.
- **Algorithm Selection:** Choose appropriate anomaly detection algorithms based on the nature of the data and the project objectives. This may include statistical methods, machine learning algorithms (e.g., Isolation Forest, One-Class SVM), deep learning models, or a combination of techniques.
- **Model Training and Evaluation:** Develop and train the anomaly detection models using historical data, validate the models using validation datasets, and evaluate their performance based on metrics such as accuracy, precision, recall, and F1 score.
- **Integration and Deployment:** Plan for integrating the anomaly detection system into existing systems or applications, ensuring compatibility, scalability, and performance. Consider deployment options such as on-premises or cloud-based solutions.

- **Documentation and Reporting:** Document the entire process, including data sources, preprocessing steps, algorithm selection, model training details, evaluation results, deployment procedures, and maintenance protocols. Generate reports and visualizations to communicate findings and insights to stakeholders.
- **Compliance and Security:** Ensure compliance with data privacy regulations and security measures throughout the project lifecycle, including data encryption, access controls, and audit trails.
- **Timeline and Resources:** Develop a project timeline with milestones, allocate resources (human, financial, computational), and establish roles and responsibilities for team members involved in the anomaly detection project.

**Limitations:**

- **Labeling of Anomalies:** One of the primary challenges is the lack of labeled data for anomalies. In many cases, anomalies are rare and unexpected events, making it difficult to obtain sufficient labeled data for training anomaly detection models. This can lead to issues such as imbalanced datasets and difficulties in accurately identifying anomalies.
- **Complexity of Anomalies:** Anomalies can vary widely in their complexity and nature. Simple anomalies, such as spikes or outliers, are relatively easy to detect, but more complex anomalies, such as subtle fraud patterns or sophisticated cyber-attacks, can be challenging to identify using traditional anomaly detection techniques.

- **Scalability and Performance:** Scalability is a concern, especially for large-scale systems or high-dimensional data. Some anomaly detection algorithms may struggle to handle large volumes of data efficiently, impacting their performance and response times.
- **Domain-specific Challenges:** Anomaly detection techniques may not be universally applicable across all domains. Different industries and use cases have unique characteristics and requirements, requiring tailored approaches and domain-specific knowledge for effective anomaly detection.
- **Adversarial Attacks:** In certain applications, such as cybersecurity or fraud detection, adversaries may deliberately manipulate data to evade detection by anomaly detection systems. Adversarial attacks can undermine the effectiveness of anomaly detection algorithms and require robust defenses.

# **CHAPTER 2**

# **BACKGROUND**

# **WORK**

## **CHAPTER 2**

### **BACKGROUND WORK**

#### **2.1. Camera based surveillance system**

##### **2.1.1. Introduction**

Nowadays, security is measure concern in every organization. To this satisfy issue the organizations use surveillance cameras. The limitation in using them is that there must be an operator to watch the stream from the cameras and take respective decisions. The use of camera based surveillance has extended from security to tracking, environment and threat analysis and many more. By using the power of modern computing and hardware it is possible to automate the process. Storage by using various technologies. The methods used are elaborated further. 2.1 Implementation of Closed-circuit Television (CCTV) Using Wireless Internet Protocol (IP) Camera [1]: The author Michael F.Adaramola in this paper presents three techniques for configuring, interfacing and networking of a wireless IP-based camera for real time security surveillance systems design. The three different real-time implementations techniques proposed for configuring, interfacing and networking the IP camera are: 1) Accessing the IP-based camera by using the WANSCAM or XXCAM vendor software, 2) Accessing the IP-based camera by Firefox® web browser, and 3) Accessing the IP camera by MATLAB with SIMULINK on an internet system.

The live streaming of video based on the proposed techniques can be adapted for image detection, recognition and tracking for real-time intelligent security surveillance systems design. The paper also carried out a thorough comparative analysis of the three methods of achieving video streaming resulting from the output of the IP-based cameras. The analysis shows that the WANSCAM or XXCAM software displays the best video animations from the IP based cameras when compared with the performance of the other methods. The emergence of machine learning, Deep learning, and computer vision tools have made this process efficient and feasible for general purpose use. So instead of using human support for monitoring and insight generation, we can let the processor and machine learning system do the task in a more efficient and errorless way. Here we have mentioned few approaches which had helped us in solving this problem.



### 2.1.2. Merits, Demerits and Challenges

#### **Merits:**

- **Enhanced Security**-Camera-based surveillance systems provide a visible deterrent to potential threats and intruders, enhancing overall security and safety in monitored areas. The presence of cameras can deter criminal activities and unauthorized access.
- **Real-Time Monitoring**-Surveillance cameras enable real-time monitoring of activities, allowing security personnel to respond promptly to incidents, emergencies, or suspicious behavior. This real-time monitoring capability enhances situational awareness and enables rapid decision-making.
- **Evidence Collection**-Cameras capture visual evidence of events, incidents, and activities, which can be crucial for investigations, legal proceedings, and evidence collection. High-definition video footage can provide clear documentation of events and aid in identifying perpetrators.
- **Remote Monitoring**-Many camera-based surveillance systems offer remote monitoring capabilities, allowing authorized personnel to access live video feeds and recordings from anywhere with an internet connection. This flexibility enables remote management and monitoring of multiple locations.

**Demerits:**

- **Privacy Concerns**-One of the primary challenges of camera-based surveillance systems is the potential invasion of privacy. Continuous monitoring through cameras can raise concerns about personal privacy and data protection, especially in public spaces or areas where individuals expect privacy.
- **Data Storage and Management**-Camera-based systems generate vast amounts of data, leading to challenges in data storage, management, and processing. Storing high-resolution video feeds requires significant storage capacity, and efficient data management strategies are needed to handle the volume of data generated.
- **Bandwidth and Network Issues**-Streaming high-definition video feeds from multiple cameras over a network can strain bandwidth and cause network congestion. This can lead to delays in data transmission, buffering issues, and reduced system performance, especially in large-scale surveillance deployments.
- **Cost of Infrastructure**-Implementing a comprehensive camera-based surveillance system requires significant investment in infrastructure, including cameras, network equipment, storage servers, and analytics software. The initial setup and ongoing maintenance costs can be substantial.
- **False Alarms and Accuracy**-Camera-based systems may generate false alarms due to factors such as environmental changes, lighting conditions, and movement of non-threatening objects. Ensuring the accuracy of anomaly detection and reducing false positives/negatives is a key challenge.

### **Challenges:**

- **Privacy Concerns**-One of the primary challenges is balancing security needs with individual privacy rights. Surveillance cameras may capture sensitive information or invade personal privacy, leading to concerns among the public and regulatory scrutiny.
- **Data Storage and Management**-Managing and storing large volumes of video data generated by surveillance cameras pose significant challenges. It requires robust storage infrastructure, data retention policies, backup procedures, and efficient data management practices to handle the data effectively.
- **Bandwidth and Network Constraints**-Streaming high-definition video feeds from multiple cameras over a network can strain bandwidth and cause network congestion. This can lead to delays in data transmission, buffering issues, and reduced system performance, especially in large-scale deployments.
- **Cost of Infrastructure**-Implementing a comprehensive camera-based surveillance system requires substantial investment in infrastructure, including cameras, network equipment, storage servers, monitoring software, and maintenance. The initial setup costs and ongoing maintenance expenses can be significant.
- **False Alarms and Accuracy**-Surveillance cameras may generate false alarms due to factors such as environmental changes, lighting conditions, movement of non-threatening objects, or misinterpretation of activities. Ensuring the accuracy of anomaly detection algorithms and reducing false positives/negatives is a key challenge.
- **Cybersecurity Risks**-Surveillance systems are vulnerable to cybersecurity risks such as hacking, unauthorized access to video feeds, tampering with cameras or recordings, and data breaches. Robust cybersecurity measures, encryption, access controls, and regular security audits are essential to mitigate these risks.

### **2.1.3. Implementation of Camera based surveillance system**

- **Assessment and Planning**-Conduct a thorough assessment of the surveillance needs and objectives, including the areas to be monitored, the types of threats or risks, and the desired features and functionalities of the surveillance system.
- **Camera Selection**-Choose suitable surveillance cameras based on factors such as resolution, field of view, night vision capabilities, weatherproofing (for outdoor cameras), PTZ (pan-tilt-zoom) functionality, and analytics capabilities (e.g., facial recognition, object detection).
- **Network Infrastructure**-Set up a robust network infrastructure to support the surveillance system, including wired or wireless connections, PoE (Power over Ethernet) switches for IP cameras, network bandwidth optimization, and VLAN (Virtual Local Area Network) configurations for security segmentation.
- **Installation and Configuration**-Install surveillance cameras at strategic locations according to the surveillance plan, ensuring optimal coverage of target areas and minimizing blind spots. Configure camera settings such as resolution, frame rate, motion detection sensitivity, recording modes (continuous recording or event-triggered recording), and storage locations (on-premises storage or cloud storage).
- **Data Storage and Management**-Set up a centralized data storage solution for storing video footage and recordings, considering factors such as storage capacity, redundancy, backup procedures, and data retention policies compliant with legal requirements.

## **2.2. Suspicious Human activity detection**

### **2.2.1. Introduction**

Suspicious Human Activity Recognition from Video Surveillance is an active research area of image processing and computer vision which involves recognition of human activity and categorizes them into normal and abnormal activities. Abnormal activities are the unusual or suspicious activities rarely performed by the human at public places, such as left luggage for explosive attacks, theft, running crowd, fights and attacks, vandalism and crossing borders. Normal activities are the usual activities performed by the human at public places, such as running, boxing, jogging and walking, hand waving and clapping. Now-a-days, use of video surveillance is increasing day by day to monitor the human activity which prevents the suspicious activities of the human.

Video Surveillance captures images of moving objects in order to watch assault and fraud, comings and goings, prevent theft, as well as manage crowd movements and incidents. In public places, human performs normal (usual) and abnormal (suspicious or unusual) activities. Normal activities are the usual activities that are not dangerous for the human world but abnormal activities may be dangerous for all over the world. Therefore, an intelligent surveillance system is required that can recognize all the activities and identify the more dangerous and suspicious activities performed by a human being. There are two types of surveillance system-first is semi-autonomous in which video is recorded and sent for analysis by human expert. Non-intelligent video surveillance requires the continuous monitoring by human, which is very costly, problematic and also very difficult and challenging to watch over all the videos continuously by a guard to prevent the suspicious human activity.

### 2.2.2. Merits, Demerits and Challenges

#### **Merits:**

- **Early Threat Detection**-Suspicious human activity detection systems can identify potential threats or abnormal behaviors early, allowing security personnel to respond promptly and mitigate risks before they escalate into serious incidents.
- **Crime Prevention**-By detecting suspicious activities such as unauthorized access, loitering, aggressive behavior, or vandalism, these systems contribute to crime prevention and deterrence, creating a safer environment for individuals and assets.
- **Enhanced Security Monitoring**-Suspicious activity detection systems provide continuous and real-time monitoring of areas of interest, enhancing overall security surveillance capabilities and situational awareness for security personnel.
- **Efficient Resource Allocation**-These systems help in efficient resource allocation by directing security personnel to specific areas or incidents based on detected suspicious activities, optimizing response times and operational effectiveness.
- **Automated Alerting and Notifications**-Suspicious activity detection systems can automatically generate alerts, notifications, or alarms when suspicious behaviors are detected, enabling immediate action and response by security teams.
- **Video Evidence and Forensics**-By capturing video footage of suspicious activities, these systems provide valuable evidence for investigations, forensic analysis, and legal proceedings, aiding in identifying perpetrators and resolving incidents.

**Demerits:**

- **False Positives**-One of the primary challenges is the potential for false positives, where normal behaviors or activities are incorrectly flagged as suspicious. This can lead to unnecessary alerts, wasted resources, and decreased trust in the system's accuracy.
- **Privacy Concerns**-Suspicious activity detection systems may raise privacy concerns, especially in public spaces or areas where individuals expect privacy. Continuous monitoring and recording of activities can infringe on privacy rights and lead to backlash from the public or regulatory authorities.
- **Ethical Considerations**-Ethical considerations arise regarding the use of surveillance technology and the potential for misuse or abuse. Questions about consent, transparency, data protection, and discrimination based on behavioral analysis need to be addressed.
- **Technical Limitations**-The effectiveness of suspicious activity detection systems can be affected by technical limitations such as limited camera coverage, poor lighting conditions, camera blind spots, occlusions, and environmental factors (e.g., weather, noise).
- **Cost of Implementation**-Implementing and maintaining sophisticated suspicious activity detection systems can be costly, requiring investment in high-quality cameras, analytics software, network infrastructure, storage solutions, and ongoing maintenance and upgrades.

### **Challenges:**

- **False Positives and False Negatives** -One of the primary challenges is minimizing false positives (incorrectly flagging normal behavior as suspicious) and false negatives (missing genuinely suspicious activities). Balancing accuracy while reducing these errors requires sophisticated algorithms and continuous tuning.
- **Complexity of Human Behavior**-Human behavior is complex and diverse, making it challenging to accurately detect suspicious activities. The variability in gestures, movements, actions, and context adds complexity to algorithm development and training.
- **Privacy and Ethical Concerns**-Suspicious activity detection systems raise privacy and ethical concerns regarding individual rights, consent, data protection, and potential misuse of surveillance technology. Striking a balance between security needs and privacy rights is crucial.
- **Context Awareness**-Understanding the context of activities is essential for accurate detection of suspicious behaviors. Factors such as time of day, location, environmental conditions, social norms, and historical patterns need to be considered to avoid misinterpretations.
- **Adaptability to Environment**-Surveillance environments can be dynamic, with changes in lighting, weather, crowd density, and background noise. Systems must adapt to these variations to maintain detection accuracy and reliability.



### **2.2.3. Implementation of Suspicious Human Activity detection**

- **Define Objectives and Requirements**-Clearly define the objectives of suspicious human activity detection, including the types of suspicious behaviors to be detected (e.g., loitering, unauthorized access, aggressive behavior) and the areas or environments to be monitored.
- **Select Suitable Technologies** -Choose appropriate technologies for suspicious activity detection, including surveillance cameras with the required resolution, field of view, night vision capabilities, and analytics software for behavior analysis.
- **Design System Architecture**-Design the system architecture for suspicious activity detection, including camera placement, network infrastructure (wired or wireless), data storage solutions (on-premises or cloud-based), and integration with existing security infrastructure.
- **Install and Configure Hardware**-Install surveillance cameras at strategic locations based on the defined objectives and requirements, ensuring optimal coverage of target areas and minimizing blind spots.
- **Implement Analytics and Algorithms**-Implement analytics algorithms for suspicious activity detection, such as behavior analysis, motion detection, object tracking, crowd monitoring, and anomaly detection.

## **2.3. CCTV surveillance system**

### **2.3.1. Introduction**

CCTV (Closed Circuit Tele-Vision) is one of the most widely used physical security technologies. A surveillance camera is a video collection device installed at a particular location and utilized for a variety of purposes. As CCTV performance has become enhanced recently, technology is being developed that attempts to perform automated processing through facial recognition using the facial information acquired from a CCTV system. However, if these technologies are exploited maliciously, privacy may be seriously violated. A set of communication equipment devices that collect image information from a surveillance camera device installed at a particular location, and transmit the images via an opened wire/wireless communication channel, so that only specified persons can receive it.

Image monitoring control server a server that stores, manages, and monitors the image information received from a surveillance camera.

The image monitoring server is composed of several modules such as encryption, decryption, facial area detection, privacy protected image, image saving, and monitoring. The monitoring module can be located behind the en/decryption module or privacy protected image module, depending upon whether privacy protection is to be applied or not, while monitoring the image. Client a system or user that seeks to receive and use the CCTV image from the image monitoring and control server. Desktops, laptops, and mobile phones can be an example of clients. The client is composed of the en/decryption, facial recognition, and image utilization modules. However, if these technologies are exploited maliciously, privacy may be seriously violated.

### 2.3.2. Merits, Demerits and Challenges

#### **Merits:**

- **Crime Deterrence**-CCTV cameras act as a visible deterrent to criminal activities such as theft, vandalism, trespassing, and other unlawful behaviors. The presence of cameras can discourage potential offenders from committing crimes.
- **Enhanced Security**-CCTV surveillance systems enhance overall security by providing continuous monitoring of premises, assets, and individuals. They help in detecting and responding to security threats, unauthorized access, and suspicious activities in real time.
- **Evidence Collection**-CCTV cameras capture video footage of events, incidents, and activities, providing valuable evidence for investigations, forensic analysis, and legal proceedings.
- **Remote Monitoring**-Many CCTV systems offer remote monitoring capabilities, allowing authorized personnel to access live video feeds and recordings from anywhere with an internet connection. This enables real-time monitoring and management of security operations.
- **Incident Response**-CCTV surveillance systems facilitate quick and effective incident response by providing visual confirmation of security alerts, alarms, or anomalies. Security personnel can assess situations, dispatch responders, and take appropriate actions based on video evidence.

**Demerits:**

- **Privacy Concerns**-One of the primary demerits is the potential invasion of privacy. CCTV cameras continuously monitor public and private spaces, raising concerns about individual privacy rights, data protection, and surveillance overreach.
- **Cost of Implementation and Maintenance**-Implementing and maintaining a comprehensive CCTV surveillance system can be costly, requiring investment in cameras, recording equipment, monitoring software, network infrastructure, storage solutions, and ongoing maintenance.
- **Data Security Risks**-CCTV systems are susceptible to data security risks such as hacking, unauthorized access, data breaches, and tampering with video footage. Ensuring robust cybersecurity measures, encryption, access controls, and secure data storage is essential.
- **Limited Coverage and Blind Spots**-CCTV cameras have limited coverage areas and may have blind spots or areas outside their field of view. Achieving comprehensive coverage may require deploying multiple cameras, which can increase costs and complexity.
- **Reliability and Technical Issues**-CCTV systems can experience technical issues such as camera malfunctions, recording errors, network connectivity issues, power outages, and system failures. Regular maintenance, troubleshooting, and proactive monitoring are necessary to ensure system reliability.
- **False Alarms and Misinterpretations**-CCTV systems may generate false alarms due to factors such as environmental changes, lighting conditions.

### **Challenges:**

- **Data Security**-Protecting CCTV data from unauthorized access, hacking, data breaches, and tampering is a critical challenge. Implementing robust cybersecurity measures, encryption, access controls, and secure data storage is essential to mitigate data security risks.
- **Technical Limitations**-CCTV systems may face technical limitations such as limited camera coverage, blind spots, poor lighting conditions, image quality issues, network connectivity problems, and hardware failures. Addressing these technical challenges requires regular maintenance, upgrades, and troubleshooting.
- **Cost of Implementation and Maintenance**-The initial cost of implementing CCTV systems, including cameras, recording equipment, network infrastructure, storage solutions, monitoring software, and installation, can be substantial. Ongoing maintenance, upgrades, and operational costs also add to the overall expenses.
- **Integration with Existing Systems**-Integrating CCTV systems with existing security infrastructure, access control systems, alarms, analytics software, and incident response protocols can be complex. Ensuring seamless interoperability, data sharing, and synchronization across integrated systems is challenging.
- **False Alarms and False Positives**-CCTV systems may generate false alarms and false positives due to factors such as environmental changes, motion detection errors, misinterpretation of activities, or technical glitches.
- **Training and Education**-Providing adequate training and education to security personnel, operators, and stakeholders on how to use CCTV systems effectively, interpret video footage, respond to incidents, and follow protocols.

### **2.3.3. Implementation of CCTV surveillance system**

- **Define Objectives and Requirements**-Clearly define the objectives of the CCTV surveillance system, such as monitoring security, enhancing safety, preventing theft, or improving operational efficiency.
- **Site Survey and Planning**-Conduct a site survey to assess the physical layout, environmental conditions, lighting levels, potential security risks, and areas requiring surveillance coverage.
- **Select and Install Cameras**-Choose appropriate CCTV cameras based on the defined requirements, considering factors such as resolution (e.g., HD, 4K), lens types (e.g., fixed, varifocal), night vision capabilities, weatherproofing (for outdoor cameras), and special features (e.g., motion detection, PTZ).
- **Setup Network Infrastructure**-Set up the necessary network infrastructure to connect CCTV cameras, recording devices, and monitoring stations. This may include Ethernet cables, PoE (Power over Ethernet) switches, routers, and network cables.
- **Install Recording and Storage Devices**-Install recording devices such as DVRs (Digital Video Recorders) or NVRs (Network Video Recorders) to capture and store video footage from CCTV cameras.
- **Configure Monitoring Software**-Install and configure CCTV monitoring software on designated monitoring stations or computers to access live video feeds, playback recorded footage, and manage camera settings.

# **CHAPTER 3**

## **PROPOSED SYSTEM**

## **CHAPTER 3**

### **PROPOSED SYSTEM**

#### **3.1. Objective of proposed model:**

- **Risk Mitigation:** Mitigate risks associated with anomalies by taking proactive measures, such as alerting stakeholders, triggering automated responses, or initiating corrective actions.
- **Improved Security:** Enhance security by detecting suspicious activities, intrusions, fraud attempts, or malicious behavior that may pose threats to systems, networks, or data.
- **Optimization:** Identify inefficiencies, outliers, or unusual patterns in operations, performance metrics, or business processes to optimize resources, workflows, or decision-making.
- **Performance Monitoring:** Monitor performance metrics, key performance indicators (KPIs), or service levels to detect anomalies that may impact operational efficiency or customer satisfaction.
- **Insight Generation:** Gain insights into data trends, patterns, or anomalies that can provide valuable information for decision-making, trend analysis, anomaly root cause analysis, or future planning.



### 3.1. Algorithms used for proposed model:

#### CNN algorithm for image classification:

```
# Import necessary libraries
import tensorflow as tf
from tensorflow.keras import layers, models
# Define CNN model architecture
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1))) #
Convolutional layer with 32 filters
model.add(layers.MaxPooling2D((2, 2))) # Max pooling layer
model.add(layers.Conv2D(64, (3, 3), activation='relu')) # Convolutional layer with 64 filters
model.add(layers.MaxPooling2D((2, 2))) # Max pooling layer
# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
# Load and preprocess dataset (e.g., MNIST dataset)
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1)) / 255.0 # Normalize pixel values
test_images = test_images.reshape((10000, 28, 28, 1)) / 255.0 # Normalize pixel values
# Train the model
model.fit(train_images, train_labels, epochs=5, batch_size=64, validation_data=(test_images,
test_labels))
# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels)
print("Test Accuracy:", test_acc)
```

### **RNN algorithm for image segmentation:**

```
# Import necessary libraries
import tensorflow as tf
from tensorflow.keras import layers, models

# Define RNN model architecture for image segmentation
model = models.Sequential()
model.add(layers.LSTM(256, return_sequences=True, input_shape=(None, 256, 256, 3))) #
LSTM layer with 256 units
model.add(layers.TimeDistributed(layers.Conv2D(64, (3, 3), activation='relu'))) # Time-
distributed Conv2D layer
model.add(layers.TimeDistributed(layers.Conv2D(64, (3, 3), activation='relu'))) # Time-
distributed Conv2D layer

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy')

# Load and preprocess image data for segmentation
# X_train, y_train, X_val, y_val = load_and_preprocess_data()

# Train the model
# model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=10, batch_size=32)

# Evaluate the model
# test_loss = model.evaluate(X_test, y_test)
# print("Test Loss:", test_loss)
```

### **GAN algorithm:**

#### **# Import necessary libraries**

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers

# Define the generator model
def build_generator(latent_dim, output_shape):
    model = models.Sequential()
    model.add(layers.Dense(128, input_dim=latent_dim, activation='relu')) # Dense layer with
    ReLU activation
    model.add(layers.Dense(256, activation='relu')) # Dense layer with ReLU activation
    model.add(layers.Dense(np.prod(output_shape), activation='sigmoid')) # Output layer with
    sigmoid activation
    model.add(layers.Reshape(output_shape)) # Reshape to output shape
    return model

# Define the discriminator model
def build_discriminator(input_shape):
    model = models.Sequential()
    model.add(layers.Flatten(input_shape=input_shape)) # Flatten layer
    return model

# Define GAN model
def build_gan(generator, discriminator):
    discriminator.trainable = False # Freeze discriminator during GAN training
    gan_model = models.Sequential()
    gan_model.add(generator) # Add generator model
    return gan_model

# Define hyperparameters and data shapes
latent_dim = 100 # Latent dimension for generator input
input_shape = (28, 28, 1) # Input shape for images
output_shape = (28, 28, 1) # Output shape for generated images
```

```
# Build and compile the discriminator
discriminator = build_discriminator(input_shape)
discriminator.compile(loss='binary_crossentropy', optimizer=optimizers.Adam(lr=0.0002,
beta_1=0.5), metrics=['accuracy'])

# Build the generator
generator = build_generator(latent_dim, output_shape)

# Build the GAN model
gan = build_gan(generator, discriminator)

# Generate random noise as input for generator
noise = np.random.normal(0, 1, (batch_size, latent_dim))

# Generate fake images using the generator
generated_images = generator.predict(noise)

# Train the discriminator on real and fake images
d_loss_real = discriminator.train_on_batch(real_images, np.ones((batch_size, 1)))
d_loss_fake = discriminator.train_on_batch(generated_images, np.zeros((batch_size, 1)))

# Train the generator via GAN model
gan_loss = gan.train_on_batch(noise, np.ones((batch_size, 1)))
```

### 3.3. Designing:

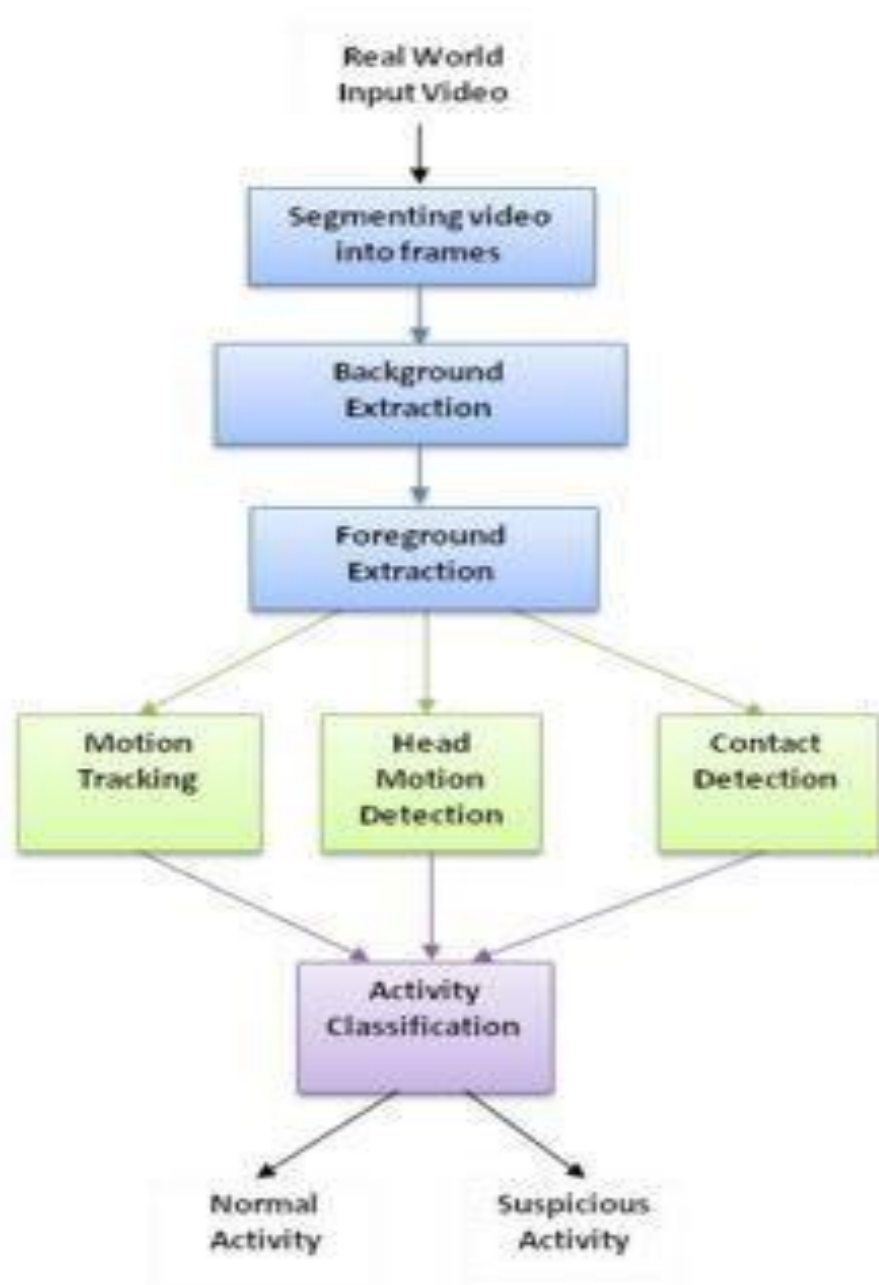


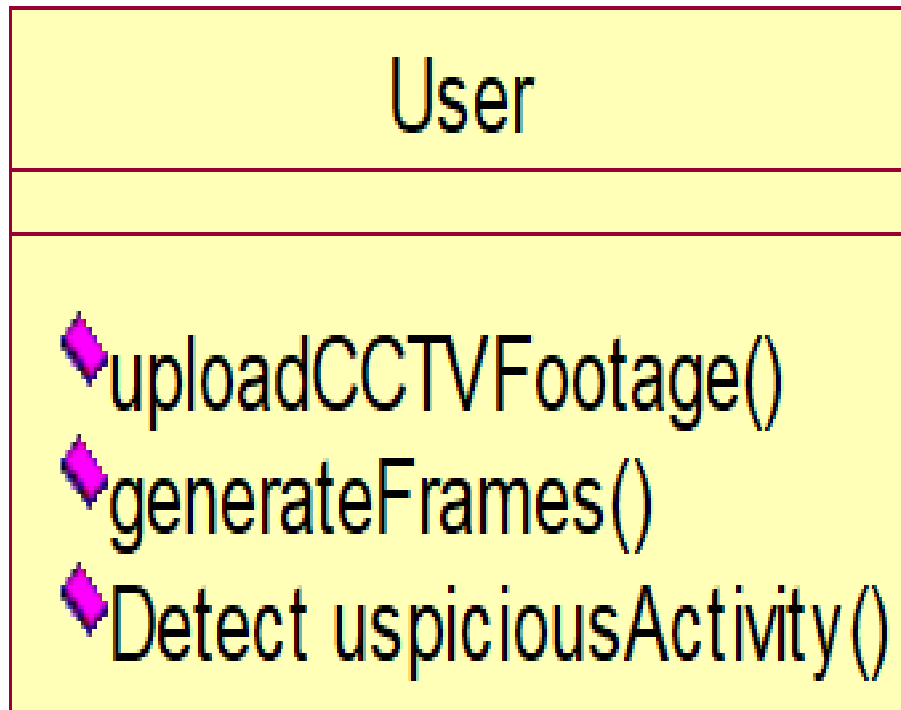
Fig. 3.3. Design

### **3.3.1. UML Diagrams:**

#### **Class diagram:**

A class diagram in the Unified Modeling Language(UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information. A class diagram is a graphical representation in Unified Modeling Language (UML) that illustrates the structure and relationships among classes within a software system. Classes, depicted as rectangles, encapsulate attributes (data) and operations (behavior) related to specific entities or concepts. Attributes describe the state of a class, while operations define its functionality. Relationships between classes are depicted using lines, such as associations that show how classes interact, inheritance indicating subclass-superclass relationships, and dependencies denoting class dependencies. Aggregation and composition represent part-whole relationships.

These classes are the building blocks of object-oriented programming, embodying real-world entities or abstract concepts and organizing them into a structured hierarchy. The relationships between classes are illustrated through various types of lines, each conveying a different aspect of their connection. Associations, indicated by lines between classes, reveal how objects of different classes interact with each other. Inheritance relationships, denoted by arrow-headed lines, showcase the hierarchical relationships between base or parent classes and derived or child classes, allowing for code reuse and promoting modular design. Dependencies, portrayed by dashed lines, signify the reliance of one class on another without direct ownership or containment. Additionally, aggregation and composition relationships further refine the understanding of how classes are composed of or connected to other classes, either as parts of a whole (aggregation) or as integral components with exclusive ownership (composition).



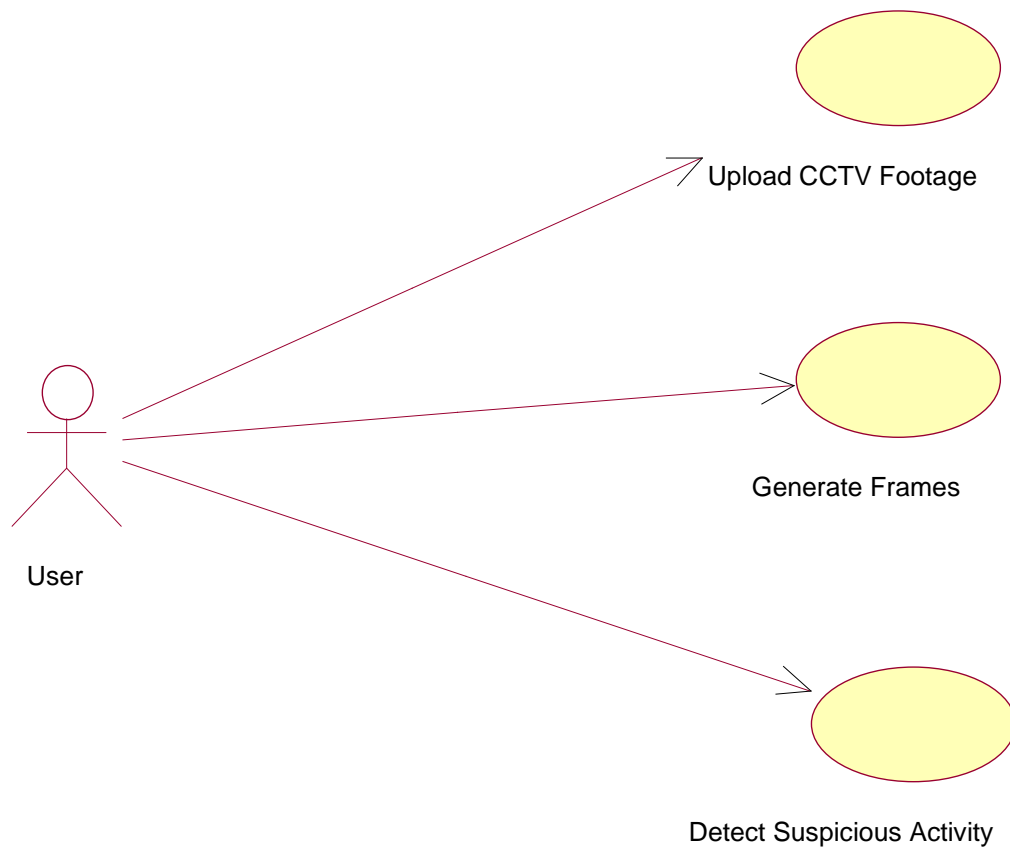
**Fig.3.3.1. Class**

### **USE CASE Diagram:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. A use case diagram is another vital component of Unified Modeling Language (UML) that visually represents the functional requirements and interactions of a system from the perspective of its users. Use cases are scenarios or sequences of actions that describe how users interact with the system to achieve specific goals or tasks. In a use case diagram, actors (such as users, external systems, or entities) are represented as stick figures, and use cases are depicted as ovals or ellipses connected by lines to actors.

These lines show the relationships between actors and use cases, indicating which actors are involved in each use case. Additionally, include extends and includes relationships that depict optional or conditional behaviors within use cases. Use case diagrams are valuable tools for understanding user requirements, defining system functionalities, identifying system boundaries, and facilitating communication between stakeholders and development teams throughout the software development lifecycle. The lines connecting actors to use cases in a use case diagram illustrate the relationships and interactions between them. These connections indicate which actors are involved in each use case and help clarify the roles and responsibilities of different actors within the system. Additionally, use case diagrams can include extends and includes relationships to capture optional or conditional behaviors within use cases.



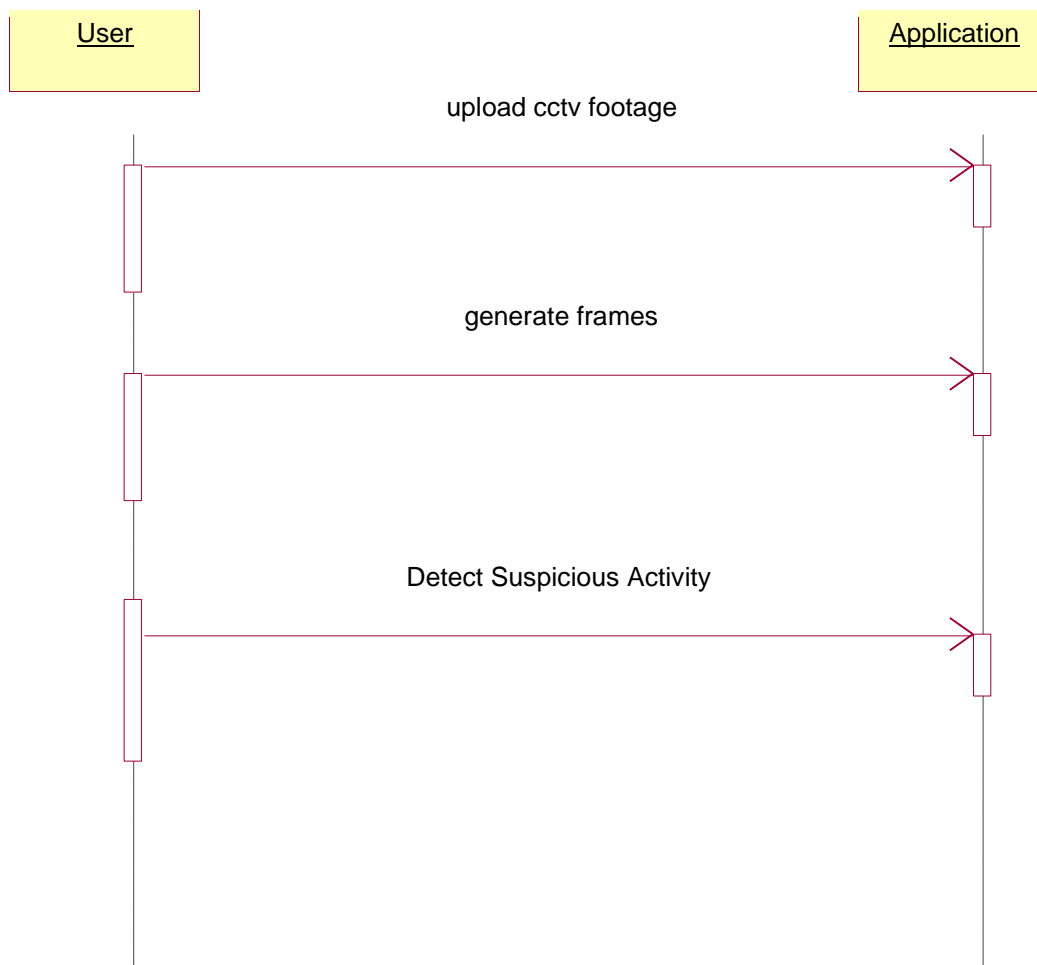


**Fig.3.3.2. Use Case**

### **Sequence Diagram:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams. A sequence diagram is a type of interaction diagram in Unified Modeling Language (UML) that illustrates the interactions and message flow between objects or components in a system over time. It shows the sequence of messages exchanged between objects or components in chronological order, helping to visualize the dynamic behavior of a system or software application.

In a sequence diagram, objects are represented as boxes or rectangles, and the messages exchanged between them are shown as arrows with sequence numbers to indicate the order of execution. Lifelines, represented as vertical dashed lines extending from objects, depict the lifespan of objects during the interaction. Additionally, activation bars show when an object is active and processing a message. Sequence diagrams are valuable for understanding the flow of control, collaboration among objects, timing constraints, and the overall behavior of a system's components during runtime. They are widely used in software design, development, and communication among stakeholders to ensure that the system functions correctly and efficiently.

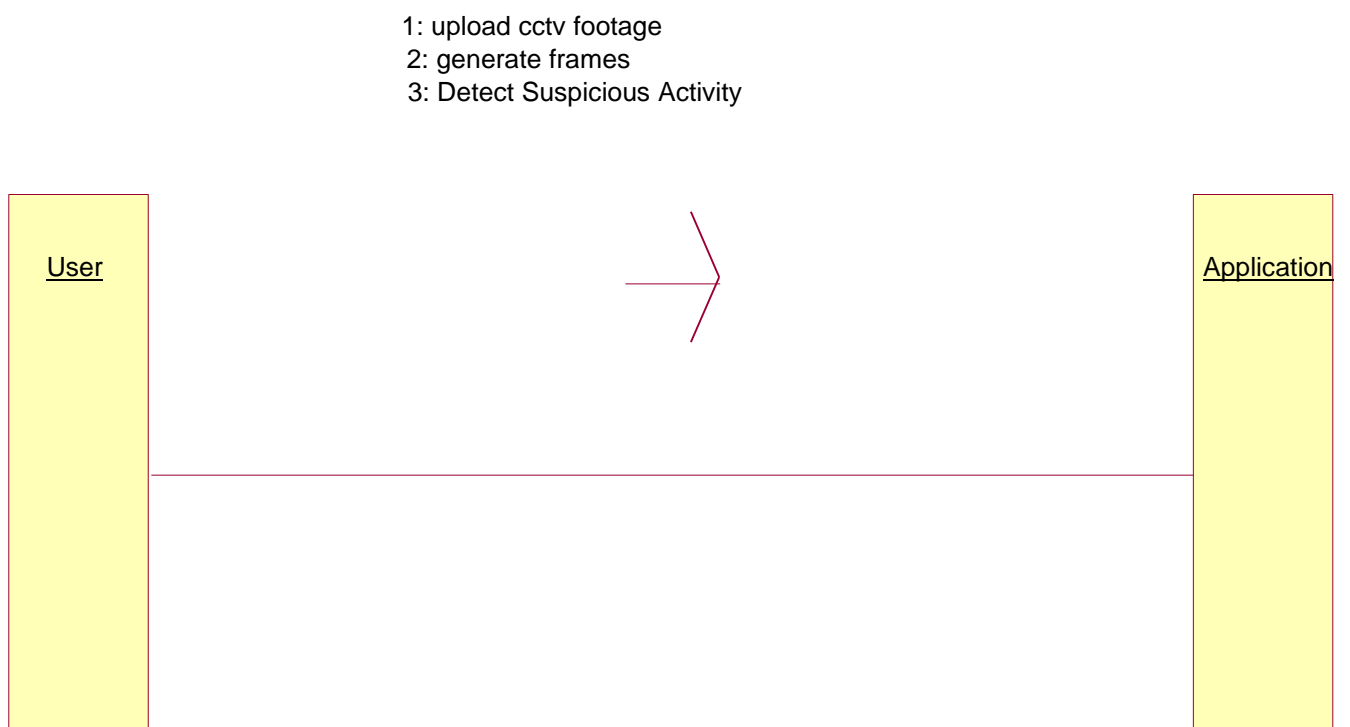


**Fig.3.3.3. Sequence**

### **Collaboration diagram:**

A collaboration diagram, also known as a communication diagram, is a type of UML diagram used in software engineering to illustrate how objects or components within a system interact and collaborate to achieve specific tasks or goals. Unlike sequence diagrams that focus on the chronological order of message exchanges, collaboration diagrams emphasize the structural aspects of interactions. In a collaboration diagram, objects are represented as rectangles or ellipses, each depicting an instance of a class in the system. These objects are interconnected with lines that represent communication channels or associations between them. The lines between objects indicate the flow of messages, requests, or responses exchanged during the collaboration process. These messages can be labeled to specify the type of communication, such as method invocations, data transfers, or dependencies.

The diagram also includes roles, which define the responsibilities or functions that objects perform within the collaboration. Roles provide context and clarity regarding the specific contributions of each object to the overall system functionality. Additionally, associations in the diagram depict the relationships between objects, showing how they are connected or related to each other. Multiplicity annotations indicate the cardinality or number of instances participating in a particular association, specifying the quantity of objects involved in the relationship. Collaboration diagrams are valuable tools for system designers, developers, and stakeholders as they provide a visual representation of system interactions and relationships. They aid in understanding the communication patterns, identifying dependencies or bottlenecks, and refining system designs for improved performance and efficiency. Moreover, collaboration diagrams facilitate effective communication and collaboration among team members by offering a common visual language to discuss system architectures, interactions, and design.



**Fig.3.3.4.Collaboration**

### 3.4. Stepwise Implementation and code:

#### **Stepwise Explanation:**

- **Define Project Objectives and Scope:**-Objective: Detect anomalous network activities indicative of potential cyber threats, such as intrusion attempts or unusual data transfers.
- **Scope:** Monitor network traffic logs from routers, firewalls, and intrusion detection systems (IDS) to identify anomalies.
- **Data Collection and Preparation:**-Collect network traffic logs containing information such as source IP, destination IP, port numbers, protocol, and timestamp.Clean the data by removing irrelevant or duplicate entries, handling missing values, and ensuring data consistency.
- **Exploratory Data Analysis (EDA):** Analyse the data distribution, traffic patterns, and typical behaviors using descriptive statistics and visualizations.
- **Identify potential features** such as traffic volume, packet rates, and communication patterns that may indicate anomalies.
- **Feature Engineering:** Extract features such as average packet size, number of connections per IP, traffic spikes, and protocol anomalies.
- **Normalize and scale the features** to ensure uniformity and effectiveness in anomaly detection algorithms.
- **Select Anomaly Detection Algorithms:**Choose anomaly detection algorithms suitable for network traffic data, such as Isolation Forest, Local Outlier Factor (LOF), or clustering methods like k-means.

- **Train Anomaly Detection Models:** Split the data into training and testing sets, ensuring that anomalous and normal traffic samples are represented in both sets.
- Train the selected anomaly detection models using the training data, tuning hyper parameters for optimal performance. Evaluate model performance using metrics like accuracy, precision, recall, and F1 score on the testing data.
- **Integration and Deployment:** Integrate the trained anomaly detection models into the network monitoring infrastructure. Set up automated pipelines to ingest real-time network traffic data and feed it into the anomaly detection system.
- Deploy the system to continuously monitor and analyze incoming network traffic for anomalies.
- **Monitoring and Evaluation:** Monitor the system's performance in real-time, generating alerts or notifications for detected anomalies. Evaluate the effectiveness of the anomaly detection system by analyzing false positives, false negatives, and overall detection rates.
- Fine-tune the models and algorithms based on feedback and performance metrics to reduce false alarms and improve detection accuracy.
- **Maintenance and Updates:** Regularly maintain and update the anomaly detection system to adapt to new cyber threats, evolving network patterns, and changes in data characteristics.
- Incorporate feedback from security analysts, update threat intelligence feeds, and enhance anomaly detection rules to improve the system's effectiveness over time.

### **Stepwise implementation and testing:**

#### **SYSTEM TEST:**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

#### **TYPES OF TESTS:**

##### **Unit testing:**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

##### **Functional test:**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is based on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.



Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### **System Test:**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### **White Box Testing:**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

### **Black Box Testing:**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Test strategy and approach:**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

**Features to be tested**

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

**Integration Testing:**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing:**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Code:**

```
#pip install ImageAi==2.0.3
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter from
tkinter import filedialog
from imutils import paths
import matplotlib.pyplot as plt
import datetime
from tkinter.filedialog
import askopenfilename
import cv2
import shutil
import os
from imageai.Prediction.Custom import CustomImagePrediction
import os
import winsound

main = tkinter.Tk() main.title("Suspicious Activity Detection") main.geometry("1200x1200")

global filename

execution_path = os.getcwd()
prediction = CustomImagePrediction()
prediction.setModelTypeAsResNet()
prediction.setModelPath("model.h5")
prediction.setJsonPath("model_class.json")
prediction.loadModel(num_objects=2)

def beep():
    frequency = 2500
# Set Frequency To 2500 Hertz    duration = 1000
# Set Duration To 1000 ms == 1 second
```

```

winsound.Beep(frequency, duration)

def upload():
global filename
filename = askopenfilename(initialdir = "videos") pathlabel.config(text=filename)

def generateFrame(): global filename
text.delete('1.0', END)
if not os.path.exists('frames'):
    os.mkdir('frames') else:
    shutil.rmtree('frames') os.mkdir('frames') vidObj = cv2.VideoCapture(filename)
count = 0 success = 1
while success:
    success, image = vidObj.read()
if count < 500:
    cv2.imwrite("frames/frame%d.jpg" % count, image)
text.insert(END, "frames/frame."+str(count)+" saved\n")
print("frames/frame."+str(count)+" saved")
    #pathlabel.config(text="frames/frame."+str(count)+" saved")
else:
    break count += 1
pathlabel.config(text="Frame generation process completed. All frames saved inside frame folder")
def detectActivity():
    imagePaths = sorted(list(paths.list_images("frames")))
count = 0 option = 0; text1.delete('1.0', END)
for imagePath in imagePaths:
    predictions, probabilities = prediction.predictImage(imagePath, result_count=1)
for eachPrediction, eachProbability in zip(predictions, probabilities):
if float(eachProbability) > 80:
    count = count + 1; if float(eachProbaity) < 80:

    "+" is predicted as "+eachPrediction+" with probability : " +str(eachProbability))
print(imagePathtext1.insert(END, imagePath+" is predicted as "+eachPrediction+" with

```

```
probability : " +str(eachProbability)+"\n\n")
    count = 0;
    print(imagePath+" processed")
    if option == 0:
        text1.insert(END,"No suspicious activity found in given footage")

font = ('times', 20, 'bold') title = Label(main, text='Suspicious Activity Detection From CCTV Footage')
title.config(bg='brown', fg='white')
title.config(font=font)
title.config(height=3, width=80)
title.place(x=5,y=5)

font1 = ('times', 14, 'bold')
upload = Button(main, text="Upload CCTV Footage", command=upload)
upload.place(x=50,y=100)
upload.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='brown', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=300,y=100)

depthbutton = Button(main, text="Generate Frames", command=generateFrame)
depthbutton.place(x=50,y=150)
depthbutton.config(font=font1)
userinterest = Button(main, text="Detect Suspicious Activity Frame", command=detectActivity)
userinterest.place(x=280,y=150)
userinterest.config(font=font1)
font1 = ('times', 12, 'bold')
text=Text(main,height=25,width=50)
scroll=Scrollbar(text)
```

```
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=200)
text.config(font=font1)
```

```
text1=Text(main,height=25,width=50) scroll=Scrollbar(text1)
text1.configure(yscrollcommand=scroll.set)
text1.place(x=550,y=200)
text1.config(font=font1)
main.config(bg='green') main.mainloop()
```

```
import tensorflow as tf
from tensorflow.keras import layers, models
```

```
# Define the CNN model architecture
```

```
model = models.Sequential()
```

```
# Convolutional layers
```

```
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

```
model.add(layers.Flatten())
```

```
model.add(layers.Dense(64, activation='relu'))
```

```
model.add(layers.Dense(10, activation='softmax')) # Assuming 10 classes for classification
```

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()

mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)

model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))

test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)

import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

```
# Define the RNN model architecture for image classification
model = models.Sequential()
model.add(layers.LSTM(128, input_shape=(None, 256), return_sequences=True))
model.add(layers.TimeDistributed(layers.Dense(10, activation='softmax')))

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

image_data = []
image_labels = []

image_data_array = np.array(image_data)
image_labels_array = np.array(image_labels)

image_data_resaped = image_data_array.reshape(image_data_array.shape[0], 1,
image_data_array.shape[1])

model.fit(image_data_resaped, image_labels_array, epochs=10, batch_size=32)

loss, accuracy = model.evaluate(image_data_resaped, image_labels_array)
print("Test loss:", loss)
print("Test accuracy:", accuracy)

import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models, optimizers
def build_generator(latent_dim, image_shape):
    generator = models.Sequential()
```



```
generator.add(layers.Dense(128 * 7 * 7, input_dim=latent_dim))
generator.add(layers.Reshape((7, 7, 128)))
generator.add(layers.Conv2DTranspose(128, kernel_size=4, strides=2, padding='same',
activation='relu'))
generator.add(layers.Conv2DTranspose(64, kernel_size=4, strides=2, padding='same',
activation='relu'))
generator.add(layers.Conv2D(1, kernel_size=7, padding='same', activation='sigmoid'))
generator.add(layers.Reshape(image_shape))
return generator
```

```
def build_discriminator(image_shape):
    discriminator = models.Sequential()
    discriminator.add(layers.Conv2D(64, kernel_size=3, strides=2, input_shape=image_shape,
padding='same', activation='relu'))
    discriminator.add(layers.Conv2D(128, kernel_size=3, strides=2, padding='same',
activation='relu'))
    discriminator.add(layers.Flatten())
    discriminator.add(layers.Dense(1, activation='sigmoid'))
    return discriminator
```

# Define the GAN model

```
def build_gan(generator, discriminator):
    discriminator.trainable = False
    gan = models.Sequential([generator, discriminator])
    gan.compile(loss='binary_crossentropy', optimizer=optimizers.Adam(lr=0.0002,
beta_1=0.5))
    return gan
```

latent\_dim = 100

image\_shape = (28, 28, 1)

```
generator = build_generator(latent_dim, image_shape)
discriminator = build_discriminator(image_shape)
gan = build_gan(generator, discriminator)

from tensorflow.keras.datasets import mnist
(train_images, _), (_, _) = mnist.load_data()
train_images = train_images.reshape(train_images.shape[0], 28, 28, 1).astype('float32')
train_images = (train_images - 127.5) / 127.5 # Normalize the images to [-1, 1]
def generate_fake_images(generator, latent_dim, n_samples):
    noise = np.random.normal(0, 1, (n_samples, latent_dim))
    fake_images = generator.predict(noise)
    return fake_images

batch_size = 128
epochs = 10000
save_interval = 1000

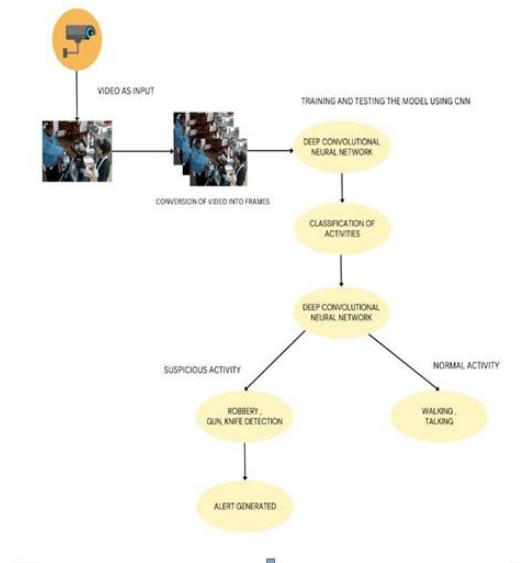
for epoch in range(epochs):
    # Generate fake images
    fake_images = generate_fake_images(generator, latent_dim, batch_size)
    discriminator_loss_real = discriminator.train_on_batch(train_images[np.random.randint(0,
train_images.shape[0], batch_size)], np.ones((batch_size, 1)))
    discriminator_loss_fake = discriminator.train_on_batch(fake_images, np.zeros((batch_size,
1)))
    discriminator_loss = 0.5 * np.add(discriminator_loss_real, discriminator_loss_fake)

    noise = np.random.normal(0, 1, (batch_size, latent_dim))
    generator_loss = gan.train_on_batch(noise, np.ones((batch_size, 1)))
```

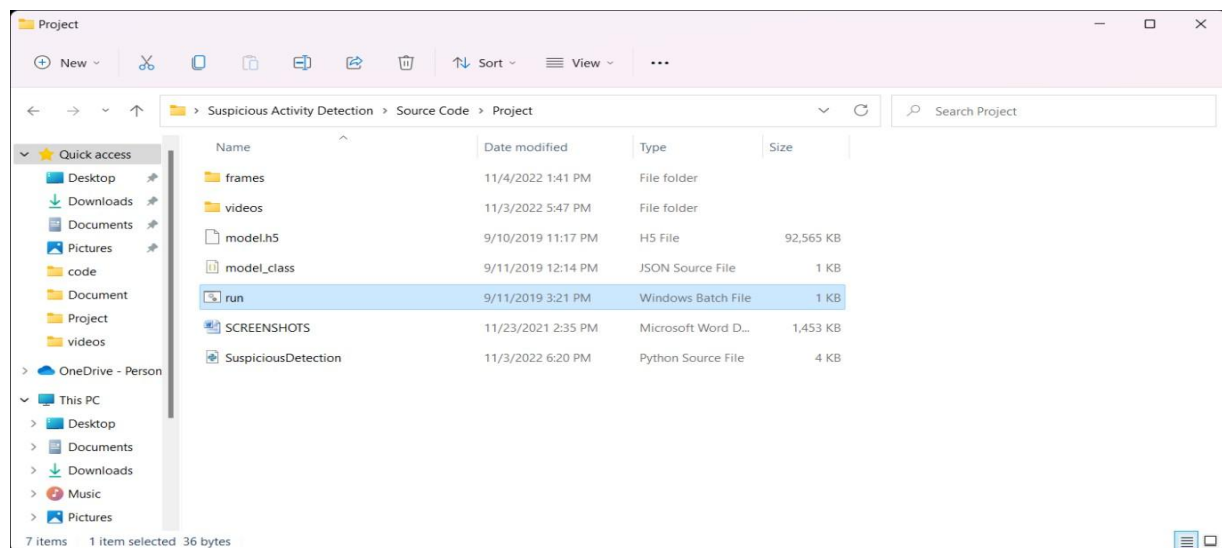
# **CHAPTER 4**

## **RESULTS AND DISCUSSION**

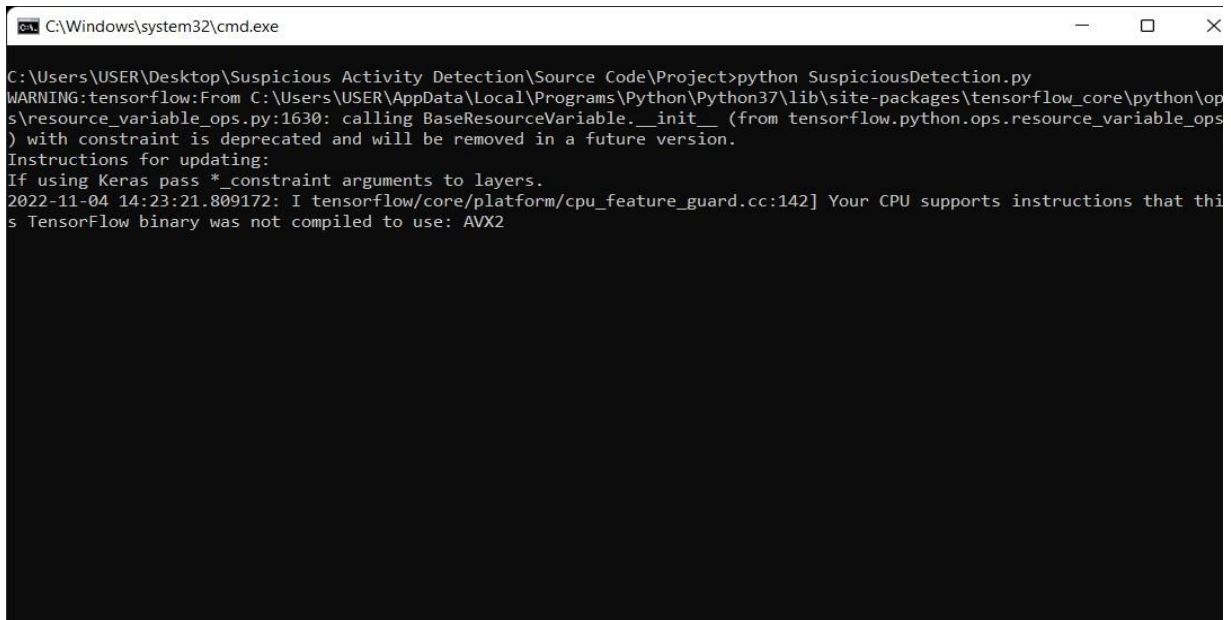
## Results:



**Fig.4.1: Prediction Model**



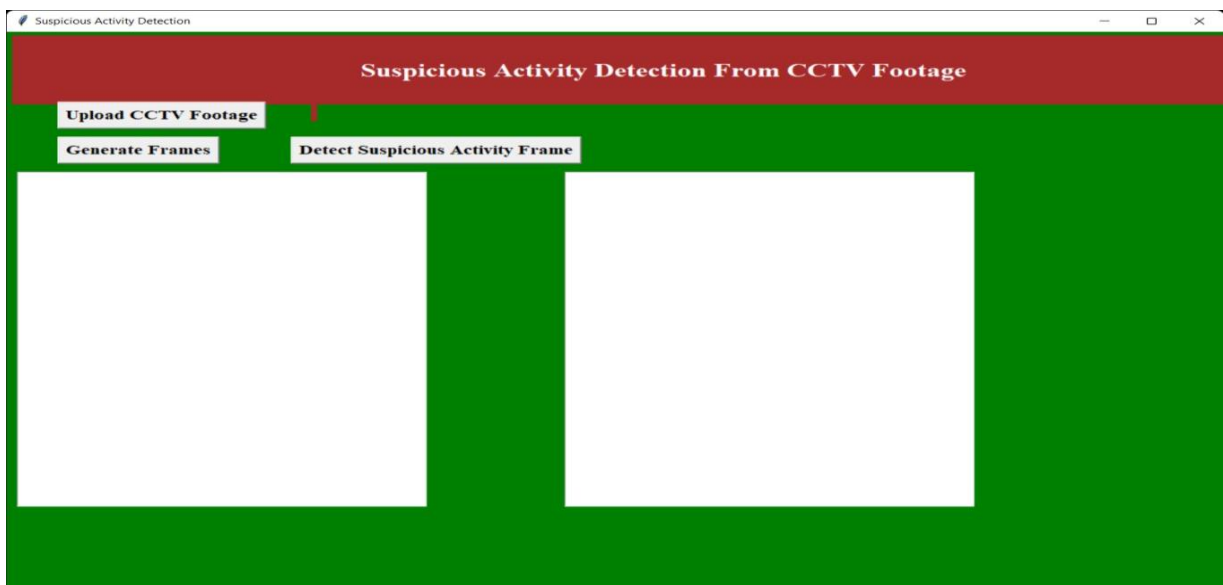
**Fig.4.2: Run**



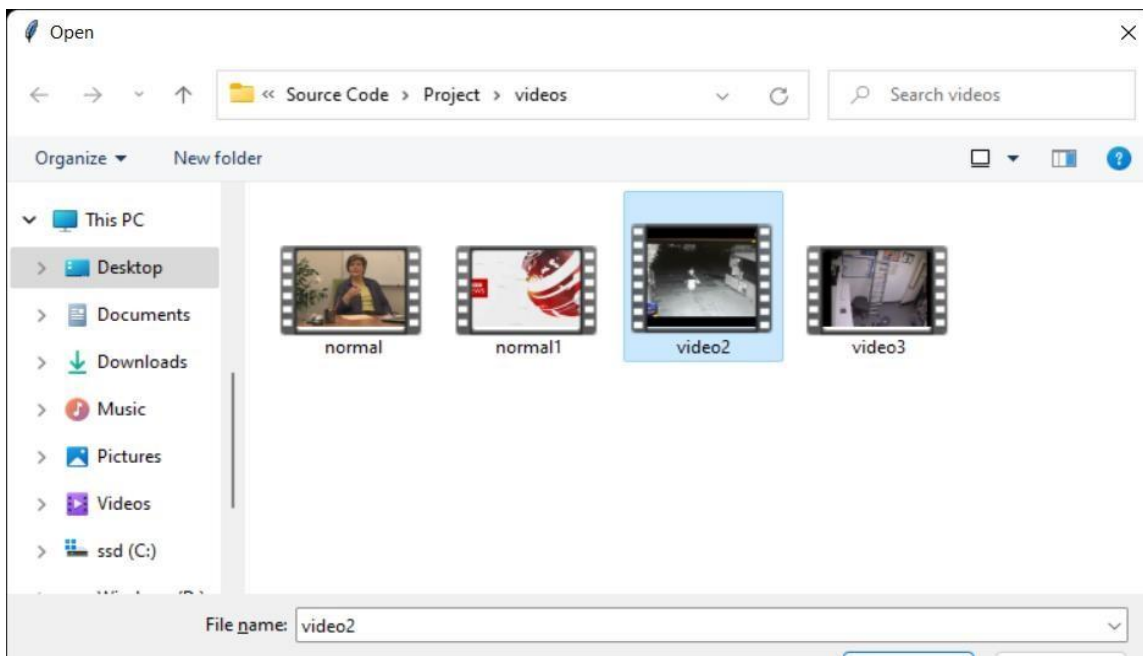
```
C:\Windows\system32\cmd.exe

C:\Users\USER\Desktop\Suspicious Activity Detection\Source Code\Project>python SuspiciousDetection.py
WARNING:tensorflow:From C:\Users\USER\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow_core\python\ops\resource_variable_ops.py:1630: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
2022-11-04 14:23:21.809172: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
```

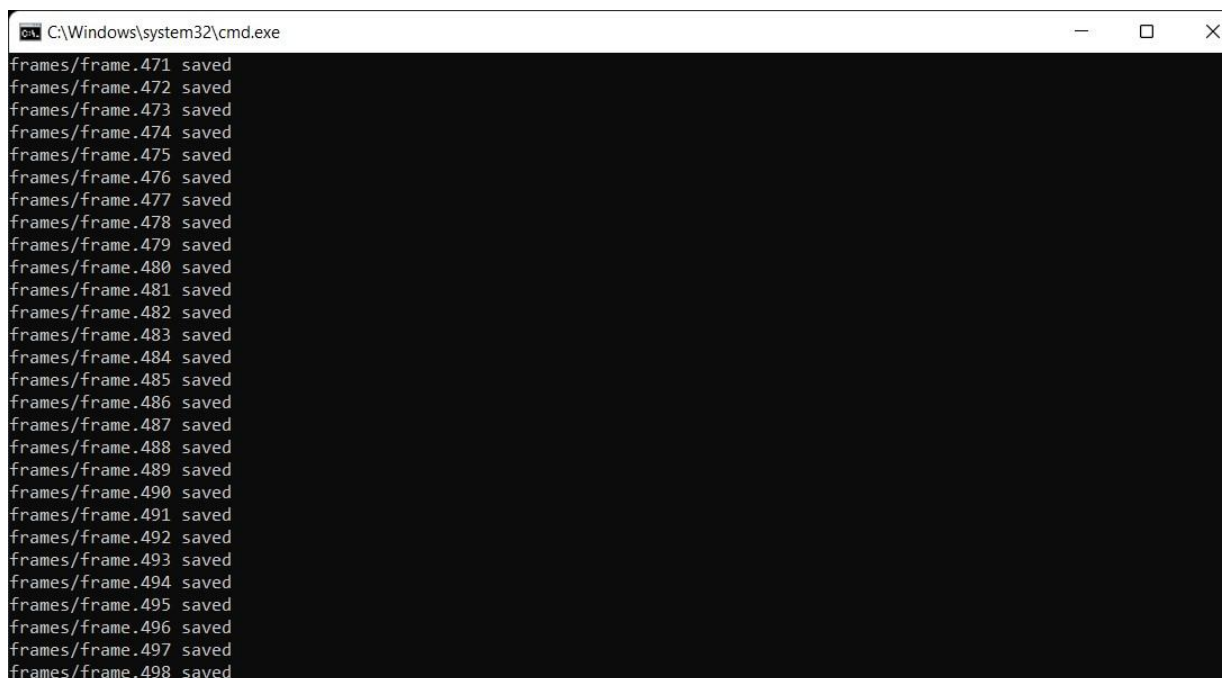
**Fig.4.3: CMD run**



**Fig.4.4: Suspicious activity**



**Fig.4.5. Footage selection**



**Fig.4.6. Frame analysis**

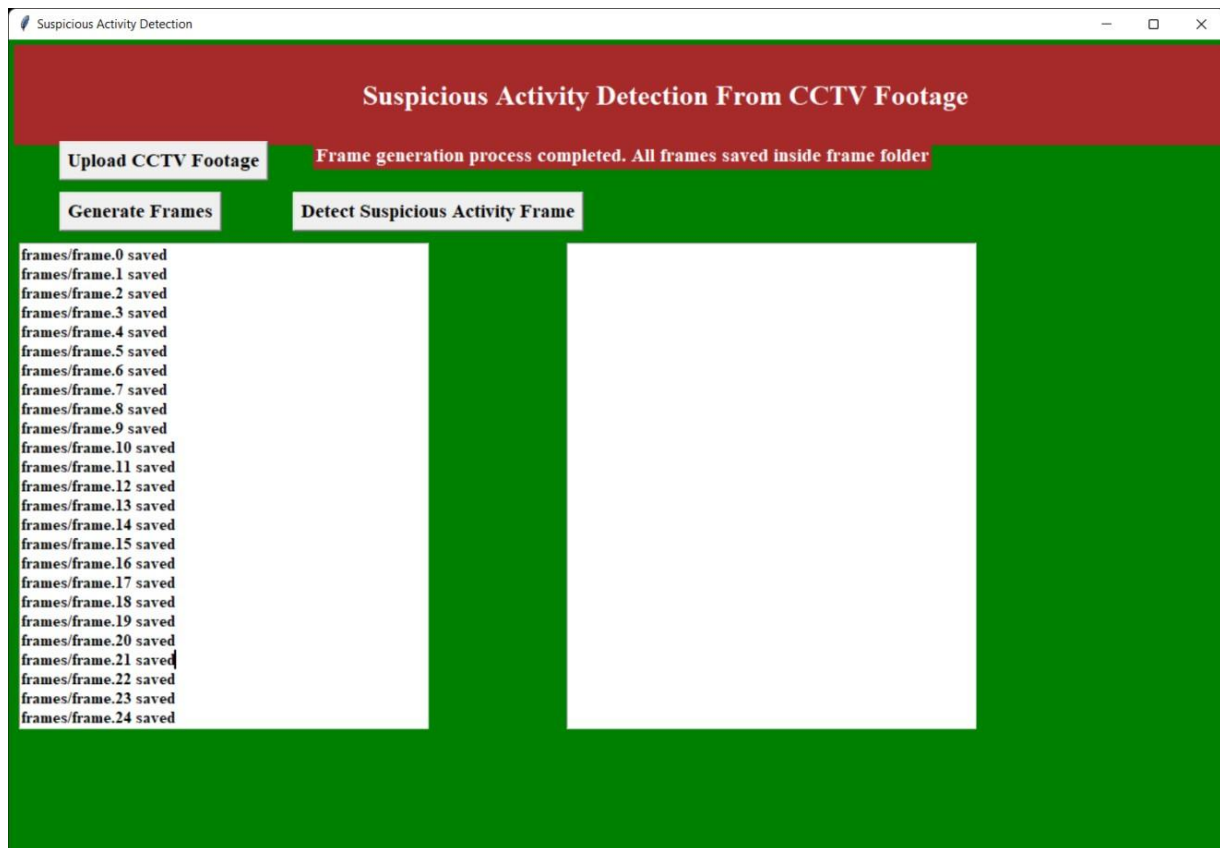


Fig4.7. Generation of frames

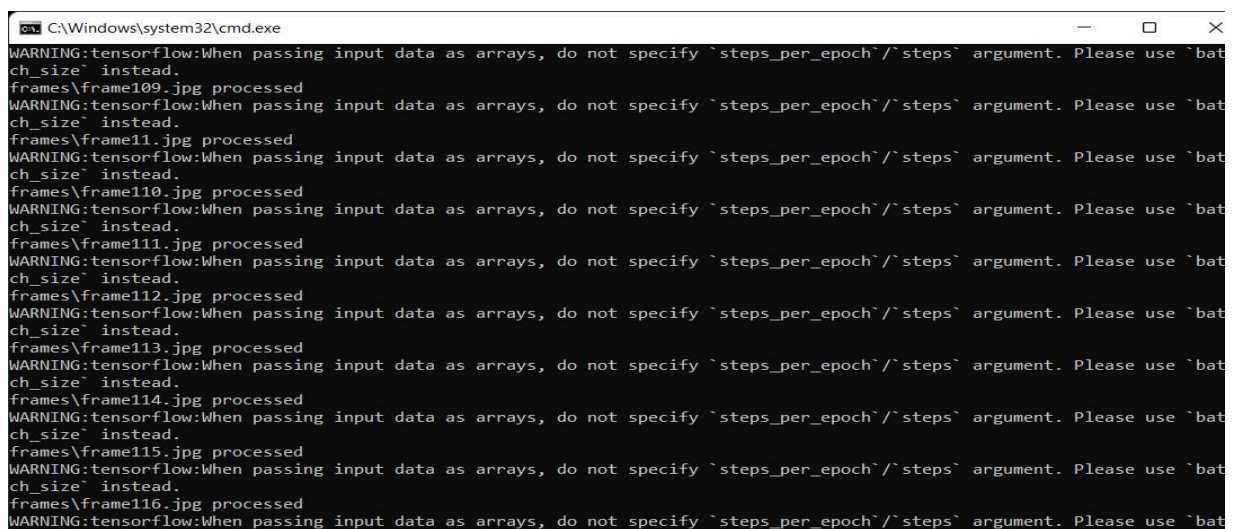


Fig4.8. Anomaly detection

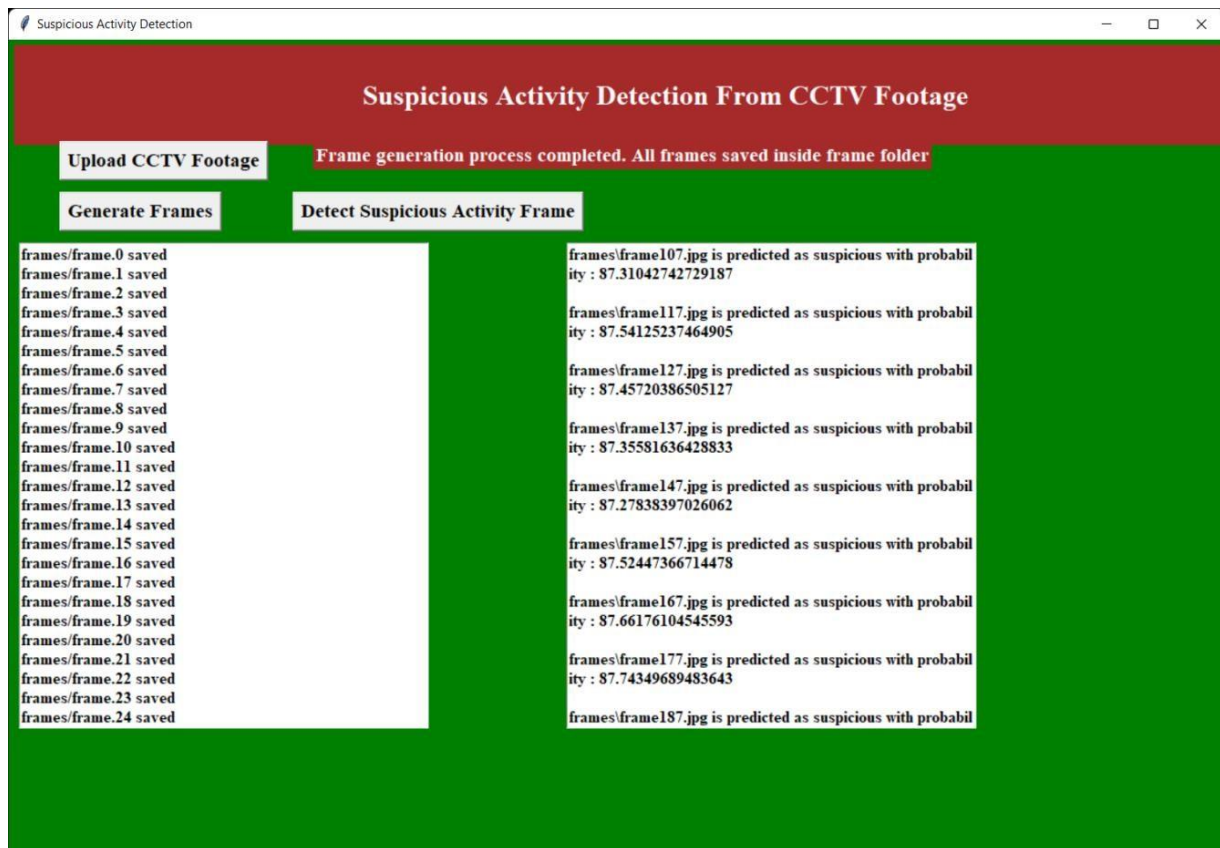


Fig.4.9. Frames detection

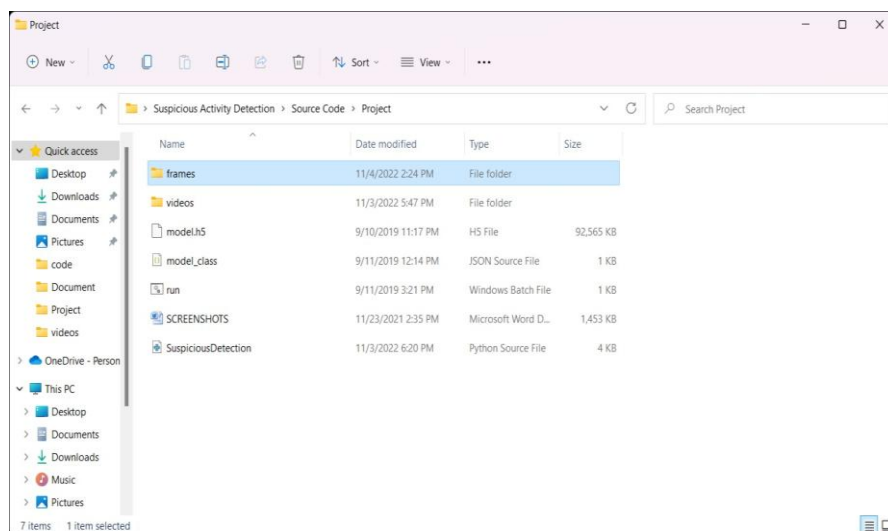
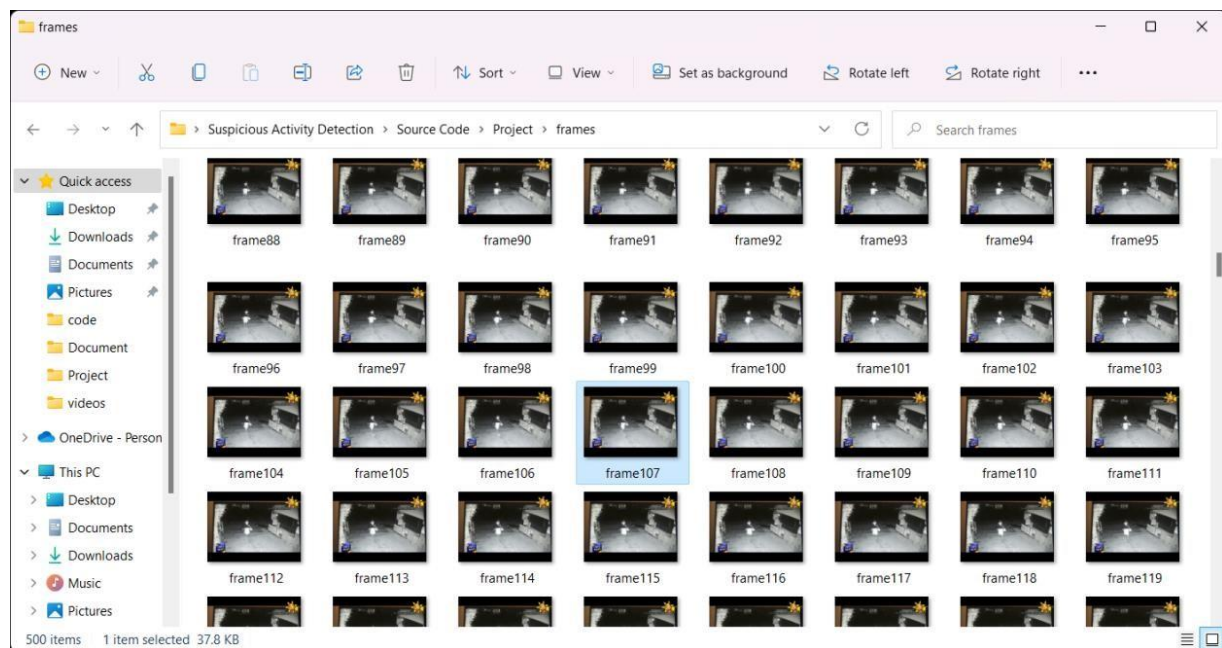
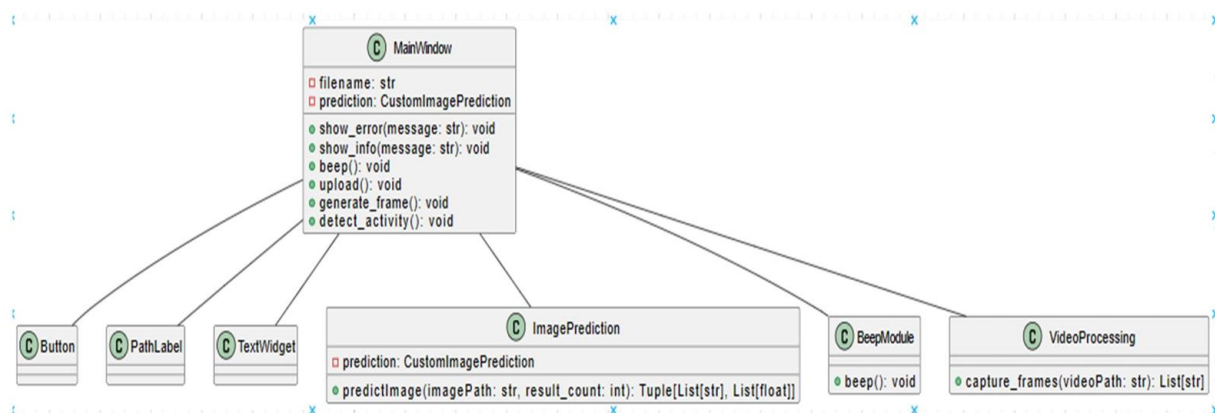


Fig.4.10. Suspicious frames folder





**Fig.4.11. Suspicious Frames**



**Fig.4.12. Image classification**

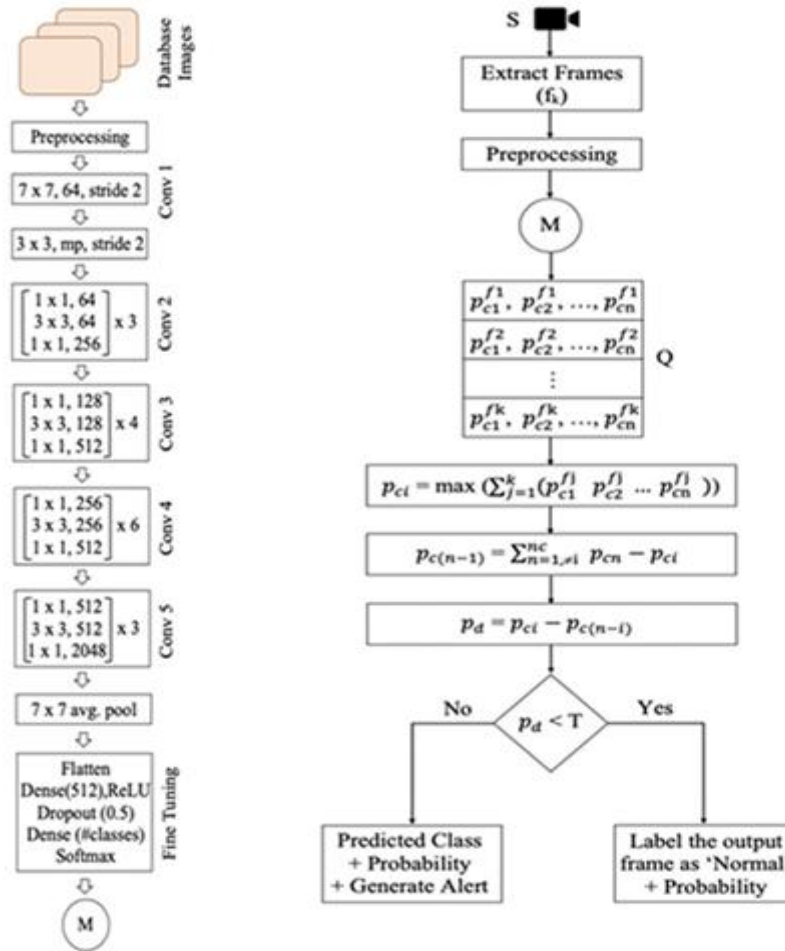


Fig.4.13. Image recognition

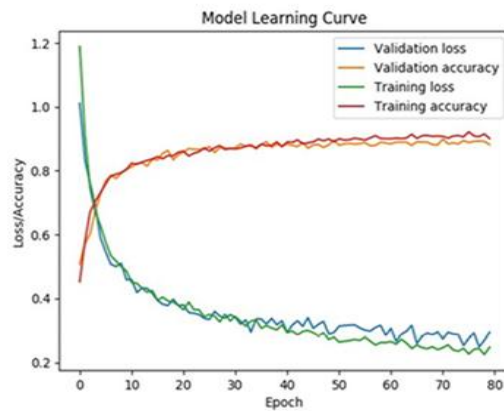


Fig.4.14. Model learning curve

**Table 4.1      Performance Metrics**

<b>Parameter</b>	<b>Performance Metrics</b>
<b>Accuracy</b>	93.63
<b>Precision</b>	89.56
<b>Recall</b>	67.23
<b>F-measure</b>	91.69
<b>Specificity</b>	97.33

# CHAPTER 5

## CONCLUSION

## **CHAPTER 5**

### **CONCLUSION**

The proposed model aims to revolutionize the use of CCTV footages by shifting the focus from post-incident investigation to real-time crime prevention. By continuously tracking and analyzing real-time CCTV footages, the model can identify suspicious activities and provide timely alerts to the respective authorities, enabling proactive intervention to prevent crimes before they occur. This proactive approach not only enhances public safety but also minimizes the impact of criminal incidents on individuals and communities. While the current implementation is limited to academic settings, the model's potential extends to various public and private environments where surveillance is crucial for security. By training the model on specific suspicious activities relevant to different scenarios, such as theft, vandalism, or violence, it can effectively detect and predict a wide range of suspicious behaviors.

Furthermore, the model can be enhanced by incorporating advanced AI techniques to not only detect suspicious activities but also identify suspicious individuals involved. This capability adds an extra layer of security by targeting potential perpetrators and enabling law enforcement to take targeted actions to prevent criminal acts.

Overall, the proposed model represents a significant step forward in leveraging CCTV technology for proactive crime prevention, enhancing public safety, and creating more secure environments for individuals and communities.

### **Future Enhancement:**

- **Integration with IoT Devices:** Incorporating Internet of Things (IoT) devices such as smart sensors, cameras, and actuators can enhance the capabilities of the surveillance system. IoT devices can provide additional data points and real-time information, improving the accuracy and responsiveness of the AI models.
- **Enhanced Real-time Analysis:** Implementing more advanced AI algorithms, such as deep learning models like recurrent neural networks (RNNs) or transformers, can improve the system's ability to analyze and interpret complex patterns in CCTV footages. This can lead to more accurate detection of suspicious behaviors and proactive prevention measures.
- **Behavioral Biometrics:** Integrating behavioral biometrics, such as gait analysis, facial recognition, and voice recognition, into the surveillance system can help in identifying and tracking suspicious individuals across different locations. This can be particularly useful in high-security environments or sensitive areas.
- **Predictive Analytics:** Leveraging predictive analytics techniques, such as time-series forecasting and anomaly prediction, can enable the system to anticipate potential criminal activities based on historical data patterns. This proactive approach can significantly reduce response times and prevent incidents before they escalate.
- **Geospatial Analysis:** Incorporating geospatial analysis capabilities can help in identifying hotspots of criminal activities or areas with higher risk levels. By analyzing spatial data in conjunction with CCTV footages, the system can prioritize surveillance efforts and allocate resources more effectively.

# REFERENCES

## REFERENCES

1. "Anomaly Detection Principles and Algorithms" by Mehmed Kantardzic: This book provides a comprehensive overview of anomaly detection techniques, algorithms, and principles, offering practical insights into their implementation and application across various domains.
2. "Practical Machine Learning for Anomaly Detection" by Ted Dunning and Ellen Friedman: This book offers practical guidance on using machine learning techniques for anomaly detection, covering topics such as data preprocessing, feature engineering, model selection, and evaluation.
3. "Anomaly Detection: A Survey" by Varun Chandola, Arindam Banerjee, and Vipin Kumar: This seminal survey paper provides a comprehensive overview of anomaly detection methods, including statistical, machine learning, and ensemble techniques, along with their applications and evaluation metrics.
4. "Isolation Forest" by Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou: This paper introduces the Isolation Forest algorithm, a popular technique for anomaly detection based on the isolation of anomalies in random subspaces, which offers advantages in terms of scalability and efficiency.
5. "One-Class SVM for Anomaly Detection" by Bernhard Schölkopf, John Platt, and John Shawe-Taylor: This paper presents the One-Class Support Vector Machine (SVM) algorithm for anomaly detection, which learns a decision boundary around normal data points to identify anomalies in high-dimensional spaces.
6. "Anomaly Detection in Time Series Data: A Survey" by Chetan Gupta and Gaurav Trivedi: This survey paper focuses on anomaly detection techniques specifically tailored for time series data, covering both traditional statistical methods and modern machine learning approaches.
7. "Deep Learning for Anomaly Detection: A Review" by Sridhar Palla and Raman Bhardwaj: This review paper provides an in-depth analysis of deep learning techniques



## **GIT HUB LINK**

<https://github.com/aravelliabhinav/Anomaly-Xpert-A-deep-learning-approach-to-anomaly-detection.git>

## PUBLICATION CERTIFICATES



ISSN No. : 2321-9653

# IJRASET

**International Journal for Research in Applied  
Science & Engineering Technology**

IJRASET is indexed with Crossref for DOI-DOI : 10.22214  
Website : [www.ijraset.com](http://www.ijraset.com), E-mail : [ijraset@gmail.com](mailto:ijraset@gmail.com)

*Certificate*

*It is here by certified that the paper ID : IJRASET59081, entitled*  
***ANOMALY XPERT: A Deep Learning Approach to Anomaly Detection***  
*by*  
***B. Gayathri***

*after review is found suitable and has been published in*  
*Volume 12, Issue III, March 2024*  
*in*

*International Journal for Research in Applied Science &  
Engineering Technology*  
*(International Peer Reviewed and Refereed Journal)*  
*Good luck for your future endeavors*

*By*   
Editor in Chief, IJRASET



ISRA Journal Impact  
Factor: 7.429



45.98  
INDEX COPERNICUS



THOMSON REUTERS  
Researcher ID: N-9681-2016



10.22214/IJRASET  
doi crossref



TOGETHER WE REACH THE GOAL  
SJIF 7.429









