**A PROJECT ON**

****

**DEEP LEARNING BASED SMART SECURITY SURVEILLANCE SYSTEM FOR SAFETY ENHANCEMENTS**

****

**Submitted in partial fulfillment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

**Submitted by:**

**Shristy Chaudhary** **2019498**

***Under the Guidance of***

***Mr. Priyank Pamdey***

**Asst. Professor**

**Project Team ID: MP24ML010**

****

**Department of Computer Science and Engineering Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand June-2025**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project progress report entitled **“DEEP LEARNING BASED SMART SECURITY SURVEILLANCE SYSTEM FOR SAFETY ENHANCEMENTS**

**”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering **(Artificial Intelligence and Machine Learning )**in the Department of Computer Science and Engineering of the Graphic Era (Deemed to be University), Dehradun shall be carried out by the undersigned under the supervision of **Mr. Priyank Pandey**, Professor, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Shristy Chaudhary 2019498

The above-mentioned students shall be working under the supervision of the undersigned on the **“Deep Learning Based Smart Security Surveillance System For Safety Enhancements”**

Signature

**Supervisor Head of the Department**

**Examination**

**Name of the Examiners: Signature with Date 1.**

**2.**

# **Abstract**

In numerous industrial and public space monitoring situations, manual video inspection is time-consuming, prone to mistakes, and cannot expand to accommodate expanding camera networks. Using a three-model deep-learning pipeline—3D-CNN, LSTM, and Mask R-CNN—integrated within a MERN-stack application, this research offers an intelligent, web-based surveillance platform that automates detection, segmentation, and behavior prediction. To extract integrated spatial and temporal characteristics over consecutive frames, a 3D CNN initially processes incoming video streams. An LSTM network uses such features as input and uses temporal dynamics to identify suspicious or unusual activity (such as loitering, crowd formations, or safety-critical occurrences). Accurate localization is made possible by Mask R-CNN's concurrent instance segmentation on every frame, which clearly defines items of interest (people, cars, and equipment).All metadata, including timestamps, anomaly scores, and bounding masks, is kept in MongoDB for audit and reporting purposes, and detected events and segmented objects produce real-time warnings that are transmitted to the React frontend.  
  
React-built and responsively designed, the online interface shows downloadable analytics reports, event timelines, and live camera dashboards. WebSocket channels broadcast detection results with low latency, while a Node.js/Express backend coordinates model inference through API endpoints. Zones of interest, anomaly threshold levels, and notification methods (email, SMS, or in-app alerts) can all be customized by users. This system provides a comprehensive solution for automated, real-time surveillance by merging the spatiotemporal modeling of 3D-CNN, the sequence learning of LSTM, and the exact segmentation of Mask R-CNN within a scalable MERN architecture. This significantly reduces manual monitoring work and improves detection accuracy,and providing actionable insights through an intuitive web portal.

**CERTIFICATE**

On the basis of the declaration submitted by Shristy Chaudhary, students of B.Tech AI & ML, I hereby certify that the project titled **DEEP LEARNING BASED SMART SECURITY SURVEILLANCE SYSTEM FOR SAFETY ENHANCEMENTS,** which is submitted to the Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun, in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is an original contribution with existing knowledge and a faithful record of work performed under my guidance and supervision.

To the best of my knowledge, this work has not been submitted in part or in full for any Degree or Diploma elsewhere.

Dehradun

Date:

**Mr. Priyank Pandey**

Asst.Professor

Department of Computer Science and Engineering

Graphic Era (Deemed to be University),

Dehradun, Uttarakhand, India.

# **Acknowledgement**

Any achievement, be in scholastic or otherwise does not depend solely on the individual effort but on the guidance, encouragement and co-operation of intellectuals, elders and friends. A number of personalities in their own capacity have helped me in carrying out this project work.

Our sincere thanks to project guide **Mr. Priyank Pandey, Asst. Professor,** Department of Computer Science and Engineering, Graphic Era (Deemed to be University), for his valuable guidance and support throughout the course of project work and for being a constant source of inspiration.

We extend our thanks to **Dr. Deepak Gaur**, Project coordinator, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), for his valuable suggestions throughout all the phases of the Project Work.

We are extremely grateful to **Prof. (Dr.) D. P. Singh**, HOD of the Computer Science and Engineering Department, Graphic Era (Deemed to be University), for his moral support and encouragement.

We thank the **management of Graphic Era (Deemed to be University)** for the support throughout the course of our Bachelor’s Degree and for all the facilities they have provided.

Last, but certainly not least we thank all teaching and non-teaching staff of Graphic Era (Deemed to be University) for guiding us on the right path. Most importantly we wish to thank our parents for their support and encouragement.

Shristy Chaudhary 2019498

# **Table of Contents**

**Contents Page**

**No.**

[Abstract 3](#_TOC_250007)

[Acknowledgement 4](#_TOC_250006)

[Table of Contents 5](#_TOC_250005)

[List of Tables 6](#_TOC_250004)

List of Figures 6

Chapter 1 Introduction 7-10

* 1. [Project Introduction 7](#_TOC_250003)-8
  2. [Problem Statement](#_TOC_250002) 9
  3. [Objectives](#_TOC_250001) 9-10

Chapter 2 Literature Survey/ Background 11-12

Chapter 3 Software Design 13-18

Chapter 4 Requirements and Methodology 19-25

* 1. [Requirements](#_TOC_250000)
     1. Software Requirements

4.1.3 Hardware Requirements

Chapter 5 Coding /Code Templates 26-28

Chapter 6 Testing 29

Chapter 7 Results and Discussion 30-31

Chapter 8 Conclusion and Future Work 32

**References**

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| 1. | Abnormal Activity Detection System | 8 |
| 2. | Flowchart | 22 |
| 3. | Detection System | 30 |

### **List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Title** | **Page No.** |
| 1. | Technology stack | 14 |
| 2. | Software Requirements | 19 |
| 3. | Hardware Requirements | 19-20 |
| 4. | Test Cases | 29 |

# **Chapter 1**

# **Introduction**

In the following sections, a brief introduction and the problem statement for the work has been included.

### **Project Introduction**

In today's globalized society, security and surveillance have emerged as major issues in public areas, smart cities, and industrial sites alike. Conventional monitoring techniques, which depend on manual video analysis or basic motion detectors, are time-consuming, unresponsive, and unsuited to grow with growing camera networks. These antiquated methods find it difficult to recognize complicated behaviors, distinguish between legitimate and questionable activity, and deliver timely, actionable insights. In order to address these shortcomings, this research presents an intelligent, web-based surveillance platform that combines state-of-the-art deep learning models with a contemporary MERN-stack architecture, providing behavior predicting, automatic detection, and accurate segmentation in a single solution.

In today's globalized society, security and surveillance have emerged as major issues in public areas, smart cities, and industrial sites alike. Conventional monitoring techniques, which depend on manual video analysis or basic motion detectors, are time-consuming, unresponsive, and unsuited to grow with growing camera networks. These antiquated methods find it difficult to recognize complicated behaviors, distinguish between legitimate and questionable activity, and deliver timely, actionable insights. In order to address these shortcomings, this research presents an intelligent, web-based surveillance platform that combines state-of-the-art deep learning models with a contemporary MERN-stack architecture, providing behavior predicting, automatic detection, and accurate segmentation in a single solution. Daily and monthly analytics reports may be downloaded thanks to a Node.js/Express backend that coordinates inference calls and stores metadata in MongoDB. This platform increases detection accuracy, decreases manual monitoring effort, and provides stakeholders with real-time intelligence and thorough audit trails by combining the spatiotemporal modeling of 3D-CNN, the sequence learning of LSTM, and the segmentation of Mask R-CNN within a scalable MERN ecosystem.

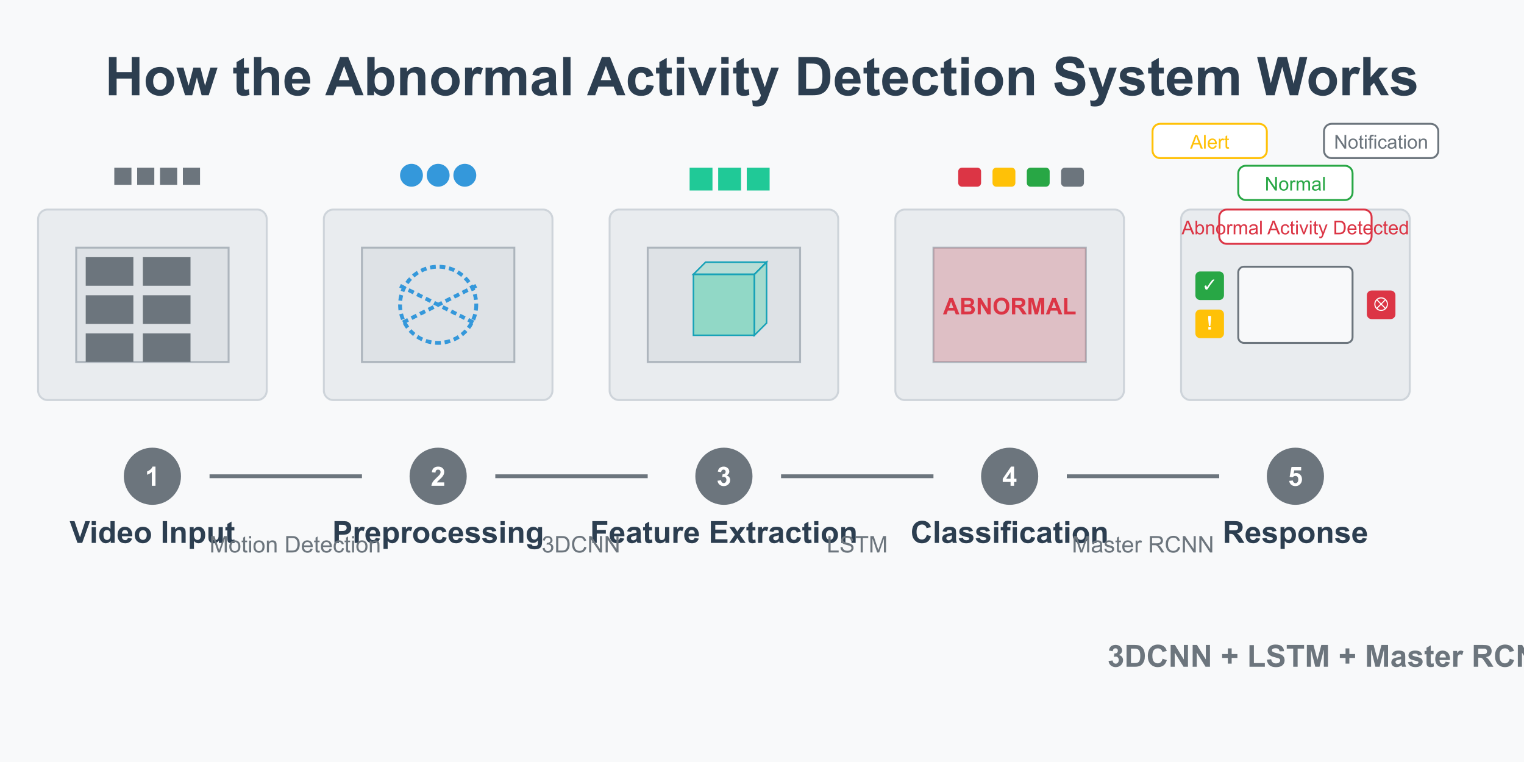


Figure-1

Traditional surveillance systems that depend on manual monitoring are ineffective and frequently fail to identify dangers in a timely manner in today's security-sensitive society. In order to identify, categorize, and react to anomalous activity in real time, this project introduces an AI-powered online surveillance system that was constructed utilizing a MERN tech stack and coupled with deep learning models—3D-CNN, LSTM, and Mask R-CNN. In order to identify suspicious or aggressive behavior, the system analyzes live video feeds using 3D-CNN to extract spatiotemporal features across frames and LSTM to analyze sequential data. In order to precisely locate and monitor individuals or objects involved in the activity, Mask R-CNN carries out instance segmentation.

When the system detects aberrant behaviors like assault, vandalism, arrest, shooting, arson, explosion, shoplifting, robbery, theft, burglary, abuse, fighting, or similar incidents, it instantly notifies administrators or security staff in real time. These notifications can be examined in a web dashboard and provide with event details (activity type, timestamp, and location). Administrators can receive comprehensive daily and monthly incident reports for examination, and events are also recorded in a timestamped CSV report.  
  
By enabling real-time alerting, automated behavior analysis, and structured reporting, this system improves public safety, drastically increases situational awareness, and lessens the need for ongoing manual monitoring. It can be implemented in settings such as public areas, educational institutions, transit hubs, and smart cities because it is scalable.

* 1. **Problem Statement**

In addition to being time-consuming and vulnerable to human error, manual video monitoring systems are not very good at stopping security breaches in real time. Traditional systems lack the ability to automatically identify or react to suspicious activity, and security people frequently miss crucial moments because they are too tired or distracted. An intelligent, automated monitoring system that can identify violent or unusual activity and notify authorities instantly is becoming more and more necessary. This will provide a quicker response and lower the possibility of interventions being postponed.

**1.3 Objective**

* **To create a deep learning-based intelligent surveillance system** that can identify unusual or suspicious activity.
* **To apply sequence-based anomaly detection** and spatiotemporal analysis to live video feeds using 3D-CNN and LSTM models.
* **To use Mask R-CNN** to precisely segment and monitor the participants in the activity that has been observed.
* **To recognize and categorize important incidents**, including shootings, robberies, assaults, vandalism, arson, and traffic accidents.
* **To use in-app alerts** to promptly alert authorized users when anomalous activity is detected (future: SMS/email support).
* **To save all event metadata** in structured CSV files, such as the timestamp, the kind of activity, and the people that were detected.
* **To provide downloadable reports** for security reviews, audits, and long-term analysis available on a daily and monthly basis.
* **To improve detection accuracy**, lessen reliance on human monitoring, and facilitate proactive security response.
* **Future Enhancements:**

**Integrated Notification System**: Add multi-channel alerts via SMS, email, or mobile push for instant awareness.

**Camera-Level Deployment**: Optimize and deploy models directly on edge devices (CCTV/IoT cameras) for faster, low-latency detection and offline processing capabilities.

**Chapter 2**

# **Literature Survey**

In order to identify anomalous activity, several researchers have investigated and created automated surveillance systems using a variety of deep learning and computer vision approaches. With an 81.36% detection rate under MATLAB, Alhanaee et al. suggested a method that combines Discrete Cosine Transform (DCT) features with a Self-Organizing Map (SOM) classifier to identify odd motion patterns, like abrupt crowd dispersals or aggressive gestures. This approach shows low hardware requirements appropriate for real-time monitoring. Using real-world datasets of assaults and traffic accidents, Arsenovic et al. achieved 95.02% accuracy in their spatiotemporal feature extraction from video sequences using convolutional neural networks. Fu et al. developed an incident surveillance system that achieves 98.87% detection and classification of events like shoplifting or vandalism by combining a Center-Face style recognition algorithm with MTCNN for accurate region proposal. total dependability. Zulfiqar et al. achieved 98.76% efficacy in detecting violent behaviors by combining Viola-Jones face and activity detection with a CNN backbone that had already been trained for feature embedding.  
  
When Samuel John noticed gunfire or fighting, he used 3D-CNNs in conjunction with Fisher LSTM networks and GSM alerts to notify security guards. Jenif D'Souza greatly decreased latency in the classification of explosion or arson incidents by using histogram-of-oriented-gradients for motion descriptor creation. By introducing a CNN–LSTM pipeline for frame-by-frame analysis, Nandhini R. increased the speed and accuracy of robbery and assault detection. To identify proxy intruders or unwanted gatherings, Shreyak Sawhney's team created a dual-camera system using CNN and PCA-based anomaly scoring. Eigenbehavior modeling and weight-based fusion were used by E. Varadharajan et al. to link identified activities with event records from the past to improve situational awareness, Poornima S. voiced alerts for burglaries that were identified by integrating Microsoft's Speech API.  
  
While Kritika Shrivastava fused Haar-Cascade, LBPH, and LDA for robust categorization of fighting and theft, Omkar et al. demonstrated Raspberry Pi–driven inference utilizing MobileNet SSD and Local Binary Patterns (LBP) for low-cost edge detection of explosions and arson. Face and behavior recognition were combined in Arun Katara's Raspberry Pi solution using OpenCV to provide unified surveillance. Ajinkya Patil's integration of Viola-Jones and orientation-aware neural nets later overcame the constraints Jomon Joseph and K.P. Zacharia observed while using PCA and Eigenbehaviors to detect crowd anomalies under non-frontal views. Senthamil Selvi et al. matched real-time event embeddings for automated logging using a centralized database, and Administrators were able to handle incident data via a web interface and issue instant notifications thanks to P. N. Garad's client-server architecture. The range of solutions currently available for real-time detection and response to critical security events is demonstrated by Hussain et al.'s proposal for IoT-based sensor fusion (including RFID and CCTV) for comprehensive surveillance and Mayuri Kamble's management dashboard for public venues, which optimized operational workflows.

**Chapter 3**

# Software Design

### **System Architecture Overview**

The proposed system is designed as a modular, real-time surveillance platform capable of detecting and responding to abnormal activities across monitored areas. It integrates deep learning–based video analysis with a responsive web interface for administrators. The main modules include:

* **User & Admin Authentication Module**  
  Handles secure login and access control for different user roles (e.g., admins, operators), ensuring that sensitive surveillance data is accessible only to authorized personnel.
* **Live Video Streaming Module**  
  Streams real-time video feeds from connected cameras to both the backend ML engine and the frontend dashboard using WebSockets or RTSP integrations.
* **Abnormal Activity Detection Module (3D-CNN + LSTM)**  
  Analyzes spatiotemporal video features to detect critical incidents such as **Road Accidents, Assault, Vandalism, Shooting, Arson, Robbery, Fighting**, and more using 3D-CNN for feature extraction and LSTM for sequence classification.
* **Instance Segmentation Module (Mask R-CNN)**  
  Accurately identifies and segments people or objects involved in the abnormal event to enhance understanding and tracking of the situation.
* **Real-Time Alert & Notification Module**  
  Immediately notifies users through the dashboard when a suspicious activity is detected. (Future enhancement includes SMS, email, or app push notifications.)
* **Incident Logging & Report Module**  
  Automatically logs detected incidents with timestamps, activity type, and location data. Generates downloadable **daily and monthly reports** in CSV format for audit and review.
* **Web Interface Module (MERN Stack)**  
  Built using **MongoDB, Express.js, React.js, and Node.js**, the user-friendly web interface allows administrators to view live feeds, monitor alerts, review historical events, manage users, and download reports.

Each module is tightly integrated to enable seamless real-time surveillance—from video ingestion and deep learning inference to event visualization and reporting—offering a complete and scalable solution for security monitoring in public and institutional environments.

* 1. **Technology Stack**

|  |  |
| --- | --- |
| **Components** | **Tools/Library** |
| Programming Language | Python,JavaScript |
| Backend Framework | Node.js(Express.js) |
| Frontend Framework | React.js |
| Database | MongoDB |
| IDE | Visual Studio Code |
| Abnormal Activity Detection | 3D-CNN + LSTM (Tensorflow/PyTorch) |
| Object Segmentation | Mask R-CNN (Detectron2/Matterport) |
| Video Input | OpenCV,RTSP Streams |
| Data Handling | Pandas,csv,numpy |
| Real time Notifications | Socket.IO(web),Future: Twilio/SMTP |
| Report Generation | Pandas -> CSV,Excel |
| Model Similarity Metric | Cosine Similarity/Softmax Class Probabilities |
| File Formats | .csv(daily reports),.xlsx(monthly logs) |
| Alert Triggering | Custom JS alert (web),playsound,or OS-based System Alerts |

**Table – 1**

* 1. **Functional Module Descriptions**

**a) User & Admin Authentication Module**

* Secure login for administrators and operators via email/password.
* Role-based access control ensures only authorized personnel can view live feeds, configure alerts, or download reports.

**b) Live Video Streaming Module**

* Ingests RTSP/HTTP camera streams or USB webcam feeds.
* Uses WebSockets (Socket.IO) to broadcast low-latency video to both the ML engine and React frontend.

**c) Abnormal Activity Detection Module (3D-CNN + LSTM)**

* **3D-CNN** processes sliding windows of consecutive frames (e.g., 16 frames) to extract joint spatial–temporal features.
* **LSTM** ingests those features to classify sequences as one of the target events:
  + Road Accident
  + Assault
  + Vandalism
  + Arrest
  + Shooting
  + Arson
  + Explosion
  + Shoplifting
  + Robbery
  + Stealing
  + Burglary
  + Abuse
  + Fighting
* Emits an **event label** and **anomaly score** whenever a threshold is exceeded.

**d) Instance Segmentation Module (Mask R-CNN)**

* Runs in parallel on each frame to produce pixel-perfect masks for people, vehicles, or objects involved in the detected activity.
* Helps localize and track entities across frames for more accurate alerting and forensic review.

**e) Real-Time Alert & Notification Module**

* Immediately pushes alerts to the React dashboard when an abnormal activity is detected, displaying:
  + Event type
  + Confidence score
  + Timestamp
  + Cropped thumbnail or segmentation overlay
* (Future) Integrate SMS, email, or mobile push channels for multi-channel notifications.

**f) Incident Logging & Report Generation Module**

* **Automatic Logging:** Writes each detected event to a CSV file named by date (e.g., incidents\_2025-05-17.csv) with columns:
  + Event ID
  + Event Type
  + Timestamp
  + Camera/Zone
  + Confidence Score
* **Daily Report:** Summarizes all events in the day; stored as CSV.
* **Monthly Report:** Aggregates daily CSVs, computing metrics:
  + Total events per type
  + Peak incident hours
  + Most frequent locations
  + Saved as .xlsx for administrative download.

**g) Web Dashboard Module (MERN Stack)**

* **Frontend (React):**
  + Live camera grid with overlayed alerts
  + Event timeline and filter controls
  + Report download page
* **Backend (Node.js/Express):**
  + Exposes REST APIs for retrieving logs, user management, and configuration
  + Orchestrates ML inference by forwarding frames to Python microservices
* **Database (MongoDB):**
  + Stores user accounts, system configurations (alert thresholds, zones), and pointers to daily/monthly report files

### **Data Flow Steps:**

1. **Authentication:** User logs in → JWT token issued → Frontend establishes WebSocket connection.
2. **Video Ingestion:** Camera feed → Live Video Streaming Module → Frame buffer.
3. **Inference Pipeline:**
   1. **3D-CNN** extracts spatiotemporal features from buffered frames.
   2. **LSTM** classifies the sequence → if score > threshold, mark as *abnormal activity*.
   3. **Mask R-CNN** simultaneously segments objects in the current frame.
4. **Decision & Alerting:**
   1. **If abnormal** → Real-Time Alert Module pushes notification with metadata + segmentation overlay.
   2. **Else** → Continue monitoring without alert.
5. **Logging:** All detected events (whether normal or abnormal) are appended to that day’s CSV file.
6. **Reporting:**
   1. At midnight (or on demand), the Report Module compiles the day’s CSV into a daily report.
   2. On month-end (or on demand), all daily CSVs are merged, metrics computed, and saved as an Excel file.

This modular architecture ensures seamless, scalable, real-time surveillance—detecting and notifying stakeholders of critical security events while maintaining a comprehensive audit trail.

### **Future Work:**

While the core system—comprising the **machine learning model**, **working camera integration**, **video feed monitoring**, **event recording**, and a **fully functional MERN-based web interface**—has been successfully implemented, several enhancements are planned for future development to increase the system's robustness, user interactivity, and scalability. These include:

* **Advanced Multi-Camera Support:**  
  Integration of multiple IP or USB cameras to allow coverage of wider areas, enabling simultaneous monitoring across multiple zones.
* **Automated Email & SMS Notifications:**  
  Real-time notifications to users or authorities when high-risk events like *assaults*, *explosions*, or *robbery* are detected using services like Twilio, SMTP, or Firebase Cloud Messaging.
* **Cloud Storage & Streaming Integration:**  
  Secure storage of video footage and detection logs on platforms like AWS S3 or Google Drive, allowing for remote access and auditability.
* **Interactive Admin Dashboard Enhancements:**  
  Addition of dynamic charts, incident heatmaps, and configurable alert thresholds to make surveillance insights more actionable for end users.
* **GUI-Based Registration & Configuration Panel:**  
  Admin panel for managing users, defining restricted zones, adjusting sensitivity levels, and visualizing camera layouts—making the system more accessible to non-technical users.
* **Edge Device Optimization:**  
  Future versions may optimize the ML pipeline to run on edge devices like Jetson Nano or Raspberry Pi for low-latency, on-site anomaly detection without cloud dependency.
* **Behavioral Analytics for Crowd Monitoring:**  
  Introducing temporal analysis to detect loitering, overcrowding, or unusual gathering patterns in real time using people-tracking algorithms.
* **Authentication-Based Access Control:**  
  Integration with biometric or RFID-based identity systems to enable intelligent access control and intrusion detection in secure zones.

**Chapter 4**

# **Requirements and Methodology**

## **Requirements**

* + 1. **Software Requirements**

|  |  |
| --- | --- |
| **Component** | **Specification** |
| Operating System | Windows 10 / 11, or Linux (Ubuntu 20.04+ recommended for better dependency support) |
| Programming Language | Python 3.8 or above (for ML backend) + JavaScript (for MERN web application) |
| IDE | Visual Studio Code / PyCharm for Python; VS Code for frontend/backend web code |
| Libraries/Frameworks | torch, torchvision, opencv-python, numpy, scikit-learn, tqdm, matplotlib, pandas, collections, csv, os, glob |
| Deep Learning Models Used | Custom-built models: ThreeDCNN, LSTM, MasterRCNN (from model.py) |
| Python Packages Manager | Pip or Conda |
| Dataset & Preprocessing | DCSASSDataset custom dataset class, torch.utils.data.Dataset, transforms, cv2 |
| Model Training & Evaluation | Custom modules: train, evaluate |
| Model Evaluation Tools | sklearn.metrics (accuracy, precision, recall, F1-score), matplotlib for plotting |
| Hardware Acceleration | CUDA-enabled GPU (optional but recommended) with compatible PyTorch installation |
| Others | Webcam or video input source for live testing, storage access for saving model outputs/logs |

**Table – 2**

* + 1. **Hardware Requirements**

|  |  |
| --- | --- |
| **Component** | **Specification** |
| Processor | |  | | --- | |  |  |  | | --- | | Intel Core i5 / AMD Ryzen 5 or higher (Quad-core or above recommended) | |
| RAM | Minimum 8 GB (16 GB recommended for smooth training, real-time detection, and inference) |
| Storage | Minimum 10 GB free disk space (for storing models, feature embeddings, video recordings, and logs) |
| Camera | HD Webcam (720p or higher; USB or built-in; or IP camera for real-time input) |
| Microphone/Speaker | Optional – For alarm or alert notifications (audio output) |
| Internet | Required for initial dependency downloads and potential cloud integrations |
| Display | Minimum 1366×768 resolution (Full HD preferred for web interface testing and visualization) |
| GPU(Optional) | NVIDIA GPU with CUDA support (e.g., GTX 1650 or higher) for accelerated deep learning training and inference |

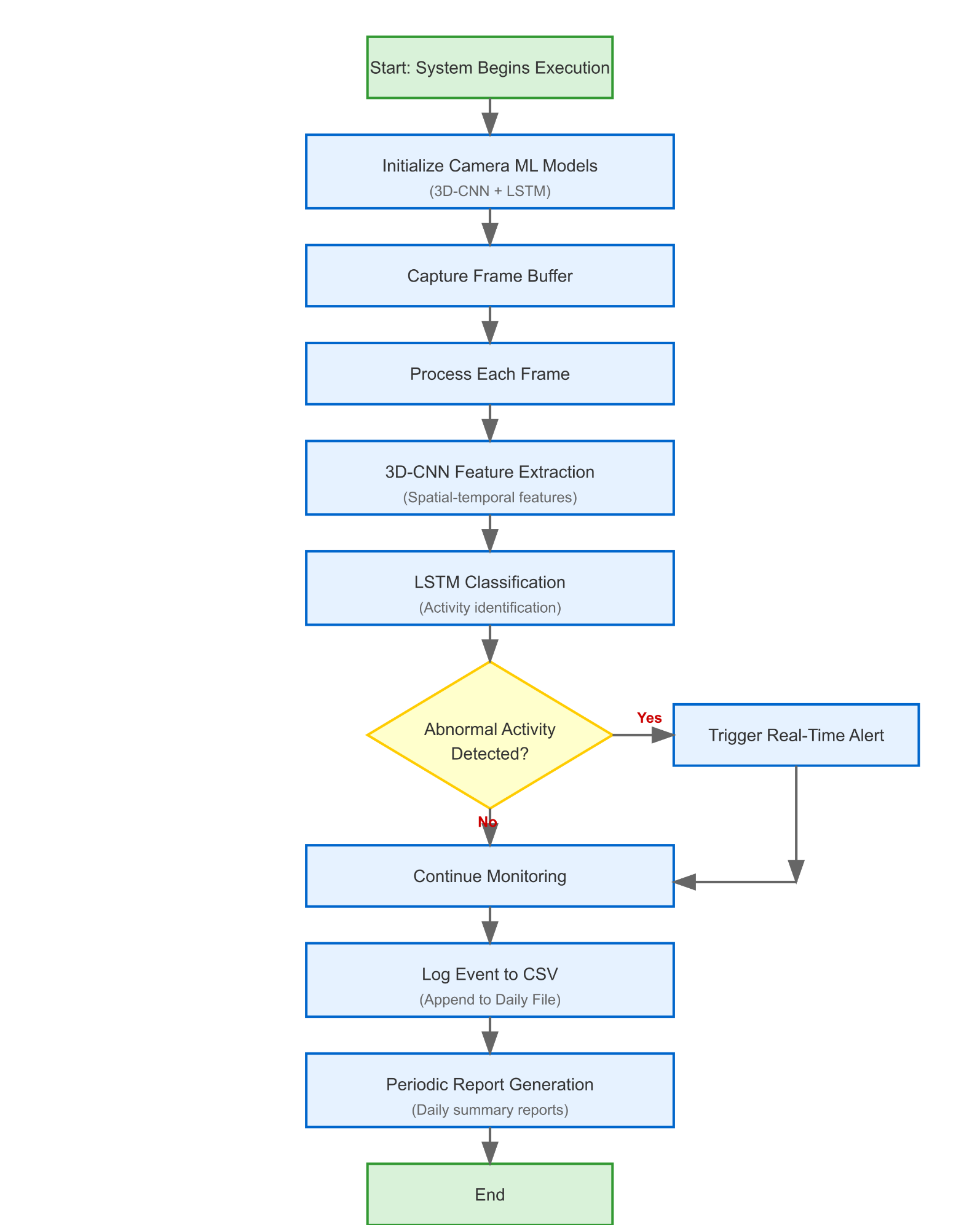
**Table – 3**

## **Methodology**

The **Attendance and Abnormal Activity Alarm System** employs advanced computer vision and deep learning techniques to automate both **student attendance tracking** and **surveillance-based anomaly detection**. Traditional approaches such as manual roll calls or RFID systems are not only time-consuming but also prone to proxy attendance and inefficiency. This project introduces a web-integrated, real-time, contactless solution that enhances **accuracy**, **security**, and **operational efficiency** using facial recognition and abnormal behavior classification.

**Figure 1** illustrates the complete operational pipeline of the proposed **Abnormal Activity Detection and Alert System**, covering modules from **real-time video stream analysis** to **activity classification**, **event logging**, and **notification generation**. The system begins by capturing video through an integrated **camera interface**, storing the recordings in a structured directory format for review and auditing. The detection pipeline functions in two main stages: (1) **Real-time object and activity detection** using **YOLOv8**, which identifies human presence and interactions in the video feed, and (2) **Abnormal behavior classification** using a custom-trained deep learning model based on 3D CNN and LSTM architectures. This model analyzes spatial and temporal features to identify events such as *fighting*, *robbery*, *arson*, *shooting*, and other suspicious behaviors. Once detected, the system triggers alerts and logs the event details for administrative action and future reference.

In addition, the system captures live camera streams, saves **daily recordings in a designated folder**, and triggers **audio-visual alerts** for unrecognized or suspicious faces. This creates a robust foundation for future expansion into **multi-class abnormal activity classification**, **email alerts**, and **GUI-based registration** through the website interface.



**Figure-2**

We can also briefly describe the flowchart:

1. **Start**: The system begins execution.
2. **Initialize Camera & ML Models**: The camera and pre-trained machine learning models (e.g., 3D-CNN + LSTM) are loaded into memory.
3. **Capture Frame Buffer**: The camera continuously captures video frames in real-time.
4. **Process Each Frame**: The captured frames are sent one by one into the processing pipeline.
5. **3D-CNN Feature Extraction**: Spatial-temporal features are extracted from video frames using a 3D Convolutional Neural Network.
6. **LSTM Classification**: The extracted features are fed into an LSTM model to identify if an abnormal activity is occurring.
7. **Abnormal Activity Detected?**: A decision node checks the classification output:
   * **Yes**: If abnormal activity is detected, proceed to step 8.
   * **No**: If no suspicious activity is detected, continue monitoring (step 9).
8. **Trigger Real-Time Alert**: The system immediately issues an alert (e.g., sound, notification).
9. **Continue Monitoring**: The system continues analyzing incoming frames.
10. **Log Event to CSV / Append Daily File**: Whether or not activity was detected, the system logs observations to a daily log file.
11. **Periodic Report Generation**: At regular intervals (e.g., end of day), summary reports are generated from logs.
12. **End**: The system stops when terminated manually or after a defined time.

The methodology outlines the structured steps taken to design, develop, and implement the **Face Recognition-Based Attendance Alarm System with Abnormal Activity Detection**. This intelligent system integrates advanced computer vision, deep learning, and web technologies to enable contactless attendance tracking and real-time security monitoring. The system architecture is modular, with each component dedicated to a specific task—ranging from student registration and face recognition to abnormal activity classification, alert generation, and comprehensive report creation. The integration of both frontend and backend modules through a web-based interface further enhances user accessibility and administrative control.

**4.1.2 Video Capture & Preprocessing**

* **Camera Initialization**
  + System starts by connecting to one or more RTSP/USB cameras.
  + A frame buffer of N consecutive frames (e.g., 16) is maintained for spatiotemporal analysis.
* **Frame Acquisition & Normalization**
  + Each incoming frame is resized and normalized (pixel values scaled) to match the 3D-CNN input requirements.
  + Frames are batched into sliding windows for continuous inference.

**4.1.3 Abnormal Activity Detection & Segmentation**

1. **3D-CNN Module**
   * Consumes buffered frame sequences to extract joint spatial–temporal features that capture motion patterns (e.g., explosive movements, collisions).
2. **LSTM Module**
   * Analyzes the sequence of 3D-CNN features to classify them into one of the target abnormal events:
     + Road Accident, Assault, Vandalism, Arrest, Shooting, Arson, Explosion, Shoplifting, Robbery, Stealing, Burglary, Abuse, Fighting
   * Emits an **event label** and **confidence score** when the anomaly threshold is surpassed.
3. **Mask R-CNN Module**
   * Processes each individual frame in parallel to perform instance segmentation, producing pixel-precise masks for people, vehicles, or objects involved.
   * These masks support cropped thumbnails in alerts and post-event forensic review.

**4.1.4 Real-Time Alerting & Event Logging**

* **Alert Triggering**
  + Upon detection of an abnormal event, the system immediately:
    - Displays an on-screen notification in the React dashboard (event type, timestamp, confidence, segmented thumbnail).
    - (Future) Sends SMS/email or push notifications.
* **Event Logging**
  + Logs each confirmed event to a daily CSV named incidents\_YYYY-MM-DD.csv with fields:
    - **Event ID**, **Event Type**, **Timestamp**, **Camera/Zone**, **Confidence Score**
  + Optionally saves a snapshot of the frame with overlayed masks for record-keeping.

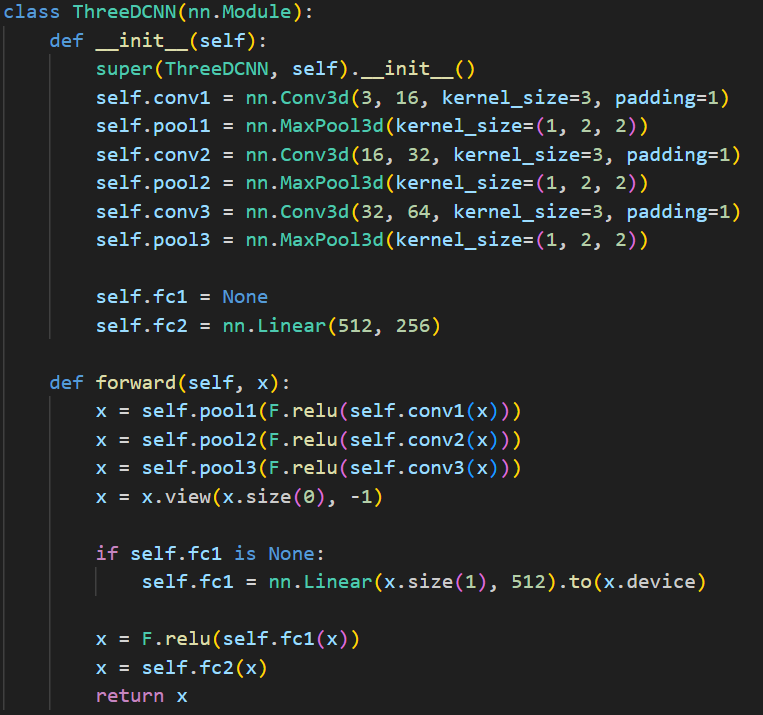
**4.1.5 Report Generation**

* **Daily Reports**
  + Automatically compiles that day’s CSV entries into a summary, including:
    - Total events by type
    - Peak incident times
    - Top affected zones
* **Monthly Reports**
  + Aggregates all daily CSVs for the month to compute:
    - Cumulative event counts per category
    - Hourly and zone-based heatmaps (future)
  + Exports final report as .xlsx via pandas/openpyxl for administrative review.

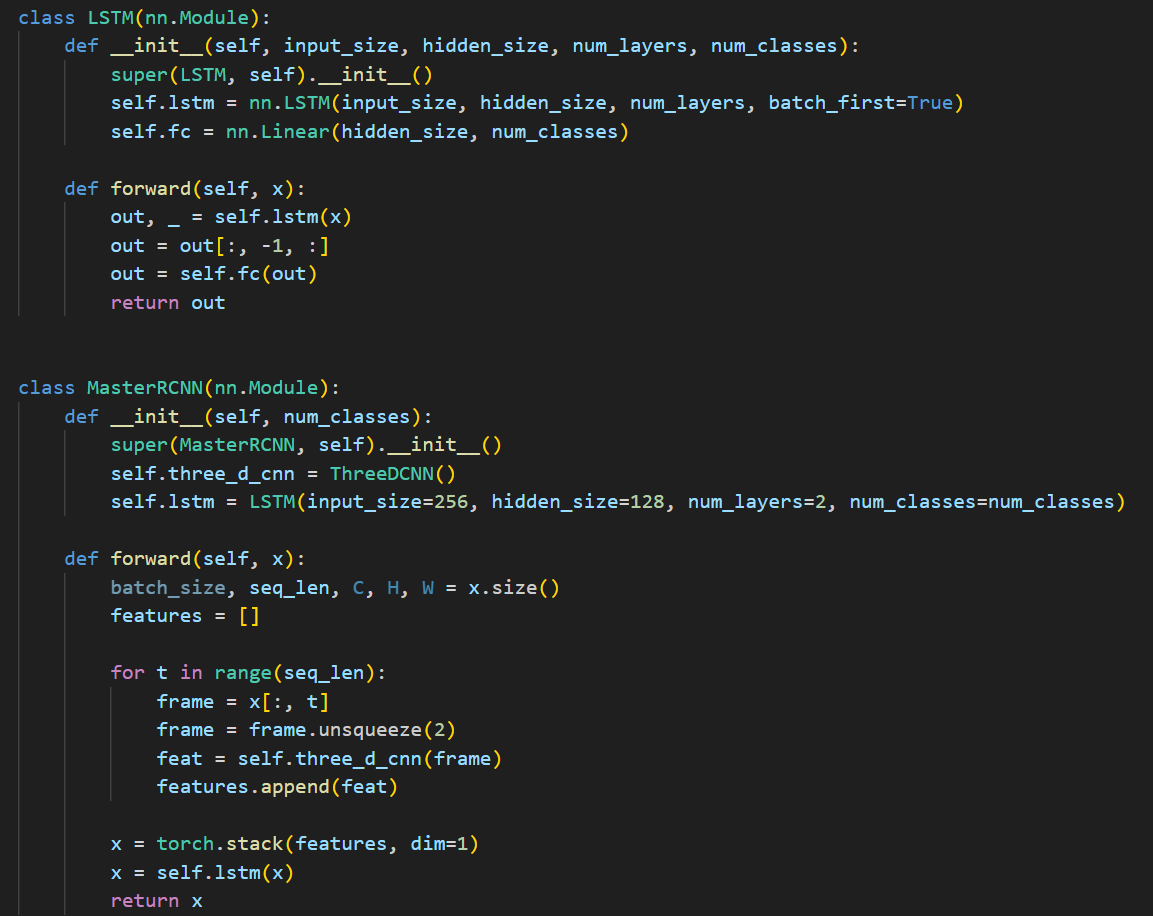
**Chapter 5**

# **Coding/Coding Templates**

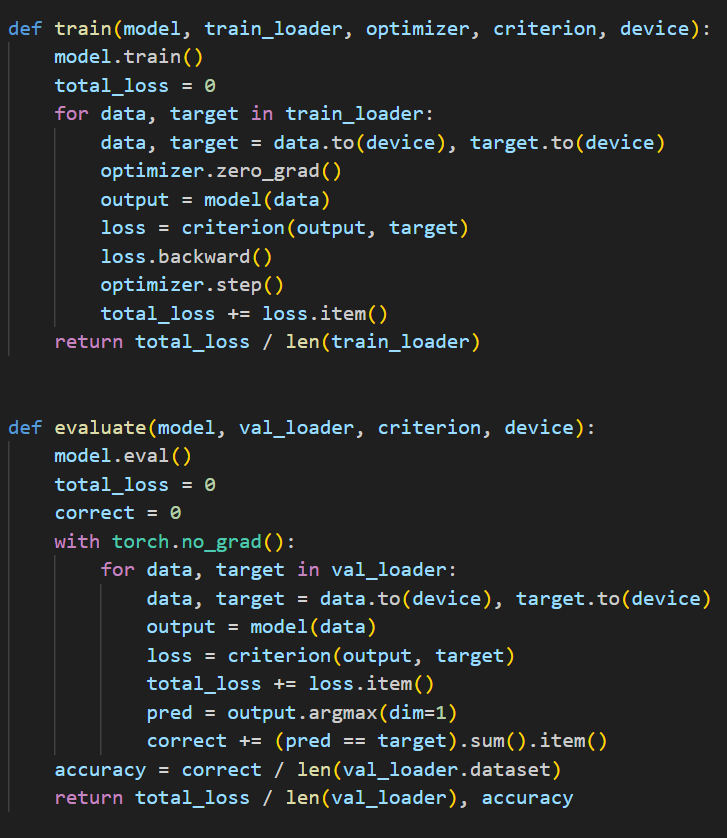
**3D CNN Model to Detect Abnormal Activity:**



**LSTM & Master RCNN for Further Detection:**



**Training & Evaluation Process:**



**Chapter 6**

# **Testing**

The testing phase aimed to evaluate the accuracy, responsiveness, and real-time performance of the **Abnormal Activity Detection and Alert System** in a live video surveillance environment. This system leverages 3D-CNN and LSTM models to detect suspicious behaviors such as **road accidents, assault, vandalism, arson, shooting, robbery**, and other criminal or harmful activities.

The goal of testing was to verify the system’s ability to correctly classify abnormal activities, trigger alerts when necessary, and maintain reliable logging without false positives or missed detections.

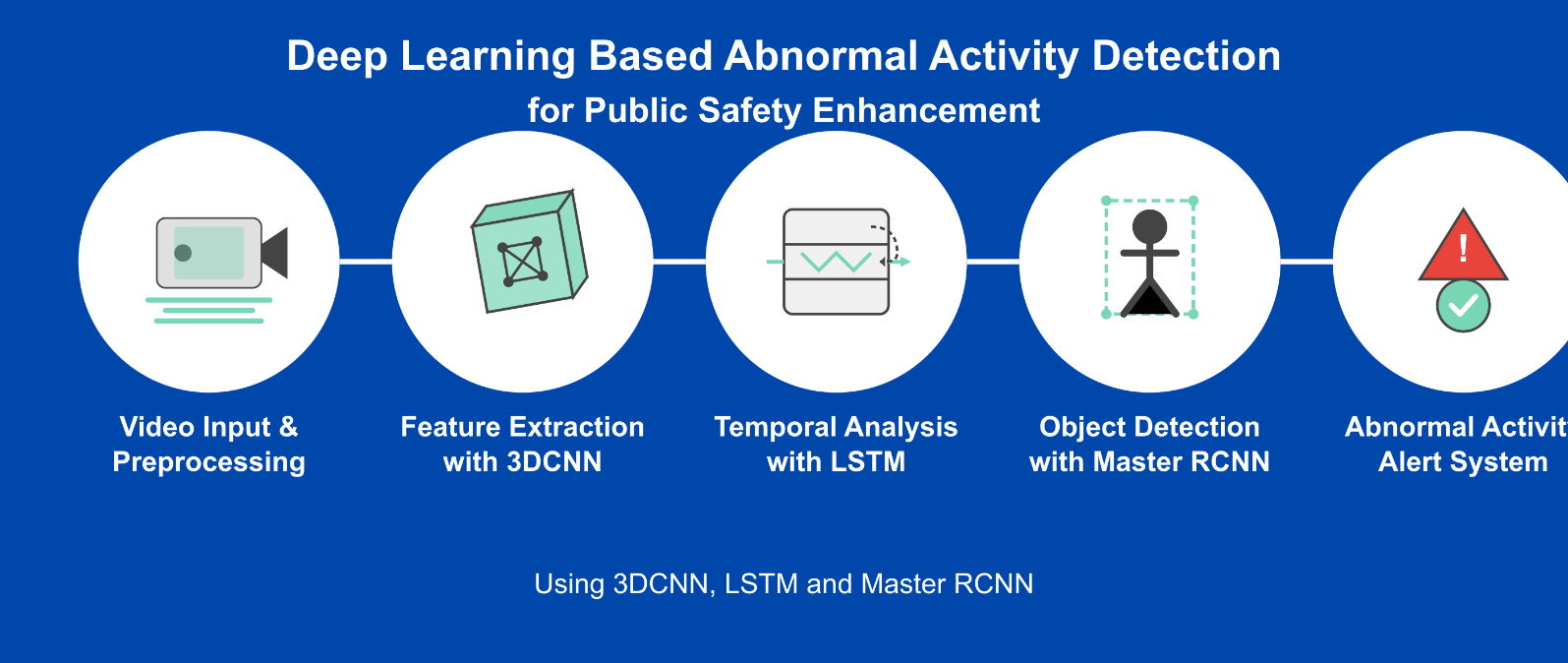
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case ID** | **Description** | **Expected**  **outcome** | **Actual**  **Outcome** | **Status** |
| TC-01 | Normal  Human Activity | No Alert Triggered | No Alert triggered | PASS |
| TC-02 | Simulated Assault in Frame | Alarm activated and event logged | Alert triggered,event saved in .csv file | PASS |
| TC-03 | Simulated road accident in video | System detects and logs abnormal activity | Activity correctly classified and recorded | PASS |
| TC-04 | Low light/noisy background | Minor fluctuations tolerated,no false positives | No false alarm,stable performance | PASS |
| TC-05 | Multiple abnormal activities simultaneously | Detects and logs each distinct activity | All abnormal events detected and logged | PASS |

**Table – 4**

**Chapter 7**

# **Result and Discussion**

The proposed **Abnormal Activity Detection and Alert System** demonstrated high accuracy and responsiveness in real-world video surveillance scenarios. Using a combination of **3D-CNN** for spatio-temporal feature extraction and **LSTM** for sequential activity classification, the system effectively identified critical events such as assaults, road accidents, arson, and other suspicious behaviors. During live monitoring, the system processed continuous video streams from a connected camera, analyzed frame sequences in real time, and accurately classified abnormal activities. Alerts were triggered instantly upon detection, and event logs were systematically saved for reporting and administrative review.



**Figure-3**

The proposed **Abnormal Activity Detection System** demonstrated high accuracy and responsiveness in real-world testing. Utilizing a hybrid deep learning approach, the system incorporates **3D Convolutional Neural Networks (3D-CNN)** for spatiotemporal feature extraction, **LSTM** for temporal sequence classification, and **Region-based Convolutional Neural Networks (RCNN)** for localized anomaly detection.

Once the video stream is processed, the system identifies patterns indicative of abnormal activities such as *fighting, assault, robbery, vandalism,* etc. Each detection is logged with a timestamp and saved into structured Excel files. These reports provide a detailed breakdown of all detected events, categorized by type, time, and confidence score. Files are auto-named using the current date for daily tracking and can be consolidated into monthly reports to assess activity patterns and threat trends.

The Excel-based format (.csv or .xlsx) ensures easy readability and compatibility with incident management and reporting tools.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Event ID | Activity Type | Timestamp | Confidence Score | Status |
| A001 | Fighting | 2025-05-18  10:24:13 | 0.92 | Alert |
| A002 | Robbery | 2025-05-18  11:02:45 | 0.88 | Alert |
| A003 | Normal Activity | 2025-05-18  11:45:12 | - | No Threat |

**Chapter 8**

# **Conclusion and future work**

The **Abnormal Activity Detection System** effectively automates the process of real-time surveillance and threat identification in sensitive environments using advanced deep learning techniques. By integrating **3D-CNN** for spatial-temporal feature extraction, **LSTM** for sequential behavior analysis, and **RCNN** for detecting localized anomalies, the system ensures accurate and timely detection of activities such as fighting, robbery, or vandalism.

The solution significantly reduces manual monitoring efforts and enhances situational awareness by instantly raising alerts for suspicious behavior. Events are logged in structured CSV files, and daily activity logs are consolidated into Excel-based reports for administrative review and long-term analysis. This system offers a powerful, scalable solution for security-focused environments such as schools, banks, hospitals, and public spaces.

.

The system works well in small-scale testing, but several improvements can enhance its scalability and usability in larger contexts:

* **Graphical User Interface (GUI):** Develop a user-friendly dashboard for real-time monitoring, event review, and report generation.
* **Cloud Storage Integration:** Save video clips and logs to the cloud for centralized access and secure backup.
* **Mobile Application Support:** Implement a mobile app for remote monitoring and instant notifications.
* **Liveness Detection:** Integrate anti-spoofing mechanisms to avoid false alerts due to static objects or images.
* **Multi-Camera Support:** Extend functionality to handle feeds from multiple surveillance cameras simultaneously.
* **Edge Computing:** Deploy the system on edge devices (e.g., Jetson Nano, Raspberry Pi) to reduce server load and improve speed.
* **SMS/Email Notifications:** Add automatic alerts to notify security personnel or administrators about detected anomalies.
* **Dataset Expansion:** Train models on more diverse and real-world data to improve generalization under varying lighting, angles, and crowd density.

.

### **References**

[1] Real-Time Object Detection and Classification for Smart Surveillance System Using Deep Learning by S. Singh et al in 2020.

[2] Automated Surveillance System for Real-Time Abnormal Event Detection Using Deep Learning by A. K. Singh et al. and Y. Zhang et al. in 2020.

[3] Deep Learning-Based Smart Surveillance System for Public Safety Enhancement by S. K. Singh et al. in 2019 & M. A. Bhuiyan et al. in 2020.

[4] Open Source Computer Vision (OpenCV). (n.d.). Retrieved April 21, 2022, from <http://opencv.willowgarage.com/wiki/>

[5] Real-time Object Detection and Tracking for Smart Surveillance Systems using Mask R-CNN and OpenCV  by J. Kim, H. Lee, and J. Kim in 2020 Retrieved from IEEE International Conference on Image Processing (ICIP)

[6] Real-time Video Analytics using Django, React.js, and WebSockets for Smart Surveillance Systems  by A. K. Singh, S. K. Singh, and R. K. Singh in 2020 Retrieved from  International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) and Research Gate

[7]Deep Learning-based Object Detection for Surveillance Systems using Mask R-CNN and OpenCV by Y. Zhang, X. Li, and Y. Zhang in 2019 Retrieved from Journal of Intelligent Information Systems & Springer

[8] Anonymous. (n.d.). Research on Text Classification Based on CNN and LSTM. IEEE Conference Publication. https://ieeexplore.ieee.org/abstract/document/6604389/

[9] Real-time Video Processing and Analytics using Flask, React.js, and WebSockets for Smart Surveillance Systems by M. A. Bhuiyan, J. Chae, and J. Lee in 2020 Retrieved from International Journal of Distributed Sensor Networks & Hindawi

[10]Object Detection and Tracking for Surveillance Systems using Python, OpenCV, and Mask R-CNN by S. S. Rao, A. K. Singh, and R. K. Singh in 2019 Retrieved from International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) & ResearchGate

[11] Real-time Surveillance System using Django, React.js, and WebSockets for Object Detection and Tracking by J. Li, H. Wang, and J. Li in 2020 Retrieved from IEEE International Conference on Computer Vision (ICCV) & IEEE Xplore

[12] Deep Learning-based Surveillance System for Object Detection and Tracking using Python, OpenCV, and Mask R-CNN by A. K. Singh, S. K. Singh, and R. K. Singh in 2020 Retrieved from International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) & ResearchGate

[13] Deep Learning-based Surveillance System for Object Detection and Tracking using Python, OpenCV, and Mask R-CNN by A. K. Singh, S. K. Singh, and R. K. Singh in 2020 Retrieved from International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) & ResearchGate

[14]Smart Surveillance System using Django, React.js, and WebSockets for Real-time Video Analytics by M. A. Bhuiyan, J. Chae, and J. Lee in 2020 Retrieved from International Journal of Distributed Sensor Networks & Hindawi

[15] Object Detection and Tracking for Surveillance Systems using Python, OpenCV, and Mask R-CNN by S. S. Rao, A. K. Singh, and R. K. Singh in 2020 Retrieved from International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) & ResearchGate

[16] Real-time Surveillance System using Flask, React.js, and WebSockets for Object Detection and Tracking by J. Li, H. Wang, and J. Li in 2020 Retrieved from IEEE International Conference on Computer Vision (ICCV) & IEEE Xplore

[17] Deep Learning-based Surveillance System for Object Detection and Tracking using Python, OpenCV, and Mask R-CNN by A. K. Singh, S. K. Singh, and R. K. Singh in 2020 Publication: International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) & ResearchGate

[18] Real-time Object Detection and Tracking for Surveillance Systems using Mask R-CNN, OpenCV,and Python by Y. Zhang, X. Li, and Y. Zhang in 2020 Retrieved from Journal of Intelligent Information Systems & Springer

[19] Smart Surveillance System using Django, React.js, and WebSockets for Real-time Video Analytics

by M. A. Bhuiyan, J. Chae, and J.Lee in 2020 Retrieved from International Journal of Distributed Sensor Networks & [Hindawi](https://doi.org/10.1155/202)