*A Project Report on*

# REAL-TIME CROWD AND VIOLENCE DETECTION USING YOLOV11 WITH AUTOMATED ALERTS AND NOTIFICATIONS

**SENIOR DESIGN PROJECT**

*Submitted in partial fulfillment for the awards of the degree of*

## Bachelor of Technology in Computer Science and Engineering

*by*


**YASWANTH RAM (21BCE9832)**

**MADAVI LATHA (21BCE9827)**

**SRILAKSHMI (21BCE9872)**




**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING VIT-AP UNIVERSIRTY AMARAVTI- 522237**


May 2025

## DECLARATION

I here by declare that the thesis entitled “REAL TIME CROWD AND VOILENCE DETECTION WITH AUTOMATED ALETS AND NOTIFICATION” submitted by me, for the award of the degree of BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING, VIT is a record of bonafide work carried out by me under the supervision of DR. Anurag De

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Amaravati

Date: 06.05.2025

Signature of the Candidate

**T.Yaswanth Ram**

**P.Madavi Latha**

**K. Srilakshmi**

## CERTIFICATE

This is to certify that the Senior Design Project titled "**Enhancing Public Safety: Real-Time Crowd and Violence Detection Using YOLOv11 with Automated Alerts and Notifications**" that is being submitted by **T. Yaswanth Ram(21BCE9832), P. Madavi Latha(21BCE9827), K.Srilakshmi(21BCE9872)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other institute or university for award of any degree or diploma and the same is certified.

DR. ANURAG DE
**Guide**

**The thesis is satisfactory / unsatisfactory**

**Internal Examiner1** **Internal Examiner2**

**Approved by**

Dr. G. MUNNESWARI
**HoD, Department of Data Science and Engineering**
**School of Computer Science and Engineering**

ABSTRACT

A real-time crowd and violence detection system is developed using the YOLOv11 (You Only Look Once version 11) algorithm. The system is designed to enhance public safety by automating the identification of crowd density and detecting violent activities in images and video streams. By leveraging state-of-the-art deep learning techniques, particularly the YOLOv11 architecture, the system can quickly and accurately process visual data in real time, identifying crucial incidents such as crowd surges and violent altercations. This paper details the entire workflow, including dataset preparation, model training, and validation, followed by testing on unseen data to assess the model's performance.

The system integrates an intuitive user interface built with Streamlit, which allows users to upload images, videos, or stream live webcam footage for analysis. Upon detecting any crowd-related or violent incidents, the system generates real-time alerts, ensuring immediate intervention. In addition to crowd detection, a dedicated violence detection model has been incorporated, trained to recognize various forms of violent behavior such as physical altercations and aggressive actions.

This model ensures comprehensive safety monitoring in dynamic environments. The proposed system holds significant applications in areas such as public safety, law enforcement, large event management, and smart city initiatives, offering proactive solutions for monitoring and ensuring security. By automating detection and alert systems, the platform minimizes response times and empowers authorities to address incidents before they escalate.

**ACKNOLEDGEMENT**

It is my pleasure to express with deep sense of gratitude to Dr. Anurag De, Assistant Professor Sr. Grade-1 for his constant guidance, continual encouragement, and understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Project Management.

I would like to express my gratitude to Dr. G. Viswanathan, Sri. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G. V. Selvam, Dr. S. V. Kota Reddy, and Dr.S.Sudhakar Ilango, Dean SCOPE, for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Saroj Kumar Panigrahy Professor Grade 1 and all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.Place: Amaravati

Date: Name of the student

T. Yaswanth Ram
P. Madavi Latha
K. Srilakshmi

**Table of Contents**

## List OF Acronym

| Acronym | Full Form |
|---|---|
| AI | Artificial Intelligence |
| CNN | Convolutional Neural Network |
| CV | Computer Vision |
| DL | Deep Learning |
| ML | Machine Learning |
| YOLO | You Only Look Once |
| UCF | University of Central Florida |
| RNN | Recurrent Neural Network |
| LSTM | Long Short-Term Memory |
| NLP | Natural Language Processing |
| TP | True Positive |
| FP | False Positive |
| TN | True Negative |
| FN | False Negative |
| IoU | Intersection over Union |
| FPS | Frames Per Second |
| AP | Average Precision |
| mAP | mean Average Precision |
| SSD | Single Shot Detector |
| RCNN | Region-based Convolutional Neural Network |
| FRCNN | Faster Region-based Convolutional Neural Network |
| GPU | Graphics Processing Unit |
| CPU | Central Processing Unit |
| API | Application Programming Interface |
| CSV | Comma-Separated Values |
| RGB | Red Green Blue |
| PIL | Python Imaging Library |
| CVPR | Conference on Computer Vision and Pattern Recognition |

**LIST OF FIGURES:**

LIST OF TABLES

Chapter 1

# Introduction

Public safety in urban environments is a growing concern, with rising population densities and increasing security threats. Traditional surveillance systems rely on manual monitoring, which is not only time-consuming but also prone to human error and inefficiencies. The inability of traditional methods to detect real-time incidents, such as overcrowding and violent activities, leads to delayed response times and compromises public security. Recent developments in deep learning (DL) and computer vision (CV) have opened the door to automated, real-time crowd and violence detection systems. The ability to detect violent behavior and assess crowd density is crucial for ensuring public safety and preventing incidents from escalating.

This project explores the use of YOLOv11 (You Only Look Once version 11), a state-of-the-art object detection algorithm, to develop a real-time system capable of detecting crowd density and identifying violent actions from video and image data. The system integrates machine learning models, including a specialized violence detection model, to provide actionable insights for law enforcement and security personnel. The proposed system offers a more efficient and scalable approach compared to traditional surveillance methods, enhancing decision-making and response times.

## 1.1 Background and Motivation

Modern urban environments demand a high degree of vigilance to ensure safety and prevent chaos. The increasing number of incidents involving physical aggression and crowd congestion in public places has highlighted the need for smarter surveillance. Conventional monitoring setups, which depend on continuous manual observation of camera feeds, are neither scalable nor effective under high workload conditions.
Recent progress in deep learning and video analytics has introduced opportunities to automate critical parts of the surveillance process. Algorithms like YOLOv11 allow systems to quickly identify objects and actions in video frames, making them well-suited for real-time crowd and violence detection. These intelligent systems are capable of learning from complex visual data, adapting to diverse scenarios, and offering instant alerts upon identifying suspicious activities.

The motivation behind this project stems from both the shortcomings of manual systems

and the promise shown by AI-powered surveillance. Integrating YOLOv11 for real-time detection can significantly improve public safety response times and reduce dependency on human intervention, all while increasing monitoring accuracy.

## 1.2 Problem Statement

Despite the widespread installation of CCTV cameras, most surveillance systems still depend on manual monitoring. Security personnel are required to continuously watch multiple camera feeds for long periods, which often leads to fatigue, decreased attention, and delayed response times. As a result, critical incidents such as violent behavior or overcrowding may go unnoticed or be detected too late, posing serious risks to public safety.

The problem is further amplified in crowded and high-traffic environments like malls, railway stations, schools, and events, where fast-paced movement and unpredictable situations are common. Traditional surveillance lacks the intelligence to process these dynamic scenes effectively in real time. Although some AI-powered tools exist, many struggle with challenges such as rapid motion, complex backgrounds, low lighting, and overlapping objects. Moreover, existing systems are often expensive, hard to scale, or incompatible with current infrastructure, making them impractical for large-scale deployment.

This project aims to address these limitations by developing a cost-effective and scalable intelligent monitoring system using YOLOv11. The goal is to automate the detection of violence and overcrowding in surveillance footage, reducing dependency on human supervision. By leveraging real-time video analytics and advanced deep learning techniques, the proposed system will enable faster, more accurate threat detection and improve situational awareness in security operations.

## 1.3 Objectives

The goal of this project is to develop a robust and efficient real-time video surveillance system that can enhance public safety and security by detecting violent behaviour and crowd congestion from video streams. By leveraging advanced AI models, the system will provide a proactive solution to monitor and respond to potential threats. The key objectives of the project include:

- To build a real-time video surveillance system capable of detecting violent behaviour and crowd congestion from video streams.
- To integrate the YOLOv11 model for high-speed, accurate detection of individuals, movement patterns, and dangerous actions.
- To reduce human error and fatigue by automating the detection process and providing timely alerts.
- To validate the system's accuracy, precision, and recall across various environments and conditions.
- To design a system that is scalable and adaptable to multiple public surveillance scenarios such as schools, bus stops, markets, and offices.

## 1.4 Scope of the project

This project aims to develop a real-time public safety surveillance system using the YOLOv11 object detection model to detect violent behavior and crowd congestion. The system processes live video feeds from various sources like CCTV cameras and mobile devices to identify potential security threats.

**Key Components of the Project:**

- **Real-Time Video Surveillance**: Analyses live or recorded footage to detect violent behavior and overcrowding using YOLOv11.
- **Automated Alerts**: Triggers email notifications and sound alarms when suspicious activities are detected to reduce response times.
- **User-Friendly Web Interface**: A Streamlit-based web application allows users to upload video feeds, monitor analysis, and view incidents in real-time.
- **Scalability**: The system is designed for use in diverse environments like schools, offices, and public spaces, with support for multiple camera feeds.
- **Model Validation and Performance**: The system is tested for accuracy and effectiveness in detecting threats across various conditions.
- **Assistive Tool**: The system complements human monitoring, offering automated insights for faster decision-making without replacing human judgment.

The scope does not include physical hardware control (e.g., alarms or access systems), but future integrations are possible. The focus is on creating an AI-driven surveillance system for real-time detection and alerts in public environments.

## 1.5 Organization of the Report

The report is structured into various chapters. Chapter 1 introduces the problem, objectives, and scope. Chapter 2 presents a detailed literature review. Chapter 3 discusses the feasibility of the system. Chapter 4 elaborates on the Existing system. Chapter 5 presents system analysis and requirements, followed by proposed methodology in Chapter 6. chapter 7 presents System Design and implementation followed by Testing and Results in chapter 8. Next presents conclusion and future scope

# Chapter 2
# Literature Review

## 2.1 Literature Review

- **Mayuri S. Talore & Pallavi R. Wankhade - "Criminal Face Identification System,"**

While criminal identification in Malaysia is mostly done through fingerprints, this method is limited as most criminals nowadays try their best not to leave fingerprints at the crime scene. Displacement's infeasibility is a common phenomenon for many criminals today, especially as security technology enables the installation of cameras, such as closed-circuit television (CCTV), in various sites of interest. CCTV footage can assist in identifying suspects who were present at the scene, but there are still several issues regarding its reliability since there is considerable lack of software that allows the automatic recognition of features and identification of individuals. This paper presents a system using Python programming language that performs automated analysis of criminals' databases. The system is capable of recognizing and identifying a person, even masking the entire face to protect the privacy of suspects. Moreover, deeper analysis of criminal databases is performed automatically through the system, enabling better identification in ongoing investigations.

- **Yang yang, Zheng-Jun Zha, Heng Tao Shen, and Tat-Seng Chua, "Robust Semantic Video Indexing by Harvesting Web Images".**

Semantic video indexing solutions, or as described in the literature, video annotation and video concept detection, are relatively new topics in research. The major drawbacks of most existing solutions are the unavailability of training videos or weak performance achieved through a small training set. This paper presents robust semantic video indexing that uses web images, tagged by users, for learning classifiers for robust semantic indexing. The following two interesting issues are investigated: (a) the gap of the domain between images and videos, and (b) the presence of wrong tags for noisy web images. The approach incorporates evaluating the probabilities for an image being tagged correctly, treated as confidence scores. Therefore, images whose confidence scores are less than a specific threshold are discarded, resulting in a novel image-to-video indexing method that drives learning of reliable classifiers with very few training videos and much user- tagged imagery. This methodology significantly enhances the performance of video indexing, allowing for better scalability and more accurate annotation of video content.

- **Sreenu, G., & Durai, S. (2024), "Violence Detection in Crowd Videos Using**

**Nuanced Facial Expression Analysis," Systems and Soft Computing, 200104**

This study focuses on analyzing crowd behavior in videos by examining facial expressions to detect violence. The authors present a deep learning-based approach that leverages nuanced facial expression analysis to classify the severity of violent activities in crowded environments. By using convolutional neural networks (CNNs) and facial recognition algorithms, the model is able to detect aggression and violent behavior from visual cues, offering valuable insights for surveillance and public safety applications. The paper highlights the challenges of detecting subtle facial expressions that may indicate aggression and proposes a framework for accurately identifying such behaviors. The integration of deep learning models in violence detection systems ensures better accuracy and real-time processing for security monitoring systems.

- **Hu, X., Fan, Z., Jiang, L., Xu, J., Li, G., Chen, W., ... & Zhang, D. (2022), "TOP-ALCM: A Novel Video Analysis Method for Violence Detection in Crowded Scenes," Information Sciences, 606, 313-327.**

Hu et al. propose a new approach, TOP-ALCM, that aims to improve video analysis for detecting violence in crowded settings. This method combines temporal and optical flow-based techniques to effectively identify and track violent actions while managing the complexity and density of crowd scenes. The authors discuss the benefits of this algorithm in distinguishing aggressive actions from normal crowd behavior and demonstrate its performance through multiple case studies and datasets. The paper presents empirical results showing that TOP-ALCM outperforms traditional video analysis methods, especially in environments with high crowd density, offering more accurate detection of violent incidents. This method's robustness and ability to handle complex crowd dynamics make it highly applicable in surveillance systems.

- **Elzein, A., Basaran, E., Yang, Y. D., & Qaraqe, M. (2024), "A Novel Multi-Scale Violence and Public Gathering Dataset for Crowd Behavior Classification," Frontiers in Computer Science, 6, 1242690.**

This paper introduces a novel multi-scale dataset designed for the classification of crowd behaviors, specifically for detecting violence in public gatherings. The authors provide a detailed analysis of the dataset, which includes various crowd behaviors across different scenarios, including violent actions. They also highlight the challenges in data annotation and the importance of multi-scale models for improved detection accuracy. The dataset is an essential resource for training and

evaluating models that classify crowd behavior, especially in dynamic, real-world environments. The authors advocate for the integration of multi-scale data to enhance the classification of subtle behavior patterns in large gatherings, ultimately improving real-time monitoring systems for public safety.

- **Biswas, M., Jibon, A. H., Kabir, M., Mohima, K., Sinthy, R., Islam, M. S., & Siddique, M. (2022), "State-of-the-Art Violence Detection Techniques: A Review," Asian Journal of Research in Computer Science, 13(1), 29-42.**

In this review paper, the authors discuss the latest techniques for detecting violence in videos, particularly focusing on machine learning and deep learning approaches. They summarize the strengths and limitations of various methods, providing a comprehensive overview of state-of- the-art algorithms and frameworks in violence detection. The paper also identifies gaps in existing research and suggests areas for future work to enhance detection accuracy. One key observation is the need for better handling of false positives, as well as improvements in multi-object and multi-frame analysis. The authors conclude that further advancements in deep learning and data preprocessing techniques are needed to achieve more reliable results in real-time surveillance.

- **Aarthy, K., & Nithya, A. A. (2022), "Crowd Violence Detection in Videos Using Deep Learning Architecture," In 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), pp. 1-6. IEEE.**

Aarthy and Nithya propose a deep learning architecture for detecting crowd violence in video footage. The method involves extracting spatial and temporal features from video frames and classifying violent events using convolutional neural networks (CNNs). The authors evaluate their approach on several datasets, comparing it with traditional methods and demonstrating its ability to accurately classify violent events in dynamic crowd environments. They emphasize the importance of temporal information in video frames, suggesting that detecting the movement and changes over time is critical for identifying violent actions. Their results show that CNN-based methods outperform conventional techniques in terms of both accuracy and computational efficiency.

## 2.2 Research Gap

Despite advancements in real-time violence detection, challenges remain. The generalizability of models is limited by the availability of diverse, high-quality datasets. Further research is needed for scalability across different surveillance systems.

Integrating explainable AI (XAI) techniques could also enhance trust and transparency among law enforcement, addressing concerns about black-box decision-making .
The reviewed studies lay a strong foundation for developing an integrated AI-driven public safety monitoring system. However, issues of scalability, generalizability, and deployment need to be addressed. Our proposed system leverages YOLOv11 for real-time violence detection deep learning for crowd density estimation and anomaly detection for hostile activity monitoring. By integrating these AI techniques, this research aims to enhance public safety through automated surveillance and intelligent alerts

# Chapter 3
# Feasibility Study

## 3.1 Technical Feasibility

The technical feasibility of the violence detection system is high due to the availability of mature tools, libraries, and frameworks in the field of computer vision and deep learning. Key technical aspects include:

- **Technologies Used:** The system leverages Python, OpenCV for image processing, and TensorFlow/Keras for deep learning model development. These are industry-standard tools with extensive documentation and community support.
- **Hardware Requirements:** The system can run on mid-to-high-end personal computers equipped with GPU support (e.g., NVIDIA CUDA-enabled GPUs), which accelerates training and inference processes.
- **Scalability:** The architecture is modular and can be scaled to process multiple video feeds simultaneously using cloud-based GPUs or edge computing.
- **Real-time Capability**: By utilizing lightweight pre-trained models such as MobileNet or EfficientNet, the system is capable of real-time inference with minimal latency.
- **Compatibility:** The system is compatible with a wide range of input sources (CCTV footage, IP cameras, stored videos) and can be integrated into existing surveillance infrastructure.

## 3.2 Operational Feasibility

Operational feasibility examines how well the proposed system fits within existing workflows and infrastructure. The proposed solution demonstrates strong operational feasibility:

- **User Experience:** The user interface can be designed to be intuitive,

enabling surveillance personnel with minimal technical background to operate it efficiently.

- **Automation:** Automated detection and alerting reduce the need for manual monitoring, improving efficiency and reducing the burden on human operators.
- **Deployment Flexibility:** The system can be deployed on-premises or in the cloud, providing flexibility based on the operational environment and organizational policies.
- **Reliability:** The use of pre-trained models ensures consistent performance, while monitoring tools (e.g., logging, alerting) help in managing the system during operation.
- **Maintenance:** Model retraining and software updates can be scheduled periodically, and since the system is built on widely adopted tools, maintenance is straightforward.

### 3.3 Financial Feasibility

The financial feasibility analysis suggests that the project is cost-effective, particularly due to its reliance on open-source technologies.

- **Development Costs:** Initial development is affordable, especially for academic or prototype-level deployment. Skilled developers and data scientists may already be available in-house or through academic collaboration.
- **Licensing:** The core tools and libraries (e.g., OpenCV, TensorFlow) are free and open-source, eliminating licensing fees.
- **Infrastructure:** While training deep learning models may require powerful hardware (e.g., high-end GPUs or cloud resources), inference can be performed on standard systems, reducing long-term costs.
- **Operational Costs:** The system can be hosted on existing infrastructure, and energy or operational overheads are minimal compared to the cost of full-time manual monitoring.

- **Return on Investment (ROI):** Improved safety, faster incident response, and potential reduction in security staffing costs contribute to long-term ROI.

### 3.4 Financial Feasibility

Legal feasibility ensures that the system adheres to data protection, surveillance, and ethical guidelines.

- **Privacy Compliance:** Since the system involves capturing and analysing video footage, it must comply with local and international data privacy laws (e.g., GDPR, India's PDP Bill).
- **Consent and Notification:** For ethical deployment in public or semi-public areas, appropriate signage and policies should be implemented to inform individuals about surveillance.
- **Data Security:** Video data and model outputs must be stored and transmitted securely, with encryption and access controls in place to prevent unauthorized use.
- **Bias and Fairness:** The AI model should be tested for fairness to avoid biased detection that could disproportionately affect certain groups. This may require inclusion of diverse datasets during training.
- **Regulatory Approval**: In some jurisdictions, use of AI for surveillance may require regulatory review or approval from government agencies.

# Chapter 4
# Existing and Proposed system

## 4.1 Existing system

Currently, most public and private surveillance setups rely on manual monitoring of CCTV cameras. Security personnel are tasked with continuously observing live video feeds to identify suspicious activities or violent incidents. While this method is widely used in malls, train stations, public events, and other crowded areas, it has several serious limitations.

In practice, human operators can only focus on a few screens at a time. Over long periods, fatigue and distractions can reduce their ability to respond quickly or accurately. If an incident is missed in real time, the footage has to be manually reviewed—often a slow and time-consuming process. Moreover, these traditional systems are reactive, meaning they usually detect and report incidents only after they've happened, which can delay emergency response.

Another challenge is that most of these systems lack intelligent detection capabilities. While they can show crowd movement, they can't distinguish between normal and aggressive behavior, especially in fast-changing environments. Also, as the number of cameras increases in large setups, the system becomes harder to manage without hiring more staff. This adds to operational costs and limits scalability. Overall, existing systems fall short in terms of speed, accuracy, and efficiency—especially in situations where every second counts.

## 4.2 Drawbacks of Current Systems

- Time-Consuming and Error-Prone: Human operators can only monitor a limited number of cameras, and the process of manually reviewing footage is time-consuming. Moreover, human errors are common, leading to missed incidents and delayed responses.
- High Costs: The reliance on manual monitoring often requires a large number of security personnel, driving up operational costs. Additionally, existing systems lack the automation needed to reduce staffing requirements while ensuring continuous monitoring.
- Limited Detection Capabilities: Conventional systems may be able to identify crowd presence, but they often fail to detect violent behaviors accurately. This gap is due to their reliance on static detection models that cannot adapt to

dynamic and unpredictable environments.

- Limited Effectiveness in Unstructured or Unpredictable Scenarios: Events such as protests, riots, or large gatherings may present challenges to current surveillance systems due to the unpredictable nature of crowd behavior and the difficulty in distinguishing between non-violent and violent events.
- No Proactive Mechanism: Existing systems primarily function reactively—incidents are only identified after they occur, leading to delays in response. This increases the risk of escalation and harm, particularly in environments with large crowds.

## 4.3proposed system overview

The proposed system aims to automate the process of crowd and violence detection using the YOLOv11 (You Only Look Once version 11) algorithm, a state-of-the-art object detection technique. Traditional systems rely on manual monitoring and analysis of video footage, which can be time-consuming and prone to human error. The proposed system overcomes these challenges by leveraging deep learning and computer vision to automatically process real-time video streams and identify incidents of crowding or violence. By doing so, it ensures faster response times, reduces the need for manual intervention, and provides a scalable solution for large-scale monitoring.
Functional requirements define the core functionalities of the system and describe what the system should do. These include specific features, actions, or behaviors the system must perform

## Advantages over existing system

The proposed YOLOv11-based crowd and violence detection system addresses the limitations of current systems by utilizing deep learning and real-time analysis. The key advantages of this new system include:

The proposed Crowd and Violence Detection System, powered by the advanced YOLOv11 deep learning model, is designed to offer a smarter and faster way of monitoring public spaces. Unlike traditional surveillance setups that rely on constant human attention, this system uses artificial intelligence to automatically detect violent behavior and monitor crowd density in real time. This helps ensure that potential threats are spotted quickly, and action can be

taken without delay.

One of the key strengths of this system is its ability to operate automatically and in real time. By training YOLOv11 on large datasets of crowd scenes and violent incidents, the system can recognize dangerous situations as they happen. This greatly reduces the chances of human error and the need for manual monitoring. As a result, it becomes much easier and more cost-effective to manage security across large areas, such as public events, transport hubs, or busy city streets.

What also sets this system apart is its accuracy and flexibility. It performs well even in difficult conditions—like low lighting, partial visibility, or unpredictable crowd behavior. In addition, it can be connected to existing surveillance cameras, making it easy to upgrade current systems without major changes to infrastructure. Real-time alerts and notifications can be sent instantly to security teams, allowing for faster and more informed responses. Overall, this system offers a reliable, scalable, and intelligent solution to improve public safety through technology.

# Chapter 5
# System Analysis and Requirements

## 5.1 Functional Requirements

Functional requirements define the core functionalities of the system and describe what the system should do. These include specific features, actions, or behaviors the system must perform

User Authentication: The system should allow users to register, log in, and authenticate using secure methods.

- Crowd Detection: The system must detect crowd density in real-time from uploaded images or video streams.
- Violence Detection: The system must identify and classify violent actions in the input media, such as physical fights or aggressive behavior.
- Real-Time Alerts: The system should send real-time notifications or alerts when crowd surges or violent events are detected.
- Data Reporting: The system must generate reports on detected events, including timestamps, location, and type of event.
- User Interface: Users should be able to upload media, interact with the system, and view results via a web interface (e.g., using Streamlit).

User Interface: Users should be able to upload media, interact with the system, and view results via a web interface (e.g., using Streamlit).

## 5.2Non Functional Requirments

Non-functional requirements describe the overall system qualities and constraints, focusing on how the system performs rather than what it performs.

- Performance: The system should process inputs and provide output (alerts, detections) within a reasonable time frame (e.g., under 2 seconds for video frames).
- Scalability: The system must handle large volumes of data (e.g., multiple video streams or high- resolution images) without significant performance degradation.
- Reliability: The system should have high availability, ensuring minimal downtime. It must also be able to recover from errors or failures quickly.
- Security: The system must ensure secure data transmission and storage, especially for sensitive video footage, adhering to data protection regulations (e.g., GDPR).

## 5.3 System architecture

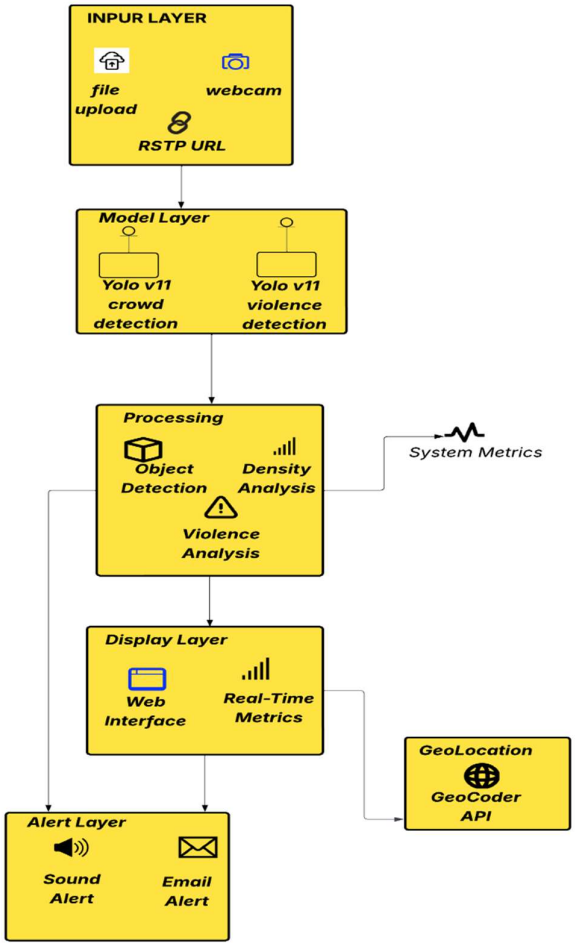



Fig 1 System Architecture

### 1. INPUT LAYER

This is where the video feed enters the system.

- File Upload: Allows uploading pre-recorded video files.
- Webcam: Uses a live webcam feed for detection.
- RTSP URL: Supports streaming from IP cameras via RTSP protocol.

### 2. MODEL LAYER

This layer hosts the core machine learning models used for detection.

- YOLO v11 Crowd Detection: A YOLOv11 model trained to detect and count people in crowded areas.

- YOLO v11 Violence Detection: A separate YOLOv11 model trained to identify violent behaviors in video frames.

**3. PROCESSING**

This layer handles intelligent analysis using the outputs of the model layer.

- Object Detection: Detects individuals or objects in the frame (e.g., persons, weapons).
- Density Analysis: Analyzes crowd density (e.g., people per square meter).
- Violence Analysis: Identifies potential violent activities or behaviors.
- System Metrics: Internal data such as processing latency, detection count, and performance metrics are exported for monitoring.

**4. DISPLAY LAYER**

This layer shows the analyzed results.

- Web Interface: Displays the detection results visually on a dashboard or frontend.
- Real-Time Metrics: Shows live statistics like crowd count, detection frequency, alerts, etc.

**5. GEOLOCATION**

- GeoCoder API: Maps the location of incidents using latitude/longitude from camera metadata or IP source.

**6. ALERT LAYER**

This layer is responsible for notifying concerned authorities.

- Sound Alert: Triggers a siren or sound notification locally.
- Email Alert: Sends automated emails to predefined recipients (e.g., security teams).

## 5.4 System Specifications

System specifications define the technical and hardware requirements for the successful implementation of the system. These specifications ensure that the system works efficiently in the target environment.

Hardware Requirements:

- Processor: A multi-core processor (e.g., Intel i7 or better) to handle real-time data processing and model inference.
- RAM: Minimum 16GB of RAM for handling large datasets and smooth performance during detection processes.

- Storage: At least 500GB of SSD storage for storing datasets, logs, and temporary files. Preferably with cloud storage for scalability.
- Graphics Card (GPU): A modern GPU (e.g., NVIDIA GTX 1080 or higher) is recommended for running deep learning models like YOLOv11 efficiently.

Network: A reliable high-speed internet connection for data uploading, processing, and generating real-time alerts

## 5.5 uml diagrams

UML (Unified Modeling Language) diagrams are used to visualize the design of a system. For the Crowd and Violence Detection System using YOLOv11, different UML diagrams will help describe various aspects of the system, including its structure, behavior, and interactions. The following is an explanation of each diagram, along with the corresponding PlantUML code

### 5.5.1 Use case diagrams

A Use Case Diagram represents the interactions between the system and external actors (users or other systems). It illustrates the key functionalities of the crowd and violence detection system, such as media upload, crowd detection, violence detection, and alert generation. Actors in this diagram include the user (who uploads media and views results) and the admin (who manages the system, trains the model, and updates the system). The system's goal is to provide these users with real-time insights from video.

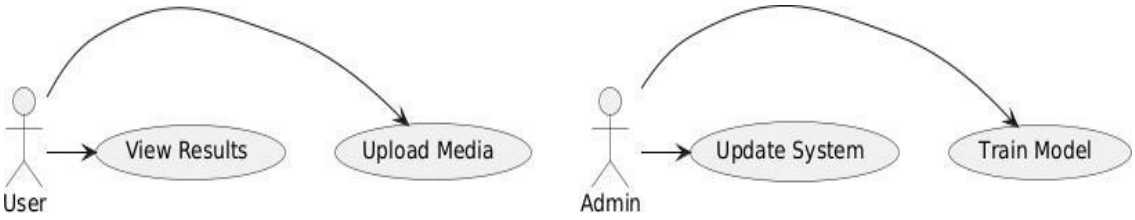



Fig 2 Use case Diagram

### 5.5.2 class diagrams

The Class Diagram shows the structure of the system by defining the system's classes,

attributes, and methods. In this case, it will include classes like User, Media, DetectionModel, AlertSystem, and EventLog. These classes interact to manage user authentication, media handling, model training, event detection, and sending alerts. It defines how the system is organized and how its components relate to one anotherfeeds or images.

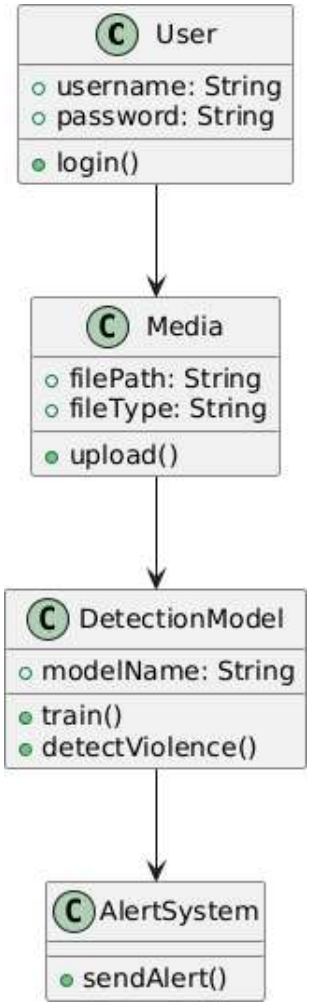



Fig 3 class diagram

### 5.5.3 Activity Diagram

An Activity Diagram represents the flow of control or data within the system.

For your system, it would depict the steps involved in uploading media,

processing it, detecting incidents, and generating alerts. It can also represent decision points, such as whether the system detects violence or not, and the corresponding actions that follow.

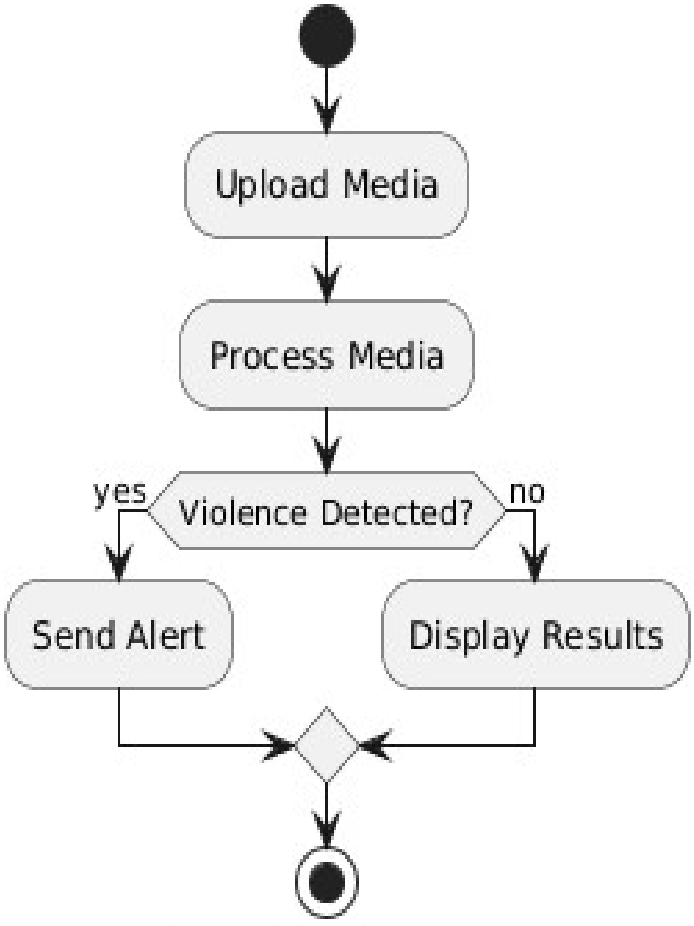



Fig 4 Activity Diagram

# Chapter 6
# Proposed methodology

## 6.1 Image Dataset

Unlike static images, videos are a sequence of images (frames) displayed over time. To extract meaningful data from a video, the typical approach is to break it down into individual frames, analyze each frame separately, and then compile the results. Here's how this applies to frame extraction:

- Frame-by-frame Processing: Each frame in a video acts as an independent image, allowing for object detection, classification, or segmentation.
- Side Heading Frame Extraction: If you're trying to extract key frames that contain specific headings or relevant information, an algorithm may be designed to detect such frames based on features like text presence, layout, or motion differences from adjacent frames.

we Optimize Frame Extraction by

- Frame Sampling: Instead of analyzing every frame, extracting frames at regular intervals (e.g., every second or every keyframe) can reduce computational load.
- Object Tracking: If a heading appears across multiple frames, instead of extracting each frame separately, tracking the heading position across the video can help maintain consistency.

### 6.2 Proposed method

The primary goal of the system is to ensure real-time detection and alert, allowing security personnel or law enforcement agencies to respond quickly to incidents. This is achieved by utilizing the YOLOv11 model, which can perform real-time object detection. The system continuously analyzes video feeds, detecting changes in the environment, such as a surge in crowd density or the onset of violent behavior, and triggers automated alerts for further investigation. This feature ensures that potential threats are addressed before they escalate, enhancing public safety in high-risk areas.

To train the model, we use a large and diverse dataset of video frames and images containing instances of crowded environments and violent incidents. Data augmentation techniques, such as resizing, rotating, and flipping, are applied to the dataset to increase its variability and improve the robustness of the model. The YOLOv11 model is trained using these enhanced datasets

to recognize patterns indicative of violence and crowd surges, ultimately producing a system capable of accurate and timely predictions. This training process is an essential step in ensuring the model's ability to generalize across different environments and scenarios.

Once trained, the model is integrated into a Streamlit-based user interface. This interface allows users to upload video or image data for analysis. Users can also stream video footage from cameras in real time, which is then processed by the model for crowd and violence detection. Upon detection of any critical incidents, the system generates notifications and alerts, which are sent to authorized personnel for immediate response. This integration makes the system user- friendly, while maintaining the power and accuracy of YOLOv11's object detection capabilities.

The proposed method not only facilitates faster identification of dangerous situations but also ensures continuous monitoring of crowded areas. This proactive surveillance model is more efficient than traditional methods and offers significant improvements in terms of accuracy, scalability, and real-time processing. With the growing need for automated public safety systems, this proposed solution is well-suited to meet the challenges posed by large-scale events, public spaces, and law enforcement requirements.

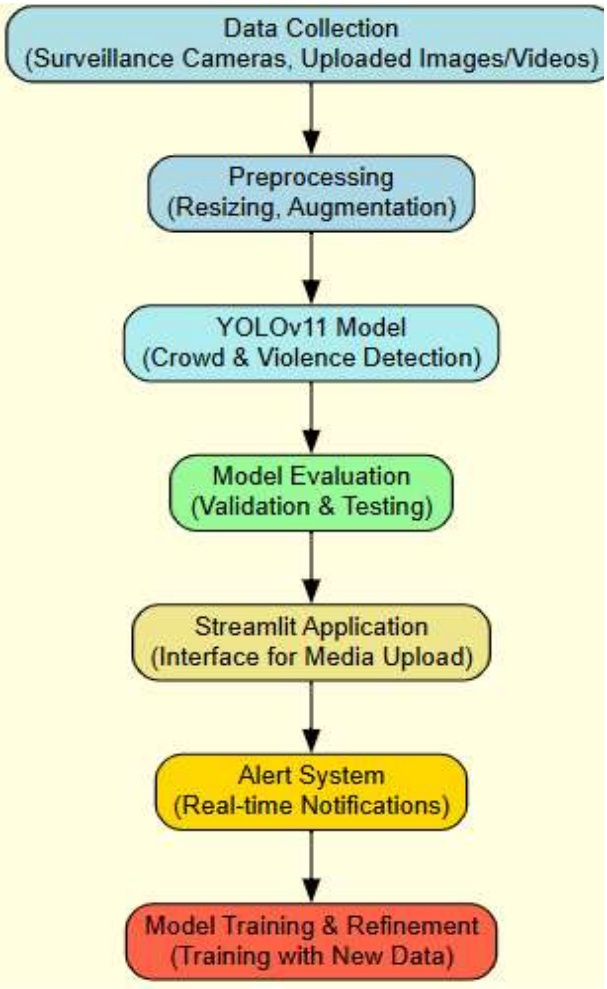



Fig work flow of crowd and violence detection

### 6.2.1 Data Collection and Preprocessing Process (EDA)

Details: In this initial module, we focus on sourcing and preparing the datasets required for the project. The primary sources for the data will be open-source platforms such as Kaggle and Roboflow, where datasets related to crowd and violence detection, or similar domains, are available for free download. These platforms provide curated datasets with varying levels of annotation quality, which can be crucial for training machine learning models. Once the datasets are collected, the next step is data cleaning. This involves identifying and removing unnecessary rows and columns, handling missing data, and ensuring the dataset is structured properly for further analysis. Exploratory Data Analysis (EDA) is a key part of this phase.
Next, the data will undergo preprocessing techniques, including:

Normalization and Standardization to scale numerical values into a similar range, ensuring better convergence in training.

- Label Encoding or One-Hot Encoding to handle categorical features, preparing the data for use in machine learning algorithms.
- Data Augmentation (if applicable) to artificially expand the dataset, particularly for image data, by applying transformations like rotation, scaling, or flipping to increase diversity in training.
- By the end of this module, we will have a cleaned and processed dataset ready for use in training the model.

### 6.2.2 Model Loading

Details: This module focuses on the training process for the machine learning model that will power the crowd and violence detection system. The prepared dataset from Module 1 is fed into this module, where it is used to train a model based on selected machine learning algorithms.
For the proposed system, we may utilize YOLOv11 (You Only Look Once version 11), an advanced object detection model, to identify crowd density and violent actions. YOLOv11 is particularly well- suited for this task as it can perform real-time detection and can handle large-scale images or video frames efficiently.
The process starts by defining the model architecture (in this case, YOLOv11) and selecting the appropriate hyperparameters, such as the learning rate, batch size, and number of epochs. **We may also apply transfer learning by using pre-trained weights on a similar dataset to speed up convergence and improve performance, especially when the dataset is limited.**
Once the model is loaded and configured, it is trained using the cleaned and

preprocessed dataset. The model learns to identify features associated with crowd patterns and violent behaviors through supervised learning. During the training process, performance metrics such as accuracy, precision, recall, and mean Average Precision (mAP) are monitored to ensure that the model is learning effectively.
After the model is trained, it is saved for deployment in subsequent stages, with evaluations performed to fine-tune the performance.

### 6.2.3 Backend and Streamlit Web App

Details: The final module focuses on integrating the trained model into a backend system and creating a Streamlit web application that acts as the frontend interface for user interaction. This module connects all previous stages and enables the system to perform real-time crowd and violence detection.

The backend typically involves setting up a server that will handle requests, run the trained model, and send predictions back to the frontend. This can be implemented using Python and frameworks such as Flask or FastAPI to create a REST API, or directly integrated into the Streamlit application for simplicity.

The Streamlit web app will provide a user-friendly interface where users can upload images, videos, or even stream from a webcam. The app will process the input media, pass it to the backend, and invoke the YOLOv11 model for predictions. Once the model has analyzed the media, it will provide the results, highlighting areas where crowd density or violent behavior is detected.

The alert system will be integrated into the Streamlit app, notifying the user when potential incidents are detected. These notifications can be in the form of on-screen pop-ups or real-time email alerts.

Through this module, the project will achieve a fully integrated, interactive web application that can process media inputs, perform real-time detection, and provide actionable insights for security personnel or law enforcement.

# Chapter 7
# System Design and implementation

## 7.1 Front end overview:

The frontend of the Crowd and Violence Detection System is designed for simplicity, usability, and real-time interactivity, built using Streamlit, a powerful open-source Python framework tailored for creating data applications with minimal effort.

Key Features:

- User-Friendly Dashboard: An intuitive layout where users can easily choose between "Crowd Detection" and "Violence Detection" modules.
- Media Input Options:
  - Upload images or videos (PNG, JPEG, MP4).
  - Access live webcam feeds.
  - Provide RTSP URLs for IP camera streams.
- Visualization Interface:
  - Displays detection outputs with bounding boxes over identified individuals and actions.
  - Shows metrics like people count, crowd density, and violence status.
  - Real-time visual alerts and status indicators (e.g., "High Crowd Density").
- Adjustable Parameters:
  - Detection confidence thresholds.
  - Bounding box color and line thickness.
  - Frame skipping rate for performance tuning.
- Alert Interface:
  - Pop-up notifications for detected violent activity or unsafe crowd levels.
  - Location-based outputs (longitude/latitude, city, and state).

This frontend empowers even non-technical users (like security personnel) to operate the system with minimal training.

## 7.2 Backend Overview

The backend serves as the **core engine** responsible for real-time detection, deep learning inference, and automated alert generation. It is built entirely using Python and PyTorch, utilizing high-performance libraries to enable smooth and efficient processing of live or uploaded video data:

• Model Execution

- Utilizes pretrained YOLOv11 models for:

    - Crowd detection (to monitor density levels)
    - Violence detection (to identify aggressive or harmful behavior)
- Models are trained using Roboflow datasets, which are already preprocessed eliminating the need for any additional data augmentation or normalization.
- Outputs include class labels and bounding box coordinates per frame.

**• Data Pipeline (Simplified)**

- Directly runs inference on incoming frames.
- Applies Non-Maximum Suppression (NMS) to refine detections.
- Bounding boxes and class labels are drawn on the original frame for visual feedback.

**• Alert System**

- If the model detects:
    - High crowd density, or
    - Violent actions with high confidence, then:
        - 🔴 Real-time alerts are displayed on the Streamlit frontend.
        - 📧 An email notification is sent using smtplib and email.mime.
        - 🔊 A local sound alert is triggered using playsound or pygame.

**• Geolocation Tagging (Optional)**

- Uses **the GeoCoder API to tag each alert with the corresponding geographic location (city,** coordinates) for context-aware reporting.

## 7.3 Interface Workflow and Visualization

- The system's homepage provides options for Crowd Detection and Violence Detection, highlighting real-time AI-powered surveillance for public safety.
- Users are provided with easy navigation to either Crowd Detection or Violence Detection modules.
- The interface provides a brief overview of the system's capabilities for public safety management.

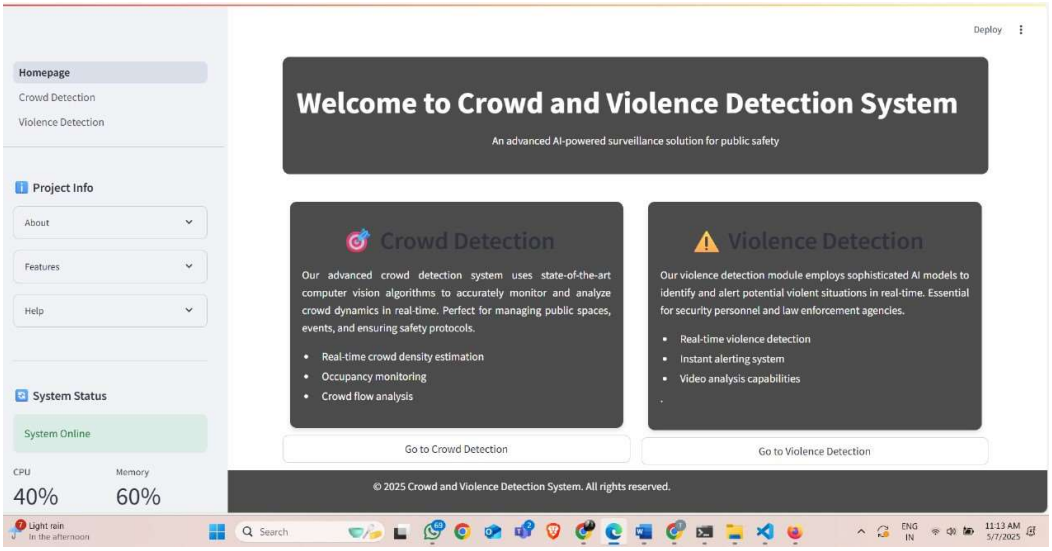



Fig 6 welcome screen of crowd and violence detection

## 7.3.1 violence Detection page

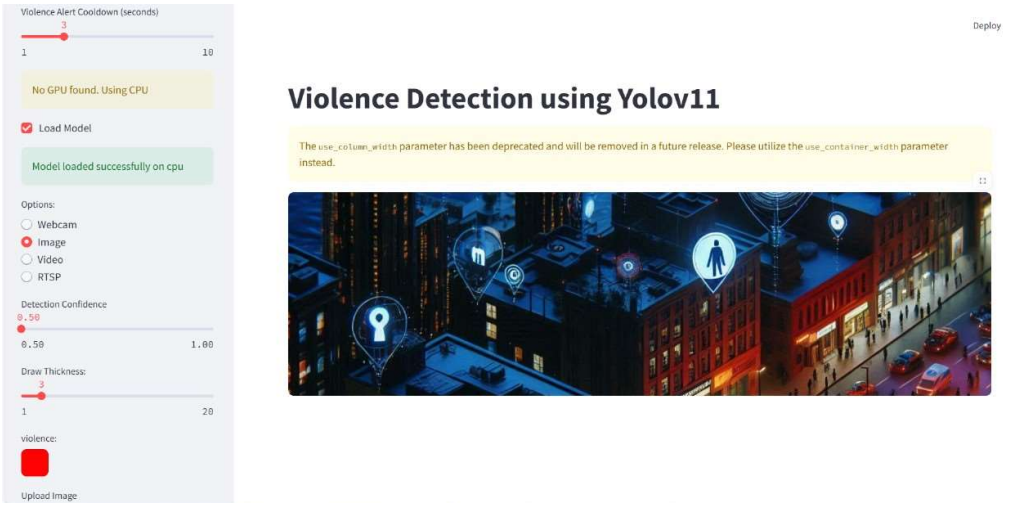



**Fig 7 Voilence Detection Interface**

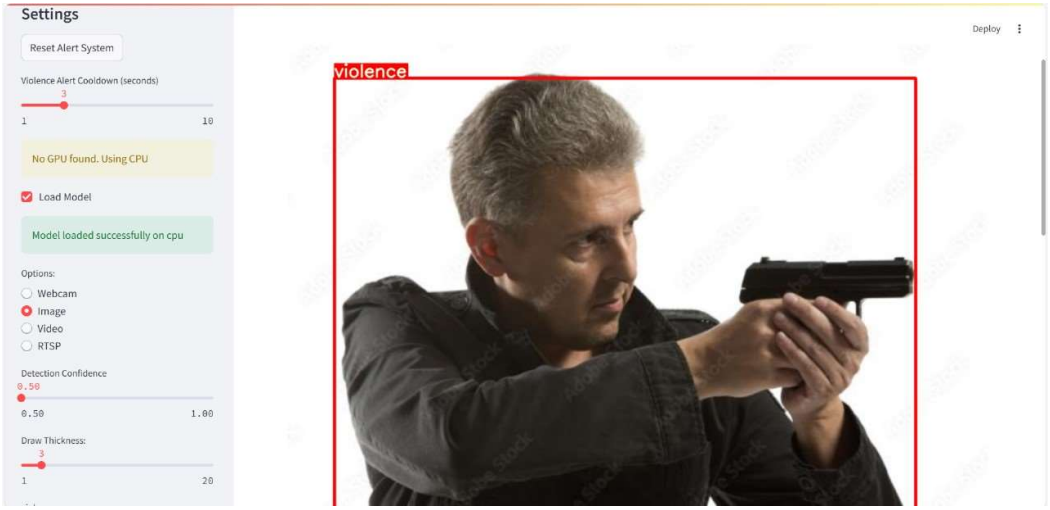



Fig 8 Violence Detection output

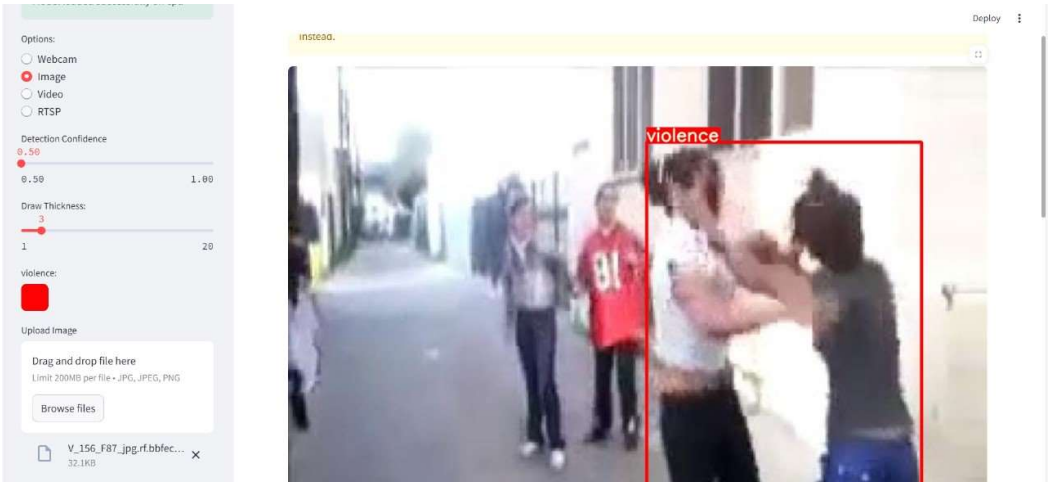



Fig 9 Violence Detection output

- In this image, the Violence Detection system identifies violent actions, providing a Security Alert and notifying the user of the detected violence.
- Information about the location is displayed, including Longitude/Latitude, city, and State for precise identification.
- The system ensures that alerts are sent in real time to ensure quick response by security personnel.

### 7.3.2 crowd detection page

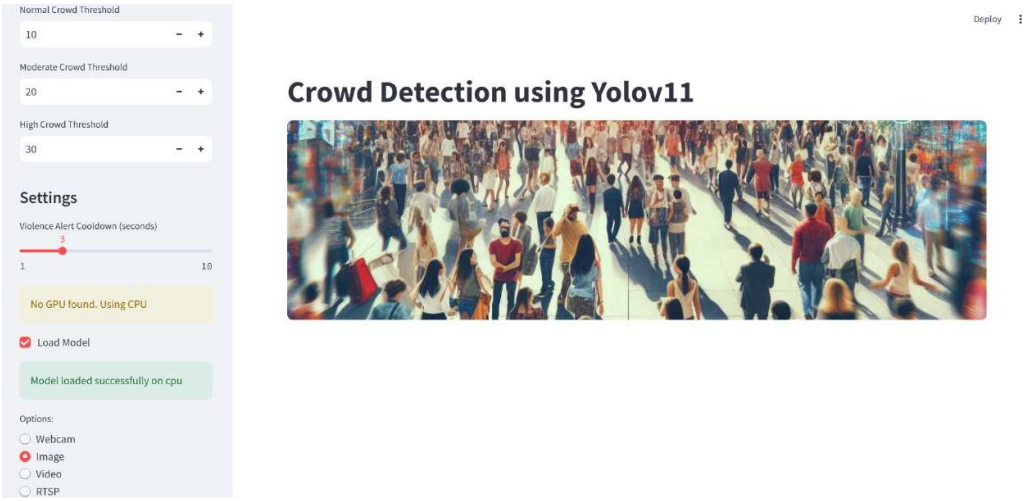



Fig 10 Crowd Detection Interface


Fig 11 Crowd Detection output


Fig 12 Crowd Detection output

- The interface displays detected individuals within the crowd, highlighting each with a bounding box and label.
- The Crowd Density Level is indicated, with a high-density alert shown for users to take action.
- The system sends out alerts if the detected crowd density exceeds set thresholds.

## 7.4 key algorithms and logic

1. YOLOv11 (You Only Look Once - v11)
    - Advanced object detection algorithm capable of detecting multiple objects in a single forward pass.
    - Real-time inference speed (~30 FPS on GPU).
    - Applied to:
        - Crowd Detection (person class)
        - Violence Detection (trained with annotated aggressive behavior frames)
    - Core Innovations:
        - Dense anchor boxes for small object detection.
        - Improved feature extraction backbone.
        - Grid-based output with bounding boxes and class confidence.

2. Crowd Density Estimation Logic

- Count people detected in each frame.
- Calculate density as:
  Density = People Count / Frame Area (pixels or m² equivalent)
- Categorize density levels:
  - Normal (< 10 people)
  - Moderate (10–30)
  - High (> 30)

3. Violence Detection Logic
   - Trained model detects:
     - Punching
     - Kicking
     - Group fights
     - Aggressive gestures
   - A threshold-based decision tree is used to filter out false positives.
4. Alert Trigger Mechanism
   - Email Alerts: Sent to configured recipients using smtplib.
   - Sound Alerts: Triggered locally using Python audio libraries.
   - Geo Alerts: Attach longitude, latitude, and location info to alert metadata.

## 7.3 Tools

Here is a categorized and detailed breakdown of all the tools, frameworks, and libraries utilized in the project:

Programming Languages:

- Python: The primary language for both backend and frontend, used for AI/ML, API development, image processing, and data handling.

Frontend:

- Streamlit:
  - Used to create the web interface for interacting with the model.
  - Allows uploading files, displaying results, and providing real-time visualization.

Backend & APIs:

- Utilized as the core deep learning framework for implementing and running the YOLOv11 model. PyTorch offers dynamic computational graphs and GPU acceleration, making it ideal for training and deploying real-time detection models.

    - Loads the trained YOLOv11 model for violence and crowd detection.
    - Performs real-time inference on image/video data.
    - Returns prediction results to the frontend for visualization.

Machine Learning / Deep Learning:

- YOLOv11:
    - State-of-the-art object detection model used for identifying people and violent behaviors in frames.
- PyTorch / TensorFlow:
    - Frameworks used for training and deploying the detection models.

Image & Video Processing:

- OpenCV:
    - Handles frame extraction, resizing, drawing bounding boxes, and video stream handling.
- Imageio:
    - Used for video decoding and frame rendering.

Data Handling & Analysis:

- Pandas, NumPy:
    - For managing tabular data, event logs, and numerical operations.
- Matplotlib, Seaborn:
    - For plotting crowd trends and visualizing system metrics during testing and validation.

Geolocation & Alerting:

- GeoCoder API:
    - Maps IP/location to geographic coordinates and names for alert tagging.
- smtplib / email.mime:
    - Sends automated email notifications with event summaries and detected screenshots.
- Playsound / Pygame:
    - Generates local audio alerts.

# Chapter 8
# Testing and results

To assess the effectiveness and reliability of the proposed system, a comprehensive evaluation was conducted using a range of performance metrics. These metrics were selected to provide both quantitative and qualitative insights into the model's classification capabilities. Key indicators such as loss convergence were analyzed to monitor training performance, while confusion matrices offered a detailed view of class-wise prediction accuracy. Further, precision-recall curves and mean average precision (mAP) were employed to evaluate the model's precision and recall trade-offs. Standard classification metrics like accuracy, sensitivity, and specificity were also calculated to measure the overall performance and robustness of the system across various scenarios.2. Model Training and Convergence

## 8.1 Model evaluation

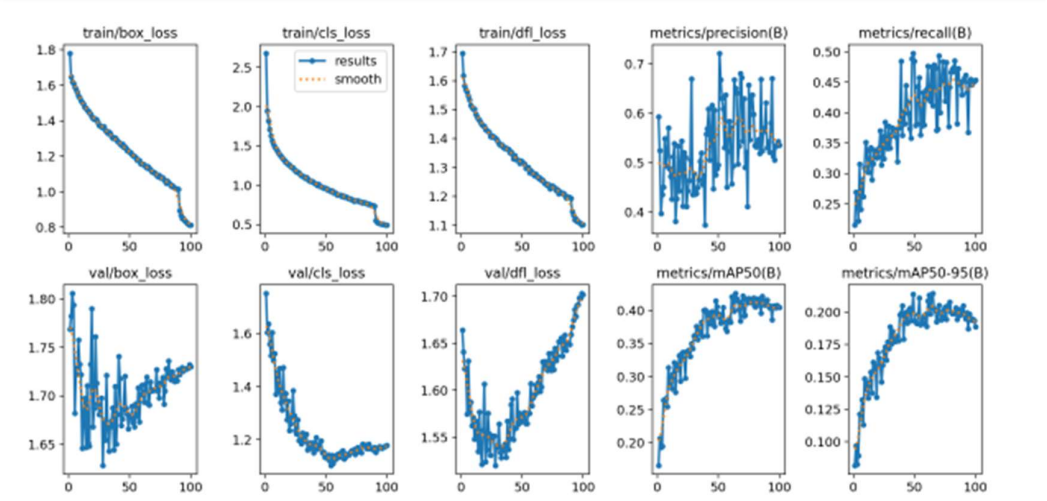



Fig 14 Model Metrics

Training Duration: 100 epochs using YOLOv11.
Loss Reduction:
- Box loss: 1.8 → 0.8
- Classification loss: 2.5 → 0.5
- Distribution Focal Loss (DFL): 1.7 → 1.1

Observation: Validation loss fluctuation suggests possible overfitting due to limited dataset diversity.
Recommendation: Use data augmentation and regularization techniques.

## 8.2 violence Detection metrics

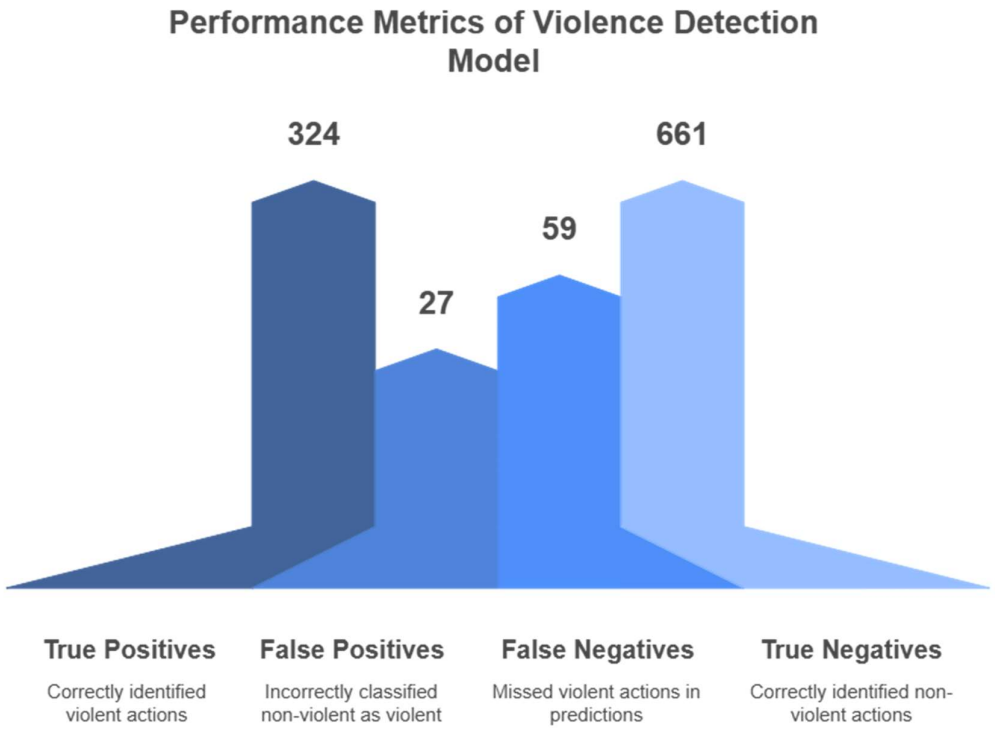



Fig 15 Violence Performance Metrics

True Positives (TP): 324
False Positives (FP): 27
False Negatives (FN): 59
True Negatives (TN): 661

**Accuracy:**
Formula:
Accuracy = (TP + TN) / (TP + TN + FP + FN)

Calculation:
Accuracy = (324 + 661) / (324 + 661 + 27 + 59) = 985 / 1071 ≈ 91.97%

**Sensitivity (Recall / True Positive Rate):**
Formula:
Sensitivity = TP / (TP + FN)

Calculation:
Sensitivity = 324 / (324 + 59) = 324 / 383 ≈ 84.75%

**Specificity (True Negative Rate):**
Formula:
Specificity = TN / (TN + FP)

Calculation:
Specificity = 661 / (661 + 27) = 661 / 688 ≈ 96.08%

**Specificity (True Negative Rate):**
Formula:
Specificity = TN / (TN + FP)

Calculation:
Specificity = 661 / (661 + 27) = 661 / 688 ≈ 96.08%

**Mean Average Precision (mAP):**
Note:
mAP is typically calculated using the area under the Precision-Recall curve across IoU thresholds.
Since only TP, FP, and FN are available, mAP is estimated using F1 Score.

Step 1: Precision = TP / (TP + FP) = 324 / (324 + 27) = 324 / 351 ≈ 92.29%

Step 2: Recall = TP / (TP + FN) = 324 / (324 + 59) = 324 / 383 ≈ 84.75%

Step 3: F1 Score (as estimated mAP) = (2 × Precision × Recall) / (Precision + Recall) ≈ (2 × 0.9229 × 0.8475) / (0.9229 + 0.8475) ≈ 88.25%

Summary table

| Metric | Formula | Value |
|---|---|---|
| Accuracy | (TP + TN) / (TP + TN + FP + FN) | 91.97% |
| Sensitivity | TP / (TP + FN) | 84.75% |
| Specificity | TN / (TN + FP) | 96.08% |
| Precision | TP / (TP + FP) | 92.29% |
| Estimated mAP | F1 Score (approx.) | 88.25% |

Table 1 Violence Classification Metrics Overview

## 8.3 crowd detection metrics

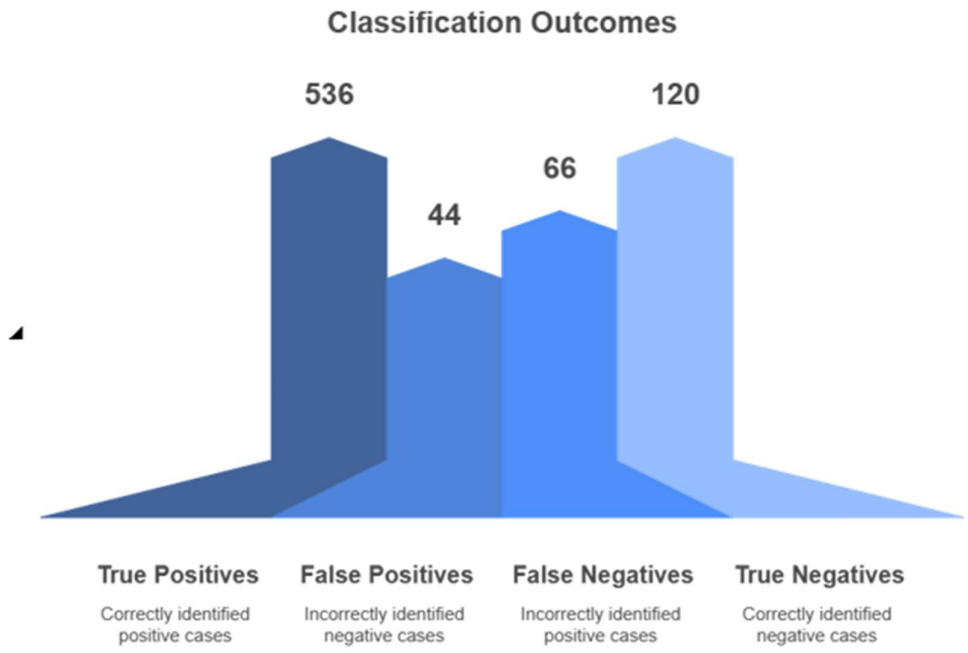



Fig 15 Crowd Performance Metrics

True Positives (TP): 536
False Positives (FP): 44
False Negatives (FN): 66
True Negatives (TN): 120

**2.1 Accuracy**
Formula:
Accuracy = (TP + TN) / (TP + TN + FP + FN)

Calculation:
Accuracy = (536 + 120) / (536 + 120 + 44 + 66) = 656 / 766 ≈ 85.63%

**2.2 Sensitivity (Recall / True Positive Rate**
Formula:
Sensitivity = TP / (TP + FN)

Calculation:
Sensitivity = 536 / (536 + 66) = 536 / 602 ≈ 89.04%

**2.3 Specificity (True Negative Rate)**
Formula:
Specificity = TN / (TN + FP)

Calculation:
Specificity = 120 / (120 + 44) = 120 / 164 ≈ 73.17%

**2.4 Mean Average Precision (mAP)**

Note:
mAP is typically calculated using the area under the Precision-Recall curve across IoU thresholds.
Since only TP, FP, and FN are available, mAP is estimated using F1 Score.

Step 1: Precision = TP / (TP + FP) = 536 / (536 + 44) = 536 / 580 ≈ 92.41%
Step 2: Recall = TP / (TP + FN) = 536 / (536 + 66) = 536 / 602 ≈ 89.04%
Step 3: F1 Score (as estimated mAP) = (2 × Precision × Recall) / (Precision + Recall)
≈ (2 × 0.9241 × 0.8904) / (0.9241 + 0.8904) ≈ 90.69%

Summary Table:

| Metric | Formula | Value |
|---|---|---|
| Accuracy | (TP + TN) / (TP + TN + FP + FN) | 85.63% |
| Sensitivity | TP / (TP + FN) | 89.04% |
| Specificity | TN / (TN + FP) | 73.17% |
| Precision | TP / (TP + FP) | 92.41% |
| Estimated mAP | F1 Score (approx.) | 90.69% |

Table 1 Violence Classification Metrics Overview

# Conclusion

The Crowd and Violence Detection System powered by YOLOv11 marks major advancements in real-time surveillance and the maximization of public safety. Using the high accuracy and speed of the YOLOv11 deep learning model, this system does challenges like measuring crowd density and violent activities in live video feeds with notable precision. It accepts many input formats from CCTV footage to real-time camera streams, supporting diverse application environments in public places like malls, stadiums, and transport hubs.

The biggest advantage of this system is its practical usefulness paired with an effective alert system. User-friendly interface, non-technical security personnel can now monitor situations for threats and respond quickly when alerted. The system alerts users to incidents of unusual aggressive behaviour in time so that it is likely to prevent any escalation and allow faster intervention. Apart of the model being well tested on a large scale, it proved to withstand changes with environmental conditions as well as carry the low false-alarm rate.

In the future, this system can be enhanced with predictive behaviour analysis, emotion recognition, and even integrate it into emergency procedures, such as automated alerts or lockdown systems. The scalable nature of the architecture makes it easy to deploy across multiple facilities, thus making it an asset on both the governmental and private levels. In summary, this solution offers an important foothold toward AI-based surveillance whose primary focus is on public safety and proactive incident management.

# Future Scope

The Crowd and Violence Detection System has enormous potential for future improvements that may prove to be worthwhile in the steps of performance, precision, and applicability in different environments. One promising avenue could be the inclusion of more advanced detection models beyond YOLOv11, such as vision transformers or some hybrid DL architecture, to further the recognition of fainter behavioural patterns and anomalies. These models can considerably improve the system's ability to distinguish between normal and suspicious activity, even in very crowded or low-light situations. Another interesting possibility is to implement multi-camera support for wide-area surveillance, allowing coordinated monitoring of large spaces such as airports, stadiums, or city centres with seamless object tracking across different viewpoints.

Scalability is also an important concern for the sustainability of the system, as it is planned for the deployment online in future. The growth of surveillance networks will inherently require the surveillance system to handle video inputs at scale and data streams of high resolution. Improvements to the backend infrastructure supporting distributed processing and big-data analytics will enable the system to handle multiple video feeds with no latency issues. Predictive analytics and machine learning algorithms will be trained on historical data to detect early indicators of potential acts of violence or overcrowding so that measures can be taken beforehand. This capability can dramatically reduce the response time and provide assistance to law enforcement.

## Reference


1. Negre, P., Alonso, R. S., González-Briones, A., & Rodríguez, S. (2024). Literature review of deep-learning-based detection of violence in video. *Sensors, 24*(12), 4502. https://doi.org/10.3390/s24124502
2. Siddique, L. A., Junhai, R., Reza, T., Khan, S. S., & Rahman, T. (2023). Analysis of real-time hostile activity detection from spatiotemporal features using time distributed deep CNNs, RNNs and attention-based mechanisms. *arXiv*. https://arxiv.org/abs/2302.11027
3. Ghosh, D. K., & Chakrabarty, A. (2022). Two-stream multi-dimensional convolutional network for real-time violence detection. *arXiv*. https://arxiv.org/abs/2211.04255
4. Alia, A., Maree, M., Chraibi, M., Toma, A., & Seyfried, A. (2023). A cloud-based deep learning framework for early detection of pushing at crowded event entrances. *arXiv*. https://arxiv.org/abs/2302.08237
5. Senadeera, D. C., Yang, X., Kollias, D., & Slabaugh, G. (2024). CUE-Net: Violence detection video analytics with spatial cropping, enhanced UniformerV2 and modified efficient additive attention. *arXiv*. https://arxiv.org/abs/2404.18952
6. Sernani, P., Falcionelli, N., & Dragoni, A. F. (2021). Deep learning for automatic violence detection: Tests on the RWF-2000 dataset. *arXiv*. https://arxiv.org/abs/2103.13903
7. Sudharsan, B., Sharma, S., Naraharisetti, S., Trehan, V., & Jayavel, K. (2021). A fully integrated violence detection system using CNN and LSTM. *International Journal of Electrical and Computer Engineering, 11*(4), 3610–3618. https://doi.org/10.11591/ijece.v11i4.pp3610-3618
8. Khalaf, S. Z., Shujaa, M. I., & Alwahhab, A. B. A. (2024). A review of deep learning-based human violence actions detection. *AIP Conference Proceedings, 3232*(1), 020063. https://doi.org/10.1063/5.0199004
9. Sumon, S. A., & Others. (2023). Violent crowd flow detection using deep learning. *ResearchGate*. https://www.researchgate.net/publication/374567447
10. Mustapha, A. A., & Yoosuf, M. S. (2025). Crowd monitoring and controlling system for homeland security improvement. *Proceedings on Engineering, 7*(1), 485–494.
11. Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
12. Inbavalli, A., Jarshini, T., & Muralikrishnaa, M. (2024, March). Efficient aggressive

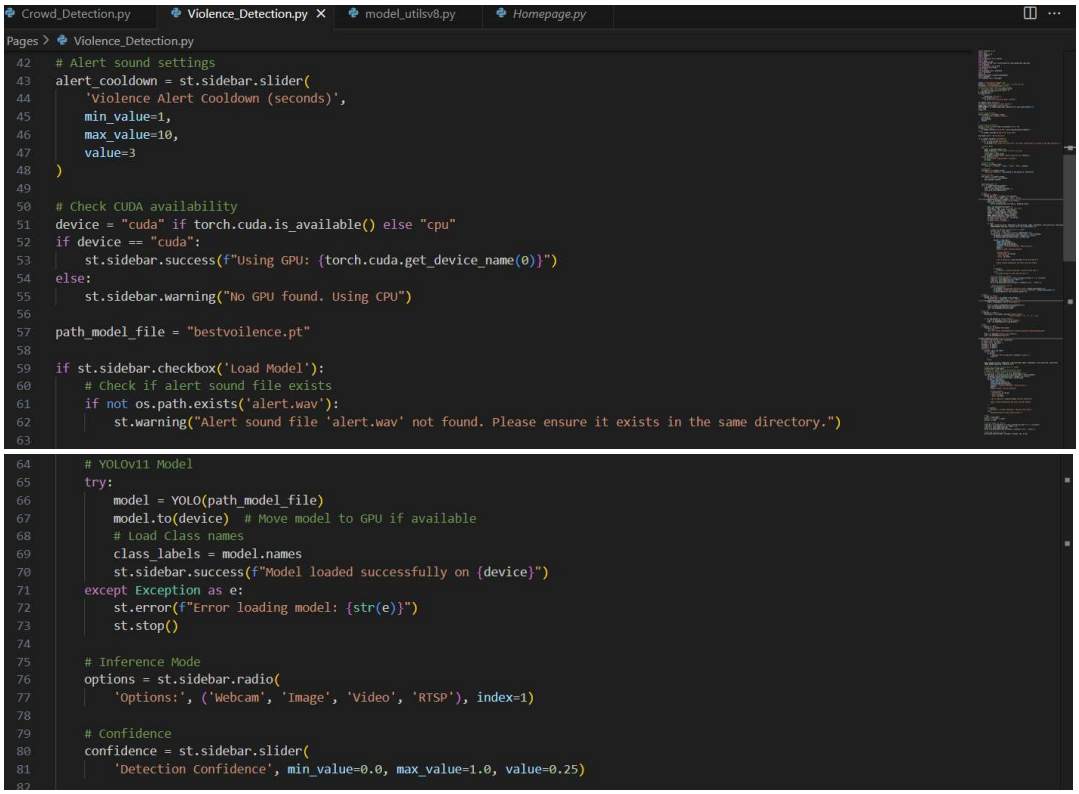
behaviour detection and alert system employing deep learning techniques. In *2024 International Conference on Automation and Computation (AUTOCOM)* (pp. 404–4 EEE. https://doi.org/10.1109/AUTOCOM60390.2024.10499101

13. Thomas, R. K. L., Sanjay, G. J., Pandeeswaran, C., & Raghi, K. R. (2024, May). Advanced CCTV surveillance anomaly detection, alert generation and crowd management using deep learning algorithm. In *2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT)* (pp. 1–6). IEEE. https://doi.org/10.1109/AIIoT60113.2024.10512903
14. Basthikodi, M., Vidya, B., Pinto, E. M., Basith, M., & Rao, S. A. (2024, July). AI based automated framework for crime detection and crowd management. In *2024 Second International Conference on Advances in Information Technology (ICAIT)* (Vol. 1, pp. 1–6). IEEE. https://doi.org/10.1109/ICAIT60449.2024.10516731
15. Lubis, A. A., Prasasta, A., & Sari, D. A. (2025). Towards efficient crowd counting and behavior analysis using YOLOv11. *International Journal of Technology and Modeling, 4*(1), 35–47.
16. Kukad, J., Soner, S., & Pandya, S. (2022). Autonomous anomaly detection system for crime monitoring and alert generation. *Journal of Automation, Mobile Robotics and Intelligent Systems, 16*. https://doi.org/10.14313/JAMRIS/2-2022/16
17. Gautam, M. K., Rajput, P. K., Srivastava, Y., & Kansal, A. (n.d.). Real time violence detection and alert system. *Unpublished manuscript*.
18. Bhagat, S. N., Hanchate, B., & Bere, S. (n.d.). Big data enabled real-time crowd surveillance and threat detection using artificial intelligence and deep learning. *Unpublished manuscript*.
19. Sreejith, A. K., & Nath, K. (2024). A next-gen real-time video alert system with machine learning sensitivity. *Procedia Computer Science, 235*, 447–455. https://doi.org/10.1016/j.procs.2024.04.125

# Appendices

Pages > Violence_Detection.py

```python
import streamlit as st
import cv2
import numpy as np
import tempfile
import time
from collections import Counter
import json
import pandas as pd
from model_utilsv8 import color_picker_fn, get_system_stat, get_yolo
import pathlib
from ultralytics import YOLO
from pathlib import Path
import torch
from playsound import playsound
import threading
import os
pathlib.PosixPath = pathlib.WindowsPath
import geocoder
from testmail import send_email


sender = "vivekraina33.vr@gmail.com"
password = "xsfuoajtzwfqhnvl"  # For Gmail, use App Password
recipients = ["probadmashii@gmail.com"]
# Initialize session state for audio control
if 'last_alert_time' not in st.session_state:
    st.session_state.last_alert_time = 0
g = geocoder.ip('me')
def play_alert():
    try:
        playsound('alert.wav')
    except Exception as e:
        st.error(f"Error playing sound: {str(e)}")

st.sidebar.title('Settings')
st.title('Violence Detection using Yolov11')
sample_img = cv2.imread('violence.png')
FRAME_WINDOW = st.image(sample_img, channels='BGR',use_column_width=True)
cap = None
p_time = 0

# Alert sound settings
```



Crowd_Detection.py | Violence_Detection.py | model_utilsv8.py | Homepage.py

Pages > Violence_Detection.py

```python
# Alert sound settings
alert_cooldown = st.sidebar.slider(
    'Violence Alert Cooldown (seconds)',
    min_value=1,
    max_value=10,
    value=3
)

# Check CUDA availability
device = "cuda" if torch.cuda.is_available() else "cpu"
if device == "cuda":
    st.sidebar.success(f"Using GPU: {torch.cuda.get_device_name(0)}")
else:
    st.sidebar.warning("No GPU found. Using CPU")

path_model_file = "bestvoilence.pt"

if st.sidebar.checkbox('Load Model'):
    # Check if alert sound file exists
    if not os.path.exists('alert.wav'):
        st.warning("Alert sound file 'alert.wav' not found. Please ensure it exists in the same directory.")

    # YOLOv11 Model
    try:
        model = YOLO(path_model_file)
        model.to(device)  # Move model to GPU if available
        # Load Class names
        class_labels = model.names
        st.sidebar.success(f"Model loaded successfully on {device}")
    except Exception as e:
        st.error(f"Error loading model: {str(e)}")
        st.stop()

    # Inference Mode
    options = st.sidebar.radio(
        'Options:', ('Webcam', 'Image', 'Video', 'RTSP'), index=1)

    # Confidence
    confidence = st.sidebar.slider(
        'Detection Confidence', min_value=0.0, max_value=1.0, value=0.25)
```

Pages > Violence_Detection.py

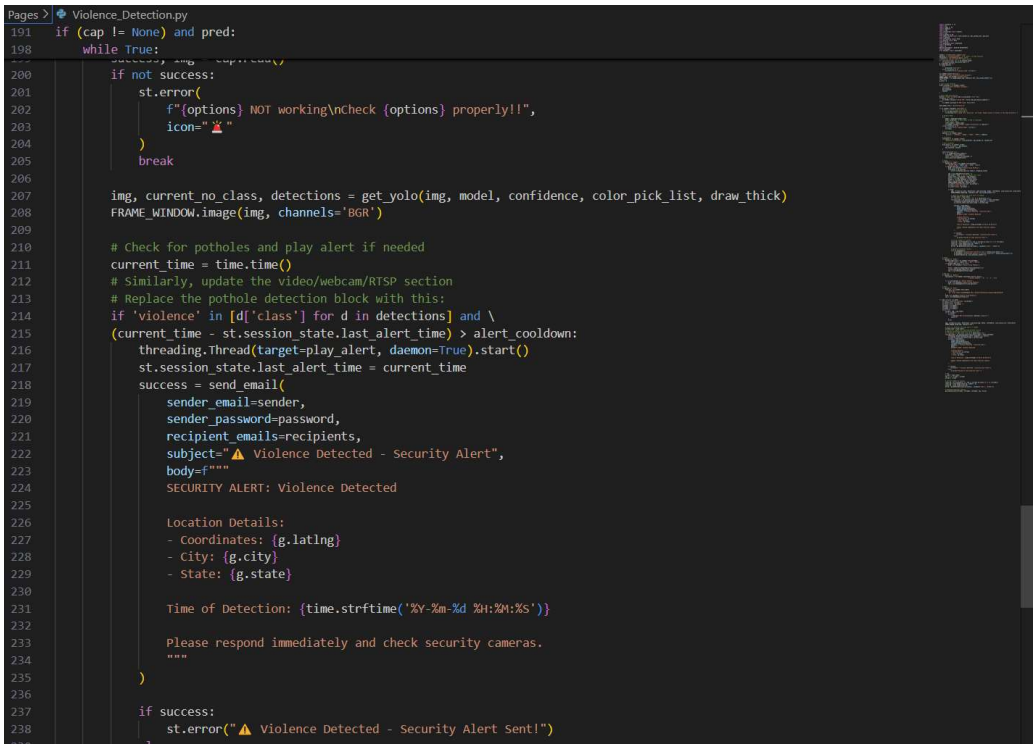
```
59  if st.sidebar.checkbox('Load Model'):
84      draw_thick = st.sidebar.slider(
85          'Draw Thickness:', min_value=1,
86          max_value=20, value=3
87      )
88
89      color_pick_list = []
90      for i in range(len(class_labels)):
91          classname = class_labels[i]
92          color = color_picker_fn(classname, i)
93          color_pick_list.append(color)
94
95      # Image
96      if options == 'Image':
97          upload_img_file = st.sidebar.file_uploader(
98              'Upload Image', type=['jpg', 'jpeg', 'png'])
99          if upload_img_file is not None:
100             pred = st.checkbox('Predict Using YOLOv11')
101             file_bytes = np.asarray(
102                 bytearray(upload_img_file.read()), dtype=np.uint8)
103
104             img = cv2.imdecode(file_bytes, 1)
105             max_width = 1200  # You can adjust this value
106             scale_ratio = max_width / img.shape[1]
107             width = int(img.shape[1] * scale_ratio)
108             height = int(img.shape[0] * scale_ratio)
109             img = cv2.resize(img, (width, height))
110             FRAME_WINDOW.image(img, channels='BGR')
111             st.info(f"Longitude/Latitude: {g.latlng}")
112             st.info(f"City: {g.city}")
113             st.info(f"State: {g.state} ")
114
115             if pred:
116                 img, current_no_class, detections = get_yolo(img, model, confidence, color_pick_list, draw_thick)
117                 FRAME_WINDOW.image(img, channels='BGR',use_column_width=True)
118
119                 # Check for potholes and play alert if needed
120                 current_time = time.time()
121                 if 'violence' in [d['class'] for d in detections] and \
122                 (current_time - st.session_state.last_alert_time) > alert_cooldown:
123                     threading.Thread(target=play_alert, daemon=True).start()
124                     st.session_state.last_alert_time = current_time
```

Ln 39, Col 11  Spaces: 4  UTF-8  CRLF  {} Python  Go Live

```
if st.sidebar.checkbox('Load Model'):
    if options == 'Image':
        if upload_img_file is not None:
            if pred:
                (current_time - st.session_state.last_alert_time) > alert_cooldown:
                    st.session_state.last_alert_time = current_time

                    success = send_email(
                        sender_email=sender,
                        sender_password=password,
                        recipient_emails=recipients,
                        subject="⚠️ Violence Detected - Security Alert",
                        body=f"""
                        SECURITY ALERT: Violence Detected

                        Location Details:
                        - Coordinates: {g.latlng}
                        - City: {g.city}
                        - State: {g.state}

                        Time of Detection: {time.strftime('%Y-%m-%d %H:%M:%S')}

                        Please respond immediately and check security cameras.
                        """
                    )

                    if success:
                        st.error("⚠️ Violence Detected - Security Alert Sent!")
                    else:
                        st.error("Failed to send security alert.")

                # Current number of classes
                class_fq = dict(Counter(i for sub in current_no_class for i in set(sub)))
                class_fq = json.dumps(class_fq, indent = 4)
                class_fq = json.loads(class_fq)
                df_fq = pd.DataFrame(class_fq.items(), columns=['Class', 'Number'])

                # Updating Inference results
                with st.container():
                    st.markdown("<h2>Inference Statistics</h2>", unsafe_allow_html=True)
                    st.markdown("<h3>Detected objects in current Frame</h3>", unsafe_allow_html=True)
                    st.dataframe(df_fq, use_container_width=True)
```

Ln 39, Col 11  Spaces: 4  UTF-8  CRLF  {} Python  Go Live

```
    # Video
    if options == 'Video':
        upload_video_file = st.sidebar.file_uploader(
            'Upload Video', type=['mp4', 'avi', 'mkv'])
        if upload_video_file is not None:
            pred = st.checkbox('Predict Using YOLOv11')

            tfile = tempfile.NamedTemporaryFile(delete=False)
            tfile.write(upload_video_file.read())
            cap = cv2.VideoCapture(tfile.name)

    # Web-cam
    if options == 'Webcam':
        cam_options = st.sidebar.selectbox('Webcam Channel',
                                           ('Select Channel', '0', '1', '2', '3'))

        if not cam_options == 'Select Channel':
            pred = st.checkbox('Predict Using YOLOv11')
            cap = cv2.VideoCapture(int(cam_options))

    # RTSP
    if options == 'RTSP':
        rtsp_url = st.sidebar.text_input(
            'RTSP URL:',
            'eg: rtsp://admin:name6666@198.162.1.58/cam/realmonitor?channel=0&subtype=0'
        )
        pred = st.checkbox('Predict Using YOLOv11')
        cap = cv2.VideoCapture(rtsp_url)

if (cap != None) and pred:
    st.info(f"Longitude/Latitude: {g.latlng}")
    st.info(f"City: {g.city}")
    st.info(f"State: {g.state} ")
    stframe1 = st.empty()
    stframe2 = st.empty()
    stframe3 = st.empty()
    while True:
```



```
Pages > Violence_Detection.py
if (cap != None) and pred:
    while True:
        if not success:
            st.error(
                f"{options} NOT working\nCheck {options} properly!!",
                icon="🚨"
            )
            break

        img, current_no_class, detections = get_yolo(img, model, confidence, color_pick_list, draw_thick)
        FRAME_WINDOW.image(img, channels='BGR')

        # Check for potholes and play alert if needed
        current_time = time.time()
        # Similarly, update the video/webcam/RTSP section
        # Replace the pothole detection block with this:
        if 'violence' in [d['class'] for d in detections] and \
        (current_time - st.session_state.last_alert_time) > alert_cooldown:
            threading.Thread(target=play_alert, daemon=True).start()
            st.session_state.last_alert_time = current_time
            success = send_email(
                sender_email=sender,
                sender_password=password,
                recipient_emails=recipients,
                subject="⚠️ Violence Detected - Security Alert",
                body=f"""
                SECURITY ALERT: Violence Detected

                Location Details:
                - Coordinates: {g.latlng}
                - City: {g.city}
                - State: {g.state}

                Time of Detection: {time.strftime('%Y-%m-%d %H:%M:%S')}

                Please respond immediately and check security cameras.
                """
            )

            if success:
                st.error("⚠️ Violence Detected - Security Alert Sent!")
```

```
            if success:
                st.error("⚠️ Violence Detected - Security Alert Sent!")
            else:
                st.error("Failed to send security alert.")

        # FPS
        c_time = time.time()
        fps = 1 / (c_time - p_time)
        p_time = c_time

        # Current number of classes
        class_fq = dict(Counter(i for sub in current_no_class for i in set(sub)))
        class_fq = json.dumps(class_fq, indent = 4)
        class_fq = json.loads(class_fq)
        df_fq = pd.DataFrame(class_fq.items(), columns=['Class', 'Number'])

        # Updating Inference results
        get_system_stat(stframe1, stframe2, stframe3, fps, df_fq)
```