

VISVESVARAYA TECHNOLOGICAL UNIVERSITY BELAGAVI



Project Work Report on

INTRUDER DETECTION AND ALERTING SYSTEM WITH MACHINE LEARNING

Submitted by

Abhishek Gowda D.S	4BB21CS002
Chandan M.H	4BB21CS009
Chandrashekhar D	4BB21CS015
Darshana S	4BB21CS016

In partial fulfilment of the requirement for the award of the
Bachelor Degree In

Computer Science and Engineering

Under the Guidance of

Mr. Faiz Aman, B.E., M. Tech.

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Bahubali College of Engineering

Shrivaniabelagola-573 135

2024-25



BAHUBALI COLLEGE OF ENGINEERING

SHRAVANABELAGOLA – 573 135

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project Work entitled **“INTRUDER DETECTION AND ALERTING SYSTEM WITH MACHINE LEARNING”** is the work carried out by Bonafide students of Bahubali College of Engineering, **ABHISHEK GOWDA D. S (USN 4BB21CS002), CHANDAN M. H (USN 4BB21CS009), CHANDRASHEKHAR D (USN 4BB21CS015), DARSHANA S (USN 4BB21CS016)** in partial fulfillment of **VIII Semester** to award the Bachelor Degree in **Computer Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year **2024-25**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report and deposited in the Department Library. The Project Report has been approved as it satisfies all the academic requirements in respect of Project Work prescribed for the Bachelor of Engineering Degree.

Mr. Faiz Aman

Assistant Professor & Guide

Dept. of CS&E

Mrs. Kavitha C. R

Associate Professor & HOD

Dept. of CS&E

Dr. Sunil Kumar D

Principal

Name of the Examiners

Signature with Date

1.

2.

ACKNOWLEDGEMENT

Inspiration and guidance are valuable in all aspects of life, especially what is academic. “Experience is the best teacher,” is an old saying. The satisfaction and pleasure that accompany the gain of experience would be incomplete without mentioning the people who made it possible.

We express our heartfelt gratitude to the project guide **Mr. Faiz Aman, Assistant Professor, Department of Computer Science and Engineering**. He being our guide has taken keen interest in the progress of the project work phase by providing facilities and guidance. We are indebted to our guide for his inspiration, support and kindness showered throughout the course.

We are extremely thankful and grateful to our Co-ordinator **Mr. Faiz Aman, Assistant Professor, Department of Computer Science and Engineering**, for providing the support for making project work possible.

We express our profound sense of gratitude to **Mrs. Kavitha C. R, Associate Professor & HOD, Department of Computer Science and Engineering** for giving us the opportunity to pursue our interest in this project work successfully.

We express our special gratitude to our Principal **Dr. Sunil Kumar D**, for providing the resources and support during project work.

Nevertheless, we express heartfelt thanks towards our parents, friends and teaching and non-teaching staff of our college for their kind co-operation and encouragement which helped us during project work .

ABHISHEK GOWDA D. S

CHANDAN M. H

CHANDRASHEKHAR D

DARSHANA S

ABSTRACT

The "Intruder Detection and Alerting System Using Machine Learning" represents a cutting- edge security solution that redefines the landscape of surveillance and alerting in high- security environments such as corporate offices, residences, and restricted zones. By seamlessly integrating advanced machine learning algorithms with sophisticated computer vision techniques, this system offers a proactive approach to threat detection, empowering security personnel to respond effectively and in real time. At the heart of this innovative solution lies a robust facial recognition model, meticulously designed to differentiate between authorized and unauthorized individuals through a comprehensive network of cameras and webcams. Each detected individual is subjected to a meticulous evaluation against a secure database of authorized personnel. Should an unrecognized individual be detected, the system captures both photos and video footage, triggering an immediate high-priority alert. This alert is accompanied by loud beep sounds to ensure prompt attention, and it is enriched with snapshots of the detected individual, dispatched to security personnel or administrators via multiple channels, including an intuitive mobile app, SMS, or email, ensuring they have visual evidence at their fingertips.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	i
ABSTRACT	ii
TABLE OF CONTENTS	iii
LIST OF FIGURES	iv
CHAPTER 1: INTRODUCTION	
	1-5
1.1 Overview	2
1.2 Problem Statement	2
1.3 Objectives	3
1.4 Project Scope	3
1.5 Existing System	4
1.6 Proposed System	4
1.7 Organization of Project Report	5
CHAPTER 2: LITERATURE SURVEY	6-12
CHAPTER 3: SOFTWARE REQUIREMENTS SPECIFICATION	13-16
3.1 Functional Requirements	13
3.2 Non-Functional Requirements	14
3.3 Resource Requirements	15
3.4 Hardware Requirements	15
3.5 Software Requirements	16
CHAPTER 4: SYSTEM ANALYSIS AND DESIGN	17-29
4.1 System Design	17
4.1.1 Design Overview	17
4.1.2 System Architecture	17
4.1.3 Detailed Design	22

CHAPTER 5: IMPLEMENTATION	30-37
5.1 Implementation Requirements	30
5.2 Visual Studio Code	30
5.3 Programming Language used	31
5.4 Key features of Python	31
5.5 Tkinter GUI	31
5.6 OpenCV-Python Tool	31
5.7 Flask	32
5.8 Google Colab	32
5.9 Packages and Pseudocodes for Preprocessing Techniques	32-37
CHAPTER 6: TESTING	38-43
CHAPTER 7: RESULTS AND SNAPSHOTS	44
CONCLUSION	48
FUTURE ENHANCEMENT	49
REFERENCES	50
Journal Certificate	53

LIST OF FIGURES

Fig no	Name	Page No
4.1	System Architecture	19
4.2	Use Case Diagram	21
4.3	Diagram of System Operation	22
4.4	Level 0 Data Flow Diagram	22
4.5	Level 1 Data Flow Diagram	23
7.1	Registration Page	34
7.2	Login Page	34
7.3	Forgery Detection and Verification	35
7.4	Genuine Signature Verify	35
7.5	Forged Signature	36
7.6	View Accuracy and Loss Graph	36

LIST OF TABLES

Table no	Table Name	Page No
6.1	Integration testing table	30
6.2	Validation testing table	31

CHAPTER 1

INTRODUCTION

In an era marked by rapid technological advancements, the landscape of security monitoring is undergoing a significant transformation. Despite these developments, challenges persist in achieving proactive, real-time surveillance, especially in environments requiring constant vigilance and immediate response. Traditional surveillance systems, which predominantly depend on human oversight, often fall short in detecting critical incidents or responding swiftly to emerging threats. Therefore, it is essential to enhance these systems with intelligent, automated solutions to improve security and operational efficiency.

This project addresses these challenges by developing a cutting-edge Intruder Detection and Alerting System that utilizes machine learning to automate threat detection, monitor activities in real-time, and facilitate instant notifications. By enabling the seamless identification of unusual behaviors and providing immediate alerts with visual evidence via mobile platforms such as SMS and email, this system enhances the capabilities of conventional surveillance methods. Moreover, it systematically archives recordings of suspicious activities, facilitating quick access, and automatically classifies detected individuals as "known" or "unknown," incorporating this information into alerts sent to security personnel.

The foundation of this system is built upon sophisticated machine learning algorithms, including neural networks and advanced computer vision techniques, which ensure high accuracy in threat detection, facial recognition, and behavioral analysis. This innovative approach not only addresses current security challenges but also paves the way for future advancements in intelligent surveillance technologies. Traditional surveillance systems, which predominantly depend on human oversight, often fall short in detecting critical incidents or responding swiftly to emerging threats. Therefore, it is essential to enhance these systems with intelligent, automated solutions to improve security and operational efficiency. Through this project, we aim to redefine security system operations by minimizing human intervention, enhancing responsiveness, and providing a more effective and comprehensive security solution for diverse environments, including public spaces, corporate offices, and residential areas.

1.1 Overview

The project focuses on enhancing security monitoring by integrating machine learning techniques into real-time surveillance. Traditional security systems rely on human oversight, making them inefficient and slow in response. The proposed system automates threat detection, activity monitoring, and alerting mechanisms, ensuring immediate notifications and improved security. It archives suspicious activity records and classifies individuals as "known" or "unknown" based on facial recognition. Machine learning models, including neural networks and computer vision techniques, enable accurate threat detection, facial recognition, and behavioral analysis.

1.2 Problem Statement

The primary challenge addressed by this project is the absence of an intelligent, real-time surveillance system capable of automatically detecting and responding to intrusions while minimizing the reliance on continuous human monitoring. Existing surveillance solutions often fall short in delivering real-time insights, accurately identifying unusual behaviors, and providing timely notifications, resulting in inefficiencies and delayed responses to potential security threats. Key issues include:

- 1. Manual Monitoring:** Reliance on human operators to continuously observe camera feeds increases the risk of errors due to fatigue and oversight.
- 2. Delayed Response:** Existing systems lack real-time detection and alert mechanisms, leading to slower reactions during critical incidents.
- 3. Inadequate Threat Detection:** Difficulty in distinguishing normal from suspicious activities results in missed threats or frequent false alarms.
- 4. Limited Modern Features:** Absence of advanced technologies like facial recognition or behavioral analysis restricts the systems.

1.3 Objectives

- **Data Collection and Integration:** Aggregating data from multiple security inputs, including cameras, sensors, and environmental devices, to create a comprehensive dataset that captures both regular and suspicious activity.

- **Machine Learning Model Development:** Creating and training a machine learning model capable of distinguishing between normal and suspicious activity patterns. The model will adapt over time, continuously improving its accuracy and reducing false positives through adaptive learning.
- **Real-Time Monitoring and Alert System:** Implementing a real-time monitoring system that sends instant alerts through various channels (such as SMS, email, or mobile app) to notify users or security personnel of potential intrusions.
- **User Interface for Customization and Control:** Developing an intuitive interface that allows users to monitor activities, adjust settings, and review alerts, enhancing user control and flexibility.

1.4 Project Scope

The project aims to develop a real-time intruder detection and alerting system that utilizes machine learning algorithms for automated threat detection. The key aspects include:

- Real-time monitoring using high-resolution cameras and machine learning models.
- Facial recognition for distinguishing authorized and unauthorized individuals.
- Instant alert system that notifies security personnel through SMS, email, or mobile applications.
- Behavioral analysis to identify suspicious movements and actions.
- Cloud-based deployment for accessibility and scalability.

1.5 Existing System

Current surveillance solutions often rely on a combination of outdated technologies and manual processes for monitoring and threat detection, resulting in inefficiencies and vulnerabilities in security management. Machine learning algorithms for facial recognition and behavior analysis need significant processing power and memory. Although various tools are available for video surveillance, real-time analytics, and alert systems, they typically lack integration, which limits their use in dynamic and high-security or highly used in this environment.

- i. **Video Surveillance:** Conventional video cameras are extensively utilized for monitoring but often lack advanced capabilities such as facial recognition and automated motion detection. Many systems still require manual oversight, leading to delayed responses to potential security threats.
- ii. **Real-Time Analytics:** Some existing surveillance systems do offer real-time analytics, including basic facial recognition and behavioral analysis. However, these technologies frequently struggle with accuracy, especially in fluctuating lighting conditions or crowded settings, and they often do not integrate seamlessly with current camera infrastructure.

1.6 Proposed System

The proposed system aims to significantly enhance security monitoring and incident response by leveraging advanced machine learning techniques. Its core components are designed to work collaboratively to provide a comprehensive surveillance solution:

1. Data Acquisition:

Video Input: The system employs high-resolution cameras integrated with sophisticated computer vision algorithms. These cameras continuously capture and analyze live video feeds, enabling the detection of suspicious activities and potential intruders in real time.

2. Processing Mechanism:

Real-Time Analytics: The system utilizes state-of-the-art machine learning models for facial recognition and behavioral analysis. This allows for the rapid identification of potential security threats, ensuring that security personnel can respond swiftly to any incidents.

3. Machine Learning Algorithms:

Convolutional Neural Networks (CNNs): The proposed system employs CNNs to enhance accuracy in image processing tasks, particularly in object and face recognition. This deep learning approach allows the system to effectively differentiate between authorized individuals and potential intruders.

4. User Interface:

Centralized Monitoring Dashboard: A user-friendly interface provides security personnel with the ability to monitor multiple camera feeds simultaneously. The dashboard facilitates real-time management of alerts, ensuring that critical information is easily accessible and actionable.

1.7 Organization of Project Report

The project report for the “Intruder Detection and Alerting System with Machine Learning” is organized to ensure comprehensive coverage and ease of navigation. It begins with a Title Page that includes the project title, any subtitle, authors, affiliations, and the date. The Abstract follows, providing a brief summary of the project, including its objectives, methods, results, and conclusions. Next, the Table of Contents lists all sections and subsections with page numbers, followed by a List of Figures and Tables for easy reference. The introduction section sets the stage by discussing the background and motivation for the project, the problem statement, objectives, scope, and structure of the report. The Literature Review provides an overview of existing methods, related work in Intruder Detection and Alerting System with Machine Learning methods. The Methodology section details the processes for data collection, preprocessing, feature extraction, and the selection and training of machine learning models. It also outlines the system architecture and the integration of various modules.

In the Implementation section, the report describes the development environment, including the software and tools used, and provides detailed explanations of the algorithms and code snippets. It also discusses the design of the user interface, highlighting its features and usability considerations. The Results and Analysis section presents performance metrics such as accuracy, precision, recall and F1-score, describes the experimental setup and interprets the results through graphs and tables. This section also includes a comparison with existing methods and case studies of verification results.

CHAPTER 2

LITERATURE SURVEY

2.1: Intrusion Detection Using Deep Learning.

Authors: Smith A., Li B., Zhao Y.

Published: January 2019

This review explores the application of deep learning techniques in intrusion detection systems (IDS), offering insights into their capability to identify complex cyber threats within large-scale networks. Traditional IDS, relying on rule-based or statistical methods, struggle with high-dimensional data and sophisticated, evolving attack methods like Advanced Persistent Threats (APTs). Deep learning approaches, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), show promise in overcoming these limitations by automatically learning hierarchical data representations and temporal patterns. The review discusses specific architectures, including CNNs for spatial pattern recognition in network traffic, and RNNs (especially LSTM variants) for analyzing sequences in time-series data, such as user behavior and network logs. The effectiveness of deep learning models hinges on the availability of large, labelled datasets—difficult to obtain in cybersecurity due to privacy concerns and the cost of manual labelling.

Hybrid models, combining CNNs and RNNs, are highlighted as particularly effective, achieving a balance between feature extraction and sequential analysis, which is crucial for detecting multi-stage attacks. The authors emphasize the role of Autoencoders for anomaly detection, using unsupervised learning to recognize deviations from normal network behavior, thus identifying zero-day threats. However, these models also come with significant challenges. High computational demands necessitate specialized hardware like GPUs, which can limit scalability, especially in resource-constrained environments. Additionally, the effectiveness of deep learning models hinges on the availability of large, labelled datasets—difficult to obtain in cybersecurity due to privacy concerns and the cost of manual labelling.

2.2: Anomaly Detection in Network Traffic Using Machine Learning.

Author: J. Patel, M. Kumar.

Published: March 2020

This paper presents a comparative analysis of machine learning techniques for anomaly detection in network traffic, focusing on their application in real-time intrusion detection systems (IDS). Anomaly detection offers a flexible approach for identifying potential threats by detecting deviations from established patterns in network behavior, which is particularly valuable for identifying novel or previously unseen attacks, such as zero-day vulnerabilities and insider threats. The authors implement two machine learning models: Support Vector Machines (SVM), known for its high accuracy in complex pattern classification, and k- Nearest Neighbors (k-NN), valued for its computational simplicity and speed. The study involves feature extraction from network traffic data, where parameters such as packet size, protocol, and connection duration are analyzed to create robust feature vectors. SVM, with its capacity to handle high-dimensional data, demonstrates superior accuracy in distinguishing between normal and anomalous traffic but requires extensive computational resources, making it challenging to deploy in high-throughput networks. Conversely, k-NN, with lower computational demands, achieves faster detection speeds, making it more suitable for real-time applications but at the cost of slightly reduced detection precision. To optimize these models, the authors employ dimensionality reduction techniques like Principal Component Analysis (PCA), which helps in enhancing processing efficiency without significantly impacting model performance.

Experimental results demonstrate that SVM achieves higher precision and recall, indicating its efficacy in detecting complex anomalies, while k-NN provides a faster, though marginally less accurate, alternative for time-sensitive applications. The paper concludes by suggesting a hybrid model that combines SVM's accuracy with k-NN's efficiency, potentially providing a balanced solution for environments requiring both precision and speed.

2.3 Intrusion Detection Using Autoencoders in High-Dimensional Data Spaces.

Author: Lopez M., Garcia B.

Published: April-2019.

This paper investigates the use of unsupervised learning, specifically autoencoders, for detecting anomalies in high-dimensional network data, providing an innovative approach to intrusion detection.

Autoencoders are designed to reconstruct input data by learning its most essential features, and deviations in this reconstruction process are flagged as potential intrusions. The model is particularly effective for identifying zero-day attacks, which are difficult to detect with traditional rule-based or supervised learning approaches due to the absence of prior labels.

This study demonstrates that autoencoders can efficiently handle complex, high-dimensional datasets, such as those found in large-scale network traffic, by compressing and reconstructing data representations. In their experimental setup, the authors evaluate the model's performance on multiple high-dimensional datasets, observing that it achieves a high detection rate with minimal false positives. However, they note that the accuracy and reliability of the autoencoder approach are heavily dependent on the quality and volume of the training data. The autoencoder requires extensive training with benign data to learn normal behavior patterns accurately. Without diverse and comprehensive datasets, there is a risk of misidentifying benign anomalies as threats, resulting in false positives.

2.4 Hybrid Models for detecting Insider Threats using Machine Learning.

Authors: Choi Y., Kim D.

Published: February 2021

This paper addresses the growing issue of insider threats within organizations by proposing a hybrid intrusion detection model that combines supervised and unsupervised machine learning techniques. Insider threats, which involve malicious or negligent activities by individuals within an organization, are particularly challenging to detect due to their subtle nature and the trusted access these individuals have to critical systems and sensitive data. The authors develop a model that integrates clustering algorithms, such as k-means, for identifying anomalies in behavior patterns, with Support Vector Machines (SVMs) for classifying identified behaviors as either normal or suspicious.

The hybrid model is designed to capture both behavioral anomalies (indicative of potential insider threats) and specific attack patterns associated with known insider threat signatures. Through unsupervised clustering, the model identifies deviations in user activity, such as unusual login times, access to sensitive files, or atypical resource usage. These detected anomalies are then processed through the SVM classifier, which has been trained on labelled insider threat data to confirm or deny malicious intent. The results show that this dual-stage approach improves detection accuracy and reduces false positives compared to traditional single-method models.

The authors conduct extensive testing on synthetic datasets and real-world insider threat datasets, demonstrating that the model achieves high accuracy in detecting insider attacks, especially those involving gradual behavioral changes that evade traditional rule-based systems.

Additionally, the paper discusses the importance of temporal data analysis in identifying insider threats that unfold over extended periods, such as data exfiltration or credential misuse, which may only appear anomalous over time. By incorporating time-series data into the model, the authors enhance its ability to detect long-term patterns that might indicate potential risks.

2.5 AI-Driven Intruder Detection Systems in Smart City Environments

Author: Silva R., Fernandez J.

Published: October 2021.

This paper explores the application of artificial intelligence (AI) in developing scalable, responsive intruder detection systems tailored for smart city environments. As smart cities integrate interconnected sensors, cameras, and IoT devices for urban monitoring, managing security threats across such complex infrastructures presents unique challenges. The proposed AI-driven system combines advanced computer vision techniques with machine learning algorithms, particularly Convolutional Neural Networks (CNNs) for object detection and Long Short-Term Memory (LSTM) networks for behavior analysis, to provide real-time detection and response to potential security threats. By utilizing a combination of fixed and mobile sensors across a city landscape, the system continuously analyses environmental and behavioral data to detect abnormal patterns indicative of intrusions or suspicious activities.

The paper details how CNNs are employed to process visual data from citywide surveillance cameras, detecting objects, faces, and potential intruders with high accuracy. These detections are then fed into LSTM models, which assess movement patterns and behaviors over time, distinguishing between typical urban activities and potential security risks. This multi-layered approach allows the system to detect complex intrusions, such as loitering near sensitive areas, unattended objects, or suspicious gathering activities, and to initiate alerts only for genuine threats, thus reducing false positives.

Challenges such as computational demand and data privacy concerns are addressed. The authors discuss edge computing solutions to offload processing from central servers, which reduces latency and enables real-time analysis directly on connected devices. Privacy protection measures, including data anonymization and selective data retention, are implemented to ensure compliance with regulations like GDPR, which is essential given the extensive surveillance involved in smart cities.

2.6 Real- Time Intruder Detection Surveillance using Computer Vision

Author: L. Smith, K. Wong.

Published: December 2021

This paper presents an innovative approach to real-time intruder detection in surveillance systems by integrating computer vision techniques with machine learning models to enhance detection accuracy and reduce response times. Traditional surveillance systems primarily rely on manual monitoring or basic motion sensors, which are often limited by high false alarm rates and reduced effectiveness in complex environments. This study introduces a dual-layered system, combining motion detection algorithms with Convolutional Neural Networks (CNNs) for facial recognition, to deliver a more robust detection mechanism. Motion detection is first used to identify and track movement within the surveillance area, enabling efficient preliminary filtering of potential intruders.

Once movement is detected, CNN-based facial recognition is employed to verify the identity of individuals, distinguishing authorized personnel from unknown or unauthorized intruders. The CNN model is trained on a diverse dataset of facial images, incorporating various lighting conditions and angles to increase recognition accuracy. Testing in controlled environments shows high precision in detecting and verifying individuals, significantly reducing the likelihood of false positives. However, challenges arise in dynamic real-world settings, where variations in lighting, background noise, and obstructions like furniture or walls can lead to reduced accuracy and increased false negatives.

To address these limitations, the authors propose several enhancements, including the incorporation of adaptive thresholding in the motion detection phase, allowing the system to adjust sensitivity based on environmental factors. Additionally, advanced image preprocessing techniques are suggested to improve facial recognition under challenging conditions, such as low light or partial occlusion. The paper further explores the integration of temporal data analysis, which could allow the system to account for suspicious behavioral patterns over time, enhancing its ability to detect persistent threats or intruders attempting to bypass security.

2.7 Secure Intrusion Detection with Homomorphic Encryption and Machine Learning

Author: Ahmed N., Youssef M.

Published: March 2022.

This paper introduces a novel approach to intrusion detection that prioritizes data privacy by employing homomorphic encryption combined with machine learning models, addressing a critical need in sectors such as healthcare, finance, and government where sensitive data is frequently processed. Traditional intrusion detection systems (IDS) analyse unencrypted network data to identify threats, which can expose sensitive information and create privacy concerns. In this approach, homomorphic encryption enables computations to be performed on encrypted data, allowing the intrusion detection process to proceed without compromising the confidentiality of the underlying information. The authors implement and evaluate the performance of Random Forest and Decision Tree classifiers in this encrypted environment, assessing their effectiveness in identifying malicious patterns while preserving privacy. Experiments reveal that the Random Forest model achieves high accuracy due to its ensemble structure, making it well-suited for detecting complex and subtle intrusions within encrypted data. However, this accuracy comes at the cost of significant computational resources, as homomorphic encryption and decryption processes are inherently resource-intensive. The Decision Tree model, while slightly less accurate, is noted for its computational efficiency, offering a feasible solution for real-time detection in environments with limited processing power, such as edge devices in IoT networks. The study's findings suggest that while both models are effective, their deployment should be carefully matched to the available resources and privacy requirements of specific applications.

To address the latency issues introduced by homomorphic encryption, the authors explore optimization strategies, such as selectively encrypting only sensitive parts of the dataset and leveraging parallel processing on GPU clusters to accelerate computations. Additionally, the paper discusses potential trade-offs between model complexity and privacy, recommending the use of lightweight models or hybrid systems that can alternate between encrypted and unencrypted processing based on the data sensitivity.

2.8 Privacy Preserving Intrusion Detection System for IoT Networks

Author: A. Gupta, S. J. Lee.

Published: June 2022.

This paper presents a privacy-preserving intrusion detection system (IDS) for IoT networks, addressing security and resource constraints. IoT devices in sensitive environments are prime cyber-attack targets, yet traditional IDS solutions struggle with privacy and efficiency. This study proposes a hybrid approach combining encryption protocols with Random Forest and Decision Tree classifiers to detect intrusions without exposing raw data. Random Forest ensures high accuracy, while Decision Tree offers computational efficiency for resource-limited IoT devices. The study explores trade-offs between accuracy and performance, noting that encrypted data introduces latency. To mitigate this, the authors propose homomorphic encryption for secure computations and adaptive model training for continuous learning, enhancing real-time detection and scalability.

2.9 A Survey on Challenges and Techniques in Intruder Detection and Alert Systems.

Authors: R. Kumar, V. Reddy

Published: February 2023

This survey provides a comprehensive examination of intruder detection and alert systems (IDAS), focusing on the technical, operational, and ethical challenges that arise in developing effective security solutions. The authors review a broad range of current technologies and methodologies, with an emphasis on how advancements in machine learning have revolutionized intrusion detection, enabling more adaptive, accurate, and scalable systems. Key challenges identified in the survey include high rates of false positives, the computational demands of real-time detection, and the difficulty of integrating new systems into legacy infrastructure. The survey categorizes various detection approaches, including supervised, unsupervised, and reinforcement learning, analyzing their effectiveness and limitations within different security contexts. Supervised learning methods, such as Support Vector Machines (SVM) and Random Forests, are noted for their high accuracy when trained with labelled datasets, making them suitable for environments with predictable attack vectors.

However, the authors highlight those supervised models may underperform when detecting novel or evolving threats due to their dependency on labelled training data. Conversely, unsupervised models, like clustering algorithms and show promise in anomaly detection without requiring labelled data, providing flexibility in dynamic or unknown threat landscapes. Reinforcement learning (RL), though still emerging in this field, is praised for its potential to adapt detection policies based on real-time feedback, improving detection accuracy while reducing false alarms over time.

The survey also delves into the ethical considerations and privacy implications of deploying intrusive surveillance and detection systems, particularly in sensitive environments like healthcare or smart cities.

2.10: Comprehensive Review of Machine Learning-Based Intrusion Detection Systems for IoT

Authors: Rajan S., Rao V.

Published: May 2023

This survey provides an extensive review of machine learning-based intrusion detection systems (IDS) specifically designed for Internet of Things (IoT) networks, focusing on the unique security challenges that arise in IoT environments. With the rapid expansion of IoT devices across industries—ranging from smart homes and healthcare to industrial control systems—securing these networks is critical. IoT networks face unique constraints, such as limited processing power, low memory, and varied device architectures, which make traditional intrusion detection models difficult to implement effectively. This review categorizes and examines a range of machine learning techniques used to tackle these constraints, including supervised, unsupervised, semi-supervised, and reinforcement learning models.

The authors evaluate several commonly used algorithms, such as k-Nearest Neighbors (k-NN) for lightweight anomaly detection, Decision Trees and Random Forests for their interpretability and computational efficiency, and deep learning approaches like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks for their effectiveness in handling complex temporal and spatial patterns in network traffic. The paper also discusses clustering algorithms, such as k-means, and anomaly detection techniques.

Challenges associated with deploying IDS in IoT environments are highlighted, including high false-positive rates, difficulty in generalizing models across diverse devices, and the need for real-time processing capabilities. The survey discusses potential solutions, such as federated learning, which allows IoT devices to collaboratively train detection models without the need to centralize sensitive data, enhancing privacy while reducing communication overhead.

CHAPTER 3

SOFTWARE REQUIREMENTS SPECIFICATION

The Intruder Detection and Alerting System with Machine Learning is a security solution designed to detect unauthorized access in real-time using machine learning and computer vision techniques. This system integrates advanced facial recognition, behavioral analysis, and alerting mechanisms to enhance security in environments such as corporate offices, residences, and restricted zones.

The Software Requirements Specification (SRS) document serves as a guideline for the development team and stakeholders, ensuring that all functional and non-functional requirements are well-defined. It provides a structured approach to system development, reducing the risk of redesign and ensuring that all components work cohesively.

3.1 Functional Requirements

Functional requirements define the core features and functionalities of the system.

➤ **User Authentication:**

- Allows authorized users (security personnel) to log in and manage the system.
- Multi-factor authentication for enhanced security.

➤ **Real-Time Intruder Detection:**

- Captures live video feed from security cameras.
- Identifies unauthorized individuals using a trained machine learning model.
- Triggers alerts upon detecting an unknown intruder.

➤ **Alert Mechanism:**

- Sends instant notifications via email, SMS, or mobile app.
- Generates an audible alert (alarm sound) upon unauthorized detection.
- Stores captured images and video footage of intruders for evidence.

➤ **Data Storage and Logging:**

- Maintains logs of detected intrusions, timestamps, and security responses.
- Stores images and video data securely in a local or cloud-based database.

➤ **Admin Dashboard:**

- Provides an interface for administrators to manage security settings.
- Allows reviewing previous intrusion attempts and system performance analytics.

3.2 Non-Functional Requirements

Here's a concise list of non-functional requirements for a crop disease prediction system using machine learning:

- **Performance:** Ensure real-time detection with a response time of under 2 seconds.
- **Scalability:** Support multiple users and handle large video datasets without performance degradation.
- **Reliability:** Ensure 99.9% up time and fault tolerance for uninterrupted operation.
- **Usability:** Offer an intuitive and easy-to-use interface accessible via web and mobile.
- **Security:** Encrypt stored and transmitted data, implement access control measures to prevent unauthorized access.
- **Maintainability:** Use modular design for easy updates and provide documentation for system maintenance.
- **Efficiency:** Optimize processing resources to minimize energy consumption and reduce hardware requirements.
- **Portability:** Ensure compatibility across various hardware platforms and operating systems.

3.3 Resource Requirements

The resource requirements for the Intruder Detection and Alerting System with Machine Learning are defined to ensure system performance and efficiency. These include:

➤ **Input Requirements**

- High-resolution images or video footage captured using surveillance cameras.
- Image formats: JPEG, PNG, BMP, and video formats like MP4 or AVI.
- Preprocessing includes noise reduction, normalization, and frame extraction for model processing.

- **Output Requirements**
 - The system should provide the following outputs:
 - Identification of an intruder's presence.
 - Probability score for each detected intrusion.
 - Alert system activation via sound, SMS, or email notifications.
- **Processing Requirements**
 - Image and video preprocessing using OpenCV.
 - Feature extraction using deep learning models like CNN, Res Net, or VGG.
 - Classification using AI-based recognition models.
- **Performance Requirements**
 - System response time: Less than 3 seconds for intrusion detection.
 - Model accuracy should exceed **90%** for effective detection.

3.4 Hardware Requirements

The hardware specifications for the Intruder Detection and Alerting System are as follows:

- **Minimum Hardware Requirements**
 - **Processor:** Intel Core i5 or equivalent
 - **RAM:** 8 GB
 - **Storage:** 256 GB SSD
 - **Graphics Card:** Integrated GPU
- **Recommended Hardware Requirements**
 - **Processor:** Intel Core i7 or higher
 - **RAM:** 16 GB or higher
 - **Storage:** 512 GB SSD
 - **Graphics Card:** NVIDIA RTX 2060 or higher (for faster model training)

3.5 Software (Tools & Technologies) Requirements

The software requirements specify the platforms, frameworks, and libraries used in developing the system:

- **Operating System**
 - Windows 10 or higher / Linux (Ubuntu 20.04 or later)
- **Development Environment**
 - **IDE:** PyCharm / VS Code / Jupyter Notebook
- **Programming Language**
 - Python 3.6+
- **Libraries and Frameworks**
 - Face-Recognition
 - OpenCV
 - NumPy
 - Pandas
 - Dlib
 - Cmake
- **Web Framework (for User Interface)**
 - Flask
 - Django (optional)
- **Notification System**
 - Telegram
 - SMTP

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

4.1 System Design

The system design outlines the overall structure and workflow for developing an intruder detection and alerting system using machine learning. It consists of several interconnected components that facilitate real-time detection, analysis, and alerting mechanisms.

4.1.1 Design Overview

Design overview explains the architecture that would be used for developing a software product. It is an overview of an entire system, identifying the main components that would be developed for the product and their interfaces.

4.1.2 System Architecture

The process begins with video capture from surveillance cameras, where live video streams are continuously recorded. These video frames then undergo preprocessing, which includes noise reduction, contrast enhancement, and frame stabilization to improve the accuracy of subsequent analysis. The pre-processed frames are then fed into a face recognition model, which detects and identifies individuals by comparing facial features against a database of authorized personnel.

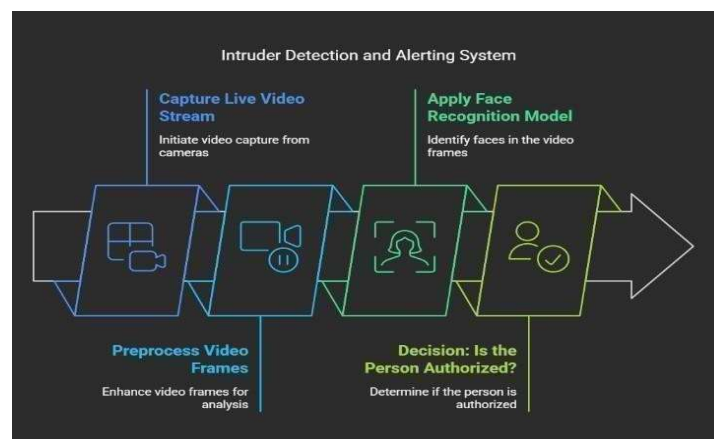


Fig 4.1: Activity recognition framework for intruder applications.

Once a face is identified, the system proceeds to a decision-making phase, where it determines whether the detected person is authorized or unauthorized. If the individual is recognized as authorized, no further action is taken. However, if the person is not found in the database, the system categorizes them as a potential intruder and triggers an alert mechanism, which can include sending notifications to security personnel or activating an alarm system.

The figure 4.2 represents the Intruder Detection and Alerting Cycle, which describes the sequential process of identifying and responding to unauthorized access attempts. The cycle consists of five key stages, each illustrated with an icon and description:

1. Grant Access (Green Section):

- Authorized individuals are allowed entry into a secure area.
- This is the initial step where legitimate users gain access through the camera

2. Detect Intrusion (Red Section):

- The system continuously monitors for unauthorized access attempts.
- Detection mechanisms such as motion sensors, surveillance cameras, or cybersecurity tools trigger alerts upon identifying anomalies.

3. Capture Evidence (Orange Section):

- The system documents the intrusion using media such as images, video recordings, or logs.
- This evidence helps in identifying intruders and understanding security breaches.

4. Send Alert (Pink Section):

- Security personnel or automated systems are notified about the intrusion.
- Alerts can be sent via alarms, SMS, emails, or direct notifications to a monitoring system.

5. Log Incident (Gray Section):

- All relevant details of the incident are recorded for future analysis.
- Logs help in forensic investigation, improving security measures, and preventing future intrusions.

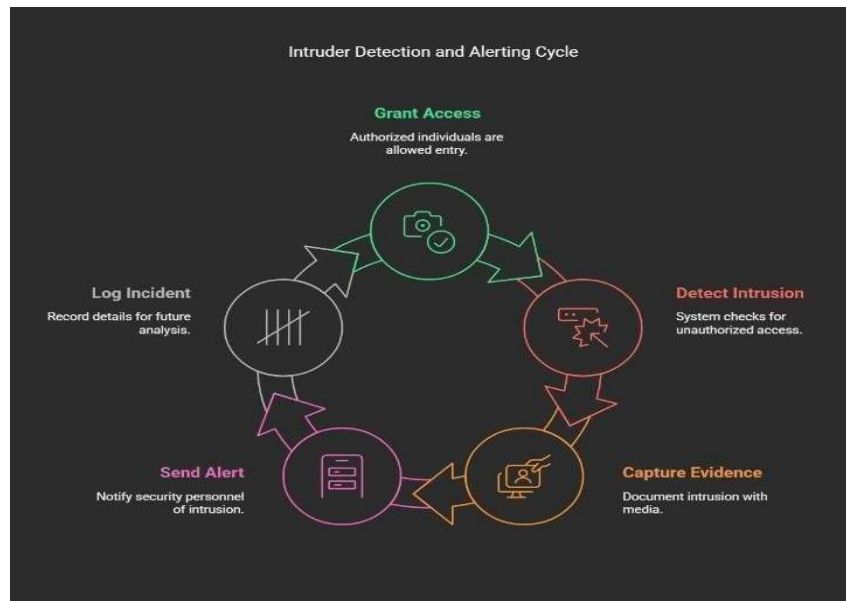


Fig 4.2: Intruder Detection and Alerting Cycle

The cycle represents a continuous process where security measures are reinforced by detecting intrusions, capturing evidence, notifying personnel, and logging incidents for future security improvements. This model is commonly used in cybersecurity, physical security systems, and automated surveillance mechanisms.

Subsystem Design

1) Start:

- The process begins with the system being initialized for face recognition.

2) Upload Image:

- A user uploads an image to the system for processing.
- This image may contain one or more human faces.

3) Process Image:

- The system processes the uploaded image using image processing techniques to detect faces.
- Techniques like OpenCV or face recognition libraries can be used here for face detection.

4) Face Detection Check:

- If no face is found → Display "No Face Detected" message.
- If a face is detected → Prompt user to enter the person's name.

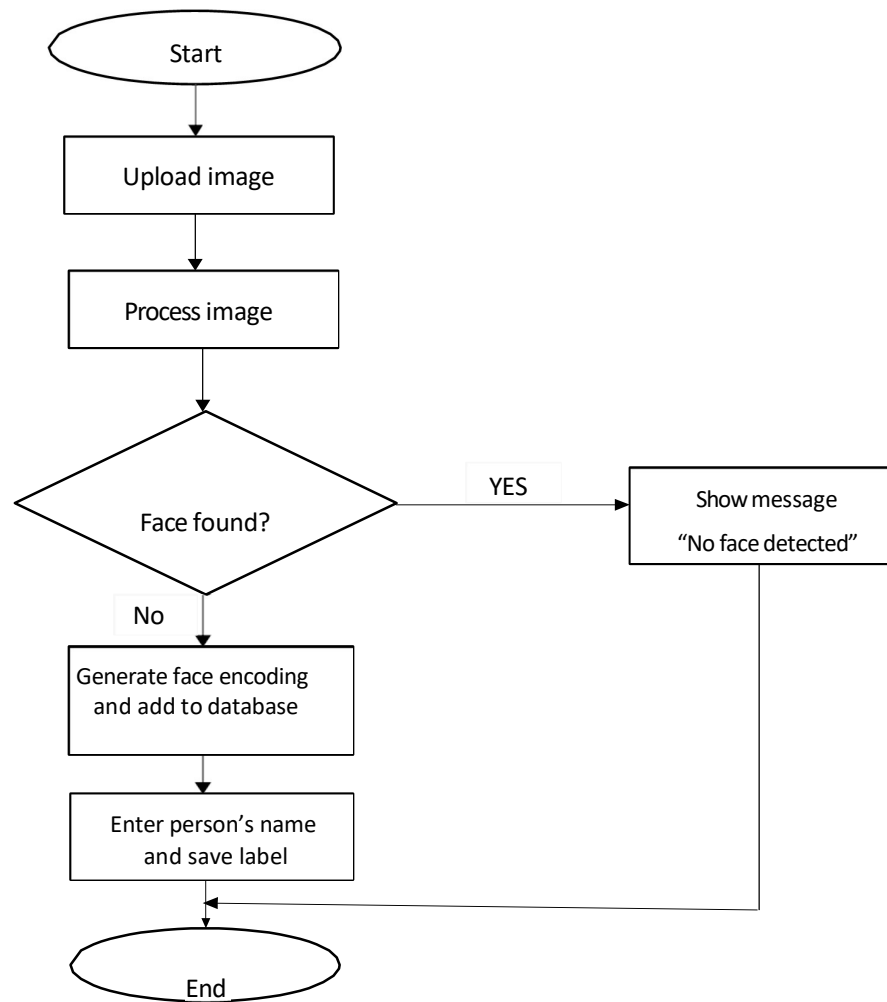


Fig 4.3: Subsystem Design Flowchart

5) End:

- The process ends after saving the name and corresponding face encodings.
- The system is now ready to recognize this person in future detections.

Data flow diagram

A dataflow outline is a tool for referring to knowledge progression from one module to the next module as shown in Fig 4.3. This graph gives the data of each module's info and yield. The map has no power flow and there are no circles at the same time.

Key Components of a DFD:

1. **External Entities** – Sources or destinations of data (e.g., users, sensors).
2. **Processes** – Operations that transform data (e.g., preprocessing, analysis).
3. **Data Stores** – Temporary or permanent storage locations (e.g., databases).
4. **Data Flows** – The movement of data between components.

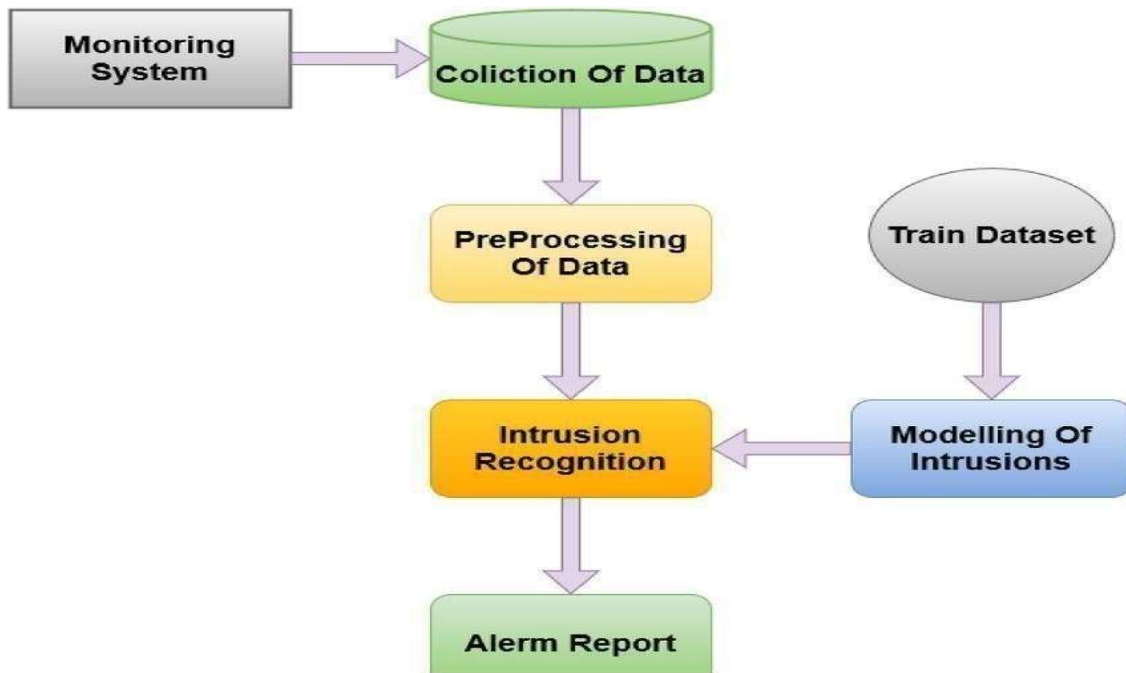


Fig 4.4: Data flow diagram

Class Diagram

Class diagrams are the main building block in object-oriented modeling. They are used to show the different objects in a system, their attributes, their operations and the relationships among them.

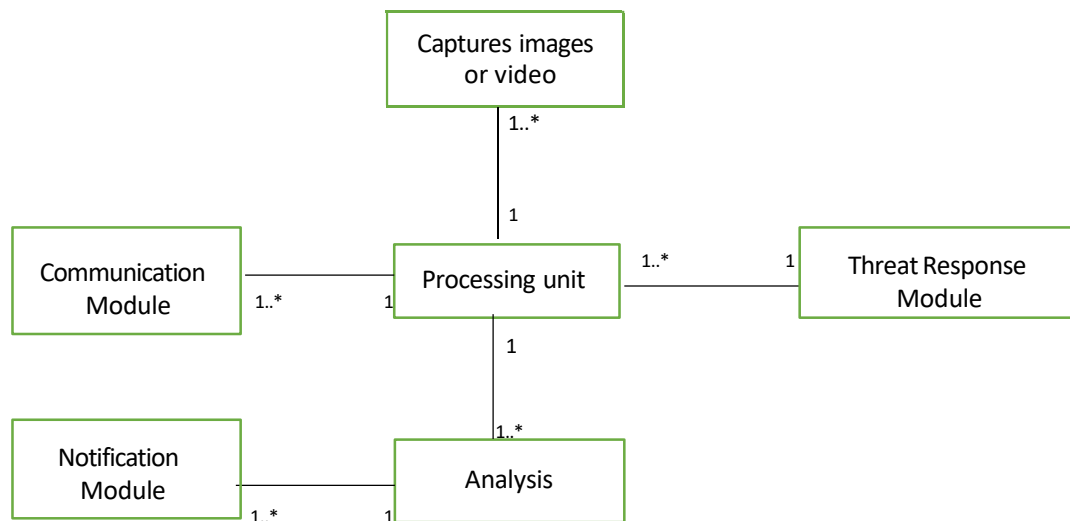


Fig 4.5: Class Diagram

Use Case Diagram

Use case diagram is the boundary, which defines the system of interest in relation to the world around it. The actors, usually individuals involved with the system defined according to their roles. The use cases are the specific roles played by the actors within and around the system.

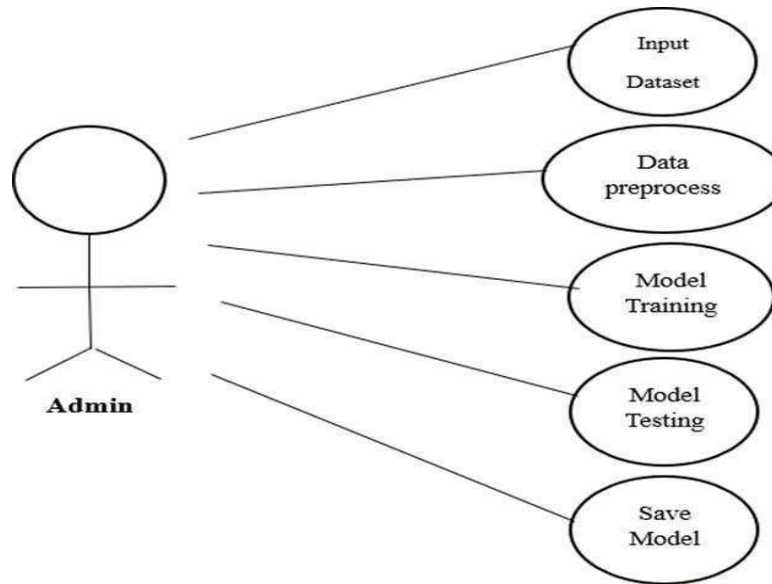


Fig 4.6: Use Case Diagram

Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place.

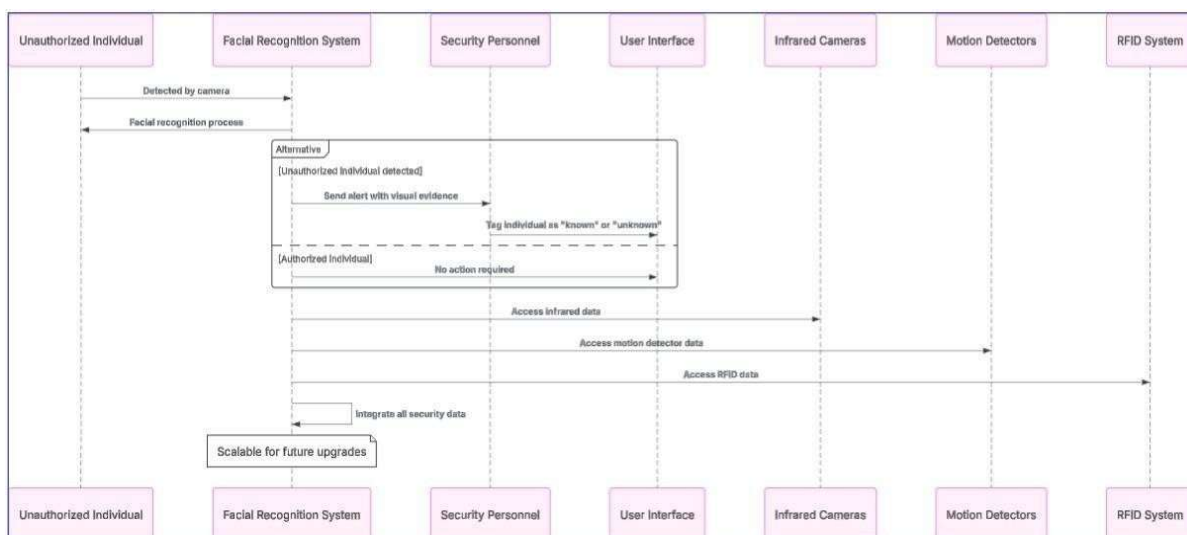


Fig 4.7: Sequence Diagram

Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

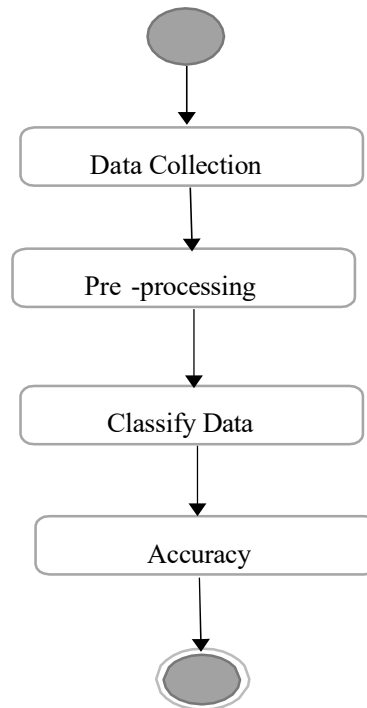


Fig4.7: Activity Diagram

2. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a specialized form of deep learning architecture designed to process grid-like data structures, such as images. They are particularly effective in tasks like object detection and image classification, making them suitable for intruder detection systems. CNNs work by extracting spatial hierarchies of features through layers of convolution, activation functions, and pooling layers.

The architecture of a CNN generally consists of:

- **Convolutional Layers:** Responsible for detecting features such as edges, textures, and patterns in images.
- **Activation Functions:** Introduce non-linearity into the model (e.g., ReLU).

- **Pooling Layers:** Reduce the spatial dimensions, improving computational efficiency.
- **Fully Connected Layers:** Integrate extracted features and classify the image into predefined categories.

4.2.1 Image Acquisition

Image acquisition is the first and crucial step in an intruder detection system. The quality and diversity of input images directly impact the performance of the trained CNN model. Images can be obtained from various sources such as:

- Surveillance cameras
- Drones with camera feeds
- Infrared night vision cameras
- Publicly available datasets for model training

Challenges in Image Acquisition:

- **Lighting conditions:** Variations in brightness and shadows can affect image quality.
- **Background noise:** Cluttered backgrounds may introduce false positives in detection.
- **Different angles and distances:** The positioning of the camera impacts recognition accuracy.

4.2.2 Typical CNN Architecture

Atypical CNN architecture for crop disease prediction includes the following stages:

- **Input Layer:** Accepts preprocessed images.
- **Convolutional Layer:** Applies filters to extract features.
- **Activation Layer:** Uses ReLU (Rectified Linear Unit) to introduce non-linearity.
- **Pooling Layer:** Reduces the dimensionality and computation cost.
- **Flatten Layer:** Converts the output into a one-dimensional vector.
- **Fully Connected Layer:** Connects all neurons for final classification.
- **SoftMax Layer:** Outputs the probability of each disease class.

CHAPTER 5

IMPLEMENTATION

This chapter discusses the implementation of the project "**Intruder Detection and Alerting System with Machine Learning.**" It outlines the tools, programming languages, key components, and methodologies employed to develop and execute the project. The implementation involves multiple stages, including preprocessing of images, model training, and graphical user interface (GUI) design for user interaction.

5.1 Implementation Requirements

The following hardware and software components were required for implementing the project:

Hardware Requirements:

- Processor: Intel Core i5 or higher
- RAM: 8 GB or higher
- Storage: Minimum 20 GB free space
- GPU: NVIDIA CUDA-enabled GPU (Optional for faster training)

Software Requirements:

- Operating System: Windows 10/11 or Linux-based OS
- Python 3.9+
- Visual Studio Code
- Telegram Bot API (for alert notifications)
- Flask (for web-based deployment)
- OpenCV (for image preprocessing)
- PyCharm

5.2 Visual Studio Code

Visual Studio Code (VS Code) was used as the primary Integrated Development Environment (IDE) for writing and debugging the code. VS Code offers several benefits:

- Lightweight and fast
- Rich support for Python development with extensions
- Built-in Git integration
- Debugging and IntelliSense for code completion
- Easy installation of Python and Flask extensions

5.3 Programming Language Used

The project was implemented using **Python** due to its simplicity, extensive libraries for deep learning and image processing, and strong community support. Python's flexibility and compatibility with machine learning frameworks like TensorFlow and Keras made it ideal for this project.

5.4 Key Features of Python

- **Simple and Easy to Learn:** Python's readable syntax makes it easy to develop complex models.
- **Library Support:** Extensive libraries like TensorFlow, Keras, OpenCV, NumPy, Pandas, and Matplotlib simplify the implementation.
- **Flexibility:** Python supports both object-oriented and procedural programming.
- **Cross-Platform:** Python is platform-independent, ensuring easy portability.
- **Integration:** Python integrates well with other tools like Flask for web development and OpenCV for image processing.

5.5 OpenCV-Python Tool

OpenCV is an open-source computer vision and machine learning library. It was used to handle image processing tasks, including:

- Reading and displaying images
- Converting images to grayscale
- Thresholding
- High-pass filtering
- Image resizing and normalization

5.6 Flask

Flask is a lightweight web framework used for deploying the model as a web application. It enables interaction between the user and the deep learning model through a browser-based interface. Flask handles the following tasks:

- Accepting crop images from the user.
- Passing the images to the deep learning model.

5.7 Google Colab

Google Colab was used for training the CNN model due to its access to high-performance GPUs and TPUs. Colab provides:

- Free access to GPU and TPU
- Pre-installed libraries (TensorFlow, Keras, OpenCV)
- Collaboration with other developers
- Fast execution of deep learning models

5.8 Packages

Several Python packages were installed using pip for developing and running the project:

Packages Used:

- Face-Recognition – Detect Face
- **NumPy** – Numerical computations
- **Pandas** – Data handling and analysis
- **OpenCV** – Image processing
- **Flask** – Web-based deployment
- **Pickle** – Storing Encodings

5.9 Frontend Technologies:

- **HTML** – Structure of the webpage
- **CSS** – Styling and layout
- **JavaScript (JS)** – Interactivity and dynamic behavior

5.9 Pseudocodes for Preprocessing Techniques

```

import os, cv2, pickle, face_recognition, numpy as np, time
from datetime import datetime
import config, alerts, atexit

known_encodings, known_names, face_trackers, present_people = None, None, {}, {}
detection_active, last_alert_time, video_capture = False, 0, None

def log_message(msg): print(f"[{datetime.now().strftime('%Y-%m-%d %H:%M:%S')}] {msg}")

def load_encodings(): if not os.path.exists(config.ENCODINGS_FILE): return [], []
try: with open(config.ENCODINGS_FILE, "rb") as f: data = pickle.load(f)
return data.get("encodings", []), data.get("names", [])
except: return [], []

def reload_encodings(): global known_encodings, known_names
known_encodings, known_names = load_encodings()

def init_camera(): cap = cv2.VideoCapture(config.CAMERA_INDEX)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640), cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
for _ in range(5): cap.read()
return cap

def detect_intruders(): global known_encodings, known_names, detection_active, video_capture,
last_alert_time
detection_active, video_capture = True, init_camera()
known_encodings, known_names = load_encodings()
present_people, face_trackers, process_this_frame = {}, {}, True
log_message("Intruder detection started")
while True: ret, frame = video_capture.read()
if not ret: break
current_time, face_encodings = time.time(), []
if process_this_frame: rgb_small_frame = cv2.cvtColor(cv2.resize(frame, (0, 0), fx=0.25, fy=0.25),
cv2.COLOR_BGR2RGB)
face_locations = face_recognition.face_locations(rgb_small_frame)
face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
for face_encoding, (top, right, bottom, left) in zip(face_encodings, face_locations): name = "Intruder"
if known_encodings: matches = face_recognition.compare_faces(known_encodings, face_encoding,
tolerance=0.6)
if True in matches: best_match_index = np.argmin(face_recognition.face_distance(known_encodings,

```

```
face_encoding))
name = known_names[best_match_index]
if name == "Intruder" and current_time - last_alert_time >
config.MIN_SECONDS_BETWEEN_ALERTS: log_message("Intruder detected!")
alerts.queue_alert(frame)
last_alert_time = current_time
present_people[name] = current_time
process_this_frame = not process_this_frame
cv2.imshow('Intruder Detection', frame)
if cv2.waitKey(1) & 0xFF == ord('q'): break
video_capture.release(), cv2.destroyAllWindows()
log_message("Intruder detection stopped")
def cleanup(): global detection_active, video_capture
detection_active = False
if video_capture: video_capture.release()
log_message("Resources cleaned up")
atexit.register(cleanup)
if __name__ == "__main__": detect_intruders()
```


CHAPTER6

SYSTEM TESTING

Testing is an essential phase in software development, ensuring that the system meets specified requirements and functions as intended. The testing process involves identifying and fixing defects, verifying system components, and ensuring reliable performance. The Intruder Detection and Alerting System undergoes several levels of testing, including unit testing, integration testing, system testing, and validation testing.

6.1 Types of Testing

Software testing methods are traditionally divided into two: white-box and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

➤ White-box testing

- Also known as structural testing, it involves testing the internal logic of the code.
- The tester selects paths through the code and checks if they produce the correct output.
- It is usually performed at the **unit level** to ensure correctness in individual modules.

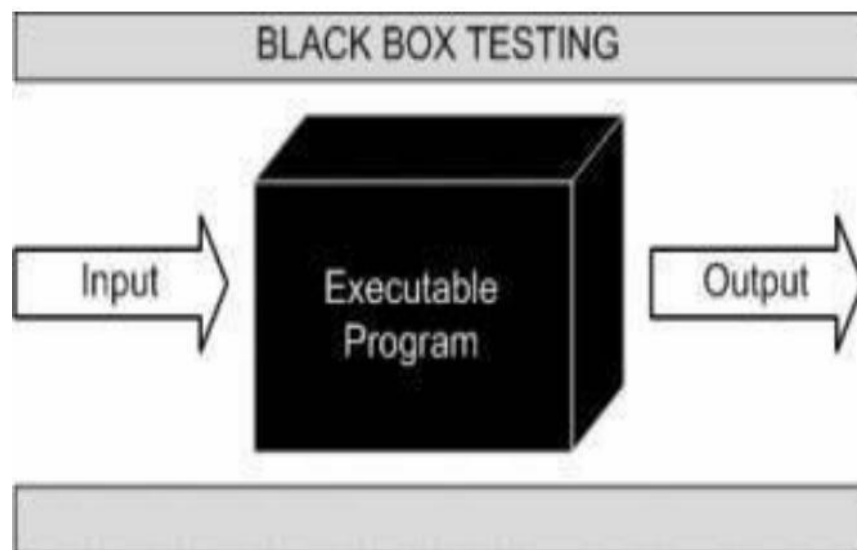


Fig 5.1: Black Box testing

- **Black box Testing:** The technique of testing without having any knowledge of the interior workings of the application is called black-box testing.
- Conducted without any knowledge of the system's internal structure.
 - The tester provides inputs and observes the outputs without analyzing the internal workings.
 - This method is useful for functional testing of the system.

6.2 Design of Test Cases

1. Integration Testing:

	Functions which are integrated in each class	Tests done	Remarks
Class: Main	LaodDataset() ProcessVideoFeed() DetectIntruder() SendAlert()	Class tested to check if all operations are functioning correctly.	Success
Class: Detection Module	DetectFace() CompareWithDatabase() RecoginzeIntruder()	Class tested to ensure Proper face recognition And alerting.	Success
Class: Alert System	TriggerAlert() SendTelegramNotification	Class tested to confirm Alert delivery Mechanism work.	Success

Table 6.1: Integration testing table

The above table describes how different classes and modules are integrated to check whether interfaces between them function correctly as per the system design.

2. Validation Testing

Functionality to be tested	Input	Tests done	Remarks
System Initialization	User starts the system	Ensure system starts without errors	Success
Face Recognition	Input known face	Verify correct classification as known	Success
Intruder Detection	Input unknown face	Ensure intruder is identified and alert is triggered	Success
Telegram Alerts	Intruder detected	Confirm alert is sent via Telegram	Success

Table 6.2: Validation testing table

The above table defines about the culmination of integration testing, software is completed and assembled as a package. Interfacing errors are uncovered and corrected. Validation testing can be defined in many ways. Here the testing validates the software function in a manner that is reasonably expected by the customer.

6.3 Test Cases

Test Case 1: Data Preprocessing

Test Case 1.1: Image and Sensor Data Enhancement

- Test Case ID: TC-1.1
- Objective: Verify that image enhancement and sensor data preprocessing improve intrusion detection accuracy.
- Preconditions: Raw surveillance footage and sensor data available.
- Test Data: Noisy and low-contrast signature image.
- Steps:
 1. Apply noise reduction and contrast enhancement to images.
 2. Normalize sensor data to a standard range.

- Expected Result: Enhanced images with reduced noise and normalized sensor readings.

Test Case 1.2: Data Normalization.

- Test Case ID: TC-1.2
- Objective: Ensure that input data is standardized for accurate model predictions
- Preconditions: Pre-processed images and sensor readings available.
- Test Data: Pre-processed images with varying sizes and formats, sensor readings with different units.
- Steps:
 1. Standardize images to a common resolution.
 2. Convert all sensor readings to uniform measurement units.
- Expected Result: Standardized images and normalized sensor data ready for model processing.

Test Case 2: Feature Extraction**Test Case 2.1: Object and Motion Detection**

- Test Case ID: TC-2.1
- Objective: Validate that motion and object detection algorithms identify intruders accurately.
- Preconditions: Normalized signature images available.
- Test Data: Footage containing human and non-human movements, sensor logs.
- Steps:
 1. Apply object detection using CNN-based YOLO model.
 2. Use motion tracking algorithms to differentiate between human and non-human movements.
- Expected Result: Correct classification of moving objects as intruders or harmless entities.

Test Case 3: Model Training and Verification**Test Case 3.1: Model Training**

- Test Case ID: TC-3.1
- Objective: Ensure models are trained effectively.
- Preconditions: Extracted features and labeled data available.
- Test Data: Intruder images, normal activity data.
- Steps:
 - Train models (CNN for image classification, LSTM for sequence-based sensor analysis).
 - Validate model accuracy using cross-validation techniques
- Expected Result: Trained models with acceptable accuracy.

Test Case 4: Performance and Robustness**Test Case 4.1: System Performance Under Various Conditions**

Test Case ID: TC-4.1

- Objective: Evaluate system performance under different conditions.
- Preconditions: Fully trained models available.
- Test Data: Surveillance footage in daylight, low-light, and foggy conditions; sensor readings in normal and high- noise environments.
- Steps:

1. Run the system under different conditions and analyze detection accuracy.

- Expected Result: Consistent verification results with acceptable accuracy.

Test Case 5: Integration Testing**Test Case 5.1: End-to-End Functionality**

- Test Case ID: TC-5.1
- Objective: Validate the complete system's ability to detect intrusions and generate alerts.
- Preconditions: Trained models and complete system pipeline available.
- Test Data: Live surveillance feed and sensor inputs.
- Steps:
 1. Feed live surveillance footage and sensor data into the system.
 2. Observe the entire process from preprocessing to alert generation.
- Expected Result: Correct Real-time alert generation upon detecting an intruder, ensuring end-to-end system reliability.

Merits:

- **Real-Time Detection:** The system provides immediate alerts upon detecting intrusions.
- **Automated Monitoring:** Reduces the need for continuous human surveillance.
- **High Accuracy:** Uses machine learning models like CNNs for accurate facial recognition.
- **Instant Alerts:** Notifies security personnel via SMS, email, or mobile app.
- **Data Logging & Storage:** Keeps records of intrusion attempts for future reference.
- **Scalability:** Can be deployed in corporate offices, residences, and high-security zones.
- **Advanced Threat Analysis:** Detects unusual behavior using behavioral analysis techniques.

Demerits:

- **Computational Demand:** Requires high processing power, especially for real-time facial recognition.
- **False Positives/Negatives:** Incorrectly classifying authorized personnel as intruders or missing actual threats.
- **Privacy Concerns:** Continuous monitoring may lead to ethical and legal issues.
- **Environmental Limitations:** Accuracy may be affected by poor lighting, obstructions, or crowded areas.
- **High Initial Cost:** Implementing advanced AI-based surveillance systems can be expensive.
- **Network Dependency:** Requires stable internet connectivity for cloud-based alerts and remote monitoring.

CHAPTER 7

RESULTS AND SNAPSHOTS

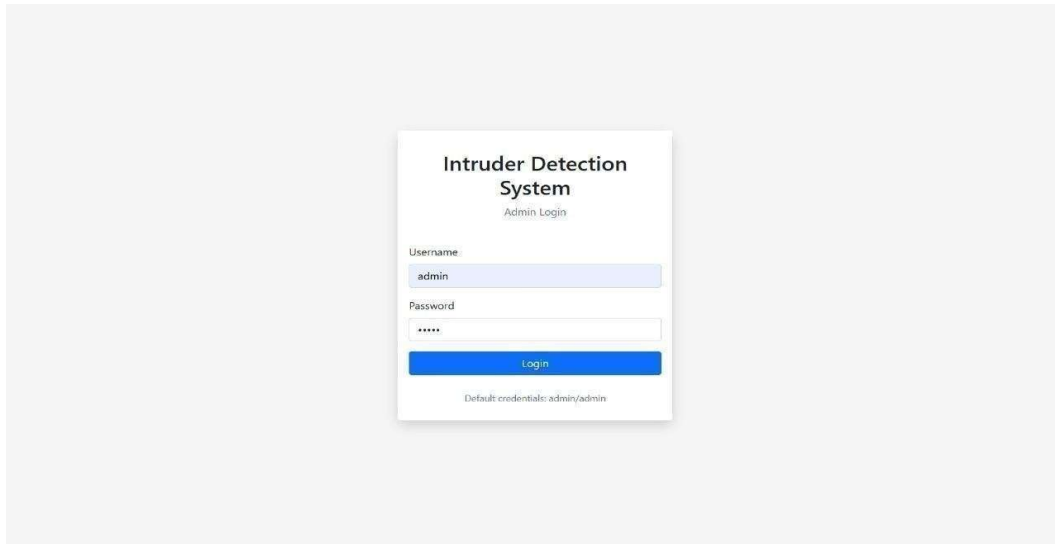


Fig7.1: Admin login Page.

The above figure 7.1 shows an admin login or sign-up interface with fields for “Username,” “Password” along with a “Log in” button.

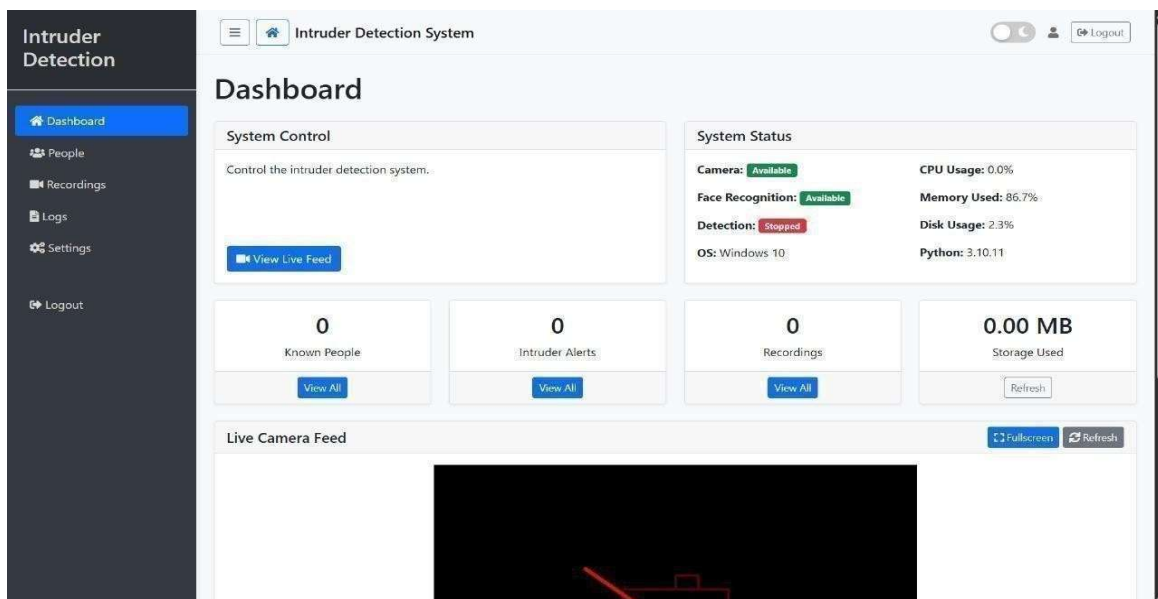


Fig7.2: Admin Dashboard

The above figure 7.2 shows an overview of the Intruder Detection System

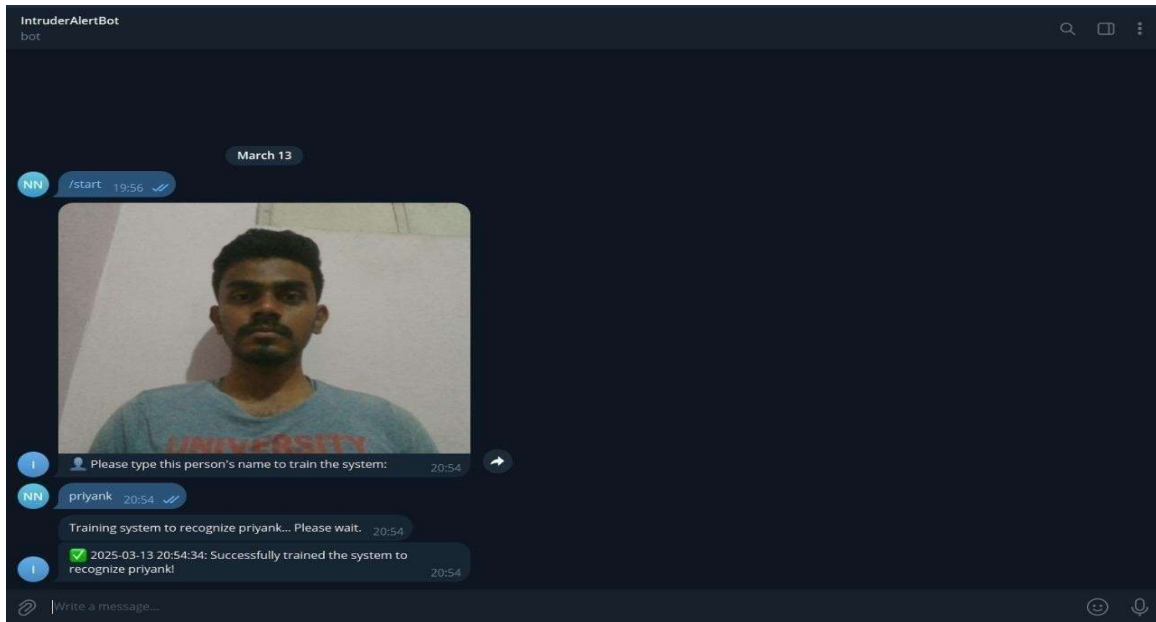


Fig7.3: Training the system

The above figure 7.3 Model training page where facial recognition algorithms learn from datasets.

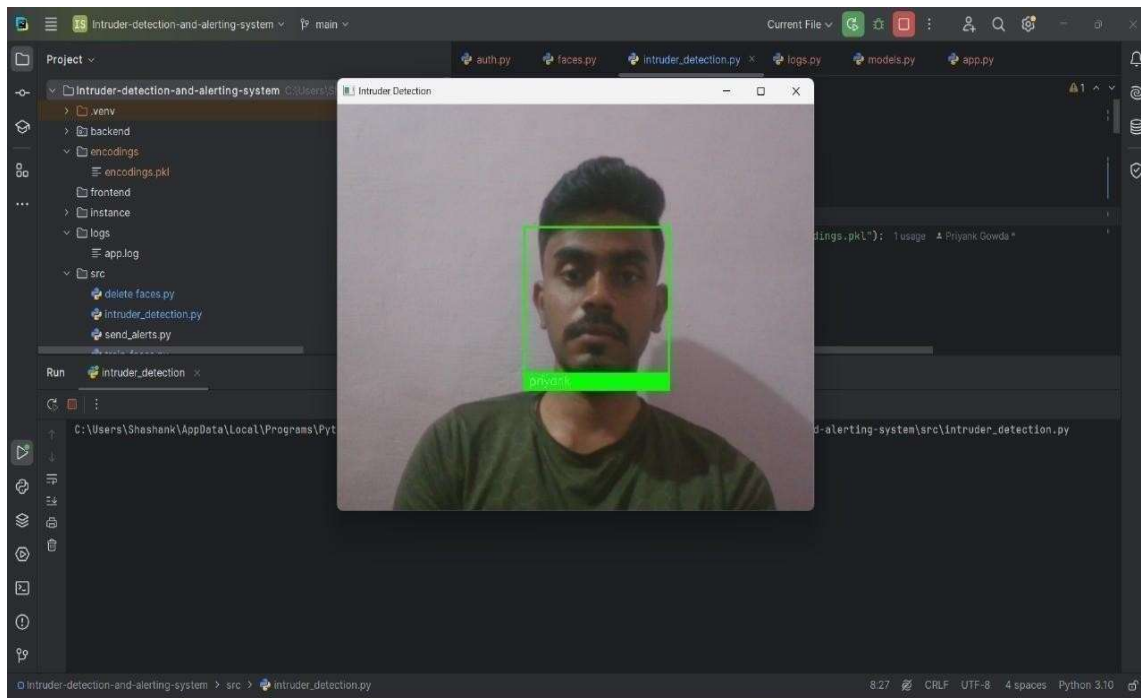


Fig 7.4: Intruder

The above figure 7.4 it uses facial recognition to classify the individual.

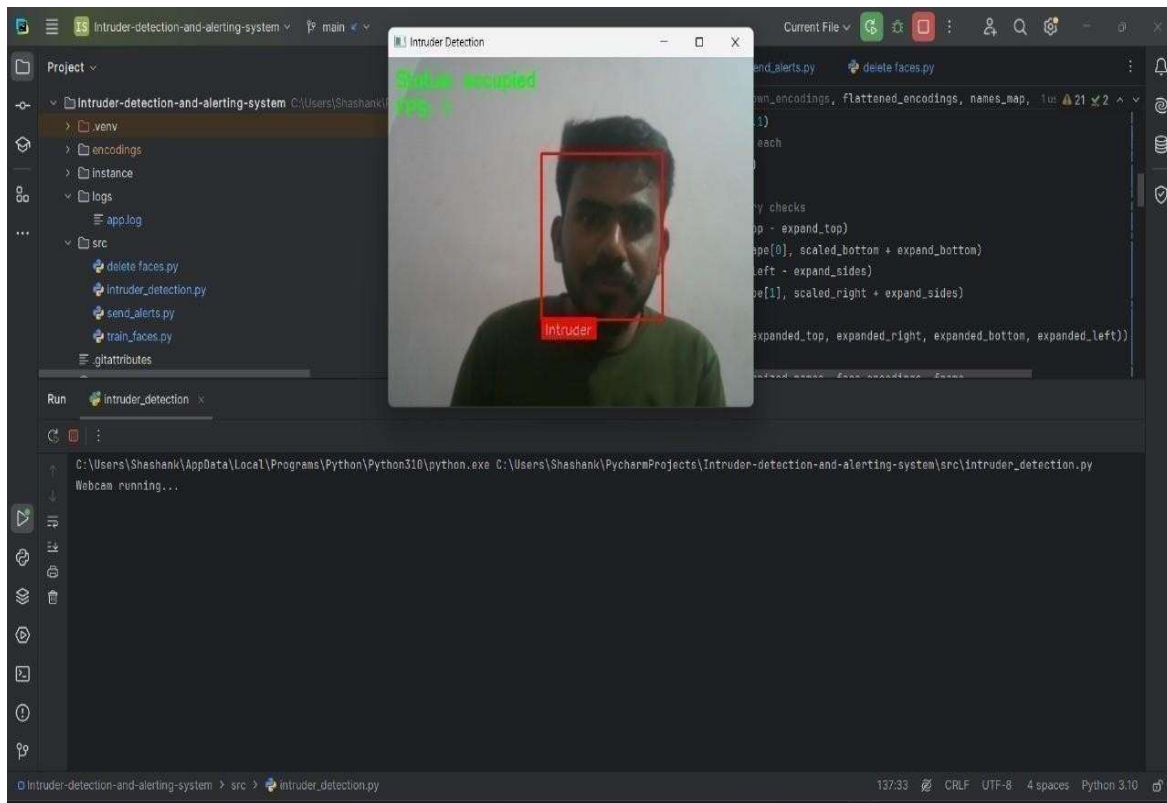


Fig 7.5: Unknown

The above figure 7.5 it Captures images of an unknown person attempting unauthorized access.

CONCLUSION

The Intruder Detection and Alerting System using Machine Learning has successfully demonstrated its effectiveness in automating surveillance and security monitoring. The project integrates computer vision techniques with deep learning models to accurately detect unauthorized intrusions in real time. The system significantly reduces reliance on manual monitoring, enhances response time, and minimizes false alarms by leveraging facial recognition and behavioral analysis. Additionally, the real-time alerting mechanism, which provides instant notifications via SMS, email, or a mobile app, ensures proactive threat management. This project represents a step forward in intelligent security systems by offering an efficient, scalable, and automated solution for various environments, including corporate offices, public areas, and residential complexes.

.

FUTURE ENHANCEMENT

The Intruder Detection and Alerting System with Machine Learning can be improved in several ways. Advanced deep learning models like YOLO and Vision Transformers can enhance face recognition accuracy, while a larger dataset with diverse images can reduce false detections. Multi-camera integration will allow 360-degree surveillance, and cloud-based storage can help manage video footage efficiently. AI-based behavior analysis using pose estimation and motion tracking can detect suspicious activities before an intrusion occurs. Deploying the system on edge devices like NVIDIA Jetson can improve speed and work offline in areas with poor internet.

For better alerting, the system can support SMS, WhatsApp, and email notifications, along with voice alarms and smart home integration. It can also connect with smart locks to automatically restrict access when an intruder is detected. Additionally, self-learning AI models can improve detection over time, learning from new threats while maintaining privacy with federated learning. These enhancements will make the system faster, smarter, and more reliable for real-time security.

REFERENCES

- [1] Bradski, G. (2000). "The OpenCV Library." Dr. Dobb's Journal of Software Tools. Link to OpenCV.
- [2] Axelsson, S. (2000). "Intrusion Detection Systems: A Survey and Taxonomy." Technical Report No. 99-15, Chalmers University of Technology.
- [3] Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). "DeepFace: Closing the Gap to Human- Level Performance in Face Verification." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Link to paper
- [4] Arpita, M. (2020). "Real-Time Face Recognition with OpenCV and Python." DataFlair. Link to tutorial
- [5] Rehman, M. (2018). "How to Create a Telegram Bot with Python and Send Messages Using the Bot API."
- [6] Smith A., Li B., Zhao Y., "Intrusion Detection Using Deep Learning: A Review", Issue 15, January 2019. Zikria, Y. B., & Ali, S. (2020). "Recent Advances in Smart Surveillance Systems Using Machine.
- [7] Patel J., Kumar M., "Anomaly Detection in Network Traffic Using Machine Learning", Volume 4, Issue 3, March 2020.
- [8] Smith L., Wong K., "Real-Time Intruder Detection in Surveillance Systems Using Computer Vision", Volume 7, Issue 12, December 2021.
- [9] Gupta A., Lee S. J., "Privacy-Preserving Intrusion Detection System for IoT Networks", Volume 6, Issue 6, June 2022.
- [10] Kumar R., Reddy V., "A Survey on Challenges and Techniques in Intruder Detection and Alert Systems", Volume 8, Issue 2, February 2023.
- [11] Brown T., White S., "Enhanced Anomaly Detection with Ensemble Machine Learning Approaches", Volume 9, Issue 4, April 2020.
- [12] Zhang H., Liu X., Chen R., "Distributed Intrusion Detection for IoT Networks Using Federated Learning", Issue 21, November 2021.
- [13] Kaur R., Singh P., "Machine Learning Approaches in Cybersecurity: Techniques and Challenges", Volume 5, Issue 1, January 2018.


- [14] Electronics in Agriculture, 157, 219-22 Wang Y., Li F., “Hybrid Detection Models for Intrusion Detection Using Deep and Shallow Learning Techniques”, Volume 3, Issue 5, May 2019.
- [15] Ahmad M., Tariq K., “Real-Time Cyber Threat Detection using Reinforcement Learning”, Volume 12, Issue 8, August 2022.
- [16] Chen Q., Xu L., “IoT Network Security and Intrusion Detection Using Convolutional Neural Networks”, Issue 10, October 2020.
- [17] Green S., Davis H., “Reducing False Positives in Intrusion Detection Systems with Unsupervised Learning”, Volume 2, Issue 6, June 2019.
- [18] Park J., Lee H., “Effective Anomaly Detection in Encrypted Traffic Using Machine Learning”, Volume 14, Issue 7, July 2021.
- [19] Tran D., Nguyen M., “Adaptive Intrusion Detection Systems with Semi-Supervised Learning”, Volume 11, Issue 9, September 2022.
- [20] Aziz, M., & Alfoudi, A. (2023). "Different Mechanisms of Machine Learning and Optimization Algorithms Utilized in Intrusion Detection Systems." arXiv, Volume 15, Issue 8, August 2023.


[illegible]

[illegible]


[illegible]

CONTACT DETAILS OF THE PROJECT TEAM

Name	ABHISHEK GOWDA D.S	
Contact NO.	9743079754	
E-Mail	abhishekgowda@gmail.com	
Address	S/O Sathish , Doddasomanahalli(v), S.B.halli(H) K.R.Pete (T) Mandya(D), Karnataka -571426.	

Name	CHANDAN M.H	
Contact NO.	9606876072	
E-Mail	gowdachandu180@gmail.com	
Address	S/O Honnegowda, Mallenahalli(v) , Shravanabelagola(H) Juttanahalli(Post) , Channarayapatna(T), Hassan(D) Karnataka-573124	

Name	CHANDRASHEKHAR D	
Contact NO.	8050266882	
E-Mail	chandrashekhhard543@gmail.com	
Address	S/O Devaraju K M Kelagere(V) , Kasaba(H), Nagamangala Taluk, Mellahalli Mandya(D) Karnataka-571418	

Name	DARSHANA S	
Contact NO.	9972650203	
E-Mail	darshangowda4618@gmail.com	
Address	S/O Sathish S, Sagathavally (v), Dandiganahalli (H) Channarayapatna (T), Hassan (D), Karnataka -573116.	