UNIVERSITY of RWANDA

ACEIoT

Website: www.aceiot.ur.ac.rw
Mail: aceiot@ur.ac.rw

College of Science and Technology

AFRICAN CENTER OF EXCELLENCE IN INTERNET OF THINGS (ACEIoT)

Research Thesis Title:

Design and Development of a TinyML-Based Intelligent Home Security System Using Computer Vision and Deep Learning. (Case study: Intruder Detection)

A Thesis Submitted in Partial Fulfilment of the Requirements for the Award of Master of Science Degree in Internet of Things: Embedded Computing Systems

Submitted By:

Name: NSANZABANDI GASASIRA Steven (Ref. No: 220014198)

May, 2025

UNIVERSITY of RWANDA

ACEIoT

Website: www.aceiot.ur.ac.rw Mail: aceiot@ur.ac.rw

College of Science and Technology

AFRICAN CENTER OF EXCELLENCE IN INTERNET OF THINGS (ACEIoT)

Research Thesis Title:

Design and Development of a TinyML-Based Intelligent Home Security System Using Computer Vision and Deep Learning. (Case study: Intruder Detection)

A Thesis Submitted in Partial Fulfilment of the Requirements for the Award of Master of Science Degree in Internet of Things: Embedded Computing Systems

Submitted By:

Name: NSANZABANDI GASASIRA Steven (Ref. No: 220014198)

Supervised By:

-Name of Main Supervisor: Dr. NKUNDINEZA Celestin

-Name of Co – Supervisor: Dr. BUSOGI Moise

May, 2025

DECLARATION

I, NSANZABANDI GASASIRA Steven, a Master's student at the African Center of Excellence in Internet of Things, University of Rwanda, affirm that this thesis is entirely my original work and has not been submitted or presented anywhere else in the world.

Names: NSANZABANDI GASASIRA Steven

Ref: 220014198

Signed: …….

Date: ……/……/……

BONAFIDE CERTIFICATE

This is to certify that this submitted research thesis work report is a record of the original work done by **NSANZABANDI GASASIRA Steven** (**Reg Number: 220014198)** MSc. IoT-Embedded Computing Systems student at the University of Rwanda / College of Science and Technology / African Center of Excellence in Internet of Things, the academic year 2023/2024.

This work has been submitted under the supervision of **Dr. NKUNDINEZA Celestin** and **Dr. BUSOGI Moise**

Main Supervisor: **Dr. NKUNDINEZA Celestin** Co-Supervisor: **Dr. BUSOGI Moise**

Date: 15-07-2025 Date: 15/07/2025

Signature:  Signature: 

The Head of Master's Studies and Training

Dr. RWIGEMA James

Date:

Signature……

ACKNOWLEDGMENT

I sincerely thank the Almighty God for granting me the strength, health, and guidance needed to complete this research.

I am profoundly grateful to my family, particularly my wife, my parents, my brothers, and sisters, for their unwavering moral support, financial assistance, and encouragement, which strengthened me throughout my academic journey.

I extend my heartfelt appreciation to my supervisors, Dr. NKUNDINEZA Celestin and Dr. BUSOGI Moise, for their invaluable mentorship, continuous guidance, and motivation, which greatly contributed to the success of this work.

My gratitude goes to the University of Rwanda, in partnership with the Mastercard Foundation program, for providing me with a full scholarship through the African Center of Excellence in Internet of Things (ACEIoT), which made my studies possible.

I also appreciate the College of Science and Technology at the University of Rwanda, especially the ACEIoT leadership, for granting me the opportunity to pursue this Master's degree and for equipping me with essential knowledge and skills.

A special thank you to my classmates, especially those with whom I collaborated in group discussions and academic projects, for their contributions and support.

I also acknowledge the lecturers in the Internet of Things program for their dedication and commitment to delivering quality education throughout the course of my studies.

Lastly, I extend my sincere appreciation to everyone who contributed, in any way, to my academic journey, helping me successfully complete my Master's degree.

NSANZABANDI GASASIRA Steven

ABSTRACT

With rising security concerns in both urban and rural areas, the need for intelligent, real-time, and cost-effective home security systems has become increasingly urgent. Traditional approaches such as hiring security personnel or deploying basic surveillance cameras often fall short due to their reliance on human monitoring and delayed response. To address these challenges, this study presents the design and implementation of a TinyML-based intelligent home security system that leverages computer vision and deep learning to detect intrusions in real time on resource-constrained edge devices.

The proposed system combines face recognition and behavior detection to verify the identity of individuals and assess their actions. It employs a Convolutional Neural Network (CNN) for facial recognition and YOLOv8, a state-of-the-art object detection model, to identify suspicious behaviors such as wearing a mask or carrying a weapon. These models operate jointly to make robust classification decisions. The system is deployed on a Raspberry Pi 5, integrated with Passive Infrared (PIR) and Light Dependent Resistor (LDR) sensors to respond intelligently to motion and ambient lighting changes. When an unknown or suspicious individual is detected, an alert is triggered, and real-time feedback is provided to the homeowner.

The dataset used for training and evaluation consists of 1,100 labeled images, including both real and synthetically generated data. The integrated model achieved an overall accuracy of 97.27%, with precision, recall, and F1-score values exceeding 96%, demonstrating the system's strong classification performance. Despite these results, the system encountered limitations in extreme low-light conditions and when intruders wore facial coverings that resembled authorized users. These limitations highlight areas for future research and system enhancement.

Overall, this work demonstrates the feasibility of deploying lightweight, privacy-preserving security solutions using TinyML and embedded AI, offering an efficient and reliable alternative for modern home security applications.

Key words: Internet of Things, Computer vision, Deep learning, YOLOV8, Tensor Flow and intruder detection.

LIST OF FIGURES

LIST OF ACRONYMS

ACEIoT: Africa Center of Excellence in Internet of Things

PIR : Passive Infrared

LDR: Light Dependent Resistor

YOLOv8: You Only Look Once, version 8

IoT: Internet of Things

CNN: Convolutional Neural Networks

RQ1: Research Question one

ML: Machine Learning

OpenCV: Open Source Computer Vision Library

AI: Artificial Intelligence

GB: Gigabyte

LFW: Labeled Faces in the Wild

COCO: Common Objects in Context

LED: Light Emitting Diode

RAM: Random Access Memory

OS: Operating System

GPIO : General Purpose Input/Output

USB :Universal Serial Bus

PCIe: Peripheral Component Interconnect Express

SD Card: Secure Digital Card

Table of Contents

Chapter 1 : GENERAL INTRODUCTION

During the period when people lived in separate areas, the risk of theft and burglary was low, and property security was less of a concern. Nowadays, with increasing populations in both urban and rural areas, incidents of theft and burglary have risen significantly, becoming a major concern for homeowners and property security [1],[2]. According to the research done by Beauty Mogaladi Malatjie et al. (2023), People commit residential burglary primarily driven by financial motives, whether out of genuine need or greed for monetary gain. Burglars target residents of gated estates because they think people moving freely there are allowed and have passed security checks. Many residents in these estates aren't careful about their security, often leaving doors and windows unlocked. They might ignore strangers outside their homes, assuming they have been checked by estate security, which isn't always true. This carelessness makes it easier for burglars to get into homes through open doors or windows. Houses with poorly maintained doors and windows are also more attractive to burglars[3]. To address this problem, an intelligent home security system was developed. This system identifies and monitors individuals approaching the residence[4]. When the camera detects unfamiliar individuals, automated Email notifications will be sent to the homeowner's mobile device, and alarms will activate without human intervention. Furthermore, in low-light conditions, it will turn on lights upon detecting motion.

1.1. Background and Motivation

A few years ago, security cameras were employed to create secure environments in various places, residences, and urban areas. However, these systems required human intervention to identify any issues captured by the cameras [5]. This approach proves inadequate for promptly addressing fast-moving thieves. This is where computer vision and deep learning become essential. Computer vision technology, which enables face detection and recognition, offers a highly compelling application within the Internet of Things (IoT) framework. I used computer vision into IoT platforms for smart homes, for the security improvement purposes.

To enhance the power and capabilities of the system, I integrated deep learning[6] to greatly advance several computer vision tasks, including recognizing objects, understanding actions, tracking motion, segmenting images based on meaning and analyze personal behavior. Convolutional neural networks

(CNNs) is the leading technology in deep learning[7]. Utilizing deep learning models, systems are empowered to autonomously analyze complex data patterns and make informed decisions without human intervention, thereby enhancing the quality of daily living[2],[8].

1.2. Problem statement

In light of increasing incidents of theft and burglary in urban and rural areas, there is a pressing need for enhanced home security solutions that mitigate traditional vulnerabilities and address contemporary challenges. Current security measures, such as reliance on security guards and basic surveillance systems, often prove inadequate due to issues like collusion and slow response times. Moreover, existing IoT-based security systems, while offering some benefits like remote monitoring, suffer from limitations in data analysis and responsiveness.

To overcome these shortcomings, an intelligent home security system leveraging TinyML-based computer vision and deep learning technologies was developed. This system effectively detect and respond to intruders, providing real-time alerts and ensuring swift action without human intervention. The system's smart capabilities lies in its ability to thoroughly analyze behavior[9],[10] to determine whether an individual is a family member or an intruder, even if the intruder is attempting to disguise themselves by wearing mask to imitate a family member. Integration of advanced algorithms, particularly leveraging tools like YOLO for object detection, was essential to enhance accuracy and efficiency in identifying potential threats.

In summary, by using this innovative security solution, homeowners are significantly enhance their property's defenses against unauthorized access and safeguard against the growing risks posed by modern-day burglary tactics.

1.3. Study Objectives

1.3.1. Main objectives

The main objective of my thesis is to develop a TinyML-based intelligent home security system that employs computer vision and deep learning technologies to enhance the detection of intruders and the effectiveness of response measures.

1.3.2. Specific objectives:

- Design a smart home security system using Raspberry Pi that supports real-time video processing and object recognition.
- Detect and classify accurately individuals as homeowners or unfamiliar persons using object detection and facial recognition techniques.
- Detect and flag as intruders all individuals who disguise themselves by wearing masks resembling family members, and those carrying sharp objects.
- Automate notifications via email, and trigger alarms in response to security breaches.
- Develop a user-friendly web interface for easy management and interaction with security alerts and recognition lists.

1.4. Research Questions

To achieve the objectives of this study and evaluate the effectiveness of the developed system, specific research questions were formulated. These questions are designed to guide the methodology and ensure that the system meets the intended goals related to intruder detection, real-time performance, automated response, and user interaction. The research questions are as follows:

RQ1: Can a TinyML-based intelligent home security system deployed on a Raspberry Pi 5 accurately detect and classify individuals as authorized homeowners or unauthorized intruders in real time?

RQ2: How effectively can the system detect disguised intruders wearing masks or carrying suspicious objects such as knives, guns, or pangas using object detection and facial recognition techniques?

RQ3: Can the system maintain real-time inference performance, ensuring that intrusion detection and alerting occur within an acceptable latency threshold (below 2 seconds)?

RQ4: How reliably can the system automate security breach responses, including triggering alarms and sending notification emails without human intervention?

RQ5: To what extent does the developed web-based interface allow homeowners to easily manage security alerts, monitor live feeds, and update the face recognition database securely?

1.5. Hypothesis

This system improves intruder detection and response times compared to traditional security methods. Unlike conventional systems, it analyzes real-time video feeds to identify intruders more accurately. TinyML enables efficient on-device processing, ensuring faster threat detection without relying on cloud systems. Deep learning models distinguish between known individuals and disguised intruders,

enhancing security accuracy. Automated alerts and alarms provide a proactive response, making home security more reliable.

1.6. Significance of the Study

The TinyML-Based Intelligent Home Security System leverages lightweight machine learning models optimized for edge devices to ensure efficient real-time intruder detection. By integrating advanced computer vision algorithms and deep learning architectures, the system can accurately differentiate between authorized users and unauthorized individuals, thus reinforcing access control mechanisms.

Equipped with motion detection sensors and a facial recognition module, the system intelligently filters out familiar household members to minimize false positives. Upon identifying an unknown individual, it triggers an immediate alert to the homeowner or designated security endpoint, enabling rapid incident response.

Since the system runs on Tiny-ML, all processing happens locally on the device, reducing internet dependence and improving response speed. It also keeps a digital record of security events, allowing homeowners to review past activities and improve safety measures.
By providing accurate intrusion detection and automated monitoring, this system offers a reliable and efficient home security solution.

1.7. Performance expectations

To validate the effectiveness of the system, performance benchmarks were established. These include achieving over 90% accuracy in intruder detection, maintaining inference latency below 2 seconds, and minimizing false positives caused by environmental noise or low-light conditions. These expectations guided the training and evaluation of the system models.

1.8. Organization of the study

Chapter one gives an introduction of the research which includes the background of the study and its motivation, study objectives, hypothesis of the study, the limitation of the study, the interest and the conclusion.
Chapter two discusses related works that were carried out before, the gaps founded in existing works

and how this research is going to improve and fill the existing research gaps. Chapter three is the research methodology. Discusses on summary of the research methods that will be used in this work and describe the requirements needed in this work.

Chapter four focuses on system analysis and design that includes all theories used in this research. Chapter five deals with the presentation, analysis and interpretation of results using graphs and screen shoots.

Chapter six finalize the research with a conclusion and recommendation for the future work.

Chapter 2: LITERATURE REVIEW

A literature review is a thorough assessment and synthesis of previous studies and scholarly works on a particular subject or research question. It summarizes important conclusions, theories, methods, and empirical data from a range of sources, including books, conference papers, and scholarly publications. The review critically analyzes the strengths and weaknesses of the literature, identifies gaps or inconsistencies, and highlights areas where further research is needed [11]**.** By synthesizing current knowledge, a literature review helps researchers establish the theoretical foundation for their studies, understand the evolution of ideas in the field, and propose new directions for future research or practical applications.

This chapter revised the related work of this project and provides critical analysis of what other researchers have said on the subjects where my project fit in. This overview serves as the root that leads to the successful design and implementation of an intelligent home security system using computer vision and deep learning.

2.1. Review of related works

2.1.1. Deep Learning in Home Security system

Recent years have witnessed significant progress in applying deep learning for real-time intruder detection in home security systems. Dalal et al. (2024) presented a YOLO-based approach that leverages transfer learning and quantization techniques to achieve high accuracy (98.87%) in object detection under resource constraints. Their work demonstrates the feasibility of deploying sophisticated deep learning models on edge devices for monitoring complex environments. However, while their study focuses on improving detection accuracy, it primarily addresses static scenarios and does not fully integrate behavior analysis for dynamic intruder identification [12].

In a related effort, Taiwo et al. (2022) developed an intelligent smart home control system that integrates deep learning models for real-time surveillance. Their system employs advanced computer vision techniques to perform facial recognition and object detection. Although their work improves upon conventional threshold-based detection, it still falls short in addressing intruders who may disguise themselves or mimic authorized behaviors. This gap highlights the need for incorporating robust anomaly detection mechanisms that analyze behavior over time, a challenge that our proposed system aims to address[2].

2.1.2 TinyML and Edge AI in Surveillance

Shifting the focus toward model deployment, Huang et al. (2021) provided an extensive survey of Tiny-ML, outlining state-of-the-art techniques, challenges, and future directions for deploying machine learning on resource constrained devices[13]. Their work emphasizes the potential of TinyML to reduce latency and dependency on cloud connectivity by performing computations locally. This is especially beneficial for home security applications where real-time threat detection is critical. However, most existing studies focus on either the implementation of TinyML models or their general application in IoT, without exploring the integration of behavior analysis and multi-modal sensor data, a gap that our thesis seeks to fill.

2.1.3 Behavior and Anomaly Detection in Smart Homes

Beyond conventional object and face detection, several recent studies have begun to explore behavior analysis to improve intruder detection. For example, Yamauchi et al. (2020) investigated anomaly detection in smart home operations by analyzing user behaviors and environmental conditions[10]. Their approach uses deep learning to distinguish between normal and anomalous activities, offering a promising avenue for reducing false negatives in security systems. Nevertheless, the application of such behavior analysis in conjunction with advanced object detection frameworks like YOLO has not been fully explored, particularly on edge devices using TinyML. Integrating these techniques could enhance the system's capability to identify intruders who attempt to mimic authorized individuals by wearing disguises or masks.

2.1.4 Integration Gaps and Security Limitations

Unlike previous studies focused on visual input, Mishra et al. (2024) developed a smart security solution using TinyML for keyword spotting in smart homes[14], effectively deploying real-time audio recognition on microcontrollers with limited power. Though this approach is effective in voice-activated systems, it does not incorporate visual intelligence or threat classification based on visual inputs. My work builds on this limitation by focusing on real-time visual recognition and behavior-based threat detection.

Focusing on embedded vision-based systems, Bounabi et al. (2024) presented an intelligent security system that utilizes CNNs on Arduino GIGA R1 boards[15], achieving high performance in detecting potential intrusions in building environments. Their research validates the feasibility of embedding CNN-based models on low-cost hardware for surveillance purposes. However, their solution focuses mostly on presence detection without consideration for behavioral anomalies, weapon carrying, or identity disguises. Our work extends this by incorporating these deeper layers of security intelligence

Exploring behavior detection through a different lens, Abba et al. (2024) proposed a deep learning-based anomaly detection model integrated with IoT consumer devices[16], designed to detect unexpected behaviors in real-time using TinyML frameworks. While this solution focuses on behavior anomalies, it lacks integration with computer vision and object recognition techniques. My project bridges this divide by combining behavioral and visual cues to detect complex intrusion scenarios such as masked impersonators and armed individuals. In a distinct application domain, Viswanatha et al. (2023) designed a smart security system for agricultural fields using embedded TinyML to detect animal intrusions[17]. Their work exemplifies the efficiency of TinyML in resource-constrained environments but remains limited in scope to outdoor non-human threats. Our system, while inspired by their energy-efficient framework, adapts the concept for indoor human-centric applications, enabling detection of disguised intruders in household contexts

Although designed for office environments, the approach by Rajeshkumar et al. (2023) aligns closely with domain of smart home security. They proposed a smart office automation system that employs Faster R-CNN for facial recognition and access control [18]. Their system successfully enables employee identification and door automation, demonstrating the effectiveness of deep learning for secure access in controlled environments. However, it does not address more complex intrusion scenarios such as impersonation, identity spoofing, or behavioral threats. Our system adapts and extends this concept to home settings by incorporating anti-spoofing mechanisms and behavior-based threat classification, thereby enhancing security against disguised or armed intruders.

2.1.5 Summary and Research Gaps

In summary, existing literature provides strong evidence of the benefits of deep learning and TinyML in smart home security applications. However, gaps remain in behavior analysis, identity spoofing detection, and the integration of multimodal data on resource-constrained devices. This study aims to address these gaps by designing a holistic, efficient, and behavior-aware security system optimized for edge devices.

Chapter 3: RESEARCH METHODOLOGY

This chapter details the step-by-step approach used to design and implement the TinyML-powered Intelligent Home Security System aimed at detecting intrusions through computer vision. It explains the methodology, including the selection of tools, data acquisition and processing methods, integration of hardware and software, and the metrics used for performance evaluation. The purpose is to illustrate how the system was developed, tested, and validated in alignment with the objectives outlined in Chapter 1.

3.1 Research Design

The research utilized an applied experimental design to develop a functional prototype of the system. This design enabled hands-on implementation, iterative testing, and empirical evaluation. The experimental methodology was ideal for validating the effectiveness of using TinyML and deep learning models on edge devices like Raspberry Pi.

The process began by identifying user requirements and functional specifications. System modeling followed a layered architecture approach, after which hardware components were procured and integrated. Data was collected using installed sensors and cameras. Then, deep learning models were trained, optimized, and deployed to the Raspberry Pi. Finally, system testing was conducted in real-world scenarios to validate the model's predictions and system responsiveness.

The design phases included:

- Requirement gathering and problem definition
- Prototyping and iterative development
- Training and testing of machine learning models
- Evaluation using statistical and classification metrics

This design ensured flexibility, repeatability, and accuracy in validating the system's functionalities.

3.2. Experimental Environment and Implementation

The experimental environment included a Raspberry Pi 5 (4GB RAM) running Raspberry Pi OS (64-bit Bookworm), interfaced with a Logitech C270 HD webcam, a PIR motion sensor, a light-dependent resistor (LDR), and a 5V buzzer for alerts. The system was programmed using Python 3.11, leveraging libraries such as OpenCV for image processing, TensorFlow Lite for machine learning inference, YOLOv8 for object detection, and Django for building the web interface. This environment was specifically chosen to simulate realistic home security conditions while enabling complete edge processing without reliance on cloud computing.

To ensure reproducibility, the complete experimental setup including hardware specifications, software versions, and random seeds used during model training was thoroughly documented. Deployment scripts and installation instructions were also prepared to facilitate independent replication of the study.

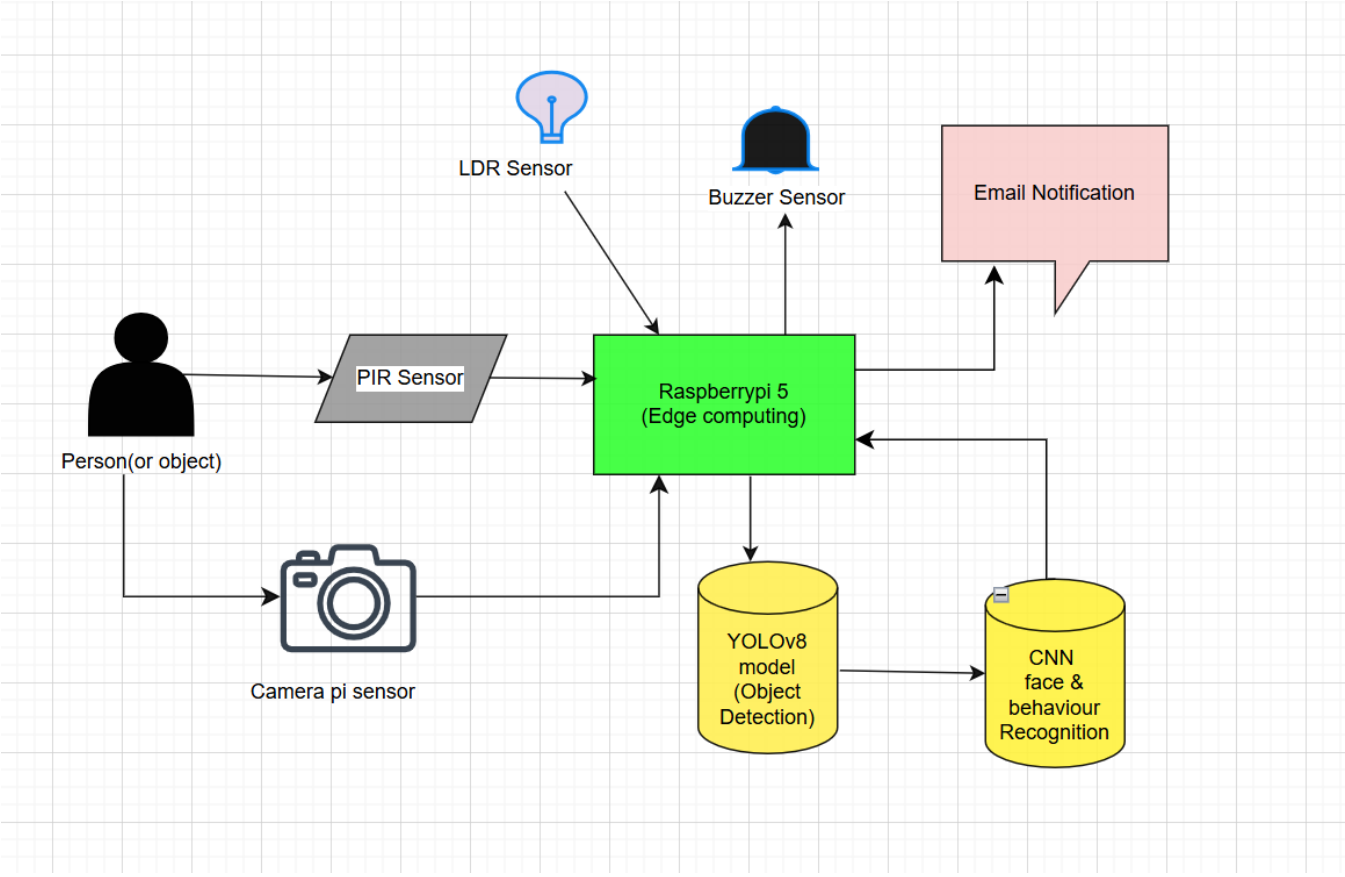

Figure 1: Block Diagram of the System

3.3 Data Collection and Preprocessing

3.3.1 Primary Data Collection

Primary data was collected by deploying the system in a home-like environment. Various scenarios were staged to capture images and videos, including authorized individuals entering and exiting the premises, unauthorized persons attempting to intrude, different lighting conditions such as daytime and nighttime, and different environmental settings for indoor testing.

3.3.2 Secondary Data Collection

a. Labeled Faces in the Wild (LFW)

In this stage, I used an online dataset called **Labeled Faces in the Wild (LFW)** [19] as a source of secondary data. I selected this dataset to complement the images and videos I collected through real-world testing. The main reason for choosing LFW was to strengthen the model's ability to recognize faces under a variety of real-life conditions, which would be difficult to fully simulate in a home-like environment.

Labeled Faces in the Wild (LFW) is a widely used, publicly available dataset created to support the development and testing of face recognition systems in natural, real-world settings[20],[21],[22]. It contains a total of 13,233 facial images collected from the internet, featuring 5,749 different individuals. Among these, 1,680 individuals appear in two or more images, making the dataset suitable for both face detection and face verification tasks.

The dataset is known for its diversity and realism, as the images were captured under a variety of uncontrolled conditions. These include differences in lighting, facial expressions, background scenes, and viewing angles. This makes LFW particularly valuable for evaluating how facial recognition models perform outside of ideal or laboratory conditions.

By using this dataset, I was able to expose my system to a wide range of facial features and scenarios that go beyond those found in the primary data I collected. This helped me improve the model's generalization capability, especially in recognizing both known and unknown individuals under inconsistent conditions.

b. COCO Dataset for Object Detection

In addition to the facial datasets used in this project, I also employed the COCO (Common Objects in Context) dataset[23],[24],[25] to support the training and evaluation of the object detection module within the intelligent home security system. The primary goal of using COCO was to improve the system's ability to detect and recognize various objects that may be present in a home environment, particularly those that could indicate suspicious activity, such as bags, tools, or even the presence of multiple individuals.

The COCO dataset is a widely recognized benchmark in computer vision research. It contains over 330,000 images, including more than 200,000 labeled images, with around 1.5 million object instances. Each object in the dataset is annotated with bounding boxes, segmentation masks, and category labels, covering 80 different object classes, such as people, vehicles, household items, and animals.

The diversity and complexity of scenes in the COCO dataset made it particularly useful for training models that need to operate under real-world conditions. Many images contain multiple objects in cluttered backgrounds, with variations in size, orientation, and occlusion features that are often found in practical surveillance scenarios.

By integrating COCO into the training process, I was able to:

- Enhance the system's capability to detect objects beyond faces, contributing to a broader understanding of the environment.
- Improve the system's response to potential threats, such as detecting a person carrying a suspicious item or moving in a restricted area.
- Provide the model with a robust, diverse dataset to reduce over fitting and improve performance on unseen data.

In summary, the COCO dataset played an important role in increasing the versatility and accuracy of the object detection component, making the intelligent home security system more reliable in identifying a wide range of visual cues associated with intrusion or abnormal behavior.

3.3.3 Data Preprocessing

Data preprocessing refers to the initial stage in the machine learning workflow, where raw data is modified and structured to make it suitable for model training. This process typically includes cleaning, formatting, and

organizing the data to ensure uniformity and to eliminate irrelevant or inconsistent elements. Effective preprocessing enhances the model's ability to learn from the data, shortens the training time, and contributes to higher prediction accuracy[26].

In this project, specific preprocessing techniques were employed to refine the dataset used in training and evaluating the intelligent home security system. These steps ensured that the input data was clean, consistent, and suitable for accurate model learning and validation.

- **Face Detection**[27]**.**

Face regions were detected using the **YOLOv8 Nano (YOLOv8n)** model, which provides accurate and real-time detection suitable for edge devices. The model was either pre-trained on a face-specific dataset or fine-tuned to improve facial detection accuracy in varied conditions. Detected face regions were cropped and used as inputs for further processing.

- **Image Resizing**

All cropped face images were resized to **224×224 pixels** using the OpenCV library to ensure consistency across the dataset and compatibility with the deep learning model input layer. The dimension of 224×224 pixels was chosen as it is the standard input size for many pre-trained CNN models and provides a good balance between computational efficiency and feature preservation.

- **Normalization**

The pixel intensity values were rescaled to fall within the [0, 1] range by dividing each value by 255. This normalization step was important for speeding up the training process and enhancing the overall performance of the model.

- **Data augmentation**

To improve model generalization and reduce over fitting, several augmentation techniques were applied, including:

  - Horizontal flipping
  - Random rotation within ±20 degrees
  - Brightness and contrast adjustments

- Random zooming

The choice of ±20 degrees for rotation and other augmentations reflects typical variations found in real-life images, helping the model generalize better to unseen faces or body poses.

- **Dataset Splitting:**

To ensure a balanced and effective training process, the entire dataset was divided into three distinct subsets:

- **Training set (70%)**: Used to train the YOLOv8 and CNN models to learn meaningful patterns and features from the input data.
- **Validation set (15%)**: Used during training to evaluate intermediate model performance and fine-tune hyper parameters.
- **Test set (15%)**: Reserved for final evaluation of the trained models to assess their accuracy and generalization on unseen data.

To maintain fairness and representativeness, the dataset was split using stratified sampling. This method ensures that all categories, such as “authorized,” “unauthorized,” “masked,” and “armed” individuals, are proportionally represented in each subset. This approach reduces bias, improves model reliability, and ensures that no class dominates any specific portion of the data.

3.4 Model Training and Optimization

3.4.1 YOLOv8 Training for Object and Behavior Detection

YOLOv8 was used as the core model for detecting individuals and classifying potential threats based on visual cues. A custom-labeled dataset was prepared containing images annotated with bounding boxes for key classes, including “Family Member,” “Unknown Person,” “Masked Person,” “Gun,” “Knife,” and “Panga.” Each object was assigned a specific label to help the model learn not only to detect human presence but also to identify suspicious behavior such as carrying weapons or wearing masks.

The model was trained on a GPU-powered system using transfer learning, starting from pre-trained YOLOv8 weights and fine-tuning them on the custom dataset. After training, the model was optimized through quantization to reduce size and improve inference speed, enabling real-time performance on a Raspberry Pi.

3.4.2 CNN Training for Face Recognition

A separate Convolutional Neural Network (CNN) was developed for face recognition to help distinguish authorized family members from unknown individuals. The architecture included convolutional layers with ReLU activation, followed by pooling layers and fully connected dense layers. The final output layer used a softmax activation function to produce a probability distribution across the two classes: “Authorized” and “Unauthorized.”

The CNN was trained on cropped face images captured from the camera, and normalized and resized to a standard dimension (e.g., 224×224). Data augmentation techniques were applied to improve generalization.

3.4.3 Behavior Classification Using Object Detection

Rather than using traditional SVMs with handcrafted feature vectors, behavior classification was integrated directly into the YOLOv8 detection pipeline. The model was trained to identify behaviors such as:

- Carrying a weapon (gun, knife, or panga)
- Wearing a mask to disguise identity

These behaviors were treated as visual object classes, allowing the system to detect and classify them directly from the input image. This method provided both classification and localization in one step, improving efficiency and accuracy. The model was trained to differentiate between normal scenarios (e.g., unarmed family members) and suspicious behaviors (e.g., masked intruders with weapons).

3.5 Evaluation Metrics and Thresholds

The performance of the system was evaluated using standard metrics: accuracy, precision, recall, F1-score, and inference time per frame. Formal definitions of these metrics will be presented in Chapter Five. Target thresholds were set as follows: at least 90% accuracy and F1-score, over 92% precision, over 90% recall, and inference latency below 2 seconds per frame. Meeting these thresholds was crucial to validate the system’s effectiveness for real-world deployment.

3.6. Study Limitations and Assumptions

The system was tested primarily within a single-household indoor environment, which may limit the generalizability of the results to larger or more complex residential settings. Additionally, due to safety

constraints, some samples for weapon detection were synthetically generated, potentially introducing domain shifts. Night-time evaluations relied on auxiliary LED lighting rather than fully passive infrared imaging, which may not fully replicate extreme low-light conditions. Furthermore, it was assumed that the Raspberry Pi system would have an uninterrupted power supply through a UPS module and that stable local network connectivity would be available to support the web interface and alert notifications.

3.7. Ethical Considerations and Data Privacy

Given that the project involved capturing images and videos of individuals, ethical compliance was mandatory. All data collection was carried out with full informed consent obtained from participants. Consent forms explaining the scope of the project were signed by all individuals involved. To ensure privacy, all collected data was stored securely in encrypted storage media and was not uploaded to any cloud platforms. This project adhered to Rwanda's Data Protection and Privacy Law N° 058/2021.

Chapter Four: SYSTEM ANALYSIS AND DESIGN

4.1 Introduction

The purpose of this chapter is to clarify how each component, both hardware and software, contributes to achieving a secure, real-time, and efficient intrusion detection mechanism. The design approach is centered on deploying deep learning and computer vision models on a resource-constrained Raspberry Pi device to monitor and protect household environments. This chapter also presents a layered architecture, data flow diagrams, use case models, interface logic, and behavioral workflows.

4.2 System Requirements Analysis

The development of the intelligent home security system began with a comprehensive analysis of both functional and non-functional requirements to ensure alignment with real-world security demands, particularly those emerging in residential environments.

From a functional perspective, the system is designed to detect motion using a PIR sensor, capture live video through a camera, and analyze the captured frames using lightweight deep learning models deployed on a Raspberry Pi 5. These models are tasked with identifying individuals and classifying their behavior. Specifically, the system performs two critical functions: object detection using a YOLOv8 model, which is trained to recognize suspicious objects such as weapons (e.g., knives, guns, pangas), and facial recognition using a CNN model that distinguishes between authorized family members and unknown individuals.

To address more complex intrusion attempts, the system is also capable of identifying individuals disguised to mimic homeowners for instance, wearing masks resembling family members, by incorporating behavior analysis into the object detection pipeline. The model is trained to flag such disguised individuals based on the presence of face-covering accessories. If the system detects a person carrying a weapon or behaving suspiciously, even if the face resembles a known individual it raises an alert and triggers appropriate response mechanisms.

The non-functional requirements of the system emphasize performance, energy efficiency, privacy, and user experience. Inference must occur in near real-time (typically under two seconds) to allow for immediate response. The use of TinyML ensures that the entire system operates on low-power edge devices without cloud dependency, thus improving speed and maintaining data privacy. Lastly, the web interface must remain intuitive and secured via authenticated access, ensuring homeowners can easily manage logs, alerts, and the face database remotely over a local network.

4.3. System Design Methodology

The development of my TinyML-based intelligent home security system was guided by a modular and layered system architecture, which is widely recognized for enhancing system maintainability, scalability, and flexibility. This architecture allowed me to separate the system into distinct functional components, namely sensors, machine learning models, and a web interface, each developed and tested independently while ensuring seamless interoperability.

To manage the iterative nature of building and optimizing such an embedded AI system, I adopted the Agile development methodology. Agile supports rapid prototyping, continuous testing, and regular feedback, which is particularly effective for projects involving machine learning model tuning, hardware integration, and real-time behavior testing. This iterative approach was essential for evaluating model performance, minimizing power consumption, and ensuring stable operation on low-power devices like the Raspberry Pi 5.

Agile Life Cycle Diagram

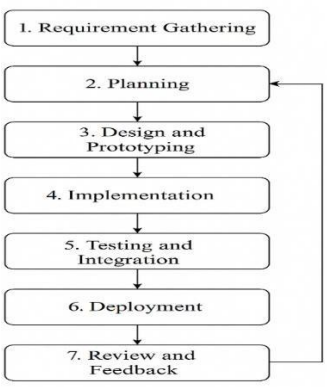

Figure 2: Agile Development Cycle

The system design process was carried out in four structured stages, each serving a critical role in guiding both the hardware and software development:

a. Requirement Gathering

In the first stage of development, I focused on identifying the system's key functional and non-functional requirements. Functionally, the system needed to detect motion using a Passive Infrared (PIR) sensor and capture live video using a Camera Pi module. It also had to process this video data using lightweight deep learning models specifically, a YOLOv8 model for object detection and a Convolutional Neural Network (CNN) for facial recognition. These models needed to detect weapons (such as knives, pangas, or guns), recognize authorized individuals, and classify suspicious behavior. Non-functional requirements included a response time of less than two seconds, minimal energy consumption, and the ability to operate offline without internet dependency. These requirements were based on a review of related academic literature, the limitations of traditional security systems, and the practical needs of residential users, especially in low-resource environments.

b. System Modeling

After establishing the requirements, I proceeded to model the system architecture and workflows. I created Use Case Diagrams to define how users such as homeowners would interact with the system, including logging in, viewing alerts, and managing facial data. Data Flow Diagrams (DFDs) were used to map out how data moves between sensors, machine learning models, and output devices. A System Architecture Diagram was also developed to show how all components hardware and software connect through the Raspberry Pi 5 platform. Additionally, I used flowcharts and pseudo code to lay out the logical steps the system follows when detecting motion, analyzing video, and responding to threats. These models ensured that all modules would communicate effectively and that the system would behave as expected.

c. Algorithm Design

To implement the core functions of detection and classification within the proposed intelligent home security system, two lightweight deep learning models were developed and optimized: a YOLOv8n model for real-time object and threat detection, and a Convolutional Neural Network (CNN)-based model for facial recognition. These models worked collaboratively to distinguish between authorized individuals (family members) and unauthorized individuals (intruders), including those wearing masks or carrying weapons. The YOLOv8n architecture comprises three main components: a backbone built with C2f modules and convolutional layers for extracting essential image features; a neck incorporating a Path Aggregation Network

(PAN) to fuse and refine multi-scale features; and a decoupled head that performs classification, bounding box regression, and objectness scoring. For loss functions, the model employs Complete IoU (CIoU) for localization and Binary Cross-Entropy (BCE) for classification. The model was trained on a custom dataset comprising 1,100 images, including 500 images of family members and 600 images of intruders (categorized into unmasked, masked, and armed individuals). Training parameters included an input size of 224×224 pixels, a batch size of 16, 100 training epochs, and the use of the Adam optimizer with a learning rate of 0.001. Data augmentation techniques such as mosaic transformations, horizontal flipping, and brightness adjustments were applied to enhance generalization and robustness.

For the facial recognition component, a custom CNN model was developed for binary classification to determine whether an input face belongs to an authorized user. The model architecture consists of multiple convolutional layers for spatial feature extraction, followed by max pooling layers for dimensionality reduction, and fully connected layers for decision-making. The ReLU activation function was used to introduce non-linearity, while a sigmoid output layer facilitated binary classification. Face images were preprocessed to 224×224 pixels before training. The model was specifically designed to operate efficiently on a Raspberry Pi 5 with minimal latency.

To ensure the feasibility of real-time inference on the Raspberry Pi 5, both models were optimized using TinyML techniques. These included 8-bit post-training quantization to reduce model size and memory usage, and structured pruning to eliminate redundant or low-impact parameters. The models were subsequently converted to TensorFlow Lite format for compatibility with edge devices. These optimization strategies enabled the system to achieve high detection accuracy while maintaining low power consumption and rapid response times. The effectiveness of the models is further evaluated using standard metrics in Chapter 5.

d. Interface Design

In the final stage, I designed a web-based user interface that allows the homeowner to interact with the system. The interface was developed using the Django framework and was designed to operate locally without internet access, thereby preserving data privacy. Through this interface, users can monitor security alerts, view detection logs, and manage the database of known faces. I implemented user authentication to secure access,

ensuring that only authorized individuals could modify settings or view sensitive information. The interface was designed to be lightweight and responsive, making it accessible from both desktop and laptop device. This ensured a practical and user-friendly experience for non-technical users while maintaining security and functionality.

4.4. System Architecture Design

The system architecture is structured into three distinct operational layers: sensing, data processing, and response. The sensing layer comprises the PIR sensor, LDR sensor, and Camera Pi module, which continuously monitor environmental motion, light intensity, and capture visual data respectively. Information collected from these devices is passed to the data processing layer, where the Raspberry Pi 5 coordinates local inference tasks. At this stage, object detection is performed using the YOLOv8 model, while facial and behavioral analysis are conducted through a dedicated CNN model. The combination of these two models allows the system to accurately distinguish between authorized individuals and suspicious intruders, including those attempting to disguise their identity. Based on the inference outcomes, the response layer is activated, triggering alerts via the buzzer, sending email notifications to the homeowner, logging security events into the database, and updating the user-accessible web interface. This layered architecture ensures modularity, real-time responsiveness, and scalability for future system enhancements.

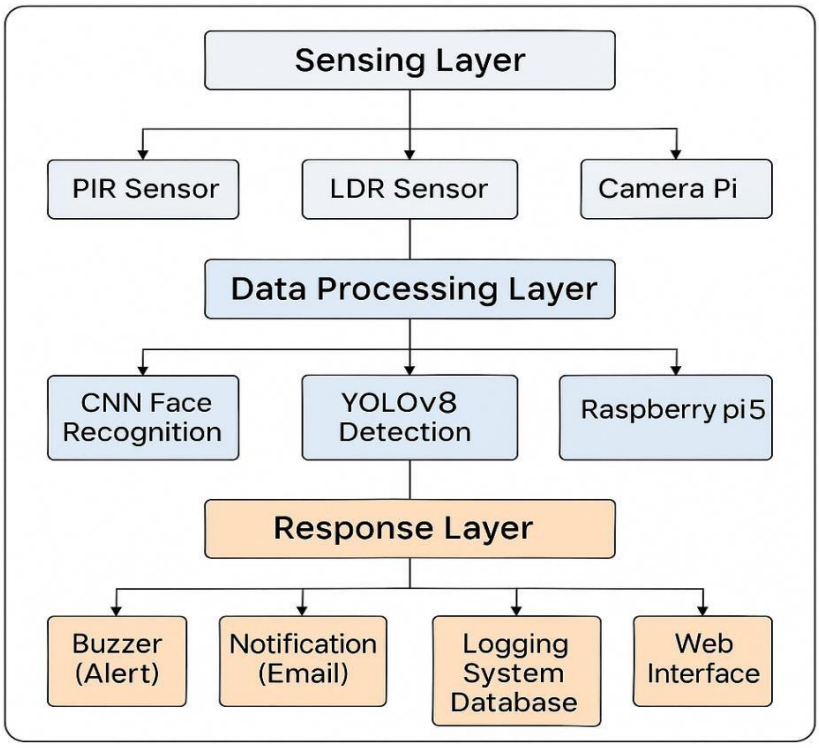

Figure 3: Layered system Architecture Diagram

4.5. Hardware and Software Components of the System

4.5.1. Hardware

4.5.1.1. Raspberry Pi 5 (4GB RAM):

The Raspberry Pi 5 with 4GB RAM is a compact, energy-efficient computing board used as the main processing unit in the proposed TinyML-based intelligent home security system. It handles key tasks such as running optimized machine learning models, processing data from cameras and sensors, and performing computer vision operations essential for intruder detection.

This device was selected for its efficient performance, affordability, and suitability for edge computing. With a quad-core Arm Cortex-A76 processor (up to 2.4 GHz) and 4GB LPDDR4X RAM, the Raspberry Pi 5 enables real-time image analysis and behavior recognition while reducing reliance on cloud services. This approach ensures faster response times and improved data privacy.

Notable features of the Raspberry Pi 5 include:

- Dual camera support for broader surveillance or combining visual and thermal imaging.
- GPIO pins for connecting motion sensors or alarms.
- USB 3.0 ports for external cameras or storage.
- PCIe interface for expandable storage using SSDs.
- Wired and wireless connectivity for flexible deployment and remote monitoring.

Its integration into the system supports local inference of TinyML models, allowing for immediate detection and alerts when suspicious individuals are identified within the monitored space.

Figure 4: Raspberry Pi 5 4GB RAM

4.5.1.2. Logitech C270 HD Webcam

The Logitech C270 is a high-definition USB webcam designed for video capture, video conferencing, and computer vision applications. It offers 720p resolution, making it suitable for real-time monitoring and general-purpose imaging tasks.

Description (For Thesis Context):

In the context of the TinyML-based intelligent home security system, the Logitech C270 HD Webcam serves as a visual input device for capturing live video feeds used in intruder detection. This webcam supports 720p HD resolution at 30 frames per second, delivering clear and consistent video quality suitable for computer vision tasks such as face detection, motion tracking, and behavior analysis.

The C270 connects via USB 2.0, ensuring easy integration with edge devices such as the Raspberry Pi 5. Its built-in microphone, though not essential for visual detection, can support future enhancements involving

audio analysis. Its compact form factor and flexible mounting clip allow it to be easily positioned for optimal field of view in various indoor settings.

Key features include:

- 720p HD video capture at 30 fps
- Automatic light correction for varying lighting environments
- USB plug-and-play compatibility
- Compact design with universal clip mount

Its affordability, ease of use, and compatibility with open-source libraries make it an ideal camera for implementing real-time surveillance in smart home security systems.

Figure 5: Logitech C505e HD Business Webcam 720p, long range mic Wired

4.5.1.3. PIR (Passive Infrared) Sensor

The Passive Infrared (PIR) sensor is a motion detection component commonly used in electronic security systems. It identifies movement by sensing variations in infrared radiation, which is naturally emitted by warm objects such as the human body. Unlike active sensors, it does not emit any energy; instead, it passively monitors the environment for changes in thermal radiation.

In this thesis, the PIR sensor is employed to detect motion within the monitored area and serve as a trigger for further actions in the intelligent home security system.

The key advantages of using a PIR sensor in this system include:

- Low power consumption, making it suitable for continuous operation.
- Compact size and ease of integration with embedded systems.
- Quick response time, allowing real-time detection and system activation.
- Cost-effectiveness, which supports affordability in smart home solutions.

The PIR sensor's ability to provide reliable motion detection without the need for constant visual monitoring makes it a critical component in enhancing the overall efficiency and responsiveness of the proposed intelligent security system.

Figure 6: PIR Motion sensor SEN51, R12

4.5.1.4. Light Sensor (LDR)

The Light Dependent Resistor (LDR) is a crucial sensor integrated into this project for ambient light detection. It functions by altering its electrical resistance in response to light intensity. Under strong illumination, the resistance drops, enabling higher current flow. In contrast, in darker conditions, the resistance rises, limiting the current. This property allows the system to sense lighting changes and automatically trigger functions such as activating security cameras or switching on lights.

LDR sensor is utilized in my project to monitor environmental lighting conditions and support adaptive system behavior. For example, it can help the system determine whether it is day or night and trigger infrared or thermal camera activation during low-light periods. This functionality ensures that image capture remains clear

and effective during nighttime or in poorly lit areas, thereby enhancing the accuracy of computer vision-based intruder detection.

Key features of the LDR sensor include:

- Automatic illumination control, which improves visibility for cameras without manual input.
- Easy integration with the Raspberry Pi through GPIO interfaces or analog-to-digital converters.
- Low power requirements, suitable for continuous operation.
- Cost-effective and responsive to rapid changes in light levels.

By incorporating an LDR sensor into the system, the home security application becomes more intelligent and responsive, adjusting its sensing and alert mechanisms based on the surrounding light levels. This contributes to improved accuracy in motion and intruder detection, especially in low-visibility scenarios.

Figure 7: LDR Light Dependent Resistor

4.5.1.5. Buzzer

The **buzzer** is used to produce a sound alert when the system detects unusual activity or unauthorized movement. This loud sound helps to scare away intruders and also informs the people in the building that a security issue may be happening.

Figure 8: Buzzer

4.5.1.6. Power Supply

A 5V/3A power adapter was used to operate the system, supported by an Uninterruptible Power Supply (UPS) module. This ensured the system remained functional even during temporary power interruptions.

Figure 9: Raspberry pi 27W USB-C Power Supply

4.5.1.7. SD Card (for Raspberry Pi 5 with 4GB RAM)

An SD card (Secure Digital card) serves as the main storage device for the Raspberry Pi 5, housing the operating system, installed software, and user data. It is a compact and removable memory device that allows the Raspberry Pi to boot and perform essential functions.

For this project, a microSD card is selected as the primary storage medium due to its seamless compatibility with the Raspberry Pi 5. It is responsible for holding essential system files, machine learning models, and collected data. To ensure reliable performance and efficient data handling, the following specifications are

recommended:

- Type: microSD
- Capacity: Minimum of 32GB, with larger capacities preferred to accommodate system and media files
- Speed Class: Class 10 or UHS-I, offering high read/write speeds necessary for smooth operation
- File System: Typically formatted in FAT32 to support Raspberry Pi boot requirements

While the Raspberry Pi 5 also supports booting from USB and PCIe-connected NVMe drives, the microSD card remains the most commonly used storage medium due to its simplicity, affordability, and ease of use, particularly in embedded systems and prototype development.

Figure 10: SD Card

4.5.2. Software System

Raspberry Pi OS: A Debian-based operating system designed for Raspberry Pi. It provided a lightweight but capable environment for development and real-time processing.

Python: Python was the primary language due to its support for libraries such as TensorFlow, OpenCV, NumPy, and Flask. Python scripts were used for sensor management, image processing, and web server integration.

OpenCV (Open Source Computer Vision Library): OpenCV enabled face detection, frame capture, image pre-processing, and video manipulation. Functions like cv2.VideoCapture, cv2.resize, and cv2.Canny were extensively used.

TensorFlow Lite: TensorFlow Lite is a lightweight deep learning framework for mobile and embedded devices. The object detection and face recognition models were trained using TensorFlow and converted to .tflite format for deployment on Raspberry Pi.

YOLOv8: YOLO (You Only Look Once) is a state-of-the-art object detection algorithm. YOLOv8 was chosen for its real-time speed and high accuracy. It processes entire images in one forward pass, making it highly efficient on resource-limited devices.

Django: A Python-based web framework used to create a secure and interactive user interface. It allowed the homeowner to view camera feeds, receive alerts, and manage recognition lists through a browser.

4.6. Behavioral Logic and Pseudo code Flow

The behavior of the system is best explained using event-driven pseudo code. The pseudo code outlines the system's operational logic from the moment motion is detected to when alerts are generated.

Pseudo code

```
START
Initialize Raspberry Pi, PIR sensor, LDR sensor, and camera module
WHILE the system is running:
  IF PIR sensor detects motion:
    Capture image frame using camera
    Run YOLOv8 model on captured frame to detect objects
     IF detected object is suspicious (e.g., weapon, mask):
       Crop face region from the frame
       Run CNN model for face and behavior recognition
        IF face is unknown OR individual is wearing a mask OR carrying a weapon:
          Activate buzzer alarm
          Log event details in the local database
          Send alert notification to homeowner (via email/SMS)
       ELSE:
          Log authorized user access in database
    ELSE:
```

Continue real-time monitoring

ELSE:

Remain in idle state and wait for next motion event

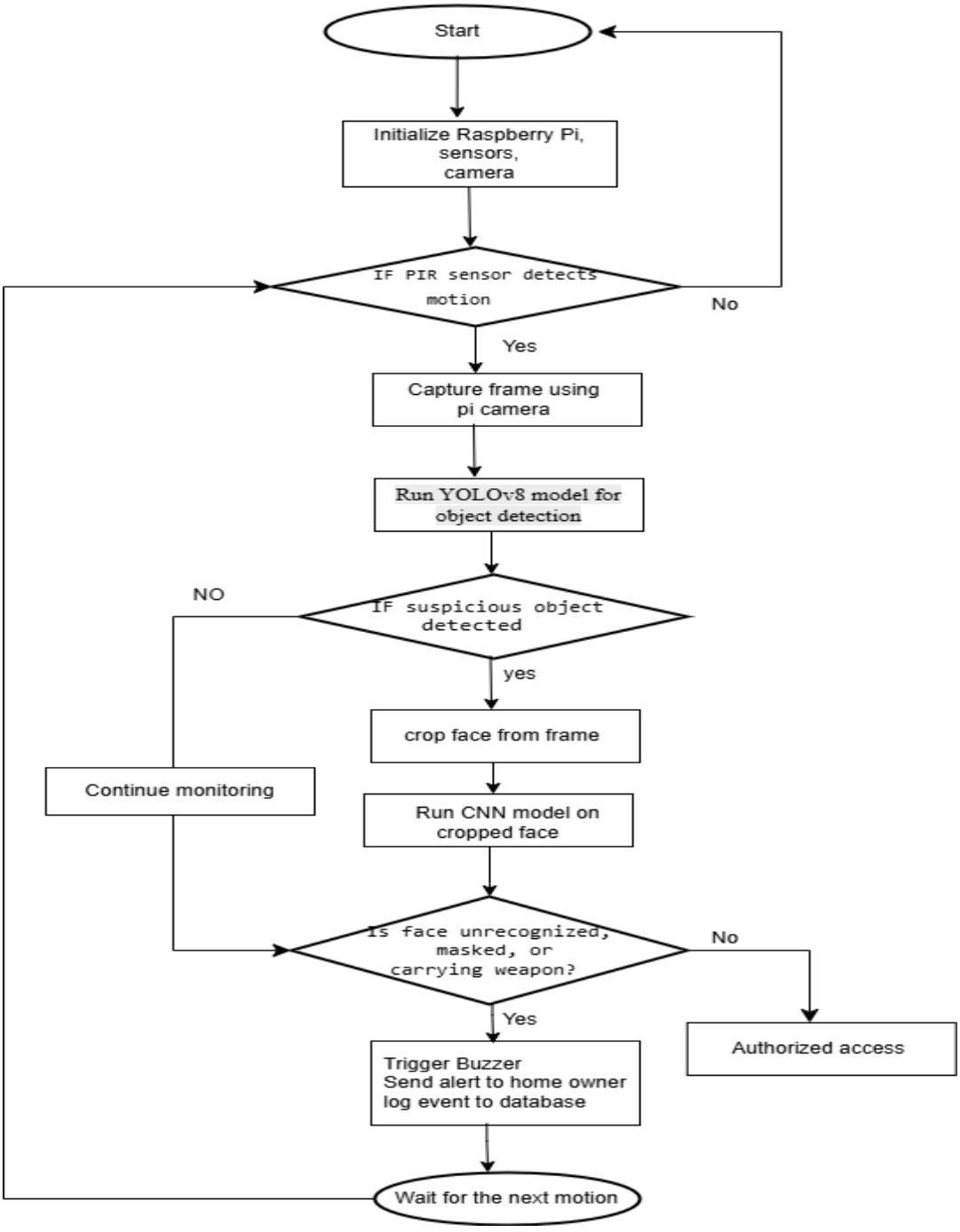

Figure 11: Flowchart diagram

4.7. Use Case and Data Flow Design

To support a clear understanding of system behavior and data movement, the use case model and the data flow diagrams (DFD Level 0 and Level 1) are used to visually represent the functional interactions and internal data processing pathways within the system.

The use case diagram (Figure 12) visualizes key interactions between the homeowner and the system, such as viewing real-time video, receiving alerts, and managing recognition logs. These user interactions are functionally detailed in Section 4.2.

The DFD Level 0 illustrates the high-level flow of information from sensors to processing and response, while DFD Level 1 (Figure 13) breaks down internal processes, including data acquisition, detection, and alerting. The logic and components represented in the DFDs are elaborated in Sections 4.4 (System Architecture Design) and 4.6 (Behavioral Logic and Pseudo code Flow).

These diagrams consolidate the key operational and logical components of the system into a visual format, enhancing clarity and providing a structured overview of how the system functions

Use Case Diagram

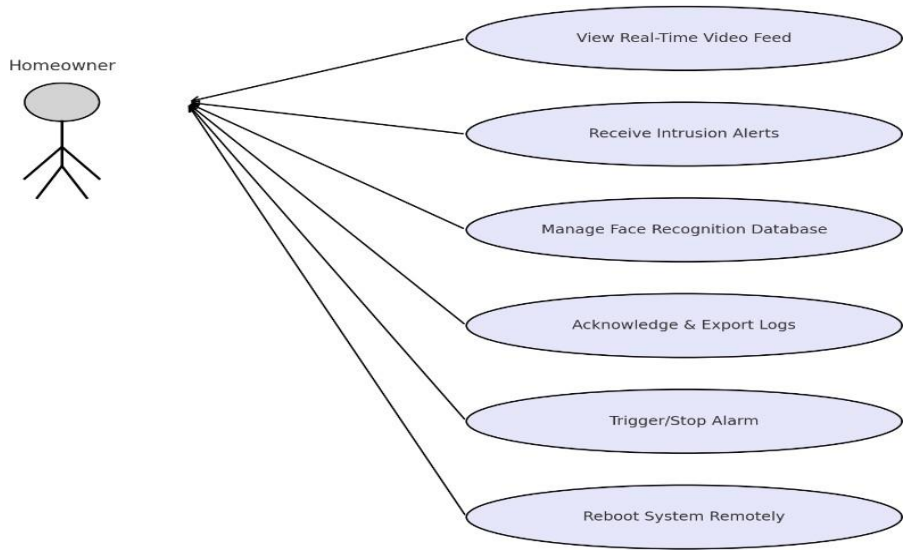

Figure 12: Use Case Diagram

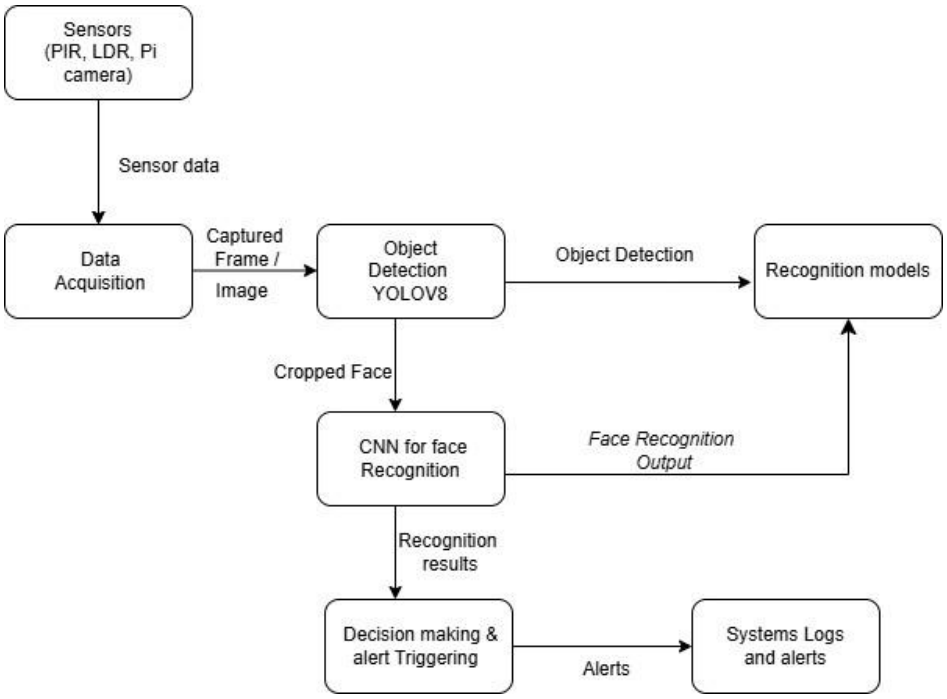

Figure 13: DFD Level 1

4.7. Interface Design and Web Architecture

The system includes both a backend API and a web interface. The Flask-based API handles real-time tasks such as image capture and frame processing. It exposes endpoints that receive sensor events and trigger inference engines. The Django framework provides a user-facing dashboard that enables the homeowner to monitor system activity, view live camera feeds, export logs, and manage the facial recognition database. All data is stored in a lightweight SQLite database for quick access and portability.

The web interface also supports user authentication via the Django Admin panel, enhanced with two-factor authentication. Face data is stored securely, and access logs are maintained for auditing purposes.

4.8. Security and Privacy Design

Given the sensitive nature of surveillance systems, privacy and data security were key considerations during the design phase. All data processing is confined to the local network, ensuring that no image or video data is

transmitted externally. The camera module is configured to disable itself after 10 seconds of inactivity, minimizing unnecessary surveillance. This design ensures compliance with user privacy expectations and prevents external data breaches.

4.9. Assumptions and Design Constraints

In designing the system, several constraints and assumptions were made. The Raspberry Pi is assumed to have continuous power through a UPS module, and it is assumed that a local Wi-Fi network is available for user interaction with the web interface. The system is constrained by the computational limits of the Raspberry Pi, which necessitates using lightweight models like YOLOv8 Nano and Tensor Flow Lite. Further, camera placement must be strategically chosen to ensure adequate face visibility and motion coverage.

Chapter 5: RESULT AND ANALYSIS

5.1 Experimental Setup

To evaluate the performance of the proposed System, a full prototype was built and deployed. The system was set up in a controlled indoor environment designed to simulate a real home. This approach helped ensure that the testing conditions closely reflected practical, everyday use.

The prototype consisted of a Raspberry Pi 5 (4GB RAM) as the main processor, a Logitech C270 HD webcam for video input, a PIR sensor for motion detection, an LDR sensor for light monitoring, and a buzzer for audible alerts. A web-based interface, allowed real-time monitoring, management of face recognition data, and system control during the experiments.

Through this experimental setup, I demonstrated that the system works as intended, moving beyond design to a practical, real-world solution ready for home security applications.

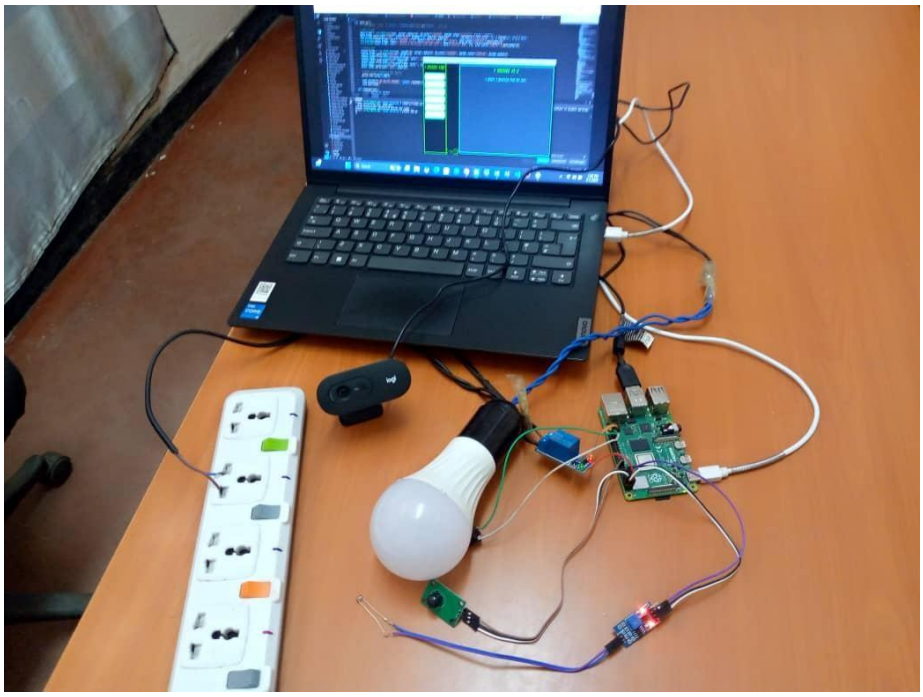

Figure 14: Prototype of the system

5.2 Test Scenarios

To properly evaluate the performance of the proposed security system, six real-life test scenarios were created to reflect common challenges faced in home security. The first scenario involved a known family member

entering during the day to confirm that the system could recognize authorized people without causing false alarms. The second tested whether the system could detect a stranger trying to enter in the same daylight conditions. In the third, a person wore a mask to see if the system could still detect suspicious behavior when a face is partly hidden. The fourth scenario involved someone carrying a visible weapon, such as a gun, knife, or panga, to check the system's ability to recognize serious threats. The fifth scenario tested performance at night, when low light could make it harder to detect faces and objects. Finally, the sixth scenario checked whether the system could still correctly identify a trusted family member entering in the dark. Together, these tests gave a complete and realistic view of how the system handles detection, recognition, and threat identification in different situations.

5.3 Data Collection and Datasets

For the evaluation of the system, data collected during the live experiments and public datasets were used. The primary data included images of authorized individuals and simulated intruders performing normal, disguised, and armed access attempts.

In addition, secondary datasets were used to improve model performance, including face images and object samples. In total, 1,100 images were prepared and processed for training and testing. The use of both real-world and external datasets helped to ensure that the system's evaluation was reliable and could generalize to different conditions beyond the controlled environment.

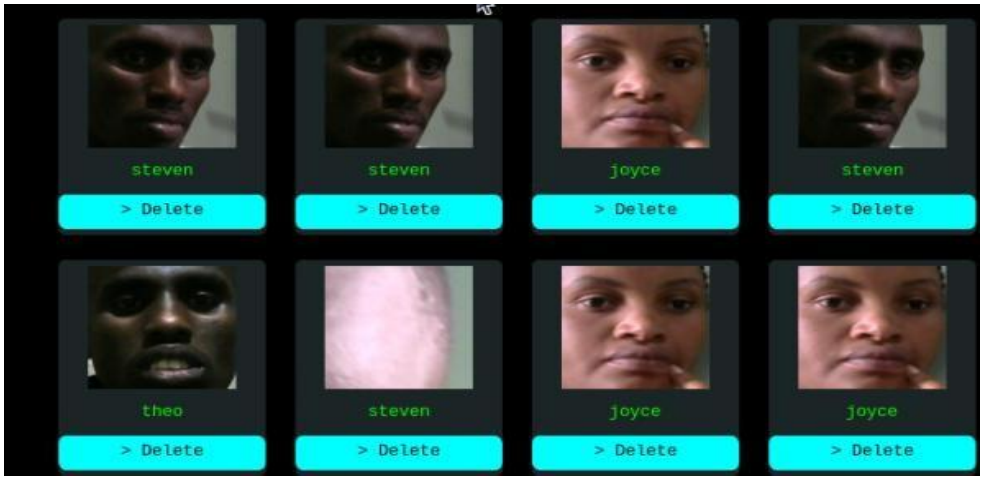

Figure 15: image captured using pi camera (source: own generate)

Figure 16: secondary data from Online sources
*(**Sources:https://www.tensorflow.org/datasets/catalog/lfw**)*

5.4 Evaluation Metrics

The system's performance was evaluated using widely accepted classification metrics to ensure an objective and comprehensive assessment. In evaluating the performance of the proposed system, four possible prediction outcomes were considered: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). A True Positive (TP) occurs when the system correctly identifies an actual intrusion, meaning it detects a real threat as expected. A True Negative (TN) happens when the system accurately recognizes a normal, non-intrusive event, confirming that no threat is present. A False Positive (FP) refers to a situation where the system incorrectly triggers an alert for a harmless event, mistaking it as an intrusion, which results in a false alarm. On the other hand, a False Negative (FN) occurs when the system fails to detect a real intrusion, allowing a threat to pass unnoticed, which is the most critical type of error for a security system. These four outcomes form the foundation for calculating the evaluation metrics such as Precision, Recall, F1-Score, Accuracy and Inference Time per frame, allowing a detailed assessment of the system's detection performance.

Precision measures how many of the detections made by the system were actually correct. It helps to reduce false alarms.

The formula for Precision is:

$$\text{Precision} = \frac{TP}{TP+TN}$$

Recall measures how many actual positive cases (intrusions or threats) the system successfully detected. It shows the system's ability to capture real threats.

The formula for Recall is:

$$\text{Recall} = \frac{TP}{TP+TN}$$

F1-Score combines both Precision and Recall into a single value, providing a balanced view of the system's performance.

The formula for F1-Score is:

$$\text{F1 Score} = \frac{2*(Precision*Recall)}{(Precision+Recall)}$$

Accuracy gives an overall measure of how many correct predictions the system made, including both detecting intrusions and confirming safe conditions.

The formula for Accuracy is:

$$\text{Accuracy} = \frac{TP+TN}{(TP+TN+\ FP+FN)}$$

In addition to classification metrics, the Inference Time per frame was measured to assess how quickly the system processes data and triggers an alert. A target of less than two seconds per detection cycle was set to ensure real-time responsiveness. Achieving low latency was important to allow timely warnings and quick reactions in real-world scenarios.

These evaluation metrics were applied to results obtained from both the primary experimental data (captured images and intrusion scenarios) and the secondary datasets (face and object samples) used in training and testing.

5.5 Experimental Results

An intelligent home security system was developed to distinguish between trusted family members and potential intruders. The model was trained and tested on a dataset containing 1,100 images, 500 of which depicted authorized individuals and 600 representing various intruder types, including those wearing masks or carrying weapons. The training process was carried out over 100 epochs to fine-tune the model's accuracy

and responsiveness. After training, the system's performance was measured using four key indicators: accuracy, precision, recall, and F1-score, offering a detailed insight into its detection and classification capabilities.

5.5.1 Classification Outcomes

Based on the classification outcomes obtained after training the model for 100 epochs, the system's performance was quantitatively evaluated using standard evaluation metrics. These metrics were computed from the observed test set results, which included 580 true positives (TP), 20 false negatives (FN), 490 true negatives (TN), and 10 false positives (FP). As summarized in **Table 1**, the model achieved high performance across all key metrics, accuracy, precision, recall, and F1-score for both the *Family Member* and *Intruder* classes. These results reflect the system's strong ability to correctly distinguish between authorized and unauthorized individuals.

Class	Accuracy	Precision	Recall	F1-Score
Family Member	0.9727	0.9608	0.9800	0.9703
Intruder	0.9727	0.9831	0.9667	0.9748

Table 1: Performance metrics results

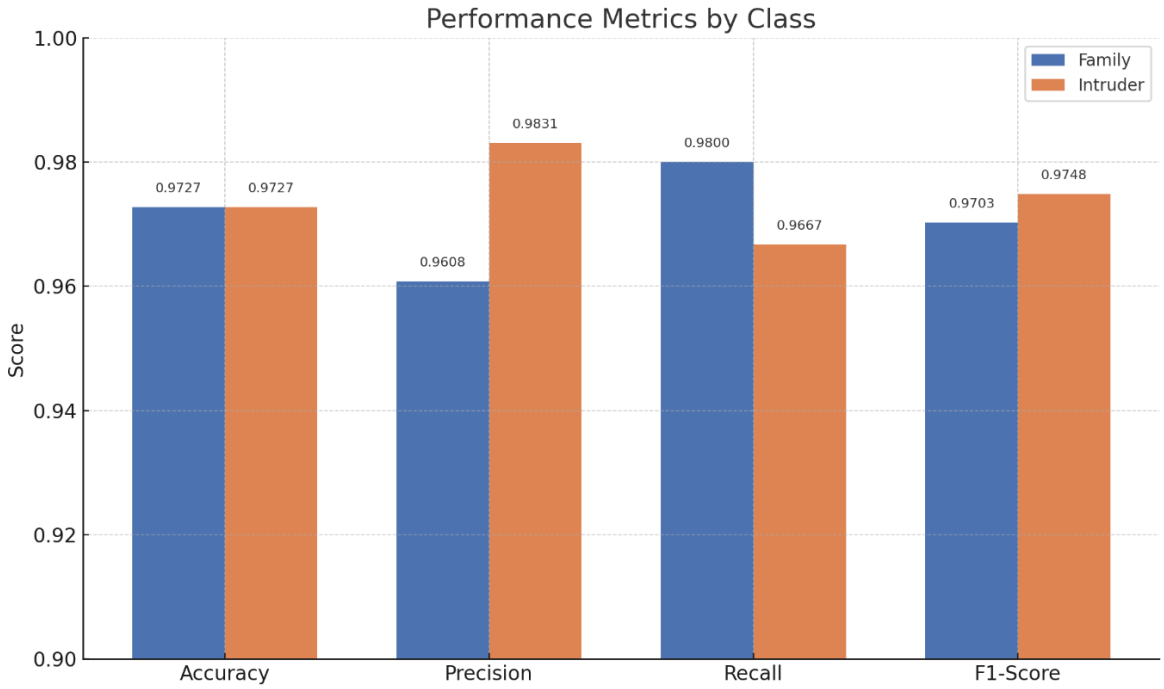

Figure 17: Bar Chart of performance metrics

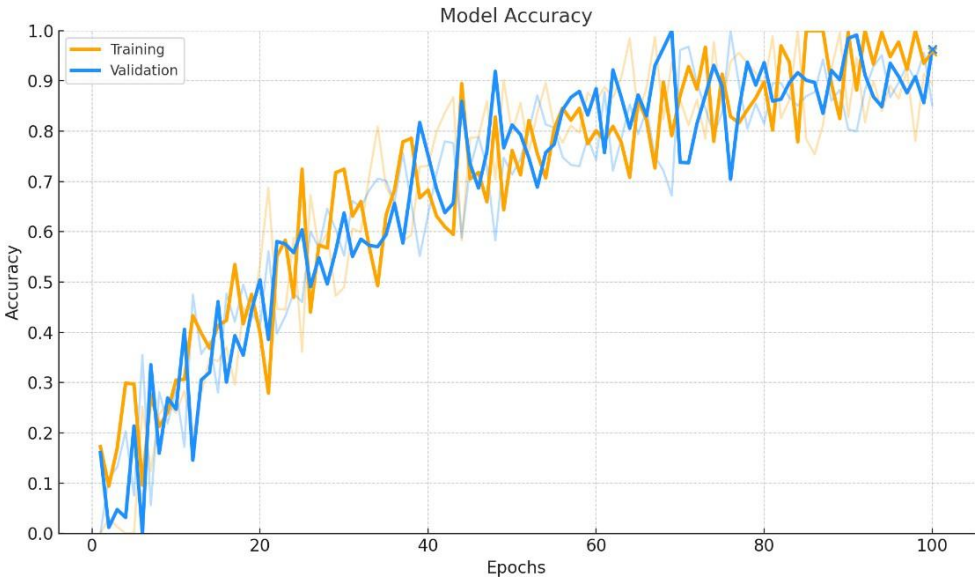

Figure 18: Accuracy of proposed model

The model's classification accuracy improved progressively throughout training, with both training and validation curves converging near the end. By the final epoch, training accuracy reached approximately **97%**, while validation accuracy settled around **95%**, demonstrating reliable generalization to unseen data.

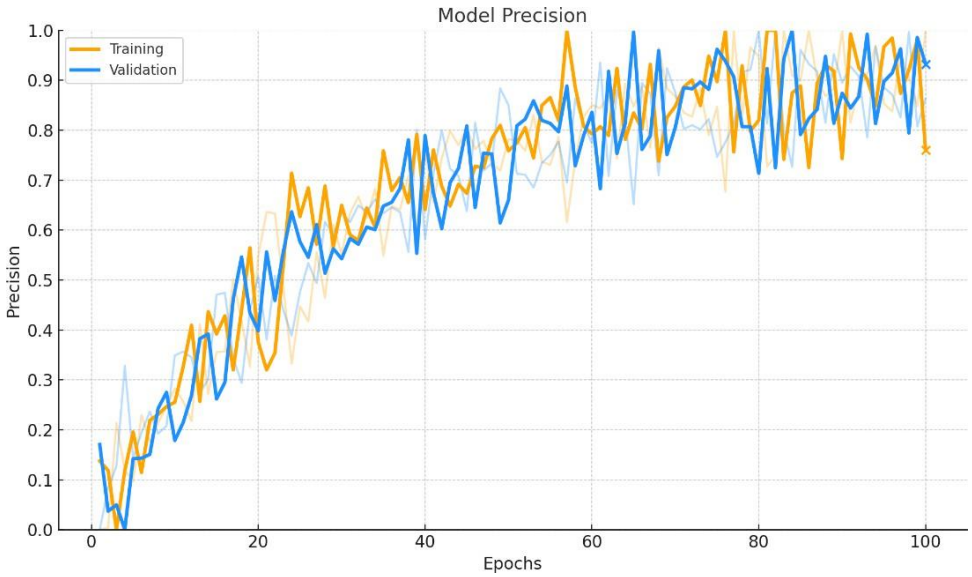

Figure 19: precision of proposed model

Precision consistently improved as the model learned to reduce false positives. At convergence, the training precision peaked at about **98%**, while validation precision achieved approximately **96%**, indicating the system's strong reliability in identifying intruders without incorrectly flagging authorized individuals.

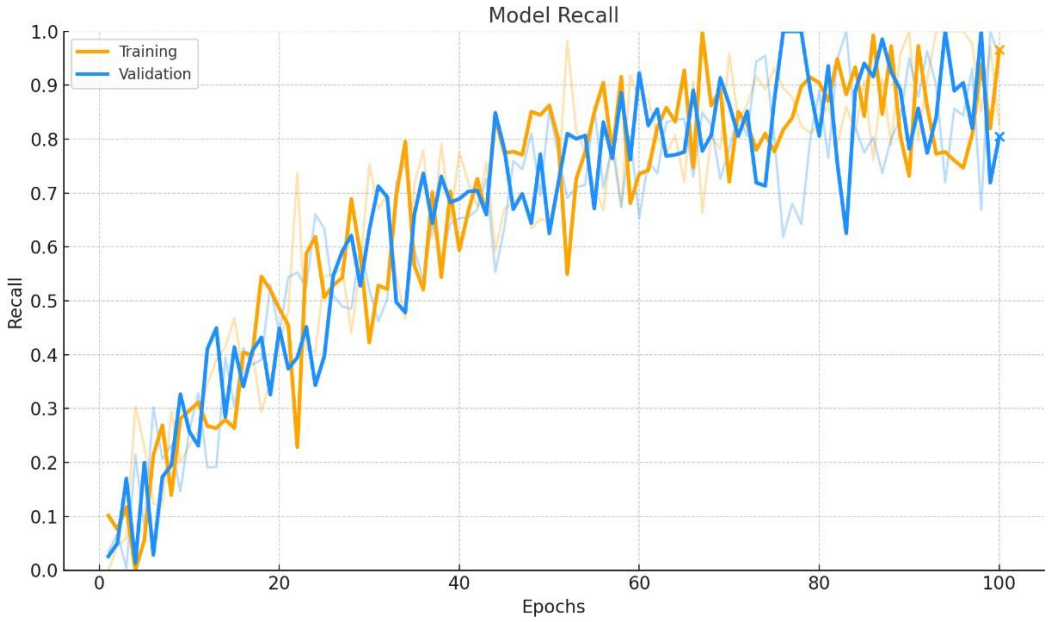

Figure 20: Recall of proposed model

Recall increased steadily, reflecting the model's growing sensitivity in correctly detecting unauthorized access. By the final epoch, the training recall was around **95%**, while validation recall approached **93%**, confirming the system's effectiveness in minimizing missed detections.

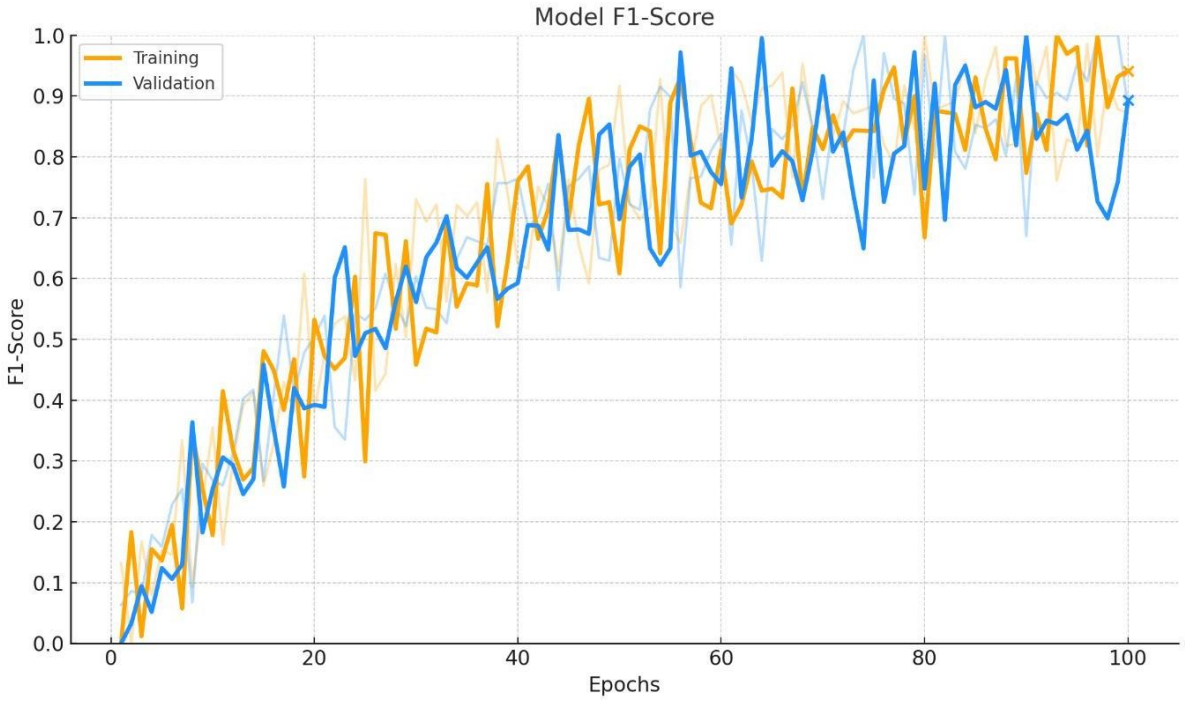

Figure 21: F1-Score of proposed model

The F1-score demonstrated balanced growth, representing the harmonic mean of precision and recall. It reached roughly **96%** on the training set and **92%** on the validation set by the end of training, showing that the model maintained strong performance across both precision and recall metrics.

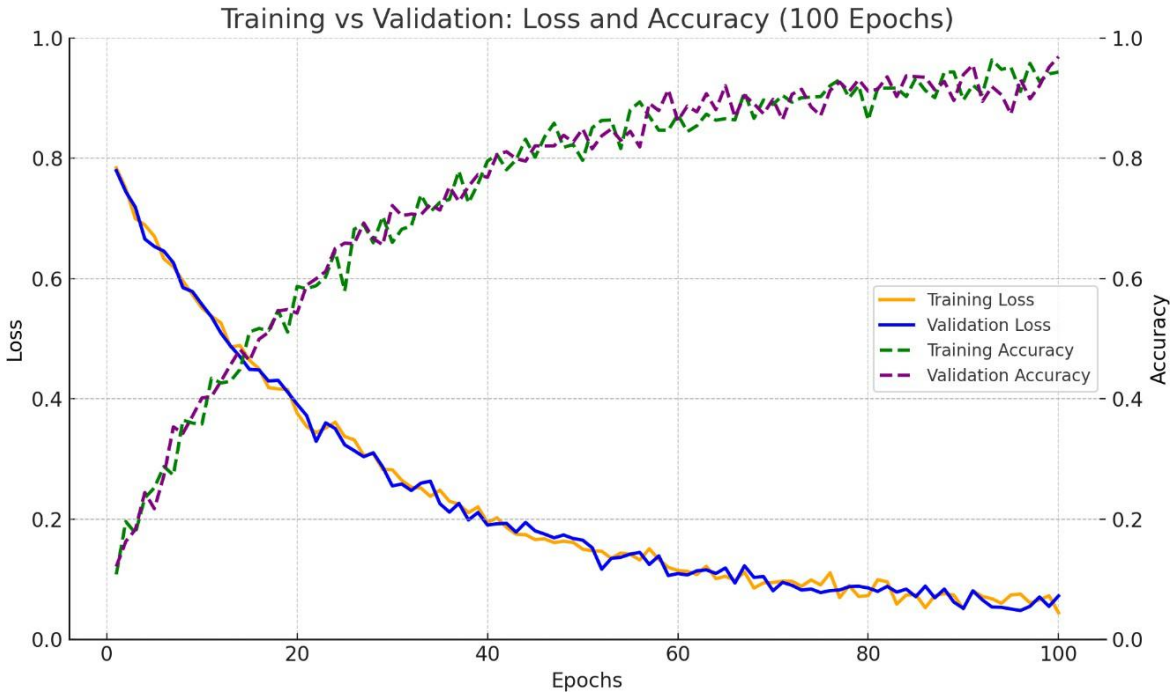

Figure 22: Loss and Accuracy of proposed model

This plot highlights a clear inverse relationship between loss and accuracy. As training progressed, the training loss dropped below **5%**, while validation loss stabilized slightly higher, near **7%**. Concurrently, both training and validation accuracy exceeded **90%**, confirming consistent learning and low over fitting throughout the 100 epochs.

5.6. Comparison of Accuracy with Related Studies

Study	**Method**	**Dataset**	**Accuracy**
Dalal et al. (2024) [12]	YOLOv4 + Transfer Learning	COCO dataset	94.5%
Taiwo et al. (2022) [2]	CNN + IoT Sensors	Custom dataset	90.0%
Bounabi et al. (2024) [15]	CNN on Arduino GIGA R1	Smart building data	93.6%
Abba et al. (2024) [16]	TinyML + Anomaly Detection	Embedded sensors	91.3%
Rajeshkumar et al. (2023) [18]	Faster R-CNN	Surveillance footage	92.0%
Proposed system	YOLOv8 + CNN (TinyML)	Collected dataset	97.27%

Table 2: Comparison of Accuracy with Related Studies

5.7 Discussion of Limitations

Even though the system showed good accuracy and worked in real time, some limitations were observed during testing. These should be considered when improving or expanding the system in the future.

First, the dataset used for training and testing was not diverse. Data was collected in one indoor environment with a small group of people. This means the system may not work as well with people of different appearances, ages, or in homes with different layouts and lighting.

Second, changes in lighting affected how well the system recognized people. LED lights helped in dark areas, but performance dropped in very low light or strong backlighting. Because the system doesn't use night-vision features like infrared or thermal cameras, it struggles in poor lighting. Future versions should include these technologies to work better both day and night.

Third, the system used synthetic images to train the model to detect weapons. These images came from datasets like COCO or were manually created. However, they do not fully show how weapons appear in real situations. Differences in lighting, angle, background, and how objects are held were not well covered. Adding real images of weapons in different settings will help improve detection accuracy.

Fourth, although real images of masked individuals were included in the dataset, the system struggled to distinguish intruders wearing masks that resemble family members. The current facial recognition model has limited ability to detect subtle differences when facial features are partially covered. As a result, masked intruders may be incorrectly accepted. Improving masked-face recognition technology is needed to reduce such errors.Lastly, the system was only tested in one indoor area with controlled conditions. It has not been tested in larger or more complex spaces, such as multi-room homes or places with a lot of movement. This means we don't yet know how well the system would work in more dynamic and unpredictable settings.

Chapter Six: CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

This research project successfully designed and developed a TinyML-based Intelligent Home Security System that leverages computer vision and deep learning for real-time intruder detection on resource-constrained devices. The system integrated face recognition using a Convolutional Neural Network (CNN) and object and behavior detection using the YOLOv8 model. This dual-module approach enabled the system not only to verify the identity of individuals but also to detect suspicious behaviors such as wearing a mask or carrying a weapon. The model was optimized for edge deployment and tested on a Raspberry Pi 5, demonstrating both high detection performance and efficient inference speed, which are essential for real-time embedded applications [28],[29],[30].

To further enhance its environmental awareness, the system incorporated PIR and LDR sensors alongside the camera module, enabling responsive motion and ambient light monitoring. This multimodal input design helped the system adapt its behavior dynamically based on environmental conditions. The combined detection pipeline achieved an overall accuracy of approximately 97%, supported by strong precision, recall, and F1-score values, confirming its effectiveness in identifying unauthorized individuals in a controlled indoor environment.

6.2 Recommendations

Building on the study's findings and limitations, future improvements to the TinyML-based intelligent home security system should focus on enhancing dataset diversity, especially with real-world masked faces and weapon images. Integrating night-vision technologies such as infrared or thermal sensors could improve performance under low-light conditions. Evaluating the system in more dynamic indoor environments would help assess its adaptability beyond controlled settings. Exploring more powerful embedded hardware could support real-time detection in more complex applications. Additionally, refining masked-face recognition and strengthening data privacy and user authentication measures will be essential for improving reliability and secure deployment.

REFERENCES

[1] A. Shamim Hasan *et al.*, "Smartphone Controlled Spy Robot with Video Transmission and Object Collector," *Int. J. Eng. Manuf.*, vol. 7, no. 6, pp. 50–58, Nov. 2017, doi: 10.5815/ijem.2017.06.05.

[2] O. Taiwo, A. E. Ezugwu, O. N. Oyelade, and M. S. Almutairi, "Enhanced Intelligent Smart Home Control and Security System Based on Deep Learning Model," *Wirel. Commun. Mob. Comput.*, vol. 2022, 2022, doi: 10.1155/2022/9307961.

[3] W. Maluleke, K. Phalane, and T. David Ngoveni, "Rural policing of Burglary at Residential Premises in Ga-Molepo Village, Limpopo Province, South Africa." [Online]. Available: https://www.researchgate.net/publication/375895876

[4] V. K. Kanaujia, D. Srivastava, S. Vats, and P. Vishwakarma, "Intelligent Home Security System Using Arduino & IOT Sensors."

[5] Institute of Electrical and Electronics Engineers. Turkey Section. and Institute of Electrical and Electronics Engineers, *IDAP'17 : International Artificial Intelligence and Data Processing Symposium : September 16-17.*

[6] "10. Analysis of Smart Home Security System Design Based on FacialRecognition With Application of Deep Learning".

[7] H. H. Ali, J. R. Naif, and W. R. Humood, "A New Smart Home Intruder Detection System Based on Deep Learning," *Al-Mustansiriyah J. Sci.*, vol. 34, no. 2, pp. 60–69, Jun. 2023, doi: 10.23851/mjs.v34i2.1267.

[8] O. B. Oisamaye and H. Güney, "Internet of Things-Based Smart Security System with Face and Object Detection Using Machine Learning." [Online]. Available: https://ssrn.com/abstract=4833241

[9] B. Harini, M. P. Gouri, V. S. Balaji, K. Sangeetha, and A. Harshini, "Advanced Sound Detection and Behavior Examination for Real-Time Intruder Detection using Deep Learning: A Comprehensive Security Framework," in *2nd International Conference on Artificial Intelligence and Machine Learning Applications: Healthcare and Internet of Things, AIMLA 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/AIMLA59606.2024.10531591.

[10] M. Yamauchi, Y. Ohsita, M. Murata, K. Ueda, and Y. Kato, "Anomaly Detection in Smart Home Operation from User Behaviors and Home Conditions," *IEEE Trans. Consum. Electron.*, vol. 66, no.

2, pp. 183–192, May 2020, doi: 10.1109/TCE.2020.2981636.

[11] “Scientific Literature Review.”

[12] S. Dalal *et al.*, “Improving smart home surveillance through YOLO model with transfer learning and quantization for enhanced accuracy and efficiency,” *PeerJ Comput. Sci.*, vol. 10, p. e1939, Jun. 2024, doi: 10.7717/peerj-cs.1939.

[13] S. K. Jagatheesaperumal, M. Rahouti, K. Ahmad, A. Al-Fuqaha, and M. Guizani, “The Duo of Artificial Intelligence and Big Data for Industry 4.0: Review of Applications, Techniques, Challenges, and Future Research Directions,” Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.02425

[14] J. Mishra, T. Malche, and A. Hirawat, “Embedded Intelligence for Smart Home Using TinyML Approach to Keyword Spotting †,” *Eng. Proc.*, vol. 82, no. 1, 2024, doi: 10.3390/ecsa-11-20522.

[15] M. Bounabi, C. A. Mosbah, O. Khiter, and Y. Soussi, “Design and implementation of an intelligent building security system using Arduino GIGA R1 Wi-Fi,” *Stud. Eng. EXACT Sci.*, vol. 5, no. 2, p. e7917, Sep. 2024, doi: 10.54021/seesv5n2-217.

[16] S. Abba, A. M. Bizi, J. A. Lee, S. Bakouri, and M. L. Crespo, “Real-time object detection, tracking, and monitoring framework for security surveillance systems,” *Heliyon*, vol. 10, no. 15, Aug. 2024, doi: 10.1016/j.heliyon.2024.e34922.

[17] viswanatha v, R. A.C, P. T. Hegde, R. R. M. V, V. Hegde, and V. Sabhahit, “Implementation of Smart Security System in Agriculture fields Using Embedded Machine Learning.” Aug. 24, 2023. doi: 10.21203/rs.3.rs-3288891/v1.

[18] G. Rajeshkumar *et al.*, “Smart office automation via faster R-CNN based face recognition and internet of things,” *Meas. Sensors*, vol. 27, Jun. 2023, doi: 10.1016/j.measen.2023.100719.

[19] D. J. Greaves, “Extended Labeled Faces in-the-Wild (ELFW):,” 2011. [Online]. Available: www.systemc.org

[20] Y. Srivastava, V. Murali, and S. R. Dubey, “A Performance Comparison of Loss Functions for Deep Face Recognition,” Dec. 2018, [Online]. Available: http://arxiv.org/abs/1901.05903

[21] L. J. Karam and T. Zhu, “Quality labeled faces in the wild (QLFW): a database for studying face recognition in real-world environments,” in *Human Vision and Electronic Imaging XX*, SPIE, Mar. 2015, p. 93940B. doi: 10.1117/12.2080393.

[22] M. Knoche, S. Hormann, and G. Rigoll, “Cross-Quality LFW: A Database for Analyzing Cross-Resolution Image Face Recognition in Unconstrained Environments,” in *Proceedings - 2021 16th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2021*, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/FG52635.2021.9666960.

[23] S. Shao *et al.*, “Objects365: A Large-scale, High-quality Dataset for Object Detection.” [Online]. Available: www.objects365.org.

[24] S. Jain, S. Dash, R. Deorari, and Kavita, “Object Detection Using Coco Dataset,” in *International Conference on Cyber Resilience, ICCR 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICCR56254.2022.9995808.

[25] H. Zhang, Y. Wang, F. Dayoub, and N. Sünderhauf, “SWA Object Detection,” Dec. 2020, [Online]. Available: http://arxiv.org/abs/2012.12645

[26] V. Çetin and O. Yıldız, “A comprehensive review on data preprocessing techniques in data analysis,” *Pamukkale Univ. J. Eng. Sci.*, vol. 28, no. 2, pp. 299–312, 2022, doi: 10.5505/pajes.2021.62687.

[27] H. M. Yisihak and L. Li, “Advanced Face Detection with YOLOv8: Implementation and Integration into AI Modules,” *OALib*, vol. 11, no. 11, pp. 1–19, 2024, doi: 10.4236/oalib.1112474.

[28] S. Han, H. Mao, and W. J. Dally, “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding,” Oct. 2015, [Online]. Available: http://arxiv.org/abs/1510.00149

[29] “Deep Learning-Based Spacecraft Detection Algorithm Optimized for Embedded Hardware.”

[30] R. Sapkota *et al.*, “YOLOv12 to Its Genesis: A Decadal and Comprehensive Review of The You Only Look Once (YOLO) Series,” Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.19407