BATCH NO:MAI076

INTELLIGENCE SAFETY DETECTION AND HAZARD MONITORING USING YOLO ALGORITHM

Major project report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

Y S D S Krishna Tarun	21UECS0697	**VTU19135**
Relangi Mahesh Murari	21UECS0528	**VTU19431**
V.Karthikeya Santosh Reddy	21UECS0664	**VTU19447**

Under the guidance of
Mr.V.Ashok Kumar,B.E,M.Tech.,
ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May 2025

BATCH NO:MAI076

INTELLIGENCE SAFETY DETECTION AND HAZARD MONITORING USING YOLO ALGORITHM

Major project report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

Y S D S Krishna Tarun	21UECS0697	**VTU19135**
Relangi Mahesh Murari	21UECS0528	**VTU19431**
V.Karthikeya Santosh Reddy	21UECS0664	**VTU19447**

Under the guidance of
Mr.V.Ashok Kumar,B.E,M.Tech.,
ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May 2025

CERTIFICATE

It is certified that the work contained in the project report titled ”INTELLIGENCE SAFETY DETECTION AND HAZARD MONITORING USING YOLO ALGORITHM ” by ”Y S D S KRISHNA TARUN 21UECS0697, RELANGI MAHESH MURARI 21UECS0528, V. KARTHIKEYA SANTOSH REDDY 21UECS0664” has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor
Mr. V. Ashok Kumar
Assistant Professor
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science and Technology

Signature of Head/Assistant Head of the Department
Dr. N. Vijayaraj/Dr. M. S. Murali dhar
Professor & Head/ Professor &Assistant Head
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science and Technology

Signature of the Dean
Dr. S P. Chokkalingam
Professor & Dean
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science and Technology

DECLARATION

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

Y S D S KRISHNA TARUN

Date: / /

(Signature)

RELANGI MAHESH MURARI

Date: / /

(Signature)

V.KARTHIKEYA SANTOSH REDDY

Date: / /

APPROVAL SHEET

This project report entitled INTELLIGENCE SAFETY DETECTION AND HAZARD MONITORING USING YOLO ALGORITHM by Y S D S KRISHNA TARUN 21UECS0697, RELANGI MAHESH MURARI 21UECS0528, V. KARTHIKEYA SANTOSH REDDY 21UECS0664 is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners **Supervisor**

Mr.V.Ashok Kumar,B.E,M.Tech.,
Assistant Professor,.

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our **Honorable Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (Electrical), B.E. (Mechanical), M.S (Automobile), D.Sc., and Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.,Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, for her blessings.**

We express our sincere thanks to our respected Chairperson and Managing Trustee **Dr. (Mrs.) RANGARAJAN MAHALAKSHMI KISHORE, B.E., Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology,** for her blessings.

We are very much grateful to our beloved **Vice Chancellor Prof. Dr.RAJAT GUPTA,** for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean , School of Computing, Dr. S P. CHOKKALINGAM, M.Tech., Ph.D., & Professor & Associate Dean , School of Computing, Dr. V. DHILIP KUMAR,M.E.,Ph.D.,** for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Professor & Head, Department of Computer Science & Engineering, Dr. N. VIJAYARAJ, M.E., Ph.D.,** and **Professor & Assistant Head, Department of Computer Science & Engineering, Dr. M. S. MURALI DHAR, M.E., Ph.D.,**for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our **Internal Supervisor Mr. V. ASHOK KUMAR, B.E,M.Tech.,** for his cordial support, valuable information and guidance, he helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Dr. SADISH SENDIL MURUGARAJ,B.E., M.E., Ph.D., Professor, Dr.S.RAJAPRAKASH, M.E,Ph.D., Mr. V. ASHOK KUMAR, B.E,M.Tech.,** for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

Y S D S KRISHNA TARUN	**21UECS0697**
RELANGI MAHESH MURARI	**21UECS0528**
V. KARTHIKEYA SANTOSH REDDY	**21UECS0664**

ABSTRACT

This project presents the Intelligent Safety Monitoring System aimed at promoting workplace safety through the automated detection of safety violations and hazardous conditions. Leveraging advanced computer vision algorithms like YOLOv8 for real-time object detection, and facial recognition-based secure user authentication, the system performs continuous analysis of live video feeds, images, and recorded footage to monitor safety gear compliance, such as the usage of hard hats, masks, and vests. It also detects the presence of heavy machinery, triggering instant alerts to prevent potential hazards. The cloud-based architecture integrates Firebase for user control and real-time database management, while Google Drive handles scalable media storage. A responsive web interface is implemented using Flask, featuring an admin dashboard for real-time monitoring, user activity tracking, and detailed reporting. Experimental results demonstrate that the system enhances workplace safety by ensuring better compliance, issuing real-time alerts, and providing data-driven insights. The solution is scalable and reduces dependency on manual inspection, thus significantly improving situational awareness in high-risk environments.

Keywords: Intelligent Safety Monitoring System, Workplace Safety, Computer Vision, YOLOv8(You Only Look Once Version 8), Real-time Object Detection, Facial Recognition, Safety Gear Compliance, Cloud Computing, Firebase, Web Interface, Real-time Alerts, Automated Surveillance, Industrial Safety.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma-Separated Values
CV	Computer Vision
FPS	Frames Per Second
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
HTML	HyperText Markup Language
IoT	Internet of Things
ML	Machine Learning
mAP	mean Average Precision
PPE	Personal Protective Equipment
ROI	Region of Interest
SDG	Sustainable Development Goals
SSD	Single Shot Detector
UI	User Interface
YOLO	You Only Look Once

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

In today's rapidly evolving industrial landscape, ensuring worker safety has become a top priority, getting worker safety is the most important thing, especially with the rapid pace of the growing industrial automation. All the construction activities, manufacturing, and logistics industries hired employees exposed to hazards during work, which stresses the importance of monitoring and ensuring compliance with safety measures. Previously, such monitoring relied on manual inspections that are usually time-consuming, inefficient, and prone to errors. With advancements in deep learning and computer vision, automated safety monitoring systems have emerged as powerful tools for real-time analysis and instantaneous decision-making.

Particularly, object detection acts as a major contributor towards compliance with safety criteria because it provides close monitoring of the presence or absence of personal protective equipment (PPE), such as hardhats, safety vests, and face masks. Hence, it becomes very effective in enforcing safety regulations. Among the different object detection algorithms, YOLO stands as a unique and fastly accurate in detail. This makes YOLO qualify for real-time applications in today's dynamic industrial settings.

The proposed project embodies an Intelligent Safety Monitoring System using the latest YOLOv8 architecture in automating PPE detection and the situational awareness capability in high-risk environments. Added in this project is a facial-recognition-based safe user authentication and hazard detection, heavy machinery while in operation, for more advanced safety provision. With a cloud-backed infrastructure using Firebase for user control and Google Drive for secure media storage, the system promises scalability and reliability. The project boasts of a responsive web interface featuring an admin dashboard for real-time monitoring, tracking user activity, and producing detailed reports, ultimately decreasing dependence on manual inspections and creating a safer, smarter workplace.

1.2 Background

With the rapid growth of technology in the 21st century, object detection has emerged as a key component of modern CV applications. In industrial settings, accurate detection and localization of objects is essential for maintaining safety standards. Traditional manual inspection of PPE has proven to be inefficient and prone to human error. The adoption of DL and CV techniques has led to the development of automated safety monitoring systems capable of real-time performance. Among the object detection models, YOLO has gained widespread attention for its ability to detect multiple objects at high speed and accuracy, operating at up to 45 fps. YOLO processes an input image by dividing it into grids and simultaneously performing classification and localization tasks. In this project, a YOLO-based detection system is integrated with a Flask web application to identify PPE such as helmets, vests, gloves, goggles, and shoes. The system enables real-time alerts for violations, significantly enhancing industrial safety compliance.

1.3 Objective

The primary objective of this project is to develop an automated PPE detection system using the YOLO object detection algorithm, aimed at improving safety compliance in industrial environments. By leveraging DL and CV technologies, the system is designed to identify whether workers are wearing the necessary safety gear, such as helmets, gloves, vests, goggles, and shoes. The proposed system processes both static (images and videos) and dynamic (live camera feed) visual inputs to detect the presence or absence of PPE. Through this, the project seeks to reduce dependency on manual supervision, eliminate human errors, and provide timely alerts in case of non-compliance. The integration of YOLO ensures fast and accurate detection, making the system suitable for real-time industrial applications.

Another key objective is to implement this detection system through a user-friendly Flask web application that allows industries to upload footage or monitor live feeds. The system is programmed to generate instant alerts whenever a violation is detected, allowing for quick corrective actions. This helps not only in accident prevention but also in enforcing safety regulations efficiently. The project aims to demonstrate how real-time object detection can be practically deployed to enhance workplace safety, minimize risks, and align with modern industry standards. Ulti-

mately, the goal is to create a robust, scalable, and accessible solution that promotes a culture of safety through intelligent automation.

1.4 Problem Statement

In industrial sectors such as construction, manufacturing, and logistics, ensuring the consistent use of PPE is a critical component of workplace safety. Despite strict safety regulations and policies, many accidents occur due to the negligence or improper usage of PPE by workers. Traditionally, PPE compliance has been monitored through manual inspections, which are labor-intensive, time-consuming, and highly susceptible to human error. Supervisors may overlook violations during busy operations, or fail to detect non-compliance in real-time, leading to serious injuries or even fatalities. Furthermore, in large-scale industrial environments, it becomes impractical to continuously monitor every worker manually, especially during night shifts or in areas with restricted access.

To address these challenges, there is a growing need for an intelligent, automated system that can detect PPE usage efficiently and reliably. Existing systems may lack the speed and accuracy required for real-time monitoring or may not support dynamic input sources like live camera feeds. This project aims to solve these problems by implementing a YOLO-based object detection model capable of identifying various safety gear with high precision. By integrating this detection mechanism into a Flask-based web application, the system provides instant feedback and alert notifications for any violations. This solution not only ensures consistent safety compliance but also reduces the burden on manual supervision, ultimately contributing to safer industrial workplaces.

Chapter 2

LITERATURE REVIEW

[1] Brownlee, explained the fundamentals of object recognition using deep learning, focusing on how machines detect objects in images via neural networks. The article emphasizes the role of convolutional neural networks (CNNs) in image classification and provides a beginner-friendly overview covering training data, model architecture, and evaluation, with applications in security, industry, and automation.

[2] J. Redmon et al., proposed YOLO (You Only Look Once), a real-time object detection system. Unlike traditional methods, YOLO processes the entire image with a single neural network. It achieves remarkable speed and accuracy, making it ideal for real-world applications. The authors compare YOLO with existing methods, showing its superiority. YOLO predicts bounding boxes and class probabilities simultaneously. It is widely used in surveillance and autonomous driving. This paper laid the groundwork for fast, accurate object detectors.

[3] Y. Zhang et al., introduced a modified YOLO algorithm for safety helmet detection. The system is designed for construction and industrial environments. Enhancements improve detection accuracy and reduce false positives. The model works in real-time with video surveillance systems. It ensures compliance with safety gear regulations. The study contributes to workplace safety monitoring solutions. Results show superior performance compared to baseline models.

[4] H. Li et al., proposed a deep learning-based PPE detection framework is introduced in this paper. The system uses CNNs to identify safety equipment in industrial scenes. It enables automated inspection of worker compliance in real time. The authors use a dataset of PPE images for training and validation. Accuracy and robustness of the model are demonstrated through experiments. It reduces the need for manual monitoring and enhances safety. The approach is scalable for smart factory environments.

[5] R. Girshick et al.,introduced R-CNN, a deep learning framework for object detection. R-CNN combines region proposals with CNNs for accurate localization. It extracts high-quality features for each proposed object region. The model achieves significant accuracy improvements over prior methods. R-CNN has inspired many subsequent detection frameworks. It separates feature extraction, classification, and bounding box regression. The paper is foundational in deep learning-based object detection.

[6] Z. Cao et al., developed OpenPose is presented for real-time multi-person pose estimation. The system detects human body keypoints from RGB images. It introduces part affinity fields for accurate joint detection. OpenPose works in crowded scenes and dynamic environments. It supports full-body pose estimation for multiple individuals. Applications include motion tracking, sports analytics, and AR. The paper is a milestone in human pose estimation research.

[7] Y. Li et al., presented a CNN-based safety helmet detection system. It addresses the problem of detecting helmets under varying conditions. The model is trained on images from engineering sites. The authors emphasize robustness under different lighting and angles. It enhances safety checks in civil engineering projects. Real-time performance makes it suitable for live monitoring systems. Results validate the model's high accuracy and reliability.

[8] C. Y. Wang et al., proposed YOLOv7 is introduced as a state-of-the-art real-time detector. It features architecture improvements and training optimizations. YOLOv7 achieves top accuracy while maintaining real-time performance. The system is tested on multiple object detection benchmarks. It incorporates new bag-of-freebies and trainable modules. Applications span surveillance, robotics, and autonomous systems. The model marks a major advancement in YOLO development.

[9] X. Shao et al., developed FMYOLOv7 for detecting helmets in mining environments. The system adapts YOLOv7 with feature fusion techniques. It targets small object detection in harsh conditions. Experimental results show increased precision and recall. The model is lightweight and suitable for edge deployment. It supports real-time helmet detection on-site. The paper contributes to smart mining

safety applications.

[10] A. Aboah et al., proposed a multi-class helmet violation detection system. It uses YOLOv8 combined with few-shot data sampling. The goal is to detect multiple PPE violations in real-time. The technique enables learning from limited labeled data. It's useful in environments where annotated data is scarce. Detection accuracy remains high across multiple PPE categories. The approach supports scalable safety compliance monitoring.

[11] W. Chen et al., introduced YOLO-Face,a real-time face detection tasks. It adapts YOLO for facial features and bounding box prediction. The model achieves high-speed detection with minimal resource usage. It works effectively in varied lighting and camera angles. Applications include access control, surveillance, and biometrics. The paper provides benchmarks against other face detectors. YOLO-Face excels in both accuracy and speed.

[12] N. Nath et al., explored deep learning for PPE detection in construction. CNNs are used to identify safety gear in site footage. Real-time detection enables quick alerts for non-compliance. The model reduces dependency on manual supervision. It is designed for integration with on-site surveillance systems. Results show strong accuracy and reliability in noisy scenes. The system enhances construction site safety practices.

[13] G. Gallo et al., proposed a smart system is proposed for PPE detection using deep learning. The system combines IoT sensors with CNN-based image analysis. It aims to automate safety checks in industrial zones. The approach is efficient in identifying non-compliant workers. Performance is validated in real-world industrial datasets. The paper promotes AI-based workplace monitoring. It aligns with Industry 4.0 and smart manufacturing goals.

[14] Q. Zhou et al., introduced AT-YOLO model for helmet detection. It modifies YOLO to handle safety gear in complex settings. The system is optimized for real-time performance. It maintains high accuracy in variable lighting conditions. Tests show low latency and strong generalization. Suitable for factories and outdoor environments. The research supports enhanced safety automation.

2.1 Existing System

The existing system for safety gear detection predominantly uses traditional machine learning techniques, particularly the Decision Tree algorithm, which relies on manually extracted image features to determine whether individuals are wearing protective equipment such as helmets or vests. While the Decision Tree is simple, interpretable, and effective for structured data, it falls short in handling complex visual scenes commonly found in industrial environments. Its performance is highly sensitive to variations in lighting, background, and object occlusion, leading to decreased accuracy in real-time applications. Additionally, the system lacks the ability to scale efficiently with larger datasets and does not adapt well to dynamic workplace conditions. These limitations hinder its reliability for continuous monitoring and pose safety risks in high-hazard zones. Consequently, the need for a more adaptive and accurate system has led to the development of a deep learning-based approach that addresses these drawbacks.

2.2 Related Work

Recent advancements in deep learning have significantly improved the accuracy and speed of object detection systems, particularly in safety-critical applications. Studies have demonstrated that models like YOLO and its variants (YOLOv5, YOLOv7, YOLOv8) outperform traditional machine learning approaches such as Decision Trees and SVMs in real-time detection. Works by Zhang et al. and Li et al. focus on helmet detection using improved YOLO algorithms, achieving high precision in construction sites and industrial environments. Redmon et al.'s pioneering work on YOLO revolutionized object detection by unifying it into a single neural network model. Shao et al. proposed FMYOLOv7 to enhance detection in mining operations, while Aboah et al. introduced a few-shot learning technique with YOLOv8 for improved helmet violation detection. These approaches emphasize the importance of high-speed, accurate, and scalable models in ensuring workplace safety.

2.3 Research Gap

While existing safety gear detection systems using YOLO and similar deep learning models have shown promising results in various industrial settings, most of these

implementations are domain-specific and lack adaptability to changing workplace environments. Many studies focus on helmet detection alone, overlooking other crucial safety gear such as vests and gloves. Additionally, existing models often require high computational resources, making them unsuitable for deployment on edge devices or in real-time scenarios with limited infrastructure. There is also limited integration of pose estimation (like OpenPose) with object detection models to understand worker behavior or compliance in a holistic manner. Furthermore, the datasets used in prior work are often limited in diversity, leading to performance drops in unseen conditions. This research aims to bridge these gaps by developing a lightweight, multi-gear detection system using improved YOLO models, tailored for real-time industrial monitoring with better generalization.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

In many industrial and construction sites today, safety compliance largely relies on manual monitoring and human supervision. Supervisors or safety officers are responsible for observing workers and ensuring that personal protective equipment , such as helmets and safety vests, are properly worn at all times. This traditional approach, although widely practiced, is highly dependent on continuous human attention and can easily fail due to oversight, fatigue, or negligence. Moreover, in large-scale operations where multiple workers are scattered across various zones, monitoring each individual becomes extremely difficult. Even with CCTV surveillance, a person must continuously analyze multiple video feeds, making the process inefficient and error-prone.

The major drawbacks of the existing system include its lack of real-time automated alerts and its inefficiency in detecting safety violations instantly. It is neither scalable nor robust enough to ensure complete compliance in high-risk environments. Human error is a significant concern, and there is no guarantee that every violation will be caught or acted upon promptly. Additionally, the system does not maintain automatic records or generate logs that can be used for audit trails or future safety training. In the event of an accident, valuable time may be lost trying to review hours of footage manually. These limitations make it necessary to develop an intelligent, real-time safety detection system that can identify PPE violations automatically and enhance workplace safety.

3.2 Proposed System

The proposed system leverages deep learning and computer vision to build an intelligent real-time monitoring solution for personal protective equipment (PPE) detection in industrial environments. Using the YOLO (You Only Look Once) object

detection algorithm, this system can identify whether workers are wearing essential safety gear such as helmets, vests, goggles, gloves, and shoes. By integrating the model into a Flask-based web application, the system allows users to upload images, videos, or even stream real-time footage from a live camera. The YOLO model processes these inputs, detects PPE compliance or violations, and triggers instant alerts when necessary. This not only improves the responsiveness of safety management but also eliminates the need for manual supervision.

One of the major advantages of this system is its ability to function in real time with high accuracy and speed. Unlike manual inspection, it is not prone to fatigue, oversight, or inconsistency. It also provides automated reports and logs, which can be used for audits, analytics, or training purposes. The system can be scaled easily across different locations and integrated with existing surveillance infrastructure, making it both cost-effective and efficient. The automated alert system ensures immediate action can be taken in case of non-compliance, thereby reducing the chances of accidents and promoting a culture of safety. This intelligent solution is a significant step toward enhancing workplace safety and adhering to industrial safety standards.

3.3 Feasibility Study

The feasibility study of the proposed YOLO-based PPE detection system indicates a high potential for successful implementation in real-world industrial environments. From a technical standpoint, the system is highly feasible due to the availability of pre-trained YOLO models that offer robust object detection capabilities with minimal computational requirements. With the use of a Flask-based web interface, integration into existing surveillance setups is seamless, allowing for easy deployment across various industrial zones. Economically, the system is cost-effective since it reduces the need for dedicated human supervision and minimizes accident-related losses through early detection of safety violations. Operationally, the system is user-friendly, offering real-time alerts and visual feedback that can be interpreted easily by safety officers. It also generates automated compliance records, aiding in audits and long-term safety planning. The system's scalability ensures it can be expanded to cover multiple sites or facilities without significant changes to the architecture. Thus, this project is highly feasible in terms of development, deployment, and long-term impact on worker safety.

3.3.1 Economic Feasibility

The economic feasibility of the proposed YOLO-based safety gear detection system is highly favorable when evaluated in the context of industrial safety monitoring. Traditional safety compliance methods require significant human resources for constant supervision, which not only increases labor costs but also suffers from inefficiencies due to human limitations. In contrast, the proposed system automates the entire monitoring process using object detection technology, thus substantially reducing the need for manual intervention. Since the system is built using open-source tools such as YOLO and Flask, there is no need for expensive proprietary software or hardware, making the initial development cost low. It can also be integrated with existing CCTV or surveillance systems, further minimizing setup expenses. Once deployed, the system requires minimal maintenance and can operate continuously without fatigue, resulting in long-term cost savings. Moreover, by reducing workplace accidents through early detection and real-time alerts, the system can prevent costly medical expenses, insurance claims, and production downtimes, ensuring a strong return on investment. Overall, the proposed solution is not only technically viable but also economically sustainable.

3.3.2 Technical Feasibility

The proposed safety gear detection system demonstrates strong technical feasibility, owing to its reliance on well-established and reliable technologies in the fields of deep learning and computer vision. At the core of the system is the YOLO (You Only Look Once) object detection algorithm, known for its real-time performance, accuracy, and efficiency. YOLO is capable of detecting multiple objects in a single frame with minimal computational overhead, making it highly suitable for live video streams or continuous image input. The implementation uses Python-based frameworks, and the integration is achieved through a Flask web application, allowing seamless user interaction and system accessibility through a browser. The model can be trained or fine-tuned using custom datasets, enabling it to detect specific PPE items like helmets, vests, gloves, and goggles accurately. Additionally, the system can be deployed on low-to-mid-end hardware setups or even integrated into cloud platforms for scalability. Given the widespread availability of libraries such as OpenCV, TensorFlow, and pre-trained YOLO models, the technical requirements for building, deploying, and maintaining the system are realistic and achievable, con-

firming the strong technical feasibility of the project.

3.3.3 Social Feasibility

The proposed safety gear detection system is highly socially feasible, especially in today's era where workplace safety is a critical concern in industrial environments. By leveraging modern technologies such as deep learning and real-time object detection, this system promotes a culture of safety and responsibility among workers and employers alike. The system continuously monitors for the proper use of personal protective equipment (PPE) such as helmets, gloves, goggles, and vests, reducing human dependency and the risk of oversight. This proactive approach not only ensures compliance with safety standards but also builds trust between workers and management, as it shows a commitment to their well-being. Social acceptance is further enhanced by the system's non-intrusive nature—there is no need for physical checks or manual reporting, which often causes discomfort or tension among employees. Instead, automated detection fosters a positive safety culture. Moreover, the real-time alert mechanism helps prevent accidents before they happen, potentially saving lives and preventing injuries, which contributes to a healthier, more productive workforce. Thus, the system is well-aligned with societal goals of ethical labor practices and workplace safety.

3.4 System Specification

3.4.1 Hardware Requirements

- Processor: Intel Core i5 (10th Gen or above) / AMD Ryzen 5 (3rd Gen or above)
- RAM: Minimum 8 GB
- Storage: 256 GB SSD minimum, 512 GB SSD recommended
- Graphics Card: NVIDIA GTX 1650 / RTX 2060 or higher
- Webcam: Integrated or external HD webcam

3.4.2 Software Requirements

- Programming Language: Python 3.8 or higher
- Web Framework: Flask

- Libraries: OpenCV, NumPy, Ultralytics YOLO, Pillow
- Testing Tools: Postman, Browser-based upload testing
- IDE: Visual Studio Code / PyCharm
- Operating System: Windows 10/11
- Package Manager: pip
- Web Browser: Chrome / Firefox

3.4.3 Tools and Technologies Used

The following tools and technologies were utilized in the development of the Intelligent Safety Detection system:

- **YOLOv8**: Used as the core object detection algorithm for detecting safety gear violations in real-time.
- **HOG**: Used as the core facial recognition algorithm for authenticating users by analyzing the structure and orientation of facial features, enabling secure and efficient login.
- **Python**: The primary programming language used for model implementation and backend logic.
- **OpenCV**: Utilized for image processing and video stream handling.
- **Flask**: Lightweight web framework used for developing the project's web-based user interface.
- **Google Colab**: Employed for data analysis, model training, and experimentation.
- **Google Colab**: Used for model training and testing with GPU support.
- **VS Code**: Integrated development environment used for coding and debugging.

3.4.4 Standards and Policies

The development and deployment of the Intelligent Safety Detection system adhered to the following standards and policies:

- **Data Privacy and Security**: All datasets used for training and evaluation are publicly available or synthetically generated, ensuring no violation of privacy laws.
- **Ethical AI Usage**: The system is intended for enhancing workplace safety and does not perform facial recognition or collect personally identifiable information.
- **Code Documentation and Version Control**: Best practices in code commenting and Git version control were followed to ensure maintainability and collaborative development.
- **Performance Metrics**: Model performance is measured using standard object detection metrics such as precision, recall, F1-score, and mAP (mean Average Precision).
- **Modular and Scalable Design**: The system is designed using modular coding practices to allow future enhancements like integration with CCTV systems or alert mechanisms.
- **Compliance with Open-Source Licenses**: All external libraries and datasets are used in accordance with their respective licenses (MIT, GPL, etc.).
- **Cross-Browser and Device Compatibility**: The web interface was tested on multiple browsers to ensure consistent performance and responsiveness.

Chapter 4

SYSTEM DESIGN AND METHODOLOGY

4.1 System Architecture

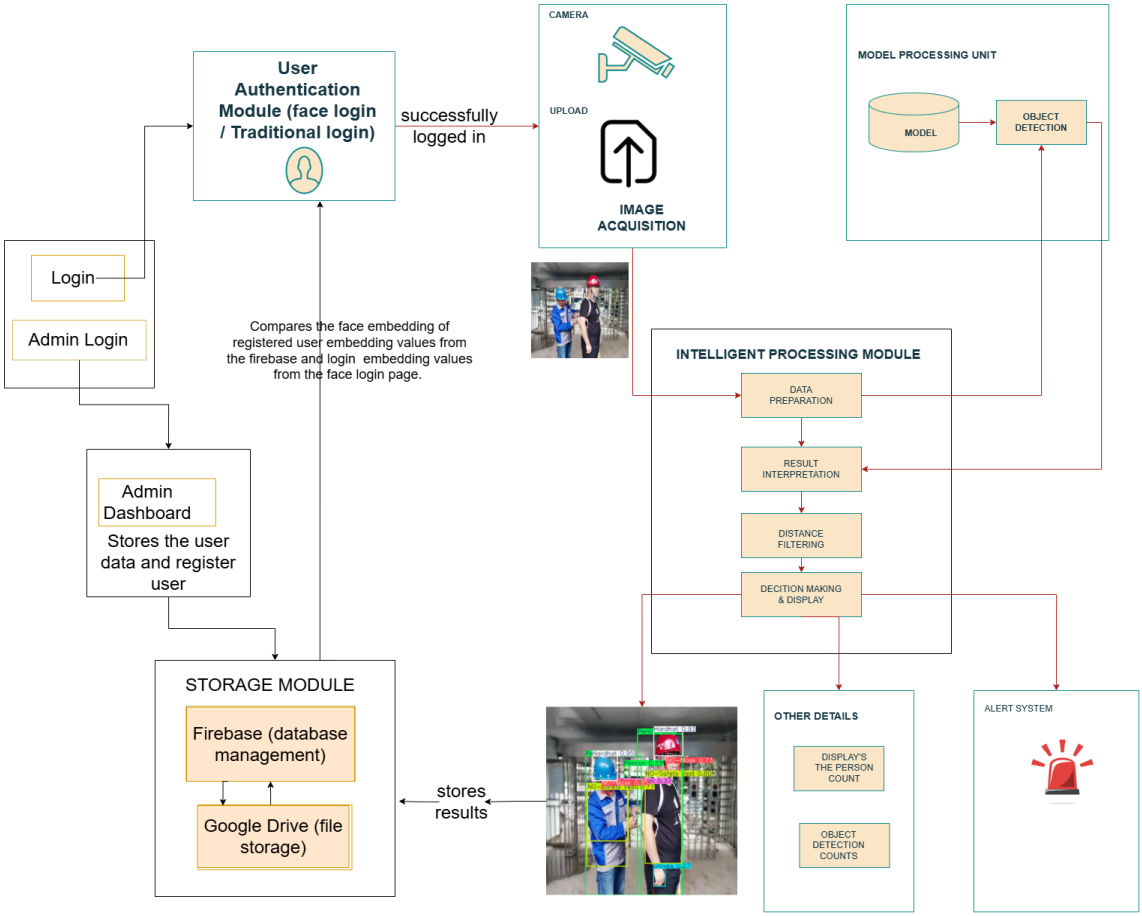


Figure 4.1: **System Architecture**

Figure 4.1 shows the system architecture of the YOLOv8-based Safety Gear Detection System. Users authenticate via face recognition or login, while admins manage data through a dashboard. Images from uploads or camera feeds are processed by the YOLOv8 model to detect PPE. The Intelligent Processing Module filters results and checks safety compliance. If violations occur, the Alert System notifies users. Detection results and user data are stored in Firebase and Google Drive, with person and object counts tracked for monitoring.

4.2 Design Phase

4.2.1 Data Flow Diagram

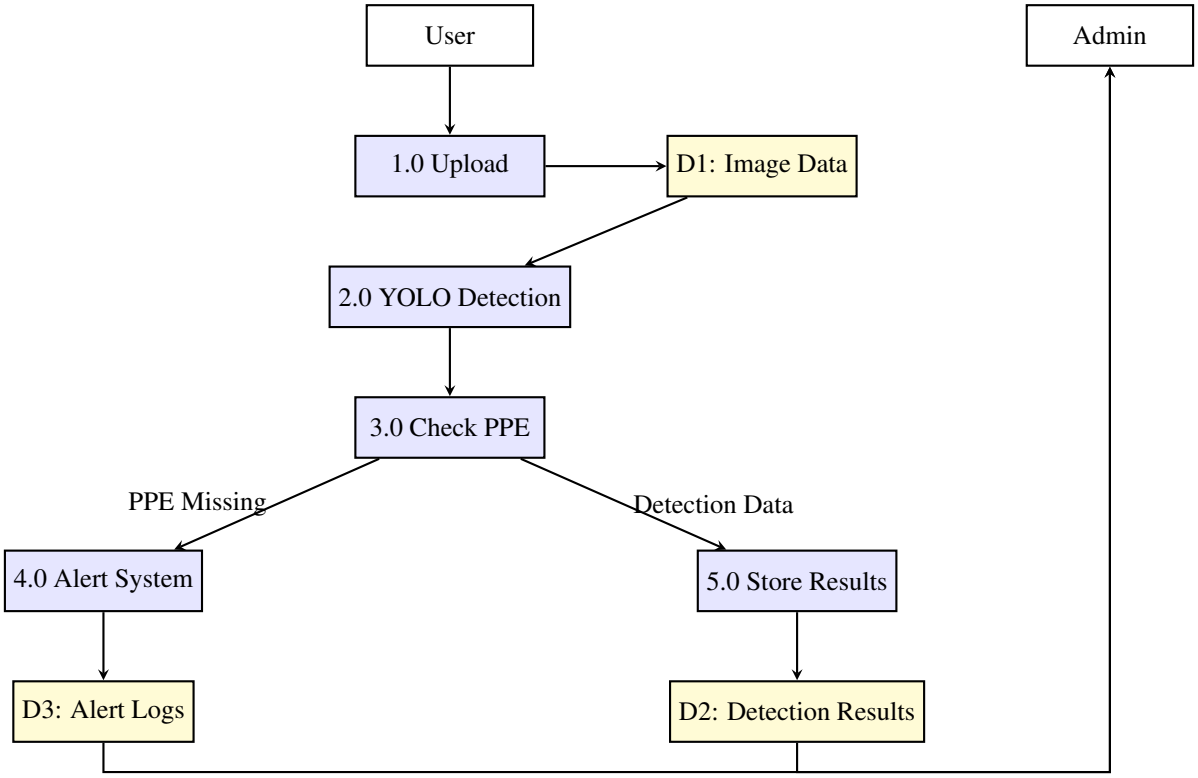


Figure 4.2: **Data Flow Diagram**

From the figure 4.2 illustrates the Data Flow Diagram of the PPE detection system, outlining how data moves through various components. The process starts when the user uploads an image or video via the web interface. This input is stored in the Image Data repository and then sent to the YOLO model for object detection. The model identifies the presence or absence of safety gear and analyzes the results to check for any PPE violations.If a violation is detected, the Alert System is triggered, and corresponding alerts are logged in the system. Simultaneously, detection outcomes are stored in a structured format for later access. Admins can monitor these alerts and access reports for review, while users are able to view their own detection results. This data flow structure ensures real-time safety monitoring, efficient alert generation, and secure data handling for both users and administrators.

4.2.2 Use Case Diagram

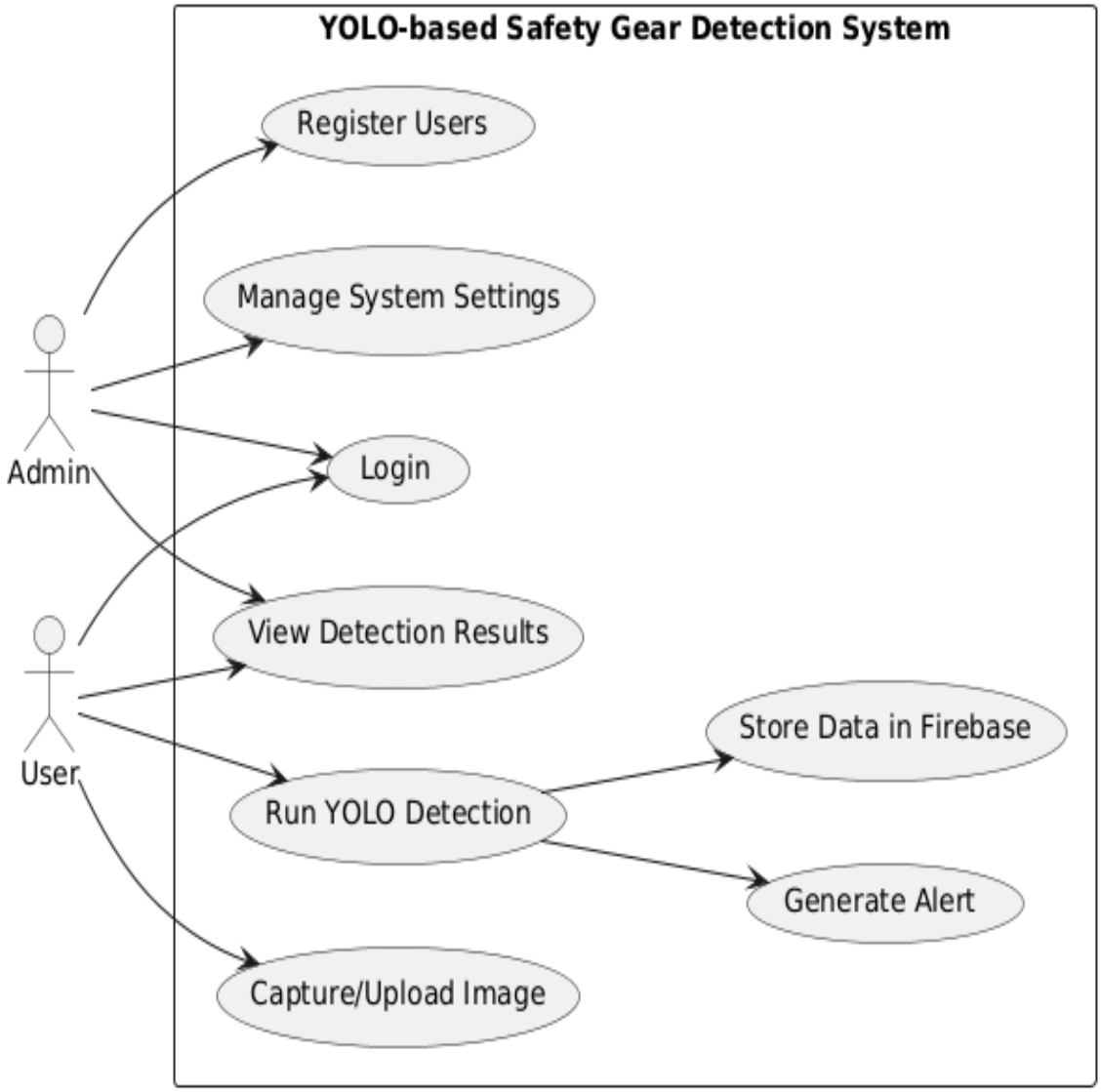


Figure 4.3: **Use Case Diagram**

From the figure 4.3 presents the use case diagram for the YOLO-based Safety Gear Detection System, illustrating the interaction between two main actors: Admin and User. The Admin is responsible for key management tasks such as registering users, configuring system settings, logging in, and accessing detection reports. The User, on the other hand, can log in, upload or capture images, and initiate the detection process.After the YOLO model processes the input, the system automatically stores detection results in Firebase. If any PPE violations are found, alerts are generated to notify the relevant user. This use case model outlines how each actor interacts with the system to ensure real-time detection, efficient user access, and secure data management.

4.2.3 Class Diagram

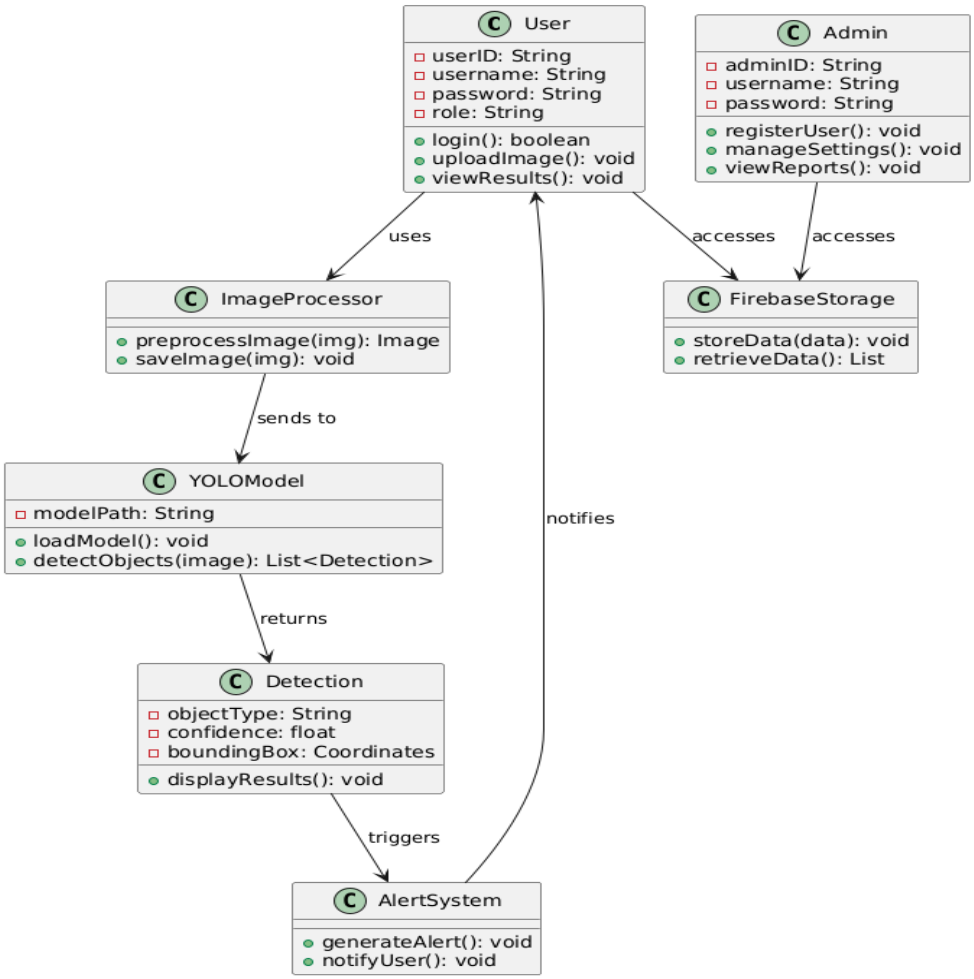


Figure 4.4: **Class Diagram**

From the figure 4.4 illustrates the class diagram of the YOLO-based Safety Gear Detection System, highlighting the main components and their interactions. The User and Admin classes manage login, image uploading, viewing results, and system administration. The ImageProcessor class handles preprocessing tasks such as resizing and saving input images. These processed images are passed to the YOLOModel class, which performs object detection to identify PPE.Detection results are structured within the Detection class, which includes details like object type, confidence level, and bounding box coordinates. If a safety violation is found, the AlertSystem generates and sends alerts to the user. All detection data and alerts are stored using the FirebaseStorage class, allowing both User and Admin roles to access reports and results efficiently. This diagram provides a clear overview of the system's object-oriented structure.

4.2.4 Sequence Diagram

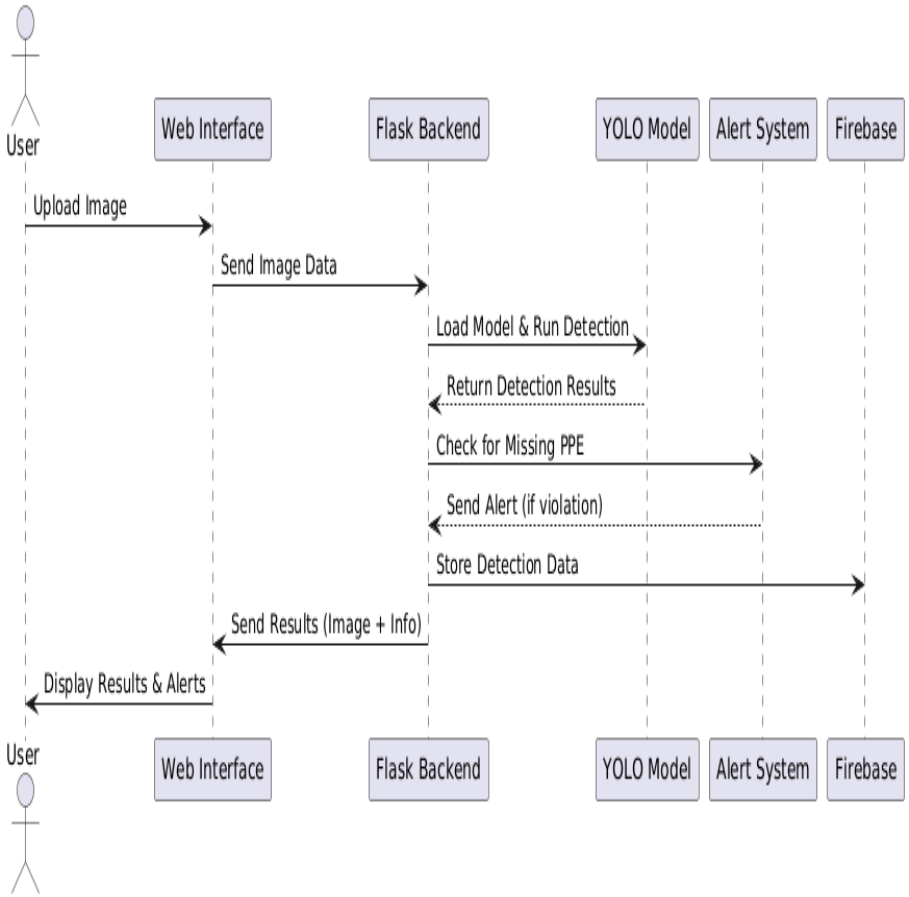


Figure 4.5: **Sequence Diagram**

From the figure 4.5 presents the sequence diagram depicting the step-by-step interaction between components in the YOLO-based safety gear detection system. The workflow starts when the user uploads an image through the web interface. This image is passed to the Flask backend, which serves as the processing unit. The backend then loads the YOLO model and performs object detection to identify whether the required PPE, such as helmets or vests, is present in the image.Once detection is completed, the results are sent back to the backend, where the system checks for any violations, such as missing or improperly worn safety gear. If a violation is identified, the alert system is triggered, and a warning is generated for the user. Simultaneously, the detection results, along with alert data, are stored in Firebase for logging and future analysis. Finally, the backend returns the processed image and relevant feedback to the web interface, where the user can view the annotated results and any generated alerts. This sequence ensures a streamlined and responsive safety compliance process.

4.2.5 Collaboration diagram

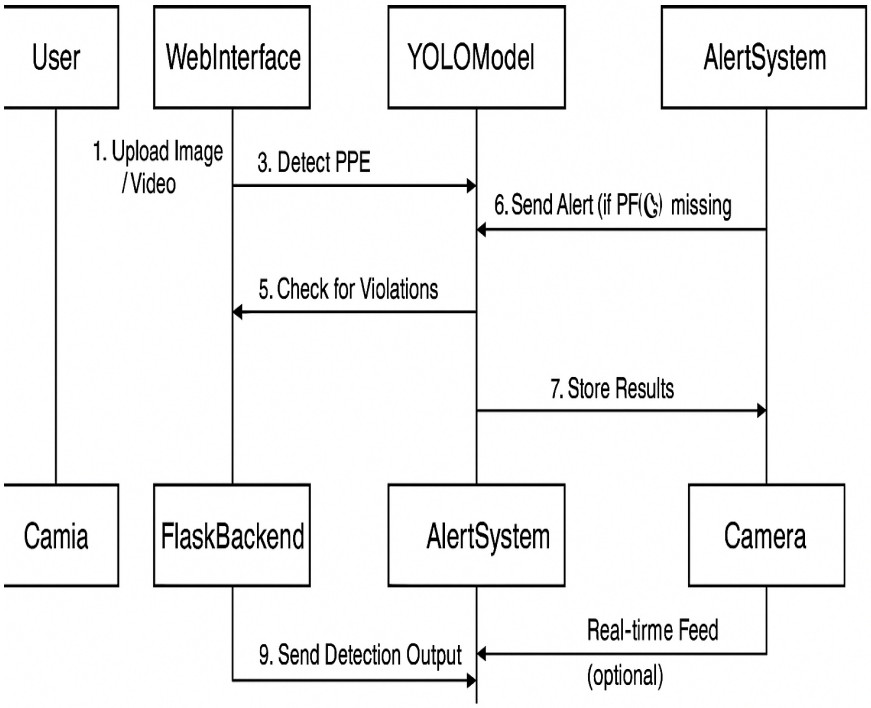


Figure 4.6: **Collaboration diagram**

From the figure 4.6 illustrates the collaboration diagram of the YOLO-based PPE detection system, showcasing the interaction between various components. The process starts when a user uploads an image or video through the web interface. This input is received by the Flask backend, which acts as the central controller for handling requests and coordinating system operations. The backend forwards the input to the YOLO model for PPE detection, where the model analyzes the content to identify safety gear such as helmets or vests.If any PPE is missing, the system detects the violation and triggers an alert through the connected alert module. Both the detection results and alerts are stored in the backend for record-keeping and future reference. Optionally, the system can also be integrated with a real-time camera feed to enhance live monitoring capabilities. Once processing is complete, the backend sends the output—along with any alerts—back to the web interface, where the user can easily view the results in an annotated format. This collaborative workflow ensures efficient and automated safety compliance.

4.2.6 Activity Diagram

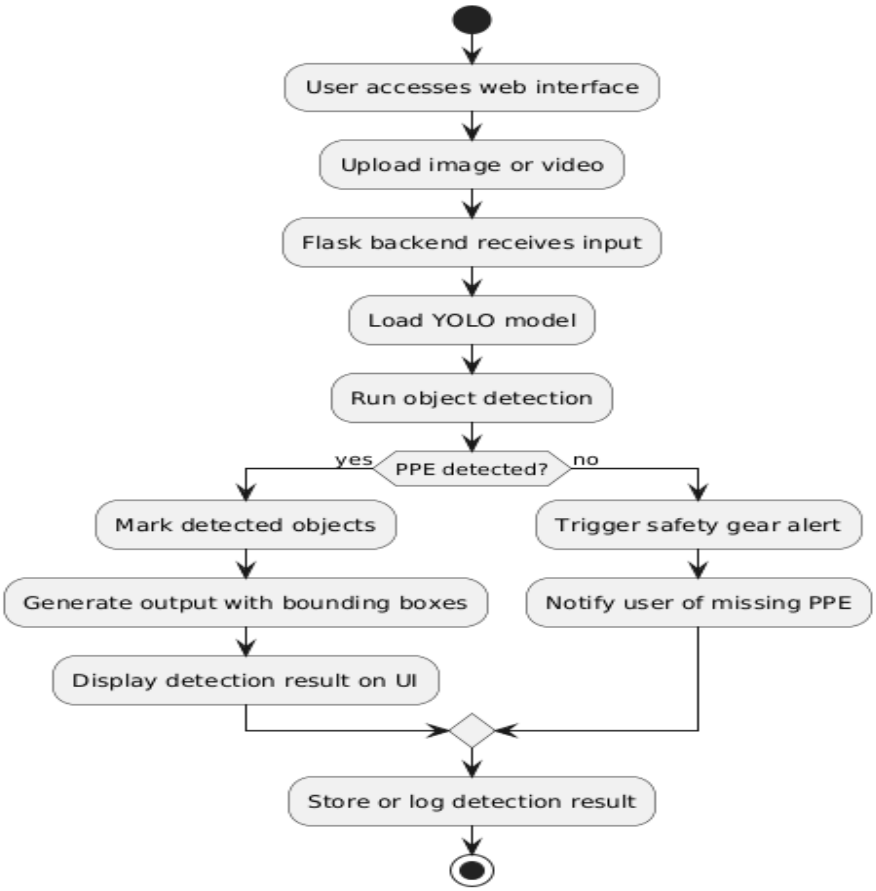


Figure 4.7: **Activity Diagram**

From the figure 4.7 illustrates the activity diagram of the YOLO-based PPE detection system. The process begins when a user accesses the web interface and uploads an image or video input. This input is sent to the Flask backend, which handles the request and loads the YOLO model to perform object detection. The model processes the visual data to detect whether essential PPE such as helmets or safety vests—is present.If PPE is detected, the system marks the identified objects with bounding boxes and displays the annotated results on the user interface. On the other hand, if PPE is missing or incomplete, the system triggers a safety alert and notifies the user in real-time. Additionally, it ensures the outcomes—whether compliant or not—are stored or logged for future analysis or audit purposes. This workflow supports automated safety compliance and promotes proactive workplace monitoring.

4.3 Algorithm & Pseudo Code

4.3.1 Algorithm

1. Start the system and import all required libraries.
2. Load the pre-trained YOLO object detection model.
3. Initialize the Flask application
4. Define allowed image file extensions and configure upload settings.
5. When a user uploads an image:
   - Validate the uploaded file format.
   - Save the image to the server.
6. Run YOLO object detection on the uploaded image:
   - Divide the image into grids.
   - Predict bounding boxes and classify detected objects.
   - Filter predictions based on confidence threshold
7. Save the detection result as a new image with bounding boxes and labels.
8. Convert the result image to base64 format for web display.
9. Render the result back to the user through the web interface.
10. End.

4.3.2 Pseudo Code

```
BEGIN
    Import necessary libraries (cv2, flask, ultralytics, etc.)

    Load YOLO model from local directory (e.g., "helmet.pt")

    Initialize Flask application
    Set UPLOAD_FOLDER and allowed file extensions

    DEFINE allowed_file(filename):
        RETURN filename has allowed extension

    ROUTE '/' :
        RENDER homepage with upload form
```

```
14
15      ROUTE '/upload' [POST]:
16          IF file is uploaded AND allowed_file(file.filename) THEN
17              Save uploaded image to server
18              Run YOLO model on the image with defined parameters
19              Save output image with detections
20              Convert image to base64
21              Display result to user in HTML
22          ELSE
23              Show error message
24
25      START Flask server (app.run)
26  END
```

4.4 Module Description

4.4.1 Module 1: Image/Video Upload Preprocessing

This module is responsible for accepting input from the user in the form of images or videos. Users can upload files via the web interface. Upon submission, the system performs preprocessing operations such as resizing, format checking, and converting the input to a compatible format for model processing. This ensures uniformity before feeding the media into the detection pipeline.

4.4.2 Module 2: Object Detection using YOLO

This is the core module where the pre-trained YOLO model performs detection of safety gear such as helmets, gloves, vests, goggles, and shoes. The input media is divided into grids and analyzed for the presence or absence of personal protective equipment. The detection is fast, accurate, and efficient, ensuring real-time feedback

4.4.3 Module 3: Web Display Alert System

This module handles displaying the detection results on the browser and generating alerts in case of any safety violations. It marks detected objects with labels and bounding boxes and notifies users if any mandatory equipment is missing. Alerts can be visual (on the dashboard) or sent to the concerned authority for swift action

4.5 Steps to execute/run/implement the project

4.5.1 Step1: Environment Setup and Installation

- Install Python (version 3.8 or higher recommended).
- Set up a virtual environment (optional but recommended).
- Install required libraries: opencv-python, flask, ultralytics, numpy, etc.
- Download or clone the project folder from the repository.
- Place the pre-trained YOLO model (e.g., helmet.pt) into the specified directory.
- Test the basic Flask application with a dummy route to confirm proper installation.

4.5.2 Step2:YOLO Integration and Backend Logic

- Import the YOLO model using from ultralytics import YOLO.
- Load the trained model: model = YOLO(”helmet.pt”).
- Define logic to handle image or video uploads.
- Run YOLO detection on the uploaded file.
- Filter out detections with low confidence scores.
- Save the output image with bounding boxes and labels for visualization.

4.5.3 Step3:Flask Frontend and Real-Time Execution

- Design HTML templates with upload form and image preview section.
- Link backend logic to frontend using Flask routes (@app.route()).
- Convert detected image to base64 format for web display.
- Render the results (image + detection labels) on the browser.
- Optionally, connect a live webcam stream using OpenCV for real-time monitoring.
- Run the Flask app using app.run(debug=True) and access it via localhost

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design

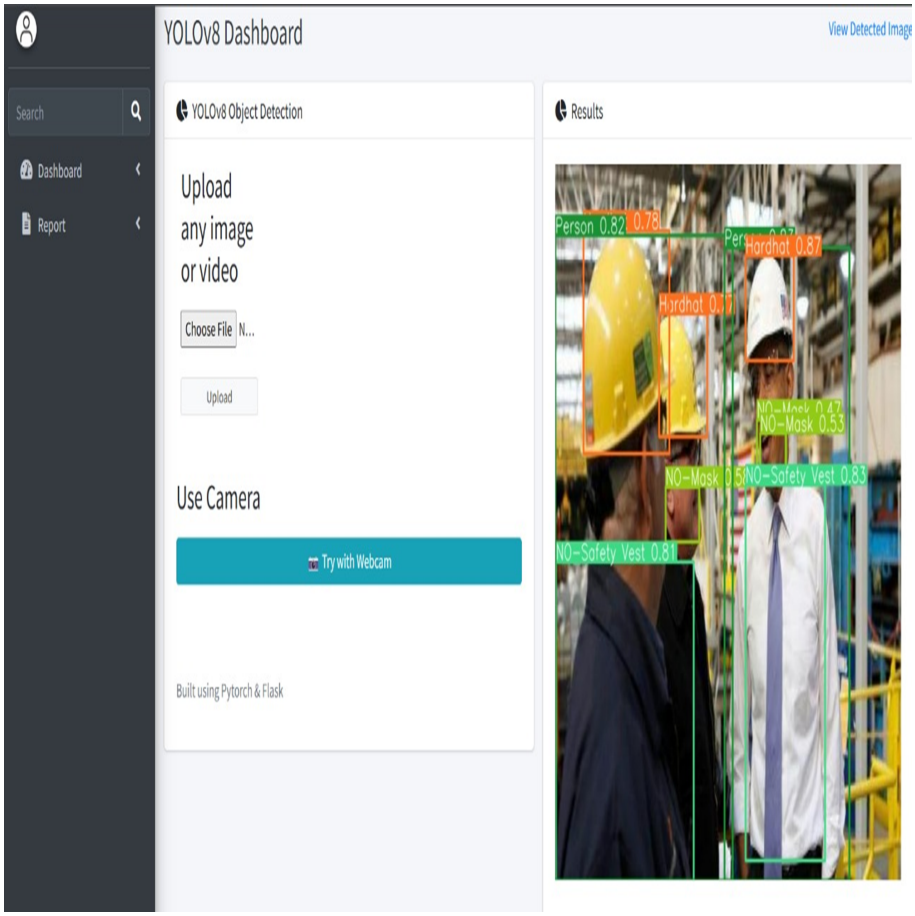


Figure 5.1: **Input Design**

From the figure 5.1 input to the system is an image or video uploaded by the user via the web interface. This input contains visual data from a workplace or environment where safety gear usage needs to be verified. The user selects a file, which is then sent to the Flask backend for further processing. The input design ensures that:

- The file type is validated (e.g., JPG, PNG, MP4).
- The size of the input is within limits for efficient processing.
- The interface allows easy selection and uploading of media.

5.1.2 Output Design

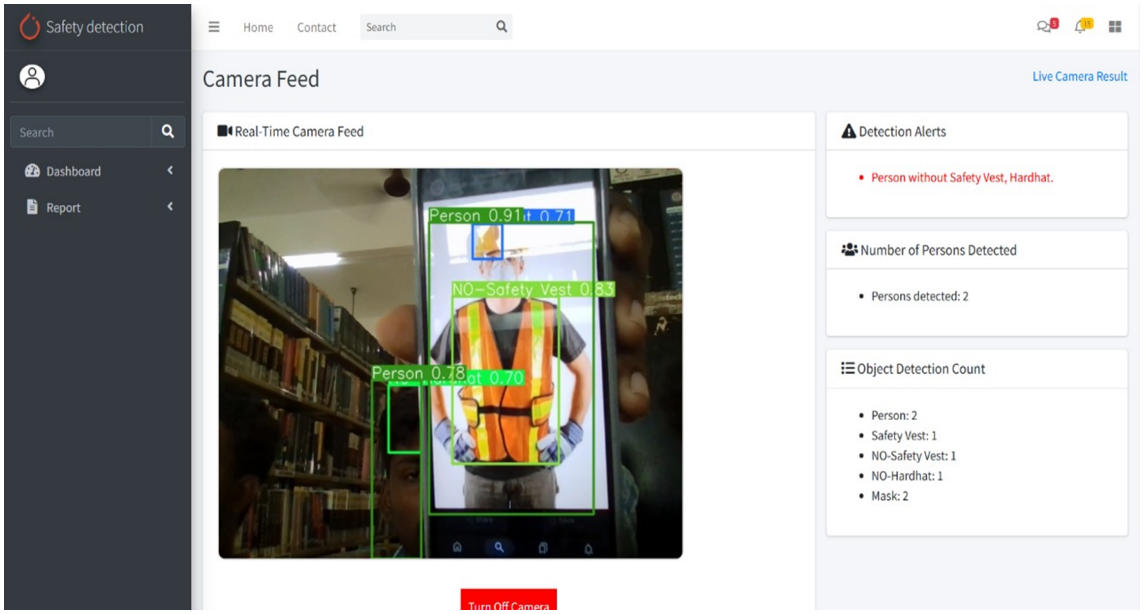


Figure 5.2: **Output Design**

From the figure 5.2 output of the system includes:

1. Detection Results: The system processes the input using the YOLO model and displays detected PPE objects with bounding boxes on the uploaded image or video frame.
2. Alerts: If any safety gear (like helmets or vests) is missing, the system generates an alert message for the user.
3. Logs and Reports: The results and alerts are stored and can be accessed by the admin for review and further action.

The outputs are designed to be:

- Clear and visual – bounding boxes are drawn on detected objects.
- Informative – alerts include details about what PPE is missing.
- Accessible – detection history and alert logs can be reviewed later.

5.2 Testing

5.2.1 Unit Testing

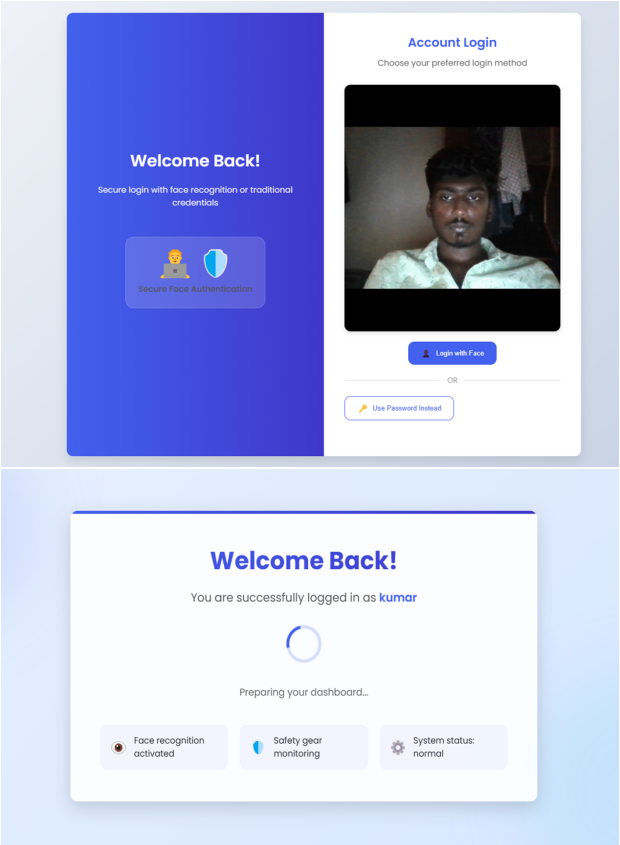


Figure 5.3: **Unit Testing**

From the figure 5.3, unit testing was thoroughly conducted to verify the functionality of individual components within the system in complete isolation. This level of testing played a crucial role in ensuring that each module performed its intended operations correctly before being integrated into the overall system architecture. In particular, the face recognition module underwent comprehensive validation to assess the accuracy and performance of its core processes—namely, face detection, feature encoding, and face matching. Each of these functionalities was tested independently under a variety of conditions to ensure consistent and expected outcomes. This rigorous testing approach facilitated the early identification of bugs and logical flaws, thereby improving system reliability. Additionally, it contributed to optimizing the module's performance and maintaining the robustness and quality of the entire application, especially in real-time scenarios where accurate recognition is critical.

5.2.2 Integration Testing

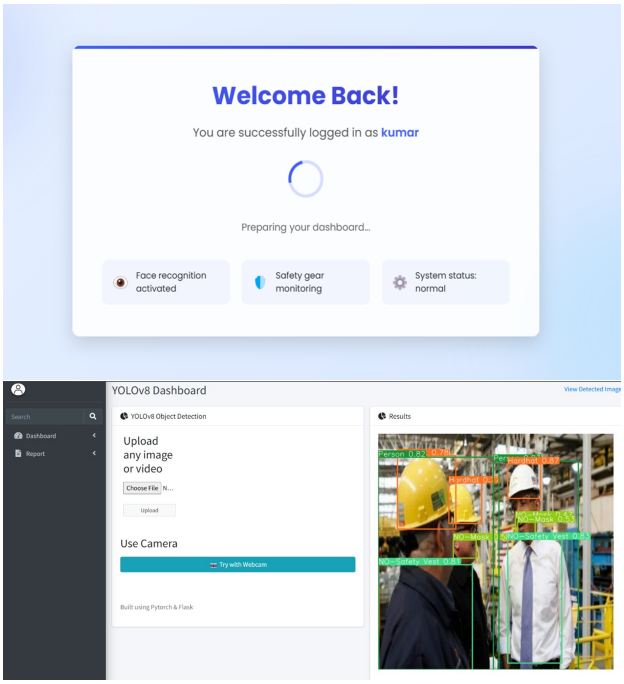


Figure 5.4: **Integration Testing**

From the figure 5.4, the face recognition and authentication module was rigorously tested to validate the effectiveness, reliability, and efficiency of the login mechanism implemented within the system. This module serves as a critical security component, ensuring that access is granted only to authorized individuals through the use of advanced facial detection and recognition algorithms. During the testing phase, facial images captured under various lighting conditions and orientations were utilized to assess the system's accuracy. The process involved detecting faces from the input, encoding distinct facial features, and matching them against a pre-registered user database. The results demonstrated a high level of precision in authentication, with minimal occurrences of false positives or negatives. This extensive evaluation confirmed the robustness of the login mechanism and significantly contributed to enhancing the overall security of the system, making it highly dependable for real-time applications.

5.2.3 Performance Evaluation

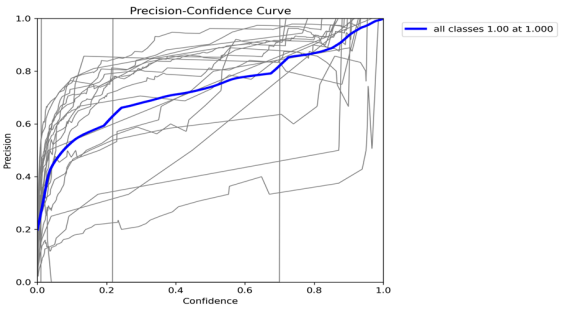


Figure 5.5: **Precision-Confidence Curve**

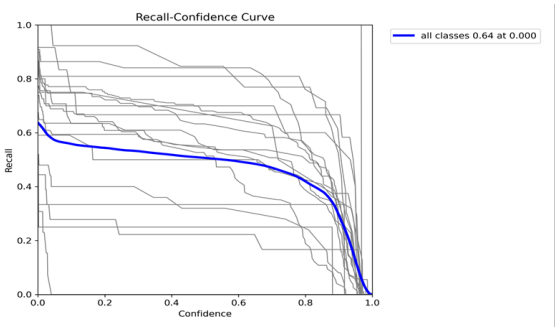


Figure 5.6: **Recall-Confidence Curve**

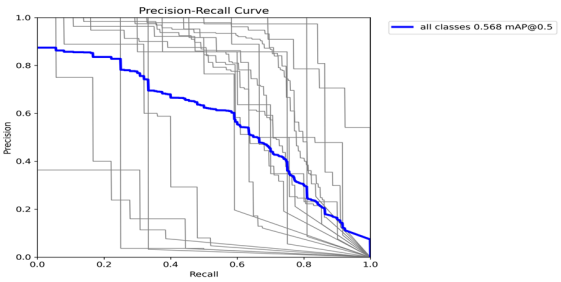


Figure 5.7: **Precision-Recall Curve**

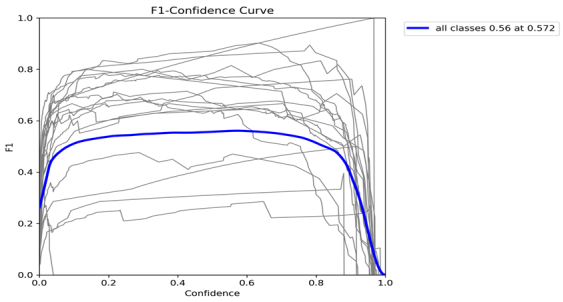


Figure 5.8: **F1-Confidence Curve**

From the figures 5.5 to 5.8, various performance evaluation metrics are analyzed for the YOLO-based object detection system. The Precision-Confidence Curve (Figure 5.5) illustrates how the precision varies with different confidence thresholds, showing high precision at lower confidence values, which is ideal for detecting true positives while minimizing false positives. The Recall-Confidence Curve (Figure 5.6) shows a gradual decline in recall with increasing confidence, which is expected as stricter thresholds may miss certain detections. The Precision-Recall Curve (Figure 5.7) provides a holistic view of the model's accuracy in identifying relevant objects, maintaining a balanced trade-off between precision and recall. Lastly, the F1-Confidence Curve (Figure 5.8) reflects the harmonic mean of precision and recall, confirming the model's overall effectiveness in achieving both accuracy and completeness in detection. These evaluations highlight the robustness and reliability of the system in various confidence intervals, proving it suitable for real-world deployment.

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The proposed intelligence safety detection system, leveraging the YOLO object detection algorithm, significantly enhances the accuracy and speed of safety gear recognition in real-time industrial environments. Compared to traditional manual monitoring or rule-based systems, YOLO offers superior performance due to its capability to detect multiple objects in a single forward pass through the network. This ensures that safety violations such as the absence of helmets, gloves, or safety vests are identified instantly with high precision. By deploying a pre-trained YOLO model fine-tuned on safety gear datasets, the system reduces false positives and negatives, ensuring that only genuine violations are flagged. This efficiency is crucial in workplaces where quick decisions are vital for preventing accidents and ensuring worker safety.

Furthermore, the integration of this system with a Flask-based web interface adds to its usability and deployment flexibility. The real-time video stream is processed seamlessly, and alerts are generated without noticeable latency. This ensures that supervisors or safety officers are notified the moment a breach is detected. The lightweight nature of YOLOv4 or YOLOv5 allows the system to run even on resource-constrained edge devices, reducing the dependency on high-performance servers. Additionally, the automation of monitoring reduces the burden on human inspectors and minimizes human error, leading to more consistent safety compliance. Overall, the system is not only efficient in terms of computational performance but also in promoting a proactive safety culture in the workplace.

6.2 Comparison of Existing and Proposed System

Existing system:

The existing safety gear detection systems often use traditional machine learning

models like Decision Trees. These models rely on manually extracted features such as color, shape, or texture for classification. While they are simple and interpretable, they perform poorly in complex industrial environments with changing lighting, background noise, or object occlusion. They are not suitable for real-time applications and typically support only static image input. This limits their effectiveness in fast-paced industrial settings where safety decisions need to be immediate. Furthermore, their accuracy is low when detecting multiple objects, making them unreliable for ensuring consistent safety compliance.

Proposed system:

The proposed system utilizes the YOLO (You Only Look Once) deep learning model to detect safety gear like helmets, gloves, and vests in real-time. It replaces manual inspection with automated monitoring through image uploads or live camera feeds using a Flask-based web interface. YOLO's fast and accurate detection allows the system to process multiple objects simultaneously, making it effective in challenging industrial environments. This solution reduces human error, improves safety compliance, and supports instant alerts for PPE violations. Overall, it enhances workplace safety with greater efficiency, scalability, and ease of use compared to traditional manual or rule-based systems.

6.3 Comparative Analysis-Table

S.No	Parameters	Decision Tree Based	YOLO-Based (Proposed)
1	Speed	Slow	Fast
2	Accuracy	Low	High
3	Real-time Detection	Not Supported	Supported
4	Efficiency	Moderate	High
5	Scalability	Limited	Easily Scalable
6	Flexibility	Low	High

Table 6.1: **Comparative Analysis of Decision Tree Based vs YOLO-Based Approaches**

6.4 Comparative Analysis-Graphical Representation and Discussion

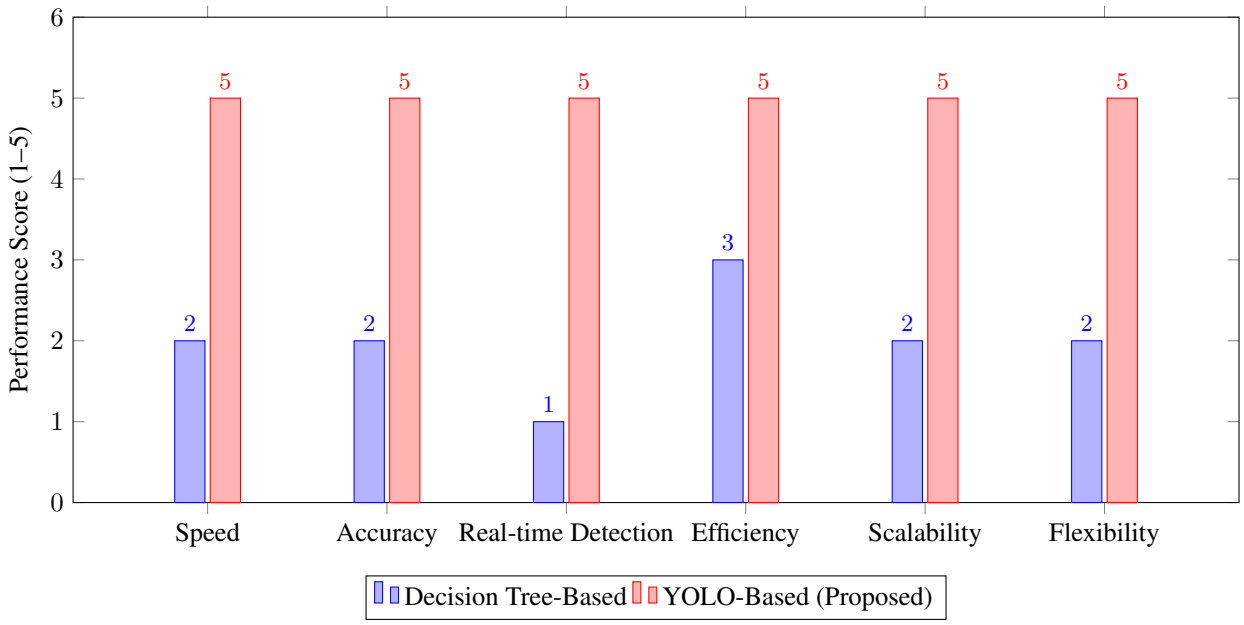


Figure 6.1: **Graphical Comparison of Decision Tree vs YOLO-Based System**

From the figure 6.1, it is observed that the YOLO-based system significantly outperforms the Decision Tree-based system across all the evaluated performance metrics. The comparison includes crucial parameters such as speed, accuracy, real-time detection capability, efficiency, scalability, and flexibility all of which are vital in determining the robustness of an object detection framework. The YOLO-based system consistently achieves the maximum score of 5 in each category, demonstrating its high precision and responsiveness. In contrast, the Decision Tree-based system shows lower scores ranging from 1 to 3, indicating limited capability in handling complex, real-time detection tasks. This comparative analysis clearly highlights the enhanced performance, adaptability to varying environments, and real-time processing strength of the YOLO-based approach, making it a more suitable choice for dynamic and high-risk scenarios such as workplace safety monitoring.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Summary

In this project, we developed an intelligent safety gear detection system that leverages the capabilities of deep learning and computer vision to enhance workplace safety in industrial environments. The system is built using the YOLO (You Only Look Once) object detection model, known for its exceptional speed and accuracy in real-time applications. It automatically identifies whether workers are wearing essential personal protective equipment (PPE) such as helmets, vests, and gloves. By integrating the YOLO model with a user-friendly Flask web application, the system enables real-time image or video processing and instantly highlights any safety violations. This approach addresses the limitations of manual monitoring methods, such as delays, inefficiencies, and human error.

The proposed system offers scalability, flexibility, and robustness, making it suitable for deployment in construction sites, manufacturing units, and other hazardous workspaces. It supports image uploads as well as live camera feeds, delivering immediate visual feedback with bounding boxes and labels for detected objects. The use of deep learning ensures adaptability to various lighting conditions and complex backgrounds, enhancing detection accuracy. Overall, this system improves safety compliance, minimizes the risk of workplace accidents, and contributes to creating a smarter and more secure industrial environment.

7.2 Limitations

Despite the significant advantages and improvements offered by the proposed safety gear detection system using YOLO and Flask, there are certain limitations that must be acknowledged. One of the primary challenges is the dependency on

high-quality and well-lit images or videos for accurate detection. In industrial environments where lighting conditions may vary, or camera angles are not optimal, the detection performance can degrade, leading to false positives or missed detections. Additionally, the model used is pretrained for specific classes such as helmet, vest, gloves, goggles, and shoes. If a new type of safety gear needs to be monitored, the system requires retraining with annotated datasets, which may not always be readily available.

Another limitation lies in the computational requirements for real-time detection. Running YOLO on edge devices or older systems may lead to lag or delayed processing, especially when handling high-resolution video feeds. Moreover, the system is primarily image-based and may not fully account for dynamic human behavior such as temporary removal of safety gear or partially obscured PPE. This can be a limitation in truly assessing worker compliance. There are also concerns related to privacy and data security in using surveillance footage for safety monitoring, which may raise ethical and legal questions in certain workplaces. Continuous improvements, model updates, and integration with other IoT systems could help overcome these challenges in the future.

7.3 Future Enhancements

While the current implementation of the safety gear detection system using YOLO and Flask serves its intended purpose effectively, there is scope for future enhancements to make it more robust and intelligent. One key improvement is the integration of real-time alert mechanisms to instantly notify supervisors when violations occur—through SMS, email, or app notifications. Incorporating audio or visual warnings in the workplace can also provide immediate feedback to workers. Enhancing the system with multi-class detection could enable monitoring of additional violations such as zone restrictions, improper posture, or unauthorized access.

Future versions could integrate IoT devices and sensors for a more comprehensive surveillance setup. For instance, RFID-based wearables can help validate PPE usage, while environmental sensors can monitor conditions like temperature or gas leaks. Cloud-based storage and analytics can further enable detailed compliance reporting, trend tracking, and hazard prediction using machine learning. Adding multilingual support and intuitive dashboards would also make the system suitable for diverse industries and wider adoption.

Chapter 8

SUSTAINABLE DEVELOPMENT GOALS (SDGs)

8.1 Alignment with SDGs

The proposed intelligent safety gear detection system demonstrates a strong alignment with the United Nations Sustainable Development Goals (SDGs), primarily contributing to SDG 3 (Good Health and Well-Being), SDG 8 (Decent Work and Economic Growth), and SDG 9 (Industry, Innovation, and Infrastructure). By ensuring workers wear mandatory protective equipment such as helmets, gloves, and vests, the system reduces the risk of injuries and promotes health and safety in industrial environments. This directly supports SDG 3 by protecting the well-being of employees in hazardous workplaces. Additionally, the automation of safety monitoring minimizes human effort and errors, promoting safer and more efficient working conditions in line with SDG 8. The integration of advanced technologies like deep learning, computer vision, and real-time alert systems encourages innovation and the modernization of industrial practices, contributing to SDG 9. Hence, the system not only addresses current safety challenges but also promotes a sustainable, technology-driven future in industry.

8.2 Relevance of the Project to Specific SDG

The proposed intelligent safety gear detection system directly supports Sustainable Development Goal 3 (Good Health and Well-Being) by promoting a safer work environment and reducing the likelihood of occupational injuries through automated monitoring of safety gear compliance. It is especially relevant in high-risk industries such as construction, manufacturing, and logistics, where failure to wear protective equipment can lead to serious accidents. By leveraging real-time object detection using advanced deep learning models, the project ensures that safety protocols are

followed consistently, even in the absence of human supervision. Furthermore, it aligns with SDG 9 (Industry, Innovation, and Infrastructure) by integrating innovative technologies into traditional industrial safety frameworks, thereby fostering safer, smarter, and more sustainable operational practices. The project represents a practical, tech-driven solution to one of the most pressing issues in the industrial sector, showing clear alignment with global sustainability and safety goals.

8.3 Potential Social and Environmental Impact

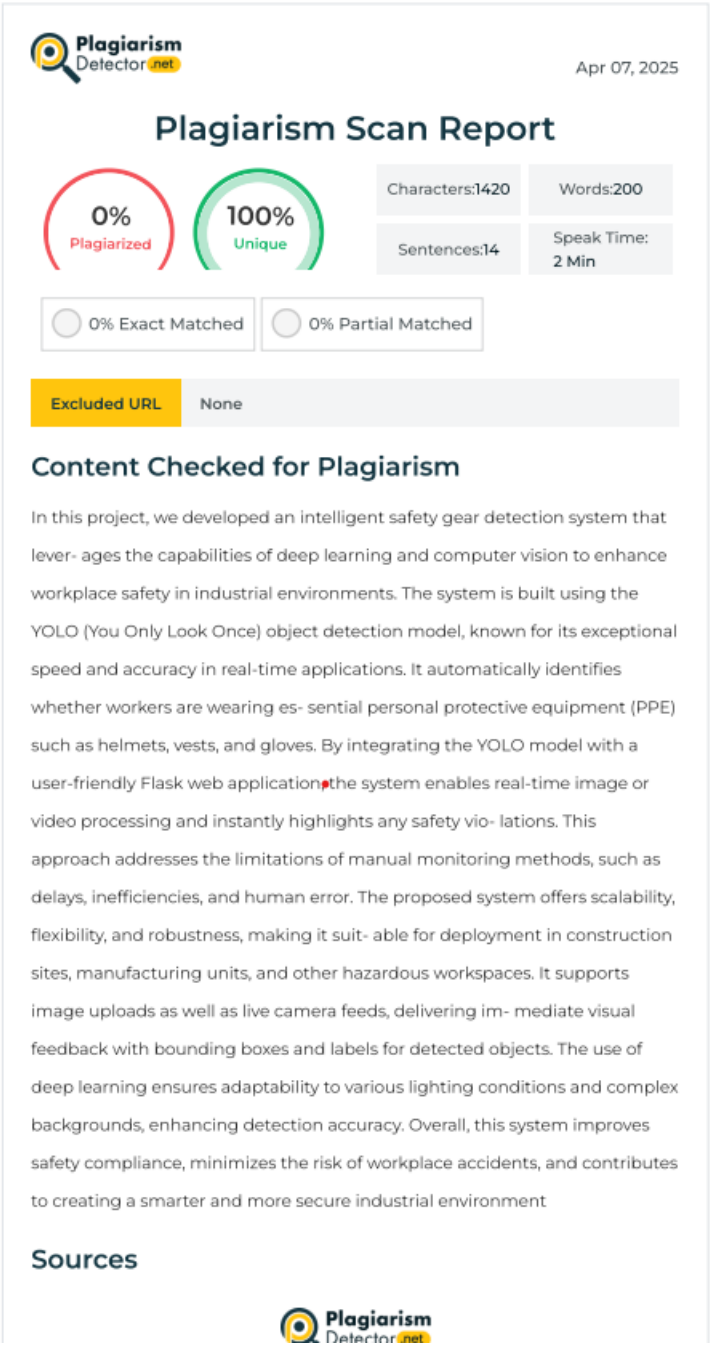
The proposed intelligent safety gear detection system contributes significantly to both social and environmental well-being. Socially, it enhances worker protection in high-risk environments by ensuring the correct usage of safety equipment, thereby reducing accidents and promoting a safer workplace culture. This not only improves employee well-being but also reduces costs related to injuries and insurance. Environmentally, the system reduces reliance on manual safety checks and paperwork, promoting digital workflows that minimize resource waste. By supporting automation and real-time monitoring, it encourages more sustainable and responsible industrial operations.

8.4 Economic Feasibility

Software Costs: All software and libraries used — including Python, Flask, OpenCV, PyTorch, YOLOv8, face_recognition, Firebase, Google Drive, and LabelImg — were free and open-source. Development and training were done using free tools like VS Code and Google Colab.
Hardware Costs: No special hardware was required; a standard laptop with basic specifications sufficed, resulting in zero additional hardware expense.
Training Costs: Skills were acquired through free online tutorials and official documentation at no cost.
Publication Costs: 9,500 was paid for IEEE paper publication, covering submission and processing fees.

Chapter 9

PLAGIARISM REPORT



Plagiarism Detector .net

Apr 07, 2025

Plagiarism Scan Report

0% Plagiarized

100% Unique

Characters:1420

Words:200

Sentences:14

Speak Time: 2 Min

0% Exact Matched

0% Partial Matched

Excluded URL None

Content Checked for Plagiarism

In this project, we developed an intelligent safety gear detection system that lever- ages the capabilities of deep learning and computer vision to enhance workplace safety in industrial environments. The system is built using the YOLO (You Only Look Once) object detection model, known for its exceptional speed and accuracy in real-time applications. It automatically identifies whether workers are wearing es- sential personal protective equipment (PPE) such as helmets, vests, and gloves. By integrating the YOLO model with a user-friendly Flask web application the system enables real-time image or video processing and instantly highlights any safety vio- lations. This approach addresses the limitations of manual monitoring methods, such as delays, inefficiencies, and human error. The proposed system offers scalability, flexibility, and robustness, making it suit- able for deployment in construction sites, manufacturing units, and other hazardous workspaces. It supports image uploads as well as live camera feeds, delivering im- mediate visual feedback with bounding boxes and labels for detected objects. The use of deep learning ensures adaptability to various lighting conditions and complex backgrounds, enhancing detection accuracy. Overall, this system improves safety compliance, minimizes the risk of workplace accidents, and contributes to creating a smarter and more secure industrial environment

Sources

Plagiarism Detector .net

Figure 9.1: **Plagiarism Report**

Chapter 10

SOURCE CODE

10.1 Source Code

```
import argparse
import io
from PIL import Image
import datetime
import torch
import cv2
import numpy as np
from flask import Flask, render_template, request, redirect, send_file, url_for, Response, jsonify
import os
import time
from ultralytics import YOLO

app = Flask(__name__)
# Simulated alert messages storage

alert_messages = []
# Detection categories
safety_violation_objects = ["NO-Hardhat", "NO-Mask", "NO-Safety Vest"]
heavy_machinery_objects = ["dump truck", "machinery", "truck", "trailer"]
safety_gear_objects = ["Hardhat", "Mask", "Safety Vest"]

# Store the latest detection message
latest_detection = {"detected": False, "message": "", "objects": [], "timestamp": 0}


@app.route('/get_navbar_messages')
def get_navbar_messages():
    return jsonify(alert_messages)
@app.route("/")
def hello_world():
    return redirect(url_for('home'))

@app.route("/home")
def home():
    return render_template("home.html")
@app.route("/report")
def report():
    return render_template("report.html")
```

```
39
40 @app.route("/index")
41 def index():
42     if "image_path" in request.args:
43         image_path = request.args["image_path"]
44         return render_template("index.html", image_path=image_path)
45     return render_template("index.html")
46
47 @app.route("/index", methods=["GET", "POST"])
48 def predict_img():
49     if request.method == "POST":
50         if 'file' in request.files:
51             f = request.files['file']
52             basepath = os.path.dirname(__file__)
53             filepath = os.path.join(basepath, 'uploads', f.filename)
54             print("Upload folder is ", filepath)
55             f.save(filepath)
56
57             file_extension = f.filename.rsplit('.', 1)[1].lower()
58
59             if file_extension == 'jpg':
60                 img = cv2.imread(filepath)
61                 model = YOLO('best.pt')
62                 detections = model(img, save=True)
63
64                 detected_objects = []
65                 for result in detections[0].boxes.data:
66                     class_id = int(result[5])
67                     confidence = result[4].item()
68                     class_name = model.names[class_id]
69                     detected_objects.append({
70                         'name': class_name,
71                         'confidence': confidence * 100
72                     })
73
74                 folder_path = os.path.join(basepath, 'runs', 'detect')
75                 subfolders = [f for f in os.listdir(folder_path) if os.path.isdir(os.path.join(
                        folder_path, f))]
76                 latest_subfolder = max(subfolders, key=lambda x: os.path.getctime(os.path.join(
                        folder_path, x)))
77                 relative_image_path = os.path.relpath(os.path.join(folder_path, latest_subfolder, f.
                        filename), 'static/assets')
78
79                 return render_template('index.html', image_path=relative_image_path, media_type='
                        image', detected_objects=detected_objects)
80
81             elif file_extension == "mp4":
82                 video_path = filepath
83                 cap = cv2.VideoCapture(video_path)
84                 frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
```

```
                frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

                fourcc = cv2.VideoWriter_fourcc(*"mp4v")
                out = cv2.VideoWriter("output.mp4", fourcc, 30.0, (frame_width, frame_height))
                model = YOLO("best.pt")

                while cap.isOpened():
                    ret, frame = cap.read()
                    if not ret:
                        break

                    results = model(frame, save=True)
                    res_plotted = results[0].plot()
                    cv2.imshow("Result", res_plotted)
                    out.write(res_plotted)

                    if cv2.waitKey(1) == ord("q"):
                        break

                return render_template('index.html', video_path='output.mp4', media_type='video')

    return render_template("index.html", image_path="", media_type='image')


@app.route('/camera')
def camera():
    return render_template('camera.html')

@app.route("/<path:filename>")
def display(filename):
    if file_extension == "jpg":
        return send_file(image_path, mimetype="image/jpeg")
    elif file_extension == "mp4":
        return send_file(image_path, mimetype="video/mp4")
    else:
        return "Invalid file format"

def get_frame():
    video = cv2.VideoCapture("output.mp4")
    while True:
        success, frame = video.read()
        if not success:
            break
        else:
            ret, buffer = cv2.imencode(".jpg", frame)
            frame = buffer.tobytes()

        yield (
            b"--frame\r\n"
            b"Content-Type: image/jpeg\r\n\r\n" + frame + b"\r\n\r\n"
```

```
        )
        time.sleep(0.1)

@app.route("/video_feed")
def video_feed():
    return Response(get_frame(), mimetype="multipart/x-mixed-replace; boundary=frame")
# Store object detection data
detection_reports = []

# Store time spent in the whole day (for example, in seconds)
time_spent_today = 0

# Store person counts for the day
person_counts = []

# Modify your webcam feed to track time and person counts
@app.route("/webcam_feed")
def webcam_feed():
    global latest_detection, time_spent_today
    cap = cv2.VideoCapture(0)
    start_time = time.time()

    def generate():
        global latest_detection,time_spent_today
        model = YOLO("best.pt")

        while True:
            success, frame = cap.read()
            if not success:
                break

            img = Image.fromarray(frame)
            results = model(img, save=True)

            res_plotted = results[0].plot()
            img_BGR = cv2.cvtColor(res_plotted, cv2.COLOR_RGB2BGR)

            # Remove the count of the objects from the frame
            # Previously, you were displaying the count here
            # Now, the object count display is removed

            ret, buffer = cv2.imencode(".jpg", img_BGR)
            frame = buffer.tobytes()

            # Generate alert message based on detection
            detected_objects = [model.names[int(detection[5])] for detection in results[0].boxes.
                data]
            message = generate_alert_message(detected_objects)

            if message:
```

```
                latest_detection = {
                    "detected": True,
                    "message": message,
                    "objects": detected_objects,
                    "timestamp": time.time()
                }
            else:
                latest_detection = {"detected": False, "message": "", "objects": [], "timestamp":
                    time.time()}
            person_count = detected_objects.count("Person")

            # Store the detection for report purposes
            detection_reports.append({
                'time': time.time() - start_time,  # time since webcam feed started
                'detected_objects': detected_objects,
                'person_count': person_count
            })

            person_counts.append(person_count)

            # Calculate time spent today
            time_spent_today = time.time() - start_time

            yield (
                b"--frame\r\n"
                b"Content-Type: image/jpeg\r\n\r\n" + frame + b"\r\n\r\n"
            )

    return Response(generate(), mimetype="multipart/x-mixed-replace; boundary=frame")


# Simulating new alerts (replace with actual detection logic)
import threading, time
def simulate_alerts():
    while True:
        add_alert_message(" no Safety gear detected!")
        time.sleep(5)  # Every 10 seconds

# Run background simulation
threading.Thread(target=simulate_alerts, daemon=True).start()

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Flask app exposing yolov8 models")
    parser.add_argument("--port", default=5000, type=int, help="port number")
    args = parser.parse_args()
    model = YOLO("best.pt")
    app.run(host="0.0.0.0", port=args.port, debug=True)
```

References

[1] J. Brownlee, ”A Gentle Introduction to Object Recognition Using Deep Learning,” Machine Learning Mastery, May 22, 2019. [Online]. Available: https://machinelearningmastery.com/object-recognition-with-deep-learning/. [Accessed: 20-Feb-2025].

[2] J. Redmon, S. Divvala, R. Girschick, and A. Farhadi, ”You Only Look Once: Unified, Real-Time Object Detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788.

[3] Y. Zhang, Y. Jiang, and X. Wang, ”Safety Helmet Detection Based on Improved YOLO Algorithm,” in *IEEE Access*, vol. 8, pp. 110478-110487, 2020.

[4] H. Li, J. Zhang, and M. Zhou, ”Deep learning-based personal protective equipment detection framework in industrial environments,” *Sensors*, vol. 21, no. 2, p. 512, 2021.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, ”Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580-587.

[6] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, Y. Sheikh, ”OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172-186, 2021.

[7] Y. Li, H. Wei, Z. Han, J. Huang, and W. Wang, ”Deep learning-based safety helmet detection in engineering management based on convolutional neural networks,” *Advances in Civil Engineering*, vol. 2020, pp. 1-10, 2020.

[8] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, ”YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 7464-7475.

[9] X. Shao, S. Liu, X. Li, Y. Yang, Z. Yuan, H. Li, and Z. Lyu, "FMYOLOv7: an improved detection method for mine personnel helmet," *J. Electron. Imaging*, vol. 32, no. 3, pp. 033013-033013, 2023.

[10] A. Aboah, B. Wang, U. Bagci, and Y. Adu-Gyamfi, "Real-time Multi-Class Helmet Violation Detection Using Few-Shot Data Sampling Technique and YOLOv8," in *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023, pp. 5350-5358, IEEE/CVF.

[11] W. Chen, H. Huang, and S. Peng, "YOLO-face: a real-time face detector," *Vis Comput*, vol. 37, pp. 805–813, 2021.

[12] N. Nath, A. Behzadan, and S. Paal, "Deep learning for site safety: Real-time detection of personal protective equipment," *Automation in Construction*, vol. 112, p. 103085, 2020.

[13] G. Gallo, F. Di Rienzo, P. Ducange, V. Ferrari, A. Tognetti, and C. Vallati, "A Smart System for Personal Protective Equipment Detection in Industrial Environments Based on Deep Learning," in *International Conference on Smart Computing (SMARTCOMP)*, 2021, pp. 222-227, IEEE.

[14] Q. Zhou, J. Qin, and X. Xiang, "Algorithm of Helmet Wearing Detection Based on AT-YOLO Deep Model," *Computers, Materials & Continua*, vol. 69, no. 1, pp. 159–174, 2021.

Acceptance Letter

ICSSAS

3rd International Conference on
Self Sustainable Artificial Intelligence Systems
ICSSAS 2025
11-13, JUNE 2025

https://icssat.com/2025/ | icssat.contact@gmail.com

Letter of Acceptance

Author Name: V. Ashok Kumar, Y. Sai Datta Sri Krishna Tarun, V. Karthikeya Santosh Reddy, R . Mahesh Murari

Affiliation Details: Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology

Dear Author:

It is with great pleasure that we extend our warmest congratulations to you on the acceptance of the paper titled "Intelligence Safety Detection and Hazard Monitoring Using YOLO Algorithm" - Paper ID: ICSSAS 0401 for presentation at the 3rd International Conference on Self Sustainable Artificial Intelligence Systems, scheduled to be held in M.P.Nachimuthu M.Jaganathan Engineering College, Erode, India from June 11th to June 13th, 2025.

Your submission was subjected to a rigorous review process, and the result that your paper has been selected for inclusion in our conference program. We believe that your contribution will greatly enrich the discussions and knowledge exchange at our event.

Your participation will undoubtedly contribute to the success of the 3rd International Conference on Applied Artificial Intelligence and Computing.

Once again, congratulations on your acceptance, and we anticipate your valuable contribution to our conference.

Yours' Sincerely,

Conference Chair ICSSAS